2016

Evidence-based Accountability Audits for Cloud Computing

Rubsamen, Thomas

Copyright Statement

EVIDENCE-BASED ACCOUNTABILITY AUDITS FOR CLOUD COMPUTING

by

THOMAS RÜBSAMEN

A thesis submitted to Plymouth University
in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

School of Computing, Electronics and Mathematics

In collaboration with
University of Applied Sciences Furtwangen, Germany

March 2016

Acknowledgements

THIS work was made possible by the funding from the European Comission and the Faculty of Computer Science at Furtwangen University. I wish to thank both organisations for their support.

I would also like to thank all team members at the Furtwangen Institute for Cloud and IT Security and Plymouth CSCAN group for their support, by critically reviewing my work and never hesitating to provide their invaluable feedback.

This thesis would not have been possible without the help and support of my Director of Studies, Prof. Dr. Christoph Reich. I thank him for his tireless support throughout the PhD process. His guidance had a lot of impact on the development of my research project and my personal development as well.

I would also like to thank my supervisors, Prof. Dr. Nathan Clarke and Prof. Dr. Martin Knahl, who have always provided me with critical feedback during my studies.

The work of this thesis was supported by multiple students, who helped me with the implemention of several prototypes and preparing conference demonstrations. Especially, I would like to thank Philipp Ruf and Timo Bayer for their hard work and tireless support.

Special thanks go to my significant other Carolin for always believing in me and supporting me especially in times when reaching my goals seemed to be very far off.

A very special *thank you* goes to my mother Edith and my father Peter for supporting me throughout my entire life.

Author's Declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award without prior agreement of the Graduate Sub-Committee.

Work submitted for this research degree at the Plymouth University has not formed part of any other degree either at Plymouth University or at another establishment

This work has been partly funded from the European Commission's Seventh Framework Programme (FP7/2007-2013), grant agreement 317550, Cloud Accountability Project - http://www.a4cloud.eu/ - (A4CLOUD).

Relevant scientific seminars and conferences were regularly attended at which work was often presented and several papers prepared for publication, details of which are listed in the appendices.

Word count of main body of thesis: **50874**

Signed: Thomas Pulsamen

Date: 18/10/2016

Abstract

Evidence-based Accountability Audits for Cloud Computing

Thomas Rübsamen, MSc

Cloud computing is known for its on-demand service provisioning and has now become mainstream. Many businesses as well as individuals are using cloud services on a daily basis. There is a big variety of services that ranges from the provision of computing resources to services such as productivity suites and social networks. The nature of these services varies heavily in terms of what kind of information is being out-sourced to the cloud provider. Often, that data is sensitive, for instance when Personal Identifiable Information (PII) is being shared by an individual. Also, businesses that move (parts of) their processes to the cloud are actively participating in a major paradigm shift from having data on-premise to transfering data to a third-party provider.

However, many new challenges come along with this trend, which are closely tied to the loss of control over data. When moving to the cloud, direct control over geographical storage location, who has access to it and how it is shared and processed is given up. Because of this loss of control, cloud customers have to trust cloud providers that they treat their data in an appropriate and responsible way. Cloud audits can be used to check how data has been processed in the cloud (i.e., by whom, for what purpose) and whether or not this happened in compliance with what has been defined in agreed-upon privacy and data storage, usage and maintenance (i.e., data handling) policies. This way, a cloud customer

can regain some of the control he has given up by moving to the cloud.

In this thesis, accountability audits are presented as a way to strengthen trust in cloud computing by providing assurance about the processing of data in the cloud according to data handling and privacy policies. In cloud accountability audits, various distributed evidence sources need to be considered. The research presented in this thesis discusses the use of various heterogeous evidence sources on all cloud layers. This way, a complete picture of the actual data handling practices that is based on hard facts can be presented to the cloud consumer. Furthermore, this strengthens transparency of data processing in the cloud, which can lead to improved trust in cloud providers, if they choose to adopt these mechanisms in order to assure their customers that their data is being handled according to their expectations. The system presented in this thesis enables continuous auditing of a cloud provider's adherence to data handling policies in an automated way that shortens audit intervals and that is based on evidence that is produced by cloud subsystems.

An important aspect of many cloud offerings is the combination of multiple distinct cloud services that are offered by independent providers. Data is thereby freuqently exchanged between the cloud providers. This also includes transborder flows of data, where one provider may be required to adhere to more strict data protection requirements than the others. The system presented in this thesis addresses such scenarios by enabling the collection of evidence at providers and evaluating it during audits.

Securing evidence quickly becomes a challenge in the system design, when information that is needed for the audit is deemed sensitive or confidential. This means that securing the evidence at-rest as well as in-transit is of utmost importance, in order not to introduce a new liability by building an insecure data heap. This research presents the identification of security and privacy protection requirements alongside proposed solutions that enable the development of an architecture for secure, automated, policy-driven and evidence-based accountability audits.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The research presented in this thesis revolves around two key issues in cloud computing: customer trust and compliance. Because of the loss of control that is inherently associated with moving data from an on-premise data center to a cloud service model, cloud customers have to trust cloud providers to handle their data appropriately, and that providers protect their data sufficiently. That trust needs to be earned by the cloud providers. This can only be achieved, if providers act transparently and according to regulation and user expectations. This includes the reporting of incidents, such as data breaches or data loss. Additionally, a cloud provider should be transparent with respect to the proper implementation of security and privacy controls. This can be achieved by providing comprehensive technical auditing capabilities that assess the level of privacy protection, security and accountability of a cloud provider. At the core of such an auditing system is the collection of data, that serves as evidence of (non-) compliance with data handling and processing policies as well as an indicator for the appropriateness of implemented privacy and security measures. The research presented in this thesis shows a novel approach to evidence-based, policy-driven and automated technical auditing of cloud environments, to enable demonstration of good data stewardship by cloud providers. The proposed system is called *Audit Agent System (AAS)*.

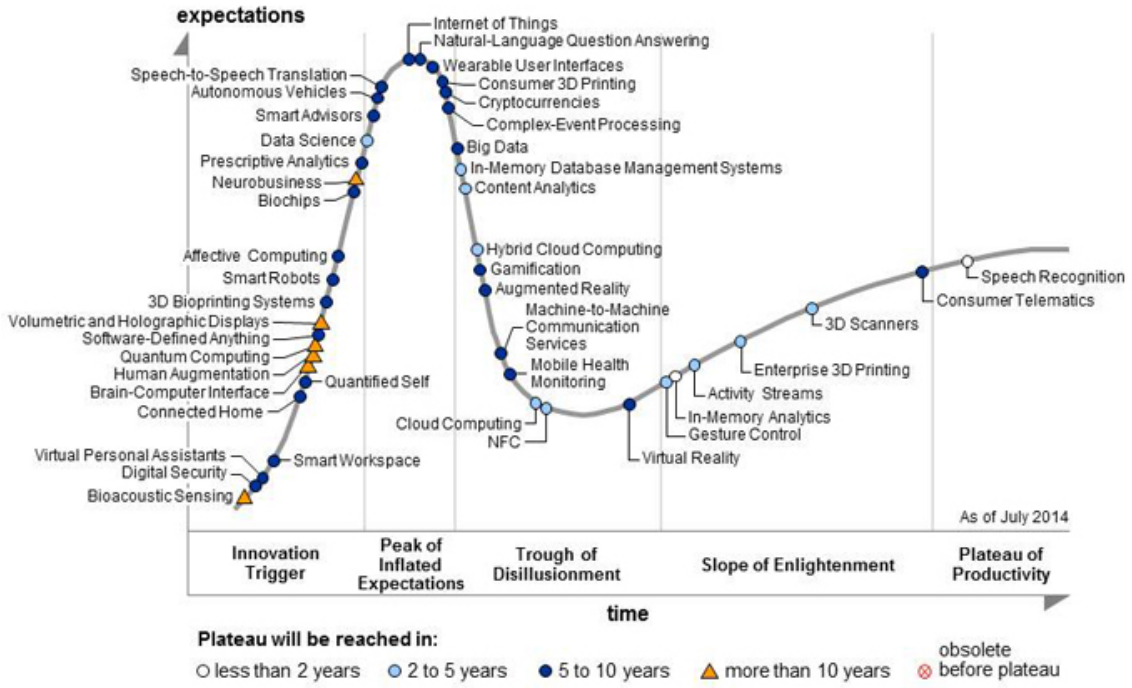

Figure 1: Hype Cycle for Emerging Technologies, 2014 [3]

1.1 State of the Cloud

This Section presents the current state of cloud adoption in the industry. Cloud Computing is known for its on-demand computing resource provisioning and has now become mainstream [1]. Whilst the level of hype in cloud computing has reduced (see Figure 1), there is still a lot of potential for market growth. For example, Forbes expect the worldwide market for public Information Technology (IT) cloud services to grow by more than 15% in 2016 [2] (as illustrated in Figure 2).

The nature of these services varies considerably in terms of what kind of information is being out-sourced to the cloud provider. Frequently, sensitive data is stored in the cloud, for instance when individuals share Personal Identifiable Information (PII), such as their contact details or medical information. Businesses that outsource (parts of) their processes to the cloud, for instance by using a Customer Relationship Management (CRM) Software as a Service (SaaS) provider (e.g., Salesforce [4]), are actively participating in a major paradigm shift

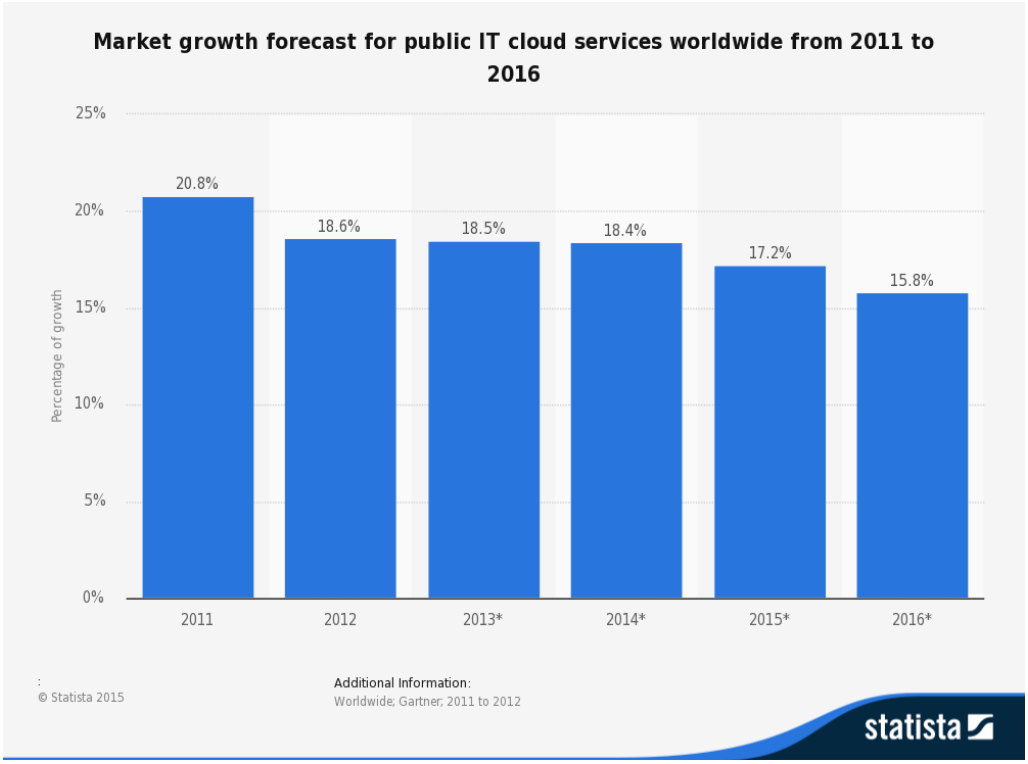

Figure 2: Cloud Services Global Market Growth 2011-2016 [2]

from having all data on-premise, and therefore under full control, to moving data to the cloud where exercising control over data and how it is used is severely limited. A recent study from 2014 on the confidentiality of data that small and large business based in the United Kingdom are storing on the Internet, has revealed that said data is usually regarded to be either confidential or even highly confidential (see [5] and Figures 3 and 4). This is particularly interesting, since data loss and data breaches are still consider to be among the top threats in cloud computing (see also Section 2.4 for a more detailed discussion security and privacy issues associated with the cloud).

However, this stands in contrast to only about half of the organisations in Europe having a security strategy in place for cloud computing (see Figure 5). As sensitive data is moved to the cloud, a well-defined cloud security strategy becomes increasingly important. A security strategy should include the identification and assessment of risks associated with moving to the cloud as well as plans for the implementation and monitoring of appropriate security and privacy controls to

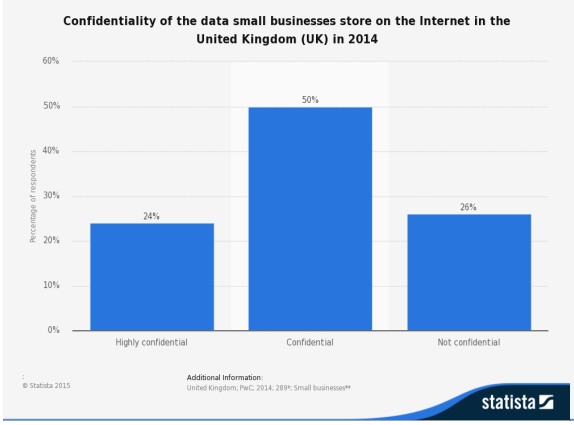

Figure 3: Confidentiality of the data small businesses store on the Internet in the UK 2014 [6]

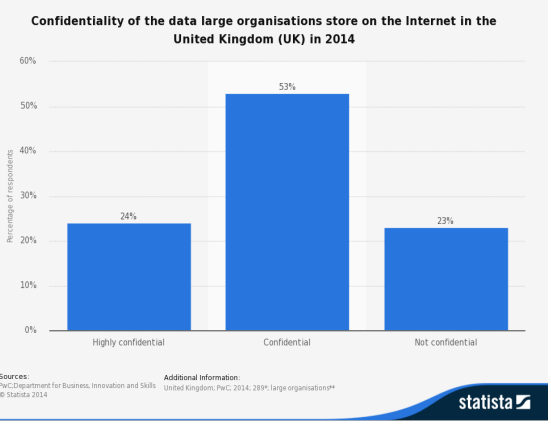

Figure 4: Confidentiality of the data large organizations store on the Internet in the UK 2014 [7]

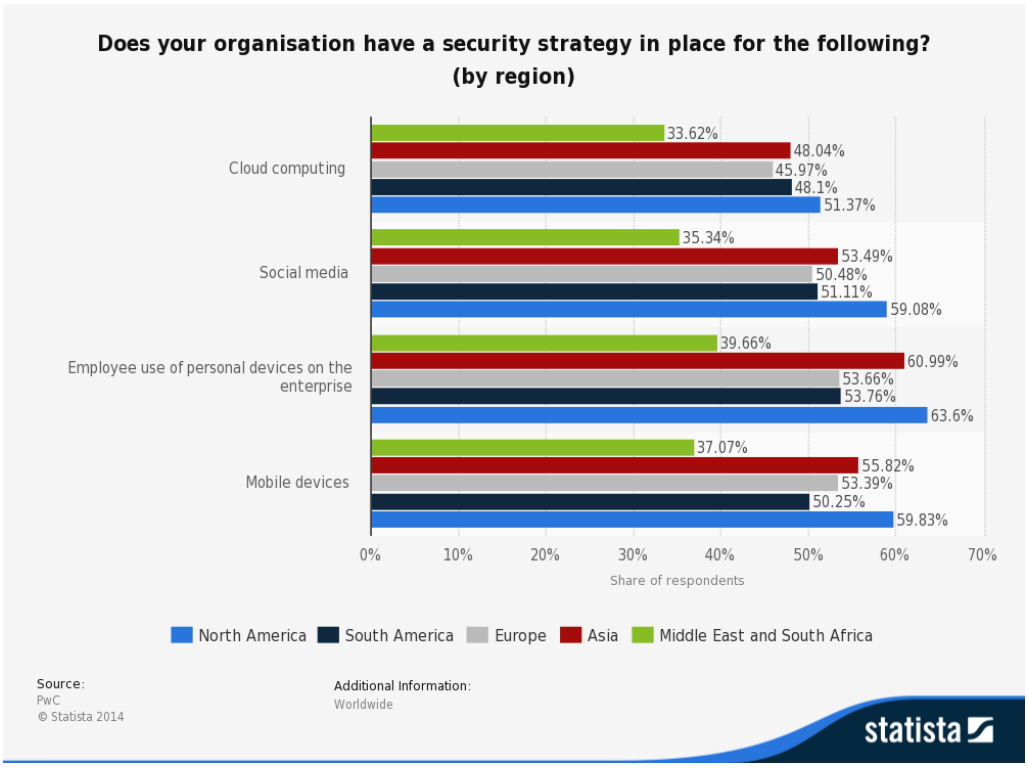

Figure 5: Company survey on information security strategies in place in 2013 by region [8]

mitigate the risks and ensure confidentiality, integrity and availability of data. This has become more complicated in the context of cloud use, since a lot of required information might not be provided by the cloud provider (e.g., details of security and privacy controls implementation at the provider's infrastructure).

1.2 Research Aims and Objectives

The research in this project aims at transparency and privacy protection issues associated with cloud computing and accountability audits as a means to strengthen trust in cloud services. By providing cloud auditors with a system for auditing compliance with data handling policies, transparency and privacy protection in the cloud shall be improved. This includes the secure and privacy-aware collection of evidence and the automated assessment and validation of that evidence to generate statements of compliance or non-compliance.

The research presented in this thesis is divided into the following series of objectives:

Objective 1: Improve trust in cloud providers by enabling automated security, privacy and accountability audits at the cloud provider
At the core of this research is the question, how the fundamental trust problem in cloud service provision scenarios can be solved to facilitate an even more widespread adoption, even if the processing of sensitive data is moved to the cloud. Furthermore, how cloud providers can be enabled to improve transparency, with respect to their infrastructures and data processing practices is investigated. This goes beyond documentation, agreements and certifications by using the concept of technical audits that are evidence-based, policy-driven and automated.

Objective 2: Enable cloud providers to show that they adhere to privacy and data security policies
Accountability is considered to be a key concept in solving the trust issue. However, the technology for demonstrating accountability is still limited. It shall be investigated, how cloud providers can be enabled to demonstrate good data processing practices and policy compliance to their customers.

Objective 3: Enable cloud provider chain auditing

Cloud service provision scenarios become increasingly complex as cloud providers start out-sourcing parts of their services or integrate additional services into their own. It is investigated, how such policy compliance auditing can be extended from single provider scenarios to multi-provider scenarios. The major challenges here are the introduction of organizational, technical and legislative heterogeneity in a single cloud service provision use case.

Objective 4: Enable end-to-end security and privacy protection of collected evidence and the audit system itself

An evidence-based approach to auditing data processing practices and policy compliance is likely to generate huge amounts of potentially sensitive information. To mitigate the risk of producing yet another potential security privacy protection nightmare, methods and techniques are examined and incorporated into the technical audit system in general and the evidence handling in particular that minimize the risks of data breaches and leaks.

1.3 The Cloud Accountability Research Project (A4Cloud)

The research that is presented in this thesis was conducted in the context of the Cloud Accountability Project and Audit Agent System (AAS) was developed as a contribution to the project's toolset. A4Cloud "focuses on the Accountability For Cloud and Other Future Internet Services as the most critical prerequisite for effective governance and control of corporate and private data processed by cloud-based IT services" [9]. The main goal of the A4Cloud project is to "increase trust in cloud computing by devising methods and tools, through which cloud stakeholders can be made accountable for the privacy and confidentiality of information held in the cloud" [9]. The interaction with other tools and concepts that are not directly contributed by the author (e.g., the interaction with other tools from the A4Cloud toolset that are developed by other partici-

pants[1]) are highlighted, clarified and delimited where necessary. The emphasis of this thesis is put exclusively on the author's contribution (the AAS design and implementation), which is concerned with enabling the automated auditing of compliance with privacy, security and accountability policies in an evidence-based and secure way. The integration of AAS with other tools from the A4Cloud toolset (e.g., most notably the Accountability PrimeLife Policy Language (A-PPL) policy language, its associated enforcement engine and the Insynd cryptographic scheme) are illustrated in the respective chapters of this thesis. AAS's role in this integrated toolset is to enable evidence collection in cloud service operations and provide automated auditing capabilities.

The A4Cloud FP7 research project [9] approach encompasses legal and regulatory mechanisms and a range of technological enhancements that can provide the necessary basis for trust. Cloud customers, providers and regulators should be supported by preventive, detective, and corrective tasks (see [10]) and, for example, give cloud customers more control over their cloud services, ensure providers meet their obligations, and enable cloud audits [11].

> "The Cloud Accountability Project (or A4Cloud for short) focuses on the Accountability For Cloud and Other Future Internet Services as the most critical prerequisite for effective governance and control of corporate and private data processed by cloud-based IT services." [9].

1.4 Thesis Structure

The remainder of this thesis is structured as follows:

Chapter 2 – Cloud Computing Fundamentals, introduces basic concepts of cloud computing in order to build a common understanding of the key concepts and roles specific to cloud computing. A review of the most important security and privacy problems that are most commonly associated with cloud computing are

[1] A4Cloud Participants: http://www.a4cloud.eu/consortium

discussed to highlight the major inhibitors of even more widespread adoption of cloud computing in the personal as well as the professional space.

Chapter 3 – Related Work, presents related work from three major topic areas. Related work from computer forensics is presented alongside work from the area of cloud forensics to build a framework for the work presented in the later parts of this thesis that relies on similar concepts and techniques. The second part presents related work that is concerned with the areas of auditing and assurance, both from a traditional and cloud-specific perspective. In the third part, related projects from the area of software agent technology are discussed, whereby a special focus is put on the application of agents in monitoring and auditing.

In *Chapter 4 – Cloud Privacy and Accountability Audits*, the first part of the main research conducted in this project is presented. Cloud computing is discussed from an evidence acquisition angle. Various evidence sources that are spread across the ecosystem on multiple architectural layers are discussed. The collection of evidence as well as the associated security and privacy risks are discussed as a basis for the following processing of evidence. Automated evidence-driven audits and their various types are presented. The chapter is concluded by an analysis of requirements that have been elicited from the previous sections. These requirements are the main drivers for the following chapters and help framing the novelty and presentation of the research.

Chapter 5 – Audit Agent System Architecture presents the main research conducted in this project. It presents an architecture that enables, based on the discussion of requirements in the preceding chapter, automated, policy-driven and evidence-based cloud auditing that improves transparency with respect to security and privacy. After the discussion of initial architectural considerations and their reasoning, an extension of the common definitions of cloud actors is presented in order to specify their functions in the latter parts of this chapter. Following that, the architecture of AAS is presented. Furthermore, the exten-

sion of the system from one-provider scenarios towards scenarios where multiple independent providers are involved in the service provision, is presented. The chapter is concluded with a discussion of audit result and evidence presentation methods and a framework for extending AAS. This discussion is performed from a practical point of view that is based upon the implementation of AAS.

A major aspect of AAS is the focus on evidence acquisition in a guided and well-defined way. *Chapter 6 – Audit Description using Software Agents* describes the integration of three different machine-readable policy languages. The main emphasis is thereby put on integration of A-PPL as the main policy language used in the current iteration of AAS. However, since there are gaps in the coverage of that language, specifically in the context of defining audits in AAS, additional candidates that can potentially cover these gaps are presented as well.

The presented research is evaluated in *Chapter 7 – Evaluation*. The evaluation of the effectiveness of AAS in achieving the aforementioned objectives is done from three different perspectives. The functional evaluation is concerned with demonstrating three audit cases that are derived from the data handling problems that are associated with cloud computing: data location, data retention and intrusion (or data breaches). The scalability focuses on management, performance and presentation aspects of AAS, whereas the last part is concerned with the security and privacy protection in AAS itself.

In *Chapter 8 – Conclusion and Future Work*, this thesis is concluded and a summary of the main achievements and limitations is presented. Based on the evaluation results, this chapter is concluded with a discussion of future work that addresses potential extensions as well as shortcomings of the current approach.

Chapter 2

Cloud Computing Fundamentals

In order to provide a common understanding of key concepts for the remainder of this thesis, basic cloud characteristics are introduced in this chapter. The major threats to information security in the context of cloud computing as well as the inhibitors of trust in this technology are also presented. Additionally, the notion of accountability in the cloud computing context is presented to provide the framework for the auditing system that AAS provides.

2.1 Actors

This section gives a brief overview of fundamental cloud computing characteristics and concepts. The National Institute for Standards and Technology (NIST) provides a comprehensive reference architecture [12, 13] for cloud computing that is widely accepted. This thesis makes use of the definitions provided by that reference architecture. NIST defines cloud computing as follows:

> **Cloud Computing:** "...a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable resources that can be rapidly provisioned and released with minimal management effort or service provider interaction" [13].

Thus, the main defining characteristics of cloud computing are

- service provision over the Internet,
- self-service without human interaction on the provider-side,
- elasticity according to resource requirements,
- shared resources among cloud consumers.

The NIST also defines several cloud actors [12]. These definitions are used throughout this thesis, if not stated explicitly otherwise:

Cloud Consumer: "represents a person or organization that maintains a business relationship with, and uses the service from a cloud provider".

Cloud Provider: "acquires and manages the computing infrastructure required for providing services, runs the cloud software that provides the services, and makes arrangement to deliver the cloud services to the Cloud Consumers through network access".

Cloud Auditor: "can perform independent examination of cloud service controls with the intent to express an opinion thereon".

The terms cloud customer and cloud consumer are used synonymously in this thesis. Depending on the specific scenario, a cloud consumer can also be a cloud provider.

The role of the auditor is also extended and refined in the context of AAS. More in-depth discussions of these adjustments to the roles can be found in Section 5.2.

Beyond these roles, the reference architecture also defines a cloud broker and cloud carrier. However, these are omitted, since they play no significant role in this research.

2.2 Service Models

The differentiation of cloud service provision scenarios with the help of service models is a useful tool to describe how a service is used, and what kind of resource is provisioned to a cloud consumer. According to the service model, audit objectives as well as sources of evidence can differ dramatically. Section 4.2 discusses these issues in more detail. It is therefore important to define a clear understanding of the different service models in order to facilitate the discussion of evidence sources and highlight their characteristics. The most fundamental and common service models that are widely agreed upon for categorization, are defined in NIST's definition of cloud computing.

> **Software as a Service (SaaS):** "The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings." [13]

This service model is probably the most common form of cloud computing, with which a typical cloud consumer (i.e., an individual person) comes into contact. In some scenarios, such services are provisioned to organizations that themselves do not host services (such as corporate e-mail or conferencing software) anymore, but rather provide their employees with outsourced services from a cloud provider.

> **Platform as a Service (PaaS):** "The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools

supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment." [13]

The border between application and underlying infrastructure is of particular interest in this scenario since there is a boundary between two, usually independent organizations. This introduces unique challenges in a globalized world, where these organisations may not necessarily fall under the same jurisdiction or regulatory regime.

Infrastructure as a Service (IaaS): "The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls)." [13]

In Infrastructure as a Service (IaaS) clouds, the consumer is often an organization that requires low-level computing resources, for instance to address peak loads on its own infrastructure (cloud bursting) or because it does not have any servers of its own. With respect to boundaries between provider and consumers, there is typically strong isolation provided by the virtualization layer. Virtualizaion thereby does not only apply to the Virtual Machine (VM) resource but also other low-level resources such as networking (e.g., by introducing concepts from Software-defined Networking (SDN) or Virtual Private Network (VPN) for network isolation) and storage.

A key property of the cloud is the shared responsibility between the provider and the customer. The actual distribution of responsibility thereby depends on the

cloud service model. For instance, in an IaaS scenario the customer has more responsibility (e.g., ensuring patch management, vulnerability management and access control on a VM level) than in a SaaS case, where the provider takes this responsibility and integrates such controls in its service.

2.3 Deployment Models

The cloud deployment models help framing the scope of the AAS project. As a basis for discussion, the well-accepted categorization by NIST is used. They present the following four deployment models:

> **Private Cloud:** "The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises." [13]

While a private cloud can be provisioned by either the consuming organization itself or a third-party, the consuming organization retains full control over the resource pool, deployment and applications. This has implications on the operation of monitoring and auditing tools and resource access control. If an organization is running its own private cloud infrastructure, it retains a certain level of control and transparency of its own data processing, since all details are available (at least internally). To some extent this is also true for outsourced private clouds, where the consuming organization retains full control over the cloud management.

Organizations can run their own private clouds using IaaS systems such as OpenStack [14] and OpenNebula [15] or SaaS systems such as ownCloud [16].

> **Public Cloud**: "The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business,

academic, or government organization, or some combination of them. It exists on the premises of the cloud provider." [13]

Typical providers of public cloud services are for example Amazon Web Services (AWS) [17] for mostly infrastructure related services, Google App Engine [18] for platform services and Google Docs [19] for software services. A common concern of users of public cloud services is data protection and isolation from other tenants. Also, the transparency problem is most distinct in public clouds, where a consumer is reliant on the technical (e.g., transparent data transfer) and organizational processes (e.g., having the right people to run the cloud) employed by the cloud provider without having any real possibility to take influence.

Community Cloud: "The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises." [13]

The community cloud model is included for showing a complete picture of the current understanding of cloud deployment models. An example of a community cloud is the Regionales Zentrum für Virtualisierung (RZV) project [20] that is run by different German universities in a collaborative effort. Thereby, the Universities of Furtwangen, Freiburg and Offenburg form a federation of institutions while following the approach of running a single cloud infrastructure for research and teaching purposes. The universities form a community of similar but independent institutions that share a common interest in running a cloud infrastructure. The actual implementation of a community cloud can share some aspects of a private cloud, where services are not provided to the general public and the technical operation of the infrastructure is the responsibility of a single partner.

Hybrid Cloud: "The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds)." [13]

The hybrid cloud is the most common way of using the cloud in enterprises, today [21]. Hybrid clouds share the advantages of the private cloud (e.g., full control) and the public cloud (e.g., nearly infinite resources on-demand). However, they also share the disadvantages from both, such as cost of ownership and lack of transparency, respectively. Furthermore, there are cases to be considered when data crosses from the private to the public domain and thereby crosses borders (e.g., by being transferred to a data center in another country).

2.4 Security, Privacy and Trust

Potential problems for data confidentiality and auditability have been expected quite early in the cloud hype cycle [22]. These expectations have proven to be accurate. In the following, the top threats to cloud computing according to Cloud Security Alliance (CSA) are presented. In the NIST cloud computing reference architecture it is acknowledged that "security is a cross-cutting aspect" [12] of the cloud architecture. Therefore, cloud providers should protect privacy, which is "the assured, proper, and consistent collection, processing, communication, use and disposition of personal information (PI) and personally identifiable information (PII)" [23].

The methodology behind the aggregation of this list of threats to cloud computing was a survey of relevant literature.

Cloud Security Alliance

The CSA maintains a well-established list of critical security and privacy problems in cloud computing as they are perceived by cloud consumers that is also updated regularly. Until now, the original report from 2010 [24] has been updated in 2012 [25] and 2013 [26].

1. **Data Breach:** In the most recent version, data breaches are listed as the top threat to cloud security. This is inherently also a threat to data protection and therefore the privacy and security of cloud consumer's data.

2. **Data Loss:** The loss of cloud consumer's data, either caused by a malicious attacker (e.g., intentional deletion) or by accident (e.g., misconfiguration or hardware failure) is listed as the second most dangerous threat.

3. **Account or Service Traffic Hijacking:** Common hijacking methods, such as phishing, can be applied to cloud scenarios as well. If, by any means, cloud credentials are compromised, an attacker can potentially gain access to the whole service, manipulate the service or simply eavesdrop on the service users.

4. **Insecure Interfaces and Application Programming Interfaces (APIs):** Typically, cloud services provide APIs for provisioning, management, orchestration, monitoring and audit. Therefore, since such APIs are at the core of a cloud service and potentially provide access to data directly or to information about data usage patterns, special care has to be taken to harden them.

5. **Denial of Service:** Problems caused by Denial of Service (DoS) attacks are not only related to service availability, but in more sophisticated scenarios can be used to exploit vulnerabilities that potentially put data stored in the cloud at risk.

6. **Malicious Insiders:** An insider at the cloud provider (e.g., a system administrator) has extended access to consumer resources, such as VMs, block

storage etc. Hence, a malicious insider can potentially gain access to consumer's data if appropriate security and data protection controls are not in place.

7. **Abuse of Cloud Services:** Abuse of a cloud service is of concern for cloud providers. However, it is hard to define, what constitutes actual abuse of a service, for instance when cloud resources are used to within the terms of service of a provider but for a potentially harmful objective (e.g., breaking encryption or launching attacks).

8. **Insufficient Due Diligence:** This threat is mainly caused by a lack of experience with cloud technology and architecture when adopting cloud services.

9. **Shared Technology Vulnerabilities:** This threat evolves around the risks introduced by multi-tenancy. Strong isolation is required between tenants to mitigate the risk of accidental (e.g., misconfiguration) or malicious data breaches (e.g., side channel attacks).

European Network and Information Security Agency (ENISA)

The threats described by the CSA are similar to those identified by European Network and Information Security Agency (ENISA) in [27], which takes a regulatory perspective on threats in cloud computing. The order of appearance does not reflect any difference in severeness of the risks.

- **Loss of Governance**: The cloud consumer gives up control over the processing of its data. This can lead to problems with the integration of security mechanisms at the cloud provider. A provider may not necessarily be able to provide sufficient protection.

- **Lock-in**: The migration from one cloud service to another is still an extremely resource intensive task and may not be feasible for all cloud con-

sumers. This effectively leads to a lock-in with the provider that was initially chosen.

- **Isolation Failure**: The multi-tenant nature of the cloud makes the introduction of suitable isolation mechanisms very important to protect consumers from one another. This also includes the protection from side-channel attacks that break isolation (see for example [28]).

- **Compliance Risk**: A consumer can put its certifications at risk by moving to the cloud, when a cloud provider cannot provide evidence of compliance or does not offer auditing capabilities to the consumer.

- **Management Interface compromise**: The management interfaces that are provided to the consumers may be at risk, because they are an interesting target for attackers since they provide access to broad resources and tenant's data.

- **Data Protection**: Effectively checking the data handling practices of the provider may prove to be very difficult for the consumer, since he is lacking the required level of insight into the cloud providers processes.

- **Insecure or Incomplete Data Deletion**: This is a risk in cloud computing since there may be multiple copies of data stored over which the consumer does not have full control. Anti-forensic methods such as secure deletion are still not feasible in cloud environments due to data location transparency and distributed storage.

- **Malicious Insider**: Administrators at the cloud provider usually have broad access rights on underlying layers of the cloud service. This puts consumer's data at risk, if there is a person with bad intentions.

ENISA explicitly lists data protection threats (the loss of being able to check data handling practices) and the loss of governance (the loss of being able to control

data) as potential threats. While the ENISA report lists risks not only with respect to security and privacy protection (e.g., vendor lock-in, loss of governance), the obvious focus is on threats similar to those described by CSA, while differing only in level of abstraction (e.g., data protection in ENISA and data breach in CSA) and terminology (e.g., isolation failure and shared technology vulnerabilities). Threats, such as hijacking attacks, insecure interfaces, malicious insiders of shared technology vulnerabilities indirectly impact a consumer's privacy, if insufficient protection and detection mechanisms are in place.

Beyond these reports, there is a substantial amount of literature and surveys that try to capture threats and risks of cloud computing on a more concrete level. For instance, in [29] or [30] more general threats to Information Security (IS) that are not focused on cloud computing and originate from standards or best practice catalogues such as ISO 27005 [31] or Open Web Application Security Project (OWASP) [32] (such as physical data center security and disaster protection) are included as well. The threats that were discussed above can be considered to be specific to cloud computing, while other more general threats need to be addressed as well.

Pearson [33] and Jansen [34] consider consumer trust, which can be linked to data protection, and compliance to be among the most important issues when it comes to data processing in the cloud. These issues are closely tied to the loss of control over data. When moving to the cloud, direct control over data is given up, for instance:

- *Data Location:* the actual geographical location of where a certain data object is stored (e.g., the data center, rack, enclosure or even disk).
- *Access Control:* the certainty of having defined access control restrictions and the enforcement of them including the implementation of assurance mechanisms.
- *Data Handling:* the fact that it is possible for a cloud provider to duplicate,

share or otherwise process data without the awareness of the owner.

Organisation for Economic Co-operation and Development (OECD)

The "Organisation for Economic Co-operation and Development (OECD) Guidelines on the Protection of Privacy and Transborder Flows of Personal Data" [35] define general guidelines for the processing of personal information. These guidelines are not limited to data processing in Information and Communication Technology (ICT) and the cloud, but apply to any kind of data processing. The principles defined in the OECD guidelines are as follows [35, Art. 7-14]:

Collection Limitation: when personal data is collected, this should happen within the boundaries of the law and with consent of the data subject.

Data Quality: personal data should not be collected arbitrarily, but always with a purpose and if the purpose changes be deleted if not required anymore.

Purpose Specification: the purpose of data collection should be always made clear at the latest when it is collected.

Use Limitation: personal data should not be disclosed, made available or used other than what has been specified, exception being by the authority of the law or when consent is given by the data subject.

Security Safeguards: data protection mechanisms should be put in place (e.g., against loss, unauthorized access, destruction, misuse, modification, disclosure).

Openness Principle: there should be a general policy of openness about developments, practices and policies with respect to personal data.

Individual Participation: a data subject should have the right to know who has data about him, access to said data and the right to challenge data including the right to have the data erased, rectified, completed or amended.

Accountability: A data controller should be accountable for complying with measures which give effect to the principles stated above.

These guidelines are applicable to cloud services as well and should govern the handling of data. However, these principles are most often only reflected in Terms of Service (TOS).

European Data Protection Directive

In the European Directive 95/46/EC [36], which is implemented in Germany by the Bundesdatenschutzgesetz (BDSG) [37], it is stated that "... the controller must implement appropriate technical and organizational measures to protect personal data against accidental or unlawful destruction or accidental loss, alteration, unauthorized disclosure or access, in particular where the processing involves the transmission of data over a network,..." [36, Art. 17]. Furthermore, "... the controller must, where processing is carried out on his behalf, choose a processor providing sufficient guarantees in respect of the technical security measures and organizational measures governing the processing to be carried out, and must ensure compliance with those measures" [36, Art. 17]. The *controller* is defined as someone who "determines the purposes and means of the processing of personal data" [36, Art. 2], whereas the *processor* is someone who "processes personal data on behalf of the controller". A new general data protection regulation [38] has been proposed, which will replace the current one and clarify some of the obligations.

From a legal perspective, cloud providers are required to comply with the above-mentioned regulations and legal frameworks and are obligated to implement adequate data protection mechanisms in order to stay compliant. Therefore, adequate data security and protection mechanisms have to be put in place. However, since these rules apply to European cloud providers and not necessarily to those from the US (e.g., if they are not doing business in Europe) or other regions, it is also important to consider the regulatory requirements depending

on the actual scenario (i.e., location of the consumer, provider and data).

2.5 Accountability

New mechanisms and technologies have to be developed to deal with the above-mentioned data protection issues, and address the two most pressing issues, when it comes to adoption of cloud computing: a lack of consumer trust and complexity of compliance [33].

Weitzner proposes information accountability as a suitable solution. It means "that information usage should be transparent so it is possible to determine whether a use is appropriate under a given set of rules" [39]. More transparency would allow to hold entities responsible and accountable for misuse of information. Transparency in this case means making failures in data handling visible.

As previously mentioned, cloud consumers do lose control over their data by moving to the cloud. Therefore, cloud consumers have to trust cloud providers that they treat their data in an appropriate and responsible way. This includes providing information about data locality, isolation, privacy controls and data processing in general. One way to enable that trust is by strengthening transparency and accountability [39, 40] of the cloud provider and its services. In the context of cloud computing, this means cloud service providers should be enabled to become responsible data stewards by following the accountability approach [33]. Another important aspect of accountability is the demonstration of the capability to achieve privacy objectives, which stem from criteria in law, self-regulation and best practices. [41].

Several research projects acknowledge the existence of privacy protection issues in today's cloud computing ecosystems and accountability as a possible solution. Distributed, automatic and enforceable logging usually serves as a foundation for developing such systems [42]. This is the first step towards more transparency of data handling in the cloud for cloud consumers. By introduc-

ing methods and technical solutions supporting accountability, the consumer is given back control over the storage and processing of its data in the cloud. However, today it is very common for cloud service providers to work on a best effort basis regarding evidence collection for availability and security of cloud services.

The consumer should be enabled to define more detailed policies on how and by whom its data may be accessed and processed in the cloud. Cloud providers, which adhere to these rules inherently become more trustworthy. However, enabling accountability in the cloud quickly becomes very complex, the more parties are involved and when applied on an international scale (e.g., third-party providers, service composition and data crossing borders). Therefore, suitable control mechanisms have to be put into place by cloud providers. This is what AAS aims to achieve by providing extensive technical audit capabilities that allow the continuous assessment of accountability obligations and rules to enable consumer trust.

2.6 Summary

In this chapter, cloud computing along with its main characteristics, typical service and deployment models have been presented to establish a basis and common understanding for the research presented in the remainder of this thesis. To frame the research conducted in this project, a special focus has been put on current security, privacy, and trust issues. The threats to cloud computing that are currently considered most problematic, mostly cover aspects of cloud security, privacy protection and accountability. Additionally, it was elaborated on how strengthening accountability and transparency of cloud providers and their data processing practices can positively influence trust in cloud services.

Chapter 3

Related Work

In this Chapter, related work is presented that is either similar to the concepts and approaches pursued in the AAS project or are integrated in this work. The focus is thereby directed towards work that has been conducted in the areas of computer forensics, cloud compliance, software agent technology and industry projects that have similarities with the auditing approach followed in this work. This review of relevant literature gives the reader an understanding of the current problems, potential solutions and shortcomings of the current state of the art in cloud accountability and auditing. It also helps to highlight the contribution that is being presented in the remainder of this thesis.

The chapter begins by elaborating on the computer forensic process and digital evidence collection in general and cloud computing in particular. Evidence collection is required to acquire meaningful information for evaluation during audits. Principles and processes from the field of computer forensics are thereby helpful in preserving the usefulness of collected information. However, some principles of digital evidence and forensics cannot be applied to the cloud directly, for example because of its dynamic nature and uncertainty of data location. A literature review in this field was performed to prepare the elicitation of requirements for this project and gain an understanding of the cloud specific problems.

In the following section, related work in the area of cloud compliance and audits as an assurance mechanism is presented. This includes a discussion of existing standards and best practices in the area of cloud security and privacy and how they cover cloud computing, cloud storage integrity, security, accountability and privacy audits. A review of research projects and literature in this area was performed to capture the shortcomings of the current state of the art.

An overview of industrial solutions for log management, security incident management and similar systems is presented as well. The reason for considering industrial solutions is their usefulness in providing evidence in the form of monitoring information that is produced by them.

Finally, since software agents have proven to be suitable for the implementation of the system, a basic introduction to the concepts of this technology is presented. Hereby, the focus is directed towards similar approaches that use software agents in the context of cloud auditing and monitoring.

3.1 Computer Forensics and Digital Evidence in the Cloud

In the following, related work regarding processes and systems for the collection of digital evidence in the cloud are presented. This includes work on cloud-specific evidence collection mechanisms as well as work on addressing the heterogeneity of digital evidence. The definition of digital evidence is “information of probative value that is stored or transmitted in binary form” [43]. This definition is general enough to capture the fact that all data that is available or produced in a cloud ecosystem can potentially serve as evidence to demonstrate a fact. Processes, methodologies and techniques from (non-cloud) computer forensics (such as descried in the well-known *Electronic Crime Scene Investigation: A Guide for First Responders* [44, 45] and *Guide to Integrating Forensic Techniques into Incident Response* [46]) also apply to the cloud to a limited extent.

3.1.1 Computer Forensic

Redfield and Date propose a system called Gringotts [47] that enables secure evidence collection from multiple different sources. Evidential data is signed by the system that produces it, before it is sent to a central server for archival using the Evidence Record Syntax (ERS) [48]. They follow a pro-active approach as well, where evidence is recorded in case it is needed later on. Their focus is on the archival of evidence for later retrieval by an investigator and omits automated evidence processing for audits. Introducing evidence encryption is not considered, since Redfield and Date focus on archival and preservation of evidence integrity. Furthermore, their approach utilizes cloud computing merely as an enabling technology for evidence storage and do not consider evidence collection in the cloud explicitly. It is therefore better described as a general computer forensics tool rather than a cloud forensics tool.

Zhang et al. [49] address potential problems when storing massive amounts of evidential data. They specifically consider possible information leaks, where sensitive information, which is not directly relevant to the incident, could be exposed. To solve these issues, they propose an encrypted database model that is supposed to minimize the potential for data leaks as well as data redundancy. However, they focus solely on the storage backend and do not provide a workflow that addresses secure evidence collection as a whole. Similarly to Redfield's approach, the cloud is hereby to be considered a vehicle for provision of their service and not necessarily the object of investigation.

Gupta [50] identifies privacy issues in the digital forensics process, when it comes to data storage devices that typically do not only contain investigation related data, but may also hold sensitive information that may breach privacy. He also identifies a lack of automation in the digital investigation process, which is typically manual and investigator-driven. To address these issues, Gupta proposes the Privacy Preserving Efficient Digital Forensic Investigation (PPEDFI) framework. PPEDFI automates the investigation process by including knowl-

edge about previous investigation cases, and which kinds of files were relevant then. With that additional information, evidence search on data storage devices is faster. However, while Gupta considers privacy issues of the data collection process, the PPEDFI framework is focused on classic digital forensics and may not be applicable to a cloud ecosystem, where there is typically no way of mapping specific data objects to storage devices.

Preserving the integrity of digital evidence is a challenge for law enforcement. Digital evidence can be easily manipulated if no additional precautions are put in place. Thus, using integrity-preserving mechanisms such as hash functions to protect evidence and detect tampering is at the core of digital evidence collection. However, the effectiveness and efficiency of these mechanisms vary. Saleem et al. [51] evaluated several integrity-preserving algorithms (e.g., SHAx, MDx and CRC) for use in digital evidence acquisition.

Schatz and Clark [52] from the Common Digital Evidence Storage Format (CDESF) working group propose an evidence framework. Their architecture focuses on Digital Evidence Bags (DEBs), a generalized method for collecting information about evidence and evidence metadata, while keeping evidence integrity. This generic data format could easily be used as a container for evidence collected for the purpose of auditing. It is also general enough to address the heterogeneity of evidence sources.

3.1.2 Cloud Forensic

There are several problems arising when the forensic process is applied to cloud computing.

Regarding data protection, NIST [53] names:

- The collection of evidence while protecting the privacy rights in a multi-tenant environment. When investigators collect evidence in a multi-tenant environment such as the cloud, this quickly becomes problematic. For in-

stance, if a VM that is executed on a server becomes the object of interest in an investigation, that server may be seized or live-monitored by law enforcement. However, co-located VMs owned by another tenant might be impacted negatively.

- The analysis of encrypted data and the linkage between evidence and a particular suspect are among the most important challenges. An increasing number of cloud storage providers follow a zero knowledge approach to storing consumer's data, where data is encrypted before it is transferred to the cloud (most commonly done, when only storage is required and no processing on that data is performed in the cloud). This leads to problems, when that data is to be analyzed as part of an investigation.

Additionally, challenges such as the following exacerbate the evidence collection process:

- Gaining access to imaging of media, since a cloud provider may not be willing to provide investigator the required access to physical machines and in some cases where multiple jurisdiction are involved, the provider may not even be obligated to do so.
- The huge volume of data that is being processed in cloud services.
- Distributed storage may break conventional investigation methods that aim at the analysis of a single storage medium.

Additionally, cloud forensics faces challenges with respect to acquisition of evidence where time synchronization and legal authority (due to distributed nature of cloud services), the preservation of evidence integrity and chain of custody as well as the availability of sufficient storage capacity [54].

New methodologies and techniques are being developed that address these problems in the cloud context.

An advanced approach for using the hypervisor for providing evidence in digital forensics is Virtual Machine Introspection (VMI) [55–57]. VMI leverages the capabilities of the hypervisor to look "inside" the VM during runtime and using information collected this way for intrusion detection (e.g., detecting malware on the introspected VM). Garfinkel et al. conclude that this method is suitable for investigating cloud infrastructures, as long as the investigator has access to the hypervisor. Dunlap et al. propose ReVirt [58], a logging and replay system for analyzing intrusions, that runs integrated in a VM and performs the logging in the host OS. After an attack, it can replay the whole VM process for analysis. These are very low-level approaches to tracing the execution of VMs. However, they provide deep insight into the operations of a VM.

Deploying additional software for evidence collection and monitoring inside virtual machines can be a problem in public IaaS cloud scenarios, where the customer has full administrative control inside a VM and therefore has the ability to manipulate the evidence source or the collection process freely. To overcome this issue, an out-of-guest-approach can be followed, like Carbone et al. present in their work [59]. By moving the monitoring tool outside the focused VM and leveraging function-call injection techniques as well as VMI, the monitoring tool can be protected from the customer, resulting in more reliable information for evidence collection.

Several projects work on making the VMI concept feasible for acquiring evidence in cloud scenarios amongst which is LibVMI [60] that provides low-level monitoring of VMs (supporting Xen [61], KVM [62] hypervisors and Qemu [63] emulator). LibVMI even provides integration with the volatility [64] framework, which is a well-known tool for memory forensics.

The approaches proposed by Garfinkel, Carbone and Dunlap provide deep insight into what happens in a VM. However, depending on the service model (i.e., how much administrative privileges a customer has) additional evidence sources outside the virtual machine need to be considered to provide a detailed view of

what happens in the cloud.

Zawoad et al. [65] propose Secure-Logging-as-a-Service (SecLaaS) to enable secure and trustworthy log collection in cloud forensics. Based on the assumption that today's cloud computing architectures lack support for cloud forensic investigations, SecLaaS provides a mechanism for log collection and verification. They consider evidence sources that are important especially in the cloud context, such as process and network logs. Also, accessing those logs is increasingly challenging for investigators due to the black box nature of the cloud and privacy protection problems originating from multi-tenancy. They also consider further challenges such as the level of control provided to investigators and the willingness to cooperate that is required by cloud providers.

The most important aspects of forensics in the cloud are addressing cloud dynamic and the preservation of evidence integrity and usefulness. The dynamic nature of cloud computing requires careful planning of evidence collection upfront, since collection after the fact might not be feasible (e.g., valuable information has already been lost because the virtual machine has been deleted, already). Also, the location transparency of data in the cloud calls for new ways of evidence collection that are similar to carefully planned monitoring of the cloud than gathering data after an incident. As in common digital forensics, the preservation of the integrity (e.g., choosing appropriate algorithms) and usefulness (e.g., preserving temporal order of events by using a timestamping authority) of evidence is also of utmost importance. The cloud does not only introduce new challenges, but also enables promising ways of evidence acquisition such as VMI that allows for a detailed reconstruction of events in case of compromised virtual machines.

3.1.3 Digital Evidence in the Cloud

In [66], Dykstra approaches the problems of digital evidence collection in cloud computing from a legal perspective. According to him, the most prevalent prob-

lems are:

- The availability of evidential data, where it may be problematic to request evidence from a cloud provider that does not have that data himself because he is using another subsequent provider (e.g., Dropbox using Amazon S3 as a storage provider).
- Access to data that is stored inside a customer's VM, where access by the cloud provider is usually limited in IaaS.
- The preservation of evidence when using mechanisms such as snapshots.
- Legal boundaries, regarding the ownership of cloud resources (e.g., a customer owning the virtual machine and everything inside it, but not additional resources it uses such as the network).
- Jurisdiction and venue, when data is stored in different jurisdictions (which is quite common in cloud computing).
- Legal basis on which to get access to the data.
- The cost of evidence acquisition in terms of time and money.

Provenance of data in the cloud (i.e., information about the history of a data object) is discussed by Zhang et al [67]. They consider the collection of provenance data in the cloud as a key to enhancing reliability, accountability, transparency and confidentiality in the cloud. If that information is produced, this of course can be valuable data for the evaluation during accountability audits, since there would be a more complete picture about the whole history of data usage. Similarly, Lu et al. [68] propose the adoption of the provenance concept in cloud computing by enabling a data object to report who created it and modified its contents. This could then also provide digital evidence for investigations by reporting on events from the object's creation until its deletion.

Flaglien et al. [69] evaluated currently used storage and exchange formats for handling digital evidence against criteria identified in recent research literature.

Formats intended for storing evidence from highly dynamic and complex systems are characterized by incorporating additional information, which can be processed by data mining tools.

The most important aspects of digital evidence in cloud computing are the heterogeneity of evidence, availability of evidence, access to evidence sources and potentially different jurisdictions with respect to globally operated cloud services. As discussed in Chapter 4.2, evidence is generated at various places in the cloud by different tools using various mechanisms and formats. Addressing this is a major challenge in a system that audits cloud operations based on digital evidence.

3.2 Audit and Assurance

Audit and assurance are two very important concepts that also need to be applied to the cloud. In the following, a literature review is presented that highlights current state of the art in the cloud auditing domain.

3.2.1 Regulatory Compliance, Standards and Best Practices

In the following, relevant standards and best practices that serve as a baseline for evaluating security and privacy controls in cloud computing are presented.

There are several frameworks considered with security and privacy of ICT in general (e.g., ISO 27001 [70], ISO 27002 [71], COBIT [72], PCI-DSS [73]). These standards describe data security and protection mechanisms including technical and organizational controls. For example for preventing data breaches (e.g., access control, encryption), ensuring integrity (e.g., hashing, backups), ensuring compliance (e.g., audits) and protecting privacy (encryption, access and usage control). Organizational aspects of information security also include management processes such as security incident management or business continuity

management.

All of the aforementioned security and data protection controls are subject to audits, which aims to ensure compliance and assure a certain level of data security. Audits are typically conducted on the basis of one of these standards as part of a certification or self-monitoring process. Typically, the intervals at which an audit is repeated are quite long (often yearly or longer). In the meantime, policy violations can potentially remain undetected for extended periods of time. One of the main goals should be to address these periods of uncertainty by enabling the continuous assessment of cloud operations with respect to policy compliance. This would be an important step towards continuous certification. Also, such audits usually lack automation and are performed manually by auditors (e.g., manual security evaluation, documentation review, interviews). One reason for this is that requirements and obligations stated by these frameworks are typically not available in a machine-readable format. There are approaches to making these requirements and obligations explicit in a machine-readable way, for example A-PPL [74] for defining data protection and data handling-related obligations for data processing in the cloud.

A special focus on the requirements of cloud computing, can be observed just recently. For instance, the US government aims to standardize the assessment and monitoring of the security of cloud services with Federal Risk and Authorization Program (FedRAMP) [75]. It is supposed to form the basis for the adoption of cloud computing in the US government. FedRAMP certification offers cloud providers a way to demonstrate good security and data protection practices and enables them to make business with the US federal institutions.

As a collection of best practices, NIST published guidelines on the security and privacy in cloud computing. In those guidelines, NIST acknowledges security and privacy considerations to be of critical importance [76], for instance:

Security and Privacy Planning: Carefully plan the security and privacy aspects of cloud computing solutions before engaging them.

Understanding Public Cloud: Understand the public cloud computing environment offered by the cloud provider.

Cloud Security and Privacy Requirements: Ensure that a cloud computing solution satisfies organizational security and privacy requirements.

Client Security and Privacy Requirements: Ensure that the client-side computing environment meets organizational security and privacy requirements for cloud computing.

Accountability: Maintain accountability over the privacy and security of data and applications implemented and deployed in public cloud computing environments.

An important institution that is concerned with the security best-practices in cloud computing is the CSA. CSA is actively working on improving cloud security by working on several projects such as the Cloud Controls Matrix (CCM) [77]. CCM aims to assist potential cloud customers in assessing security and privacy risks associated with a cloud provider. It is based upon established standards, regulations and controls frameworks, such as the aforementioned ISO27001 and 27002, COBIT, PCI-DSS but also considers cloud-focused approaches such as NIST 800-144 and FedRAMP among others.

Another project by the CSA is the Consensus Assessments Initiative Questionnaire (CAIQ) [78]. The goal of CAIQ is to provide cloud customers and auditors a list of questions and answers they might want to ask a cloud provider. Cloud customers can use this questionnaire for cloud provider evaluation.

The Security, Trust & Assurance Registry (STAR) [79] is based on CCM and CAIQ and provides a public database of cloud providers, that have completed assessment based on CCM and CAIQ. There are different levels of STAR certifications ranging from *self-assessment*, which is self-explanatory, *certification* and *attestation*, which require an assessment by a third-party, to *continuous*, which is a certification based on continuous monitoring.

EuroCloud Star Audit [80] is an initiative that aims to "facilitate acceptance for Cloud Services on the international market, as well as to support the consumer oriented provision of those services as their needs demand". It provides an audit framework that underpins the EuroCloud certification process in order to provide an assessment "provider's profile, contract and compliance including data privacy protection against local law, security, operations, environment and technical infrastructure, processes and relevant parts of the application and implementation up to interoperability and data portability" [80].

3.2.2 Cloud Auditing Frameworks

The Distributed Management Task Force (DMTF) is working on cloud auditing with the Cloud Auditing Data Federation (CADF) working group. They focus mostly on developing standardized interfaces and data formats to enable cross-provider cloud security auditing [81]. The OpenStack project [14] provides an implementation [82] of that standard which allows the export of monitoring events generated by its subsystems as audit events in CADF format.

A similar project is the Cloud Security Alliance's CSA Cloud Trust Protocol (CTP), which defines an interface for enabling cloud users to "generate confidence that everything that is claimed to be happening in the cloud is indeed happening as described, . . . , and nothing else" [83], which indicates an additional focus on providing transparency of cloud services.

The adoption of these standards is currently lacking. This is in part associated with their complexity (see CADF specification) and missing reference implementations with required maturity for being used in production systems. However, this does not mean that there is no technical implementations of systems that provide a deeper insight into the technical processes at cloud providers. There are of course proprietary solutions that can be integrated into comprehensive monitoring and auditing workflows. For instance, Amazon provides CloudTrail [84] for requesting audit records using a Representational State Transfer

(ReST) API. Other cloud providers publish similar interfaces to provide at least some auditing capabilities. Additionally, Amazon provides a check-list for auditing their services, which provides a short overview of evidence providing systems such as CloudTrail alongside relevant certifications that Amazon provides [85].

While these services provide much needed monitoring and auditing capabilities, they also increase the vendor-lockin problem associated with cloud computing due to their proprietary nature.

The work presented in this thesis relies on open standards and technologies in order to mitigate this problem, while providing the capabilities of integrating existing services using adapters.

3.2.3 Cloud Storage Integrity Audits

One of the main concerns when moving data to the cloud is the security of that data. This is closely related to the loss of control over the data. Whereas in a non-cloud scenario, a data owner (e.g., a company) can typically tell, where data is stored (e.g., on which server and which disks), this is no longer possible in the cloud, partially due to the provider hiding that information but also due to technical constraints such as the use of distributed filesystems. Therefore, new mechanisms to audit the security of data in the cloud are required. A trivial approach to assuring the integrity of a cloud-based storage service, is to download data stored at the provider and check it for completeness and integrity. However, because of the volume of data that can be stored in the cloud, it is not a feasible approach to repeat that process every time a check is made. Recent research on Proof of Data Possession (PDP) and Proof of Retrievability (PoR) has come up with mechanisms to address this problem and is described in the following.

Worku et al. define the basic requirements for data integrity proving schemes as [86]:

Unforgeability: where no cloud provider can forge an audit result by positively

influencing the outcome without being in possession of the actual data.

Privacy preserving: there should be no way for a third-party auditor to retrieve the actual content during an audit.

Recoverability: a scheme should also implement hardening techniques (e.g., error recovery).

Blockless verification: no parts of data (e.g., file blocks) should be retrievable (directly or indirectly).

Dynamic operation support: allowing file and block level verification.

Public auditability: enabling anyone else to verify data correctness (not just the data owner).

These are baseline requirements for auditing cloud storage services. Wang et al. [87, 88] extend the privacy preservation and public auditability requirements by also demanding (in case of third-party audits, where the auditor cannot necessarily be trusted):

- The auditor should be able to efficiently audit the cloud data storage without demanding the local copy of data.
- The auditing process should bring in no new vulnerabilities towards user data privacy.

These principles are addressed in recent work on PDP and PoR by several research projects, which are presented below.

Outsourcing data storage and verification of the correctness (i.e., integrity of data) is a problem that predates cloud computing. Ateniese et al. describe a model for proving data possession. [89] It allows a client to verify that a server (e.g., a cloud storage provider) possesses the original data that was uploaded, without requiring the client to download the data.

Proof of Retrievability (PoR) of data stored remotely (e.g., in the cloud) is a similar problem that also includes the notion of being able to reconstruct the contents of a file at any time at the client side. Several schemes have been proposed [90–93], which differ in security and efficiency, but in general allow for effective retrievability audit of very large files.

Similarly to PoR, there are proposed audit systems that build upon the previously described schemes and implement audit services with the introduction of performance optimization (such as reducing the computational overhead) [94] or focus explicitly on preserving privacy of audited data in a public auditing scenario, where the auditor is considered trustworthy but curious [95].

At the core of any data-centric audit stands the audit for the completeness and integrity of data stored in the cloud. The previously described approaches provide promising mechanisms to effectively and efficiently include such audits as part of AAS.

3.2.4 Cloud Security, Accountability & Privacy Audits

Security auditing is a very important part of accountability auditing of a cloud provider, since it demonstrates that required security controls are put in place and are functioning correctly (i.e., data protection mechanisms are working correctly and effectively). There are projects working on the architectural and interface level regarding the automation of security audits such as the Security Audit as a Service (SAaaS) project [96, 97]. SAaaS is used to monitor cloud environments and to detect security incidents. SAaaS is specifically designed to detect incidents in the cloud and thereby consider the dynamic nature of such ecosystems, where resources are rapidly provisioned and removed. However, the main focus of SAaaS is not to provide auditors a comprehensive way of auditing the cloud provider's compliance with privacy and accountability policies, which requires additional security and privacy measures to be considered in the data collection process.

Ryoo identify the need for data transparency in the cloud as even more critical than in conventional IT audits, since "security-relevant data is harder to obtain as cloud service providers, rather than cloud service users, control most of the data" [98].

There are also approaches that deal with checking compliance with data location policies. As stated in Section 3.2.1, it may be required from a cloud consumer to store data in distinct locations. This is against the principle of location transparency of data in the cloud (i.e., a user does not know in which server, data center or even country a specific data object is stored). Thus, new methods of enabling customer's to comply with regulation – a cloud customer may be required by law to enforce certain storage locations – are required. In [99], the authors propose a system that exposes infrastructure-level location monitoring information to the cloud customer.

A crucial part of cloud auditing is logging of actions and collection of data on all architectural levels in a cloud infrastructure. Several systems for logging in the cloud have been proposed. For instance, Marty [100] proposes a logging framework and guidelines for IaaS and SaaS logging. He considers time synchronization on log sources, reliable transport of logs to a central collector and compressed and encrypted log transfer as the main issues in such distributed environment as the cloud. The main reasons for logging are defined by business needs (e.g., tracking business metrics), operational requirements (service health monitoring), security requirements (e.g., security or privacy incident detection) and compliance (legal obligation to log certain actions).

Park et al. describe cloud auditing as the evaluation of a target based on monitoring results and pre-defined criteria [101]. Compared to cloud monitoring systems, such as Amazon's CloudWatch, a cloud audit system should be able to use a monitoring system as a data source, compare observed events against predefined criteria and come to a conclusion if the system is acting in a compliant manner. Ideally, this happens continuously. This is in contrast to conventional

audits based on ISO27002, COBIT or similar standards, which are not adopted to multi-tenant cloud environments, require too much human intervention during an audit and do not support automated continuous auditing. This is also acknowledged by Chen et al., who in their work established a checklist for auditing cloud computing [102]. The checklist includes, among others, cloud-specific criteria such as data location, isolation and cloud-focused disaster recovery.

Especially in Platform as a Service (PaaS) and SaaS clouds, which are more software-centric, the choice of what needs to be logged for evaluation during audits is of utmost importance. According to Guts et al., this choice is typically made by the software developer during service development [103]. This highlights the problem of evidence sources and whether or not logs that are produced on these layers are complete or not. In general, this depends on the quality of the logging mechanisms and monitoring / auditing APIs that are integrated into the software.

As a result, it can be said that one of the most important aspects of cloud accountability auditing is to carefully consider possible sources of data (e.g., logs) and their evaluation.

3.2.5 Security Service Level Agreements

Service Level Agreements (SLAs) are typically part of a contractual relationship between cloud providers and their customers. Security SLAs define for instance how long it may take a service provider to resolve a vulnerability depending on its severity. The vulnerability can thereby be uncovered by regular vulnerability scanning, which is typically part of an organization's baseline information security strategy. The severity and associated mandatory time to resolve requirements can be for example: high severity - 24h, medium severity - 3 business days, low severity - 5 business days. Other metrics as part of security SLAs could be the number and severity of published, unpatched security vulnerabilities or environments with missing security updates. For instance, in a

PaaS use case, the customer is typically relying on the provider's capability to manage the underlying software stack. This of course also includes basic patch and change management processes on the infrastructure as well as the platform layer. For security-critical deployments, compliance with SLAs that cover such metrics is crucial. The actual values are thereby subject to negotiation between the provider and its customer.

SLA-reporting is an important tool used by providers to report on their compliance with SLAs. Such reports serve as the basis for financial remediation of failures (e.g., time resolve a security vulnerability exceeded maximum threshold). For many customers, this is a sufficient tool that provides adequate assurance on the correct implementation of security processes and systems at a cloud provider.

However, SLAs need to be monitored and validated continuously by both the service customer (is the provider acting as expected) and provider (demonstrating compliance and contract fulfilment). A system that monitors SLA compliance has to rely on the provision of sufficient evidence in the cloud environment. Both, audits and SLAs enforce improved risk management and can therefore uncover gaps in a provider's security posture [27]. Depending on who controls whom (e.g., the provider controlling itself or a customer / auditor controlling the provider) different level of access and evidence sources are available. For instance, a cloud provider could directly integrate with the patch management system in order to monitor security patch compliance continuously. A customer is much more limited in that regard that tight integration is not possible but he has to rely on black box testing for vulnerabilities (e.g., by verifying an unpatched or insecure SSL library via remote testing). Different approaches can mitigate this problem, where either the provider grants extended access and insight into its infrastructure to the customer/auditor or provides sufficient evidence transparently via corresponding interfaces. The former approach requires increased trust by the provider to allow for a deeper insight into the inner workings of its

services, whereas the latter requires trust by the customer in the truthfulness of the provided data.

3.2.6 Overview of Industry Practices and Systems

When looking at cloud audits and the associated process of collecting evidence to assess policy compliance, it is important to look at existing industry practices regarding monitoring and auditing. Many tools, such as the well-established Nagios [104], support distributed data collection for service monitoring purposes. In more complex IT infrastructures, Security Information and Event Management (SIEM) systems are the main source of monitoring information. They provide additional means of detecting security incidents by collecting information from various sources in the infrastructure.

In the following, a list of the currently influential commercial SIEM solutions for both data center and cloud monitoring is presented. This list is based on [105], which evaluates the strengths and weaknesses of these solutions. Here, only outstanding characteristics of the solutions are presented:

- HP ArcSight [106] is part of HP's Enterprise Security Products (ESP) and targets midsize businesses with preconfigured monitoring and reporting.
- IBM Security QRadar SIEM [107] includes log management, event management, reporting and behavioral analysis of networks and applications. It aims at midsize and large enterprises.
- LogRhythm [108] includes log management and event management. Additionally, it supports network forensics in the form of packet capturing capabilities and collection of data on the major operating systems using agents.
- McAfee Enterprise Security Manager [109] provides database activity monitoring, application monitoring and global threat intelligence.

- Splunk Enterprise [110] provides log management, search alerting, real-time correlation and a powerful query language with extensive visualisation capabilities. It also has a very strong presence in IT operations, today.
- Logentries [111] collects and analyzes log files. It is provided as SaaS and integrates other technologies such as AWS.
- CloudPassage Halo [112] provides security features for cloud deployments by supporting firewall automation, vulnerability monitoring, network access control management, security event alerting. It uses a special agent to collect data and enforce policies.
- AWS CloudWatch [113] allows the monitoring of AWS services including the capability to collect metics and logs, and set alarms.
- AWS CloudTrail [84] allows the logging on the AWS API (e.g., identity of the API caller, operation etc.) including requests and responses.
- Rackspace Monitoring API [114] allows customers to query the rackspace Cloud Management System (CMS) for performance metrics on the provider infrastructure. Additionally, they provide an agent software for installation on the customer's VM that exposes similar informaion via the same API.
- Salesforce ReST API [115] exposes comprehensive monitoring information and since the *Winter 2016* release, this also includes the audit trail generation for customizations actions done by service administrators.
- Alert Logic [116] considers cloud security from a shared responsibility perspective where the cloud customer and the cloud provider have different obligations with respect to information security depending on the cloud service provision model. It thereby combines information provided by both parties in order to monitor security posture.

Many of the above mentioned systems make use of logging information that is generated in the IT infrastructure. The extent to which information is captured

from the environment is thereby dependent on the focus of the tool (e.g., network capturing, database vendor-specific adapters). Also, there are cloud-focused solutions such as AWS CloudTrail, that allows logging on the API level with tools that enhance analysis capabilities of AWS (such as Logentries). Most of the tools have in common that a variety of adaptors is required to collect security events from different sources.

3.3 Software Agent Technology in Cloud Monitoring and Audit

The concept of software agents is not a new one. Much of the literature in this field dates back to concept that were developed in the field of (distributed) artificial intelligence in the mid to late 90s. A brief introduction to the topic domain is given in Section 5.1. The focus in the following literature discussion is put on projects that leverage software agent technology similarly to AAS, especially concerning the collection of data in distributed environments.

In [117], an approach to monitoring grids using agents is presented. Their system aims at enabling the quick and dynamic deployment of monitoring sensors. In addition to sensor agents, they introduce a sensor manager that exports a subscription based interface for monitoring to consumers. The actual software that generates sensor data is thereby loaded after agent deployment from a central sensor repository. Similar monitoring systems that aim at networks are presented by Sladic [118] and Pugazendi [119]. In both systems, agents are used to integrate data sources and decentralize the processing of data and reduce load on the network by shifting workload towards the data source.

Yi et al. [120] describe an approach for information gathering in the World Wide Web (WWW) using mobile software agents that collect specific data for a client. These so called Information Gathering Agents (IGAs) thereby include cryptography based mechanisms to introduce non-repudiation and authenticity into the

gathering process.

This list of related projects is an indicator of the popularity and usefulness of agent-based systems in the context of data collection.

Similar work on adapting Multi-agent System (MAS) to cloud monitoring is done for instance in the mOSAIC project [121], where cloud users are enabled to monitor their virtual resources using an infrastructure that relies on mobile software agents to collect data and execute management operations. Beyond that, there is also the work of Dölitzscher et al. that utilizes software agents for performing security audits in a IaaS cloud infrastructure [96, 97].

AAS uses mobile software agents as an essential architectural paradigm for the implementation of the evidence collection and evaluation process.

3.4 Summary

In this Chapter, related research projects have been presented. A special focus was given to literature on the cloud digital forensics process including evidence collection and processing, since these are very important aspects of the Audit Agent System (AAS). In the second part, relevant literature on cloud audits and compliance with respect to existing information security standards and regulations has been presented. A major problem in cloud computing is the lack of transparency regarding security and privacy controls implemented by the cloud providers. In this research, it is argued that customer trust and cloud adoption can be increased, if cloud providers act accountable by implementing appropriate data protection mechanisms and demonstrate compliance by providing evidence-based audit capabilities to cloud auditors as well as customers.

From a forensics perspective, AAS combines aspects from the aforementioned approaches and integrates them in a single system. The most important factors thereby are the use of a common evidence record format that is flexible enough to address heterogeniety of evidence sources in the cloud and infor-

mation security aspects that are covered largely by implementing integrity and confidentiality-preserving mechanisms, while still maintaining a degree of scalability that enables the implementation in a cloud environment. Furthermore, standards and best practices, which are currently the basis for conventional manual audits are enabled by AAS's focus on integrating policy languages that aim at making the abstract requirements contained therein, machine-readable and evaluatable by computer systems. The AAS approach comprehensively integrates all aspects and thereby provides an approach for continuous, automated auditing of cloud provider accountability even in complex chaining scenarios.

Chapter 4

Cloud Privacy and Accountability Audits

In this chapter, the requirements for AAS are elicited. To frame the discussion in the remainder of this thesis, a general scenario is described that defines a use case that is complex enough to demonstrate particular problems of multi-provider scenarios, yet simple enough to be useful for demonstration of issues throughout this document and in the scenario-based evaluation in Section 7.

The methodology of requirements elicitation thereby follows a structured approach that starts with the discussion of potential sources in the cloud. Based on the methods and principles of digital evidence collection an overview of potential evidence sources is presented alongside a discussion of architectural layers where evidence can be produced in the cloud.

With respect to the collection process, requirements that are mandatory in the conventional non-cloud (or even non-digital) forensic process serve as a starting point from which specific requirements for AAS are derived. A simple process is proposed that serves as a framework for the implementation of the AAS prototype.

Beyond the evidence collection process and the discussion of potential evidence

sources, approaches to evidence processing and evaluation in different types of audits are discussed.

Results of the research presented in this chapter have been published in several workshop and conference papers that focus on the discussion of evidence sources in cloud computing for conducting accountability audits [122–124].

4.1 General Scenario

In this section, a scenario is described that serves as a tool for framing further research in the following chapters. The main aspects to be covered are:

Combination of cloud service delivery models: the scenario should combine at least two service delivery models (i.e., IaaS, PaaS and SaaS). The combination of different service delivery models (e.g., a SaaS provider hosting its service on-top of an IaaS provider's infrastructure, such as Dropbox hosting on Amazon EC2) provides interesting flows of data (across boundaries between providers) and levels of data abstraction (e.g., objects in memory, datasets in databases or files on block devices).

Interaction of multiple cloud service providers: the scenario should include at least two distinct cloud service providers. Transparency issues can arise, if a cloud consumer does not know of another provider taking part in the service provision.

Focus on public cloud offerings: publicly available cloud services are to be used. Private cloud models do not necessarily stress data protection and privacy problems as much as public cloud services.

Processing of sensitive information: potentially sensitive data is handled by at least one of the service providers. This is important to stress the need for data protection mechanisms that are implemented in the cloud service as well as the audit system.

Requirements regarding data privacy and security: privacy and security requirements do apply, either originating from the nature of the processed data (sensitive or confidential data) or because security and privacy controls are required by law.

An example scenario is depicted in Figure 6. The flow of data is depicted by the connections between the different stakeholders. The following roles and responsibilities are assumed in this scenario:

- A company (*E*) is the customer of a cloud-based Customer Relationship Management (CRM) provider. The end-user (an employee of company *E*) is using the CRM software. The user at *E* interacts with the CRM application using a web-interface that enables the customization of some aspects of the service.

- Cloud Provider 1 (*CP1*) provides the CRM software as a SaaS application to *E*. *CP1* does not possess computing resources itself, but uses third-party cloud providers in order to provide its own service. Primarily, CP1 has contracts with CP2 for infrastructure services and with CP3 for additional backup services.

- Cloud Provider 2 (*CP2*) is specialized in IaaS services and hosts the CRM software of *CP1* on VMs.

- Cloud Provider 3 (*CP3*) is specialized in backup services (SaaS) and provides this to CP1, which integrates this as an optional service in its CRM service.

This scenario is of particular interest, since transparency of data processing in the cloud is reduced by business relationships between cloud providers that are not necessarily openly communicated to cloud customers (here: CP1 using a third-party service for backup provision). Employees at E handle sensitive business-related data in the CRM software on a regular basis (e.g., customer

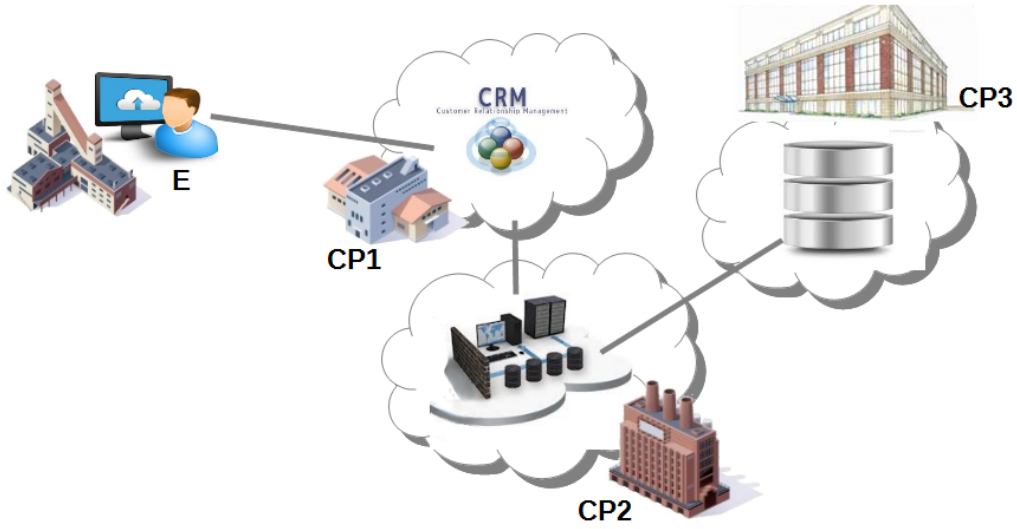

Figure 6: Abstract Overview of the Demonstration Use Case

information like names, addresses and history of business transactions). Therefore, the company is concerned about the protection of this data, because they have no direct control over the infrastructure at CP1, CP2 and CP3 where the CRM is hosted. To address this problem, E has carefully considered data protection and handling policies before finally settling on using the services provided by CP1. However, E wants to have assurance that their agreements with CP1 are being honoured and that CP1 is acting according to regulations and agreements on which they settled. Both, *CP2* and *CP3* are constantly being audited by *CP1* regarding privacy, accountability and security requirements.

Company E and CP1 have agreed to the following security and accountability-related contractual clauses and policies:

- CP1 should put mechanisms in place that effectively protect E's data. This includes that all service providers involved in the service provision chain need to put baseline security controls in place. Furthermore, CP1 shall pro-actively control the effectiveness of those mechanisms.

- CP1 should adhere to a maximum data retention time after which it is forced to delete it.

- CP1 has to fulfill compliance requirements of E that force E to store its data under European Union (EU) jurisdiction, only. In any case, no data transfers outside the EU are allowed. Therefore, data must only be stored inside the EU regardless of *CP1*'s performance and cost evaluations of sub-providers.

Furthermore, all providers CP1, CP2 and CP3 are assumed to each implement an instance of AAS as it is described in this thesis. There is also a certain level of trust between the cloud providers that enables them to cooperatively perform audits.

The Auditor in this scenario is considered to be a trusted third-party or a technically capable person at CP1 for the connections $CP1 \leftrightarrow CP2$ and $CP2 \leftrightarrow CP3$. In any case, company E is informed by the auditor about policy violations that might occur.

4.2 Sources of Evidence

Evidence in digital forensics focuses on information left behind by a (careless) intruder and reconstructing the chain of events of an incident. Evidence collection for cloud accountability audits needs a planned approach to gain control of the sheer amount of potential evidence data in the cloud and to not create a potentially dangerous information repository that may leak sensitive data. Data minimization and purpose binding (see Section 4.6 for more details) are of utmost importance. Therefore, the evidence sources and specific evidence objects to acquire need to be known beforehand to enable the appropriate monitoring of the evidence sources.

4.2.1 Types of Evidence

The sources of evidence can be diverse, ranging from business process logging to operational logging. Operational logging covers actions of a single cloud customer or critical infrastructure conditions that impact all customers. In the following, an approach of classifying evidence types for cloud privacy and accountability audits is presented based on the literature review presented in Section 3.1:

Logs: Logs are generated at many different places for all kinds of purposes such as failure handling, compliance verification or debugging. In general, logs are typically a collection of temporally ordered events.

Data Provenance: Recording where data originates, how, where and by whom it is processed, as well as where (in terms of geographical location) it is transferred, is invaluable information when auditing against data processing policies. This information is tracked typically on the software and platform layers. On the infrastructure layer it may not be possible to track single data objects due to the coarser abstraction of data in files and filesystems. On that coarser level, data can be tracked for instance using the CMS's VM image location and migration history.

Content: Any kind of data such as e-mail messages, social network messages or data that is being transmitted on the network.

Files: Especially in digital forensics, files and the contained information are valuable sources of evidence (e.g., documents, configuration files, temporary files).

Cryptographic Hashes and Proofs: Methods for attestation, such as PDP and PoR (see also Section 3.2.3) contribute as evidence to audits.

Business Process Documentation: Most commonly this kind of evidence is the central object of interest in conventional document-revision based audits.

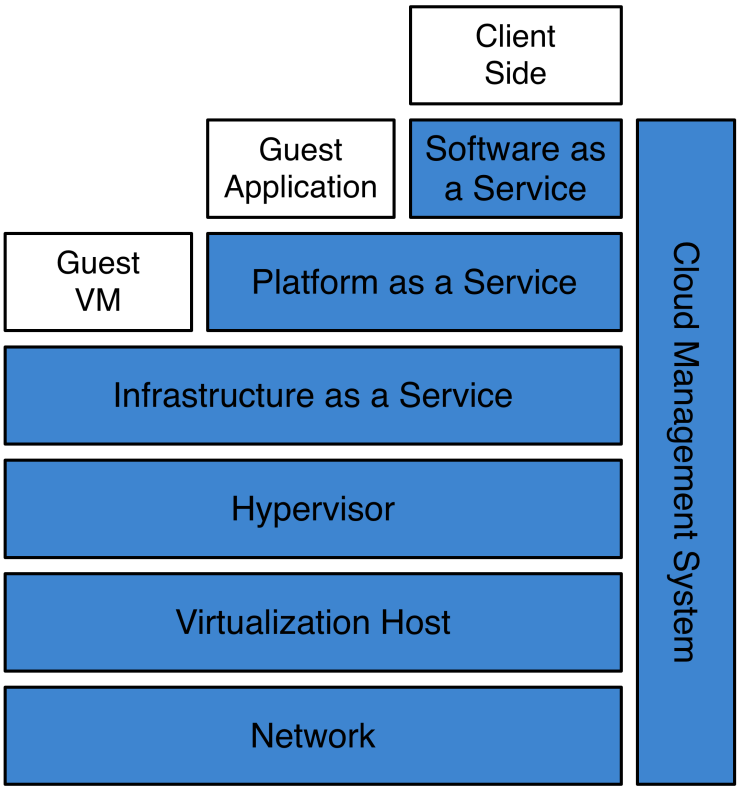

Figure 7: Sources of Evidence for Accountability Audits

Anything indicative: While the aforementioned examples for evidence types are most commonly looked at, anything that holds some information that can contribute to either stating compliance or indicating a violation can be considered evidence.

4.2.2 Sources by Cloud Layer

In the following, a classification of potential evidence sources is presented (depicted in Figure 7). The categories of evidence are presented according to the typical architectural layers of the cloud. On the lowest level, there are the network and servers that provide computing and communication resources. On top of that, there is the Hypervisor as a central component for resource allocation, virtualizaton and tenant isolation. Based on these layers are the core cloud service models, representing the use of infrastructure resources (IaaS), application

runtime environments (PaaS) and software (SaaS), with the respective boundaries between the tenant (marked as white boxes) and provider (marked as blue boxes) administrative domains. As a cross-spanning source of evidence, there is also the CMS.

Network: It is required to monitor networking resources, which are utilized by the cloud provider and by particular tenants. Networking resources can be either physical or virtual. From the provider's point of view, physical network resources are most important. This includes active network components such as switches, routers and physical network interface controllers of virtualization hosts.

Flow information that is gathered on these components can be invaluable evidence of data flows in complex cloud deployments. If a cloud provider is using SDN, that traffic flow information is usually already converging at a central node called the SDN controller. From a cloud customer's point of view (especially in IaaS) the network layer is typically a virtualized one. A common way of segmenting networks on a tenant level is by using technologies such as Virtual Local Area Networks (VLANs) [125] (which is a traditional non-cloud technique), Virtual Extensible Local Area Networks (VXLANs) [126] (which extends and adapts the VLAN concept to the cloud) or VPNs (which is typically used to bridge the gap between cloud and enterprise network). All of these can be used to virtualize networks and isolate tenants from each other.

In any case, tenants share networking resources just like CPU, RAM and storage since they are ultimately co-located on the same physical server. For instance, in IaaS, a single physical network card in a virtualization host is utilized by several VMs by bridging virtual interfaces to physical ones, despite them belonging to different tenants. This is typical in a multi-tenant cloud environment and one of the easiest ways to provide networking to a customer's VM. Distinguishing a customer's traffic on such a shared device

may be possible by either filtering using coarse attributes like logging IPs and communication endpoints on the full data stream of the shared physical device or by monitoring individual virtual network interface controllers.

On higher levels like PaaS and SaaS, distinguishing between different customers' traffic flows can become quite difficult since virtual devices are typically not a suitable identifier of different customers. The reason for this is that there is no clear mapping between a virtual resource, such as a virtual machine and its flow-relevant identifiers (e.g., source IP address) anymore. Additional identifiers from the application layer need to be considered to clearly identify flow belonging to different customers.

Nevertheless in cloud audits, traffic flow information is important to reveal violations of security policies related to data location and transfer (e.g., the detection of compromised hosts that have traffic flows to disallowed locations). Logging of traffic flows is therefore an important source of evidence, especially in IaaS deployments.

Furthermore, evidence that is coming from network device monitoring can be required as well. That includes availability monitoring and routing information. This data can be of particular importance to uncover disallowed cross-border flows of data due to routing changes that are caused by availability incidents. In any case, the information that is collected on this layer provides deep insight on the flow of data in a cloud but is at the same time very low-level information that requires extensive filtering and a clear definition of which flows are allowed and which not.

One example for this layer is the Cisco-proprietary IOS NetFlow [127] protocol, but also information that is gathered from network components (e.g., performance counters such as *packets transmitted*) using Simple Network Management Protocol (SNMP) (see RFCs 3410-3418).

Virtualization Host: Evidence collected on the virtualization host operating system level (i.e., nodes hosting VMs) includes data that are produced on the

host. Such data include operating system performance counters, system log files (e.g., from authentication and authorization logging) or other information produced by the operating system or complementary monitoring tools. These evidence sources are not necessarily cloud- or virtualization-related. Since a virtualization host is just a typical server, most server monitoring mechanisms do also apply in the cloud case. Also, this is the layer, where most of the digital forensic techniques (e.g., file carving, memory dumping) can be used for evidence collection.

Examples for this layer are files in the VM such as documents or configuration files, logs such as syslog, access log and kernel counters such as those provided by the proc filesystem. Also, it is good practice to monitor servers for specific parameters such as load, availability or intrusions. That monitoring information can be valuable as well. Typical examples of monitoring systems that collect such information were presented in Section 3.2.6.

Hypervisor: The Hypervisor (e.g., Kernel-based Virtual Machine (KVM) [62] or Xen [61]) operates VMs. It has full control over VMs and assigns physical resources of the underlying virtualization host to them. The control over the Hypervisor and the virtualization host is taken by the CMS. On this layer, runtime statistics generated by the Hypervisor, such as resource utilization, or statistics about memory ballooning, shared pages between VMs (both of which are commonly used Linux kernel-level techniques to save memory on the virtualization host and therefore allow it to run more VMs in parallel) are considered potential evidence sources.

Because of the level of control that the Hypervisor exercises over VMs, observation of virtual machines from the Hypervisor's point of view can also provide valuable evidence. A VM's memory is a mapped portion of the virtualization host's physical memory. It is therefore trivial for the Hypervisor to "look into" a VM by simply scanning the right memory area. This principle is being used in advanced intrusion detection mechanisms that have the

goal of detecting sophisticated intrusion attempts that otherwise may not be detectable due to anti-forensic countermeasures. VMI is one of these approaches that allows for deep insight into the internal procedures of a VM without modifying it or being detectable from the inside of the VM.

A review of related work in this field is presented in Section 3.1. Obviously this level of access provides nearly limitless potential for monitoring and collecting evidence.

Infrastructure as a Service: At this level, data are generated inside a VM or with the use of the CMS. One of the most interesting evidence acquisition techniques is VM snapshotting. At any time, a copy of a VM (a snapshot) can be created to preserve the precise state of a VM at that particular point in time. This includes the state of processor registers, Random Access Memory (RAM) and open network connections. This is not limited to VMs, but also includes any block storage (in this case, the state of a data object is preserved).

While the previous layers had a strong focus on the cloud provider side, it is noteworthy that at this level, there is a shift of the administrative domain towards the cloud consumer. Anything running inside a VM in an IaaS scenario, the cloud consumer has full control over. There is only very limited possibility for the cloud provider of influencing a VM's internals and neither is it in his interest to do so, since this would be against the IaaS business model.

Therefore, any evidence that is collected inside a VM in IaaS is not collected by the provider, but the consumer. The types of data are thereby very similar to those of the virtualization host layer as well as the PaaS and SaaS layers depending on what the consumer uses the VM for. From a provider's and auditor's perspective, this has direct influence on the trustworthiness and usefulness of the collected data, since a VM could have been breached due to lacking protection mechanisms or manipulated by the cloud con-

sumer. This makes information stored on it untrustworthy.

Examples for this layer are files in the VM such as documents or configuration files, logs such as syslog, access logs, kernel counters such as those provided by the proc filesystem and also full copies of VMs that were generated by snapshotting or cloning.

Platform as a Service: In a web application development scenario, evidence collection can be performed by the platform provider as well as the cloud consumer (i.e., the application developer) on the PaaS layer. On the provider side, runtime environment logging of the provided platform is of utmost importance. Of course, evidence on the underlying IaaS layer is obtained as well.

However, the virtualized resources are provisioned for the platform services and therefore owned by the cloud provider instead of the consumer (see IaaS layer for reference). The major concern regarding information collection on the PaaS layer revolves around the segregation of multi-tenant log information at the provider-side. The consumer-controlled evidence source in this layer is pushed upwards towards the application that he develops (i.e., the VM is not under the customer's control anymore, but the application's logs are). Logs and performance monitoring data can only be considered at this layer, if corresponding mechanisms are being implemented in the application and interfaces are being exported that allow access to that information.

Examples for this layer are logs produced by the webserver, Java Runtime Environment, database service, identity management and access control service and also logging that the deployed application generates.

Software as a Service: On the SaaS level, evidence may come from audit and logging APIs provided by the SaaS service provider. Such APIs include authentication, action and access logging. However, the actual content of such data is highly dependent on the cloud provider and service type. Many

cloud providers offer such APIs to consumers under the label of monitoring or audit interfaces from which they can export that data and analyze it further.

Another source of evidence on the SaaS layer is transient data stored on the client side. In most cloud usage cases, the client application is a web browser. The web browser stores information such as a browsing history or locally-cached data from a browsing session at a service. That information can potentially contain valuable information. However, evidence collection on the client side requires additional mechanisms (e.g., logging actions of users or debug logs produced by the client side logic in the browser) and sending that information back to the service.

An example of this layer is the ability to export audit logs for an organization that is using a SaaS service such as the Google Apps Activity API [128] or for instance the Reports API for Google Drive [129]. For the client side logging, a simple library such as JSNLog [130] can be sufficient.

Cloud Management System: The CMS is a major source for evidence. It is the central controlling component of a cloud infrastructure and provides information about user logins, cloud service usage, access rights, configuration, resource provisioning, location and much more.

In IaaS the CMS manages physical resources and provisions them in virtualized manner. In PaaS the platform manager is responsible for the deployment environments and uses the underlying infrastructure manager to deploy them. In SaaS the same happens, but the abstraction is even more coarse and the underlying cloud infrastructure fully transparent to the consumer. The CMS (be it IaaS, PaaS or SaaS) has therefore the most complete picture of what is happening in the cloud and is therefore one of the most important sources of evidence. However, the usefulness of the provided data is highly dependent on the amount and quality of monitoring and logging information the CMS generates as well as the APIs it provides

to export that information.

Examples for this layer are OpenStack [14] as an IaaS focused CMS, Cloud Foundry [131] as a PaaS focused CMS or any SaaS service that provides its consumers with a dashboard and API.

This categorization of evidence sources shows that a system for evidence collection primarily needs to be capable of collecting data from various heterogeneous sources. The evidence can then be evaluated during an audit and compared against rules that govern data processing in the cloud (e.g., data location evidence that is compared with restrictions on locality). The cloud's multi-layered and heterogeneous environment has significant influence on the proposed system architecture described in Section 5. For instance, logs from different tools, while being similar in general structure (e.g., typically one line per event, beginning with a time-stamp), differ very much in the used syntax (e.g., time-stamp format, order of event elements).

To address the heterogeneity of the evidence sources an adapter-based approach for evidence collection, where each type of collector provides the capability to interface with another type of source, looks promising. More precisely, for every data source there is a specialized collector, which is aware of the syntax, semantic and interface definition of the evidence source on one side and of the syntax, semantic and interface definition of AAS on the other side. For instance, the collector may use an evidence-generating tool's API to collect information, monitor log files for events or parse the output of analysis tools.

4.3 Evidence Collection

The term evidence collection usually refers to forensic methodology and techniques. Digital forensics is a technique that covers the collection and investigation of evidence that is usually tied to an incident or crime. Cloud forensics refers to digital forensics investigations performed in cloud computing environ-

ments. As discussed in Section 3.1, there are some special requirements and challenges introduced by cloud computing. In the following, the adoption of the digital forensic process for use in evidence-based cloud audits is discussed. As a basis for the view presented in this section, well-understood principles and requirements for a digital forensics process have been collected during a literature review. In the following it is shown how principles and techniques of (digital) forensics can be applied to improve the credibility of audit results.

4.3.1 Principles and Techniques

The core properties of evidence, and thus also digital evidence, are described based on [132] and [133]:

Admissibility: Evidence must conform to certain legal rules, before it can be put before a jury. It is influenced by the transparency of the collection process and data protection regulation. Typically, a judge decides upon the admissibility of evidence in court.

Authenticity: Evidence must be tieable to the incident and may not be manipulated in any way and must be protected against any kind of tampering (intentionally and accidentally).

Completeness: Evidence must be viewpoint-agnostic and tell the whole story. This needs to be ensured by the evidence collection process as a whole. It is especially important that the source from which data is collected is known. A mechanism should complement the evidence collection process by providing assurance that all stored data are made available as evidence, and are not cherry-picked.

Reliability: There cannot be any doubts about the evidence collection process and its correctness.

Believability: Evidence must be understandable by a jury. This is typically

ensured by the interpretation and presentation by an expert in court. Since judges and juries are usually non-technical, an abstracted presentation and interpretation of digital evidence is required.

These principles apply to conventional evidence in non-digital forensics as well as digital evidence. Therefore, the evidence collection process for cloud audits has to consider special requirements, which help addressing these properties and ensure best possible validity in audits.

4.3.2 Evidence Collection and Evaluation Process for Cloud Accountability Audits

The process of a digital forensics investigation can be separated into different phases as defined by NIST in [134]. Each of these 3 phases has its own specific purpose in the chain of proving that a cloud environment is accountable:

1. **Securing Phase:**

   The major intention is the preservation of evidence for analysis. Data are collected in a manner that maximizes their integrity. A bitwise copy of the original media usually achieves this in conventional digital forensics.

   As previously mentioned, this represents a huge problem in cloud computing where it is typically unknown where a specific data object is located. Additionally, there is often no access to any physical hardware for an investigator, since this is solely under the control of the cloud service provider.

   However, state-saving technologies in virtualized environments (such as snapshotting) provide powerful tools to freeze system states. Investigations similar to those conducted in non-cloud environments, at least in IaaS scenarios, are made theoretically possible by such technologies. In any recent virtualization technology, pausing a virtual machine, saving the (volatile) memory contents and (non-volatile) disc contents and therefore enabling

the "saving" of the state of a system at a specific point in time, is a trivial task.

In PaaS or SaaS scenarios, where possibly multiple distributed systems (e.g., a n-tier architecture) are involved in service delivery, securing evidence for forensics is a more complex task. The principle of pro-actively monitoring and describing beforehand which data is required as evidence becomes more important in the context of cloud audits. According to these policies it can then be decided which evidence data are important.

An alternative approach to pro-actively defining relevant data sources and event types before evidence is collected is to simply collect any data that may become relevant for an audit. However, this leads to significant amounts of data being transferred over the network and stored in central archives. Amounts that can quickly become too large to handle or efficiently process. Furthermore, building such heaps of data just in case can potentially introduce privacy protection threats if correlation with human beings is possible and the data storage is insufficiently protected against leakage or loss.

Due to the heterogeneity of potential evidence sources, the securing phase must use approaches that are flexible enough. Using adapters that transform data into a common format is a common approach to address this problem. Of course this introduces a considerable amount of effort that has to be invested into developing adapters for each potential evidence source.

2. **Analysis Phase:**

   In this phase, data from multiple sources are pulled together to create a complete picture of an incident by reconstructing the trail of events. Especially in distributed system infrastructures, like clouds, this means that bits and pieces of data are pulled together not only from a single system or source, but from various parts of the cloud. Because of the multi-tenancy of cloud infrastructures, the collected information may not only come from the administrative domain of the cloud provider (or several providers) but

also from the cloud consumer (e.g., the customer in IaaS).

In AAS the scope and extent of the evidence analysis (e.g., simple string searches over logs or feeding them to a complex ruleset in a rule engine) is decided by the audit task developer based on his decision on scope and depth required for evaluating policy compliance in an audit. Both factors are tightly coupled to cost of computation, bandwidth and storage for the collection and evaluation of evidence. With respect to resource requirements, executing the analysis in the cloud as well can be considered.

The more complex analysis techniques consider multiple evidence sources during analysis. This however, introduces the requirement of integrating and correlating sources that differ in format and semantic. Therefore, the analysing entity must be aware of the types of evidence sources and methods for processing that data.

3. **Presentation Phase:**

   The report created in this phase is a compilation of all the results, relevant documentation and accompanying evidence from the analysis stage. The main intention of such a report is that it clearly and understandably draws a picture of what happened. The timeline presented in the reports is also of high significance because of the dynamic character of cloud environments (i.e., time synchronization mechanisms are required at all evidence sources).

   The presentation of evidence and results acquired during the analysis phase is also affected by the heterogeneity of cloud evidence. While some evidences (both collected and results from the analysis phase) may be considered monitoring metrics, others can be logs or even plain text. This has to be considered in a technical system that aims at presenting results in an easily understandable way.

Overall, the requirement for integrating varying types of evidence sources has a big impact on the architectural choices for AAS. This choice relies on the use

of adapters and containers with specific tasks that are aware of the syntax and semantics of their associated evidences.

The above-mentioned process forms the basis for the accountability audits in AAS. The securing phase thereby follows a planned approach of collecting evidence in a well-defined way and storing it securely. The analysis phase reflects the performed analysis of evidence against policies during the audit. The presentation phase presents the audit result in a suitable way.

4.3.3 Security and Privacy Protection Threats

Pro-actively collecting evidential data for the purpose of auditing against data protection, security and accountability policies can introduce new risks. The main risks are, based on a literature review of [133], [65], [53] and [135] are presented below:

Legal: Collecting certain data may be against legislation and therefore require a search warrant. In cases where multiple jurisdictions are concerned, this problem is intensified.

Data breach: Piling-up a lot of data that can potentially include sensitive information can make the pile a target of attacks that aim at stealing that data. Also, the privacy of another tenant might be breached, if its data is collected.

Broad access: Evidence collection itself does require broad access privileges on the targeted systems. In live analysis, where evidence is collected pro-actively, this may have negative impact.

Collection process security: The collection process itself may become a target for manipulation.

4.4 Evidence Processing

The processing of evidence in AAS is concerned with the automated preparation and evaluation of evidence against rules that are defined in security, privacy and accountability policies. In the following, different areas of application are presented.

The processing of data includes, as a first step, the filtering and reduction of data. According to [135] this can happen either at the acquisition (securing) stage or at the examination (analysis) stage. While AAS conceptually supports both approaches, the former approach is preferred in order to adhere to the data minimization principle described in Section 3. The filtering at the collection stage has also obvious advantages with respect to scalability when load can be reduced (see Section 7.3).

Since the evidence collection process in AAS is pro-active and policy-driven, the following data processing techniques can be applied at the collection stage as a form of pre-processing:

Filtering: Evidence sources can produce a lot of data, that may not necessarily be useful or even required for evaluating policy compliance. Filtering mechanisms such as selection of events based on keywords or regular expression reduce the amount of data that is being collected.

Reduction: Depending on the evidence source, different data formats (e.g., eXtensible Markup Language (XML)-based, Javascript Object Notation (JSON)-based) are collected. Reduction mechanisms aim at reducing the size of data by removing unneeded information that is not required for the compliance check (as defined by the auditor) and overhead from the input data.

Aggregation: In cases where many events are produced that share a similar structure and maybe even redundant information, the collection information can be reduced by carefully aggregating multiple events into one or

removing redundancy.

Anonymization: Anonymization does not reduce the size of evidence but rather addresses privacy issues very early in the process by removing PII unreversably. However, the applicability of the technique during evidence collection highly depends on its impact on the usefulness of the evidence afterwards.

Compression: Compression can further reduce size of collected evidence, but introduces additional computational load and requires decompression before analysis.

4.5 Audit Types

Auditing is defined as a "systematic, independent and documented process for obtaining audit evidence and evaluating it objectively to determine the extent to which the audit criteria are fulfilled" [136]. In this Section, two aspects that allow for a more detailed categorization of audits are discussed: distribution and interval. Both have a significant influence on how an automated audit is conducted. While AAS supports all of the mentioned characteristics, it is important to highlight which type is suitable for which kind of evaluation methodology and evidence collection.

4.5.1 Event-based Analysis

The first of two general types of analysis methodologies that are considered in AAS is the *event-based* approach. The event-based approach thereby has two distinct variants.

The first variant is similar to stream processing, as the processing and analysis of evidence is pushed towards the collection to achieve immediate results. This approach is particularly interesting for rules that only require one evidence

record for the analysis, so that those can be processed continuously as they are collected.

The second variant is event-driven, which means a collector is monitoring an evidence source for state changes or other triggers (e.g., modification of a file by monitoring the filesystem) and reacts on that event by collecting evidence. This approach is particularly interesting for cases where the triggering event fires irregularly and a periodic collection would result in either too much redundant evidence data or would pick-up the evidence with too much delay.

Both approaches can be used to quickly react to events and verify changes and are therefore important enablers of continuous auditing. However, neither of those are feasible when multiple evidence records over a period of time are required for the audit or multiple sources and therefore collectors need to be integrated to build a single trail of events for analysis.

4.5.2 Analysis of History Data

The second general type of analysis methodology that is considered in AAS is the *periodic* approach.

This is similar to conventional audit practices, where there are often long periods between each interval. In AAS, periodic audits enable the audit of compliance with rules over a period of time, while considering multiple evidence records from multiple evidence sources. In fact, there is no theoretical limit on the amount of evidence sources or records that can be processed during each periodic audit but only a practical one that is defined by the length of the evaluation process, the amount of storage required and load introduced. The length of a period can be chosen arbitrarily depending on the audit task by the auditor. Typical periods are: hourly, daily, weekly, monthly or yearly.

4.5.3 Distribution Aspects of Audits

An important factor to consider in the design of a system that processes data in the cloud is its potential distribution. In AAS the evidence collection process happens in a decentralized way. When it comes to processing the collected data in an audit however, centralized and decentralized approaches are suitable for different types of audits. Key factors are the audit interval and whether an event-driven or history-based approach is chosen.

Centralized Audits

In the centralised approach, multiple evidence collectors generate evidence records and store them in a central evidence repository. From the repository, a set of evidence records can be retrieved for performing policy compliance analysis. Regarding the aforementioned scenario in Figure 4.1 this means, there are multiple evidence collectors deployed at different locations, for instance: a collector for monitoring the access control module, a collector for monitoring the service interface configuration for changes to security parameters such as the available cryptographic algorithms, key material and validity of cryptographic certificates. The hereby collected evidence is saved in the central evidence repository, from which another entity can retrieve it during the analysis phase.

Decentralized Audits

In the decentralized approach, evidence collectors still generate evidence records at the evidence source and store it in a evidence repository, however the repository is not centralized anymore. Additionally, the evidence evaluators of the analysis phase are distributed as well, to achieve a tighter coupling with the relevant collectors. By bringing collectors and evaluators closer together, the data flow between the two can happen without requiring a centralized evidence store in between. With this, the evidence repository is reduced to an archive of

evidence and audit results.

4.5.4 Audit Intervals

In AAS multiple audit intervals are considered. The suitability of a specific interval is dictated by the amount of evidence and the type of evaluation methodology that are required. There are intervals that are more suitable for audits where a constant stream of evidence is analyzed but also others for the analysis of longer periods of time that stretch multiple days, weeks or even months of evidence collection. In the following, three types of intervals are presented: periodic, continuous and on-demand.

Periodic Audits

In periodic mode, evidence is evaluated in an audit at specific intervals (e.g. hourly, daily, weekly, monthly etc.). It is similar to the on-demand audit in having the auditor setup the audit once and have it repeated at the specified points in time, without having to take action. The most useful application of this audit interval is a regular evaluation for policy compliance. For instance, simple tests aimed at the correct functioning of security controls on a daily basis are prime candidates for this interval type.

Periodic audits typically happen centralized for more complex tasks where more than one evidence collector is involved and where historic evidence (i.e., evidence that has been collected during previous periods) is required. This enables audits to perform more complex analysis of larger periods.

Continuous Audits

In continuous mode, evidence is analyzed as soon as it is collected. The continuous audit mode is very similar to monitoring with immediate notification if a violation is detected. The time between evidence being recorded and actual de-

tection of a violation or incident and its notification is minimal. However, since evidence is analysed on-the-fly, more complex evidence analysis that relies on taking a series of records or historic information into account is generally harder to implement. This is especially the case, when evidence records are being produced rapidly. This type is therefore more suitable for evidence that represents rarely happening events such as configuration changes that happen rarely if at all.

On-demand Audits

The on-demand type does not constitute a real interval but is rather a special case of the periodic audit, where the execution of an audit is done exactly once and immediately after configuration. This means, the audit is planned once, which includes the selection of evidence sources, method of evidence analysis and presentation of one result. It needs to be considered that the lifetime of the evidence collector does not last longer than that of the whole on-demand audit. In fact, collectors are being distributed in the system according to their configuration, collect as much evidence as is required and are then destroyed. Any data that is not available at that point in time at the evidence source is not collected and not considered during the audit. The most useful application of this type is when a compliance verification is required immediately and it is not possible to wait for the next evaluation interval to pass.

On demand audits can be de-centralized when a single evidence source and therefore a single evidence collector is involved.

4.6 Requirements for Handling Evidence in Audits

In this section, requirements for cloud privacy and accountability audits are described based on the preceding sections of this chapter. These requirements originate partly from the digital forensics process and partly from general privacy

ID	Description
EH1	Evidence must conform to certain legal rules, before it can be put before a jury.
EH2	Evidence must be tieable to the incident and may not be manipulated in any way and must be protected against any kind of tampering (intentionally and accidentally).
EH3	Evidence must be viewpoint-agnostic and tell the whole story.
EH4	There cannot be any doubts about the evidence collection process and its correctness.
EH5	Evidence must be understandable by a jury.

Table 1: Evidence Handling Requirements for the Audit Agent System

and security considerations.

4.6.1 Evidence Handling

The baseline requirements that arise from principles regarding the handling of digital evidence have been discussed in Section 4.3.1. These requirements are identified in the remainder of this thesis as described in Table 1.

Any system that collects data and tries to preserve its evidential value has to address these requirements to a degree, where the collected data remains useful in court. The above-mentioned evidence handling requirements can be considered equally important, while EH5 could have more impact than the others in a court.

4.6.2 Security and Privacy Protection

Beyond ensuring the usefulness in court, additional requirements were identified, that arise from regulation, standards and best practices for privacy protection as described in Sections 2.4 and 3.2.1.

The following major requirements should also be considered in the system design:

Protection of Evidence Source: On every architectural layer of the cloud data

is generated, which may potentially serve as evidence during audits. The credibility and usefulness of audit results is tightly coupled with the quality of evidence sources. Therefore, the integrity of evidence data must be guaranteed. Data collected from evidence sources must be tamper-proof (evidence cannot be manipulated) or at least tamper-evident (any manipulation is detectable). Without protection from tampering, the evidence collection system is not reliable and audits cannot be performed based on that data. Operational security mechanisms (such as tamper-proof logging) and organizational measures (such as restricting access to potential data sources and collected evidence to a minimal set of employees with authority over evidence collection) need to be put in place.

Confidentiality: Confidentiality of data revolves around mechanisms for the protection from unwanted and unauthorized access. Typically, cryptographic concepts are used to ensure confidentiality of data. By encrypting the collected evidence during storage and transit, compromising the privacy of cloud customer data that has been collected in the evidence collection processes becomes almost impossible. If done correctly, this goes as far as being able to safely outsource the evidence storage to an untrusted third-party. This has direct impact on the scalability of the whole system since storage can be outsourced and does not require a trusted environment anymore.

Data Minimization: This principle states that the collection of personal data should be minimized and limited to only what is strictly necessary. However, data minimization has to always be considered in the context of completeness of evidence. Therefore, evidence should only be collected for performing specific audit tasks, which are very limited in scope. It should never be possible to arbitrarily collect data based on the potential that it may or may not be needed later on.

Purpose Binding: This requirements entails that personal data should only be

used for the purposes it was collected for. Incorporating secure evidence collection and storage serves to differentiate data collected for auditing purposes with other data collected for other purposes (e.g., for marketing). Also, technical means such as Usage Control (UCON) [137] may be used to enforce purpose binding on collected evidence.

Data Retention: Evidential data generated at the different layers may contain sensitive information (e.g., communication logs depicting communication partners, length and date). Such data must be handled carefully and deleted as soon as it is not needed anymore or after a certain period to ensure protection from misuse. These periods are usually defined by legal and business requirements.

However, in the cloud actual data deletion poses a real challenge. Typically, the precise location of a data object is not directly available, (i.e., the actual storage medium used to store a particular block is unknown) making unrecoverable data deletion hard. However, if data has been encrypted before storage, a reasonably safe way to ensure deletion is to discard the key material required for decryption.

Another obstacle is that continuous audits continuously evaluate a system. Thus, a continuous stream of evidence may be needed, depending on the audit task, producing unclear data retention limits (e.g., when is an evidence object *not* needed anymore?)

Public API: Public audit interfaces, which serve the purpose of strengthening transparency of cloud systems and ensuring proper usage of data in the cloud, come with several privacy issues. For instance, actual customer data, usage profiles or any other potential PII should ideally not be exposed to an auditor, as long as this does not negatively impact his ability to conduct audits. Therefore, third-party auditors should be restricted in the amount of information they can request from audit interfaces.

ID	Description
SP1	Evidence source shall be protected.
SP2	Confidentiality of collected evidence should be provided.
SP3	Data minimization regarding evidence collection should be considered.
SP4	Purpose binding of the collected evidence to only be used in audits should be observed.
SP5	Observation of data retention requirements.
SP6	Provide public audit interfaces for second and third-parties.

Table 2: Security and Privacy Protection Requirements for the Audit Agent System

To summarize, from a security and privacy protection perspective, the evidence collection and processing architecture should fulfil the requirements listed in Table 2. While the requirements associated with confidentiality and integrity of evidence are of utmost importance, SP6 can be considered somewhat less important if the audit system is used in a private cloud scenario that does not consider complex service provision chains. However, today's cloud use cases quite often integrate the services of multiple cloud providers and therefore require the use publicly available and open interfaces that follow industry standards.

4.6.3 Data Processing and Performance

As discussed previously in this chapter, evidence for cloud audits is generated at a vast amount of different sources, while many types of audits require a centralized collection of data to build a consistent audit trail of events from multiple sources. With a growing number of evidence collectors, central processing (e.g., storage and evaluation) becomes increasingly difficult to handle. Therefore, ways have to be found to address these problems and make AAS scalable.

In Figure 8 the main problem regarding scalability of an evidence collection system for cloud audits is depicted. A central evidence storage facility is used to archive collected evidence records for processing. Multiple evidence collectors at different sources feed into that evidence store. Also, the evidence processing is done in a centralized way. It reads from the evidence store, processes and

Figure 8: The Evidence Collection Scalability Problem

evaluates the evidence, and produces an audit result. Assuming that there are many different collectors storing evidence, and only a single evaluator reading from the store, three major problems become immediately visible (order of appearance reflects left-to-right view in Figure 8 and not any particular order of importance):

- Event Storms: Events generated at the evidence collector are transformed into evidence records and stored in the evidence store. However, if there are no limitations on the frequency of evidence records that are created at the collector, then, depending on the characteristics of the evidence source, event storms can effectively overload the capacity of the evidence storage. For instance, if the evidence source is a logging mechanism of an authentication and authorization component in a big SaaS deployment, tens of thousands of accesses per minute are not uncommon.

- Storage bottleneck: Since there is only one entity (evidence store) taking evidence records from the collectors, the risk of overloading the system with messages (either by the amount of messages or by the total size) may lead to a bottleneck.

- Processing overload: A similar problem exists at the processing side, which can be overloaded by the sheer amount of processing tasks and evidence

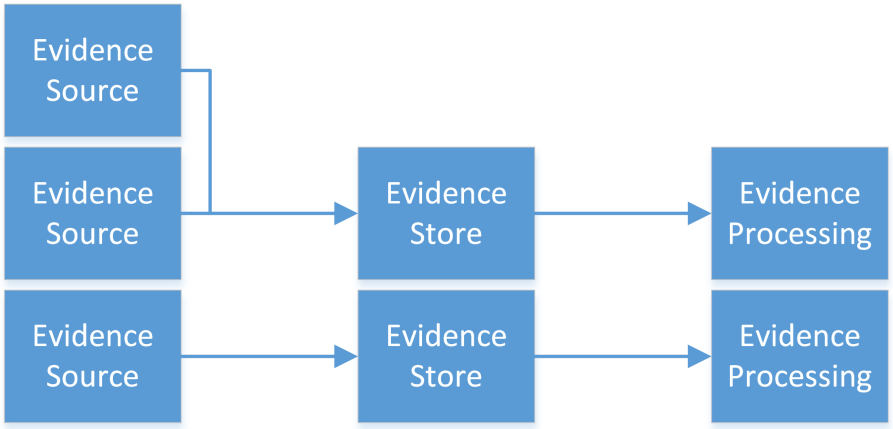

Figure 9: Distribution of Evidence Collection

records that need to be evaluated.

A way to mitigate those problems is to choose an architecture design that allows for distributing the critical components and thereby allowing for the distribution of load. Figure 9 depicts this case, where multiple evidence sources write to multiple evidence stores that again feed into multiple evidence evaluators. In this figure, two distinct audit tasks are depicted (e.g., access control auditing for two different tenants). Each of the tasks is associated with its own set of evidence collectors, evidence store and evaluators. Additionally, the underlying architecture needs to allow the distribution of those components across the cloud on different servers.

To summarize, from a data processing and performance perspective, the evidence collection and processing architecture needs to fulfil the requirements described in Table 3. All processing and performance metrics are considered equally important. A distinction can be made for the performance requirement PP2 where small cloud deployments may not have extensive scalability requirements and therefore load-balancing becomes less important.

ID	Description
PP1	Evidence collectors need to be aware of the type of evidence source to allow for pre-processing such as filtering and aggregation.
PP2	The evidence store may not be a single component, but shall be distributed across several servers to allow load balancing.
PP3	The evidence processing components shall be distributable with a clear association to the evidence store that holds the needed evidence for the audit.

Table 3: Data Processing and Performance Requirements for the Audit Agent System

4.6.4 Extensibility

Previously, it was often hinted towards all the different potential audit tasks that have to be considered in a system such as AAS. Therefore, a system architecture has to take into account the variety of evidence sources in a cloud. It is therefore of utmost importance to provide maximum flexibility and extensibility on both the evidence collector and evidence evaluator side. The evidence collector side requires a different type of adapter for each evidence source due to the inherent heterogeneity of the evidence sources and formats. The adapter can then be used to translate the source-specific format into a common format in the evidence store.

On the evaluator side, a similar approach should be taken to enable the integration of various analysis methods depending on the requirements of the audit task. For instance a simple evaluator takes input evidence from the evidence store and performs simple keyword searches on it. A more sophisticated approach can require the combination of evidence from different sources in a preparation step and feeding that into a rule engine or Artificial Neural Network (ANN). In any case, the audit system should provide the developer community with the necessary tools and architectural considerations to allow extension by adding new collectors and evaluators.

To summarize, from a extensibility and usability perspective, the evidence col-

ID	Description
E1	A simple way of adding new adapters for evidence sources.
E2	A simple way of adding new ways of processing evidence for new audit policies.
E3	A development methodology that allows the reuse of existing adapters and evaluators in new audit tasks that aim at different requirements or obligations from policies.

Table 4: Extensibility Requirements for the Audit Agent System

lection and processing architecture needs to fulfil the requirements as listed in Table 4. All extensibility requirements are considered equally important since they in combination allow a more comprehensive view of compliance based on evidence.

4.6.5 Presentation

The presentation of audit tasks and their results is a critical task. The presentation can take different forms. In more conventional (non-automated) audits, the result is often presented in document form with overview sheets and descriptions. In software supported audits such as AAS this form of presentation is not necessarily replaced but extended with a preferably real-time overview of the audit results.

The main purpose of a Graphical User Interface (GUI) in a cloud audit system is to quickly provide a summary of potential violations that happened in the system. Additionally, further information should be provided on-demand to the auditor to allow for a manual verification of the automatically generated audit result, and presentation of a violation with supporting evidence. However, considering the amount of cloud consumers and overall size of environments in a typical cloud service, challenges in handling and presenting the vast amounts of results needs to be considered. This includes choosing appropriate organization, filtering and graphing techniques in the GUI.

The last important mechanism for presentation is the integration with incident

ID	Description
P1	Support for the generation of audit result reports.
P2	Support for presentation of current audit state in an overview.
P3	Support for integration with incident management tools.
P4	Support for stakeholder notification.
P5	Audit-dependent selection of the appropriate presentation method(s).

Table 5: Presentation Requirements for the Audit Agent System

management and notification tools. While an audit system detects potential policy violations, incident response is an appropriate process to act upon them. Therefore, the integration of an audit system with incident management and response tools or direct notification of relevant stakeholders should be supported.

To summarize, from a audit result presentation perspective, the evidence collection and processing architecture should fulfill the requirements as listed in Table 5. Report generation is an important requirement since documents that describe the compliance state at a certain point in time may need to archived for compliance reasons. The presentation in a dashboard (see P2) is less important from the compliance perspective, but enables continuous tracking, alarming and notification. The integration with incident management tools is important from an incident handling perspective, which could lead to shorter reaction and remediation times.

4.7 Summary

In this chapter, a use case and its major characteristics was presented to illustrate a scenario with multiple cloud providers, different service models and complex data flows. This scenario is used to demonstrate AAS's capabilities.

Also, a general process for evidence collection as well as a classification of potential sources of digital evidence for privacy and accountability audits was presented. The classification is based on the different architectural layers of a cloud

ecosystem where evidential data can be generated.

Following the evidence discussion, methods of evidence processing during an audit were presented. Furthermore, the types of audit were differentiated based on distribution and frequency aspects that take into account the characteristics of cloud environments.

Additionally, a baseline set of guiding requirements for AAS have been elicited. The requirements for the proposed system described in this chapter were derived from requirements for digital evidence and from regulation, standards and general security principles. Together, these requirements frame the evidence collection process and its implementation in AAS.

Table 6 summarizes the elicited requirements for the audit system presented in the following chapters.

ID	**Description**	**Ref.**
SP1	Evidence source shall be protected.	5.3.5
SP2	Confidentiality of collected evidence should be provided.	5.3.5
SP3	Data minimization regarding evidence collection should be considered.	5.3.5
SP4	Purpose binding of the collected evidence to only be used in audits should be observed.	5.3.5
SP5	Observation of data retention requirements.	5.3.5
SP6	Provide public audit interfaces for second and third-parties should be provided.	5.6
PP1	Evidence collectors need to be aware of the type of evidence source to allow for pre-processing such as filtering and aggregation.	5.3.1
PP2	The evidence store may not be a single component, but shall be distributed across several servers to allow load balancing.	5.3.1
PP3	The evidence processing components shall be distributable with a clear association to the evidence store that holds the needed evidence for the audit.	5.3.1
E1	A simple way of adding new adapters for evidence sources.	5.3.1
E2	A simple way of adding new ways of processing evidence for new audit policies.	5.3.1

E3	A development methodology that allows the reuse of existing adapters and evaluators in new audit tasks that aim at different requirements or obligations from policies.	5.7
P1	Support for the generation of audit result reports.	5.3.1
P2	Support for presentation of current audit state in an overview.	5.3.1
P3	Support for integration with incident management tools.	5.3.1
P4	Support for stakeholder notification.	5.3.1
P5	Audit-dependent selection of the appropriate presentation method(s).	5.6
EH1	Evidence must conform to certain legal rules, before it can be put before a jury.	Indirectly
EH2	Evidence must be tieable to the incident and may not be manipulated in any way and must be protected against any kind of tampering (intentionally and accidentally).	5.3.5
EH3	Evidence must be viewpoint-agnostic and tell the whole story.	5.3.5
EH4	There cannot be any doubts about the evidence collection process and its correctness.	Indirectly
EH5	Evidence must be understandable by a jury.	Indirectly

Table 6: Overview of Requirements for the Audit Agent System

Chapter 5

Audit Agent System Architecture

This chapter describes the fundamental aspects of the architectural design process of AAS. It starts off by showing the basic considerations that led to the choice of using software agents as an architectural design. Following that, the main actors, components and the general flow of information from the evidence-producing sources to the audit report in AAS are described. With the basic functions in place, the architecture is extended to also address possible approaches to cloud auditing in scenarios where multiple service providers are involved in the provision of a single cloud service. The chapter is completed by the presentation of user interface considerations as well as a discussion of AAS's approach to extensibility regarding new policy rules. The discussion follows the requirements that have been elicited in Section 4.6.

It is important to note, that this chapter presents the final result of many iterations of improving and refining the architecture design. Programming of the prototype implementation of the resulting architecture was supported by students under the supervision and guidance of the author.

Results of the research presented in this chapter have been published in conference papers that focus on the AAS architecture for accountability audits [124]. Security and privacy issues, especially regarding the handling of evidence and

the integration of Insynd (a cryptographic scheme for secure messaging and logging) as a suitable means for securing evidence, have been addressed and published at conferences and in a journal paper [138, 139]. The results of extension of the AAS for supporting cloud provider chains was accepted as conference papers and will be published in the near future [140, 141].

5.1 Architectural Considerations

As a foundation for the architecture of AAS a suitable design has to be chosen. The most fundamental question to answer when defining the architectural choice for AAS is whether to aim for a monolithic or modular style. Thereby, the most influential requirements for the choice of architecture are related to efficient data processing, performance, and extensibility.

5.1.1 Multi-agent Systems Introduction

A popular example of a modular distributed architectural style are MASs (see also Section 3.3).

Multi-agent System (MAS): "MAS represent a powerful model to solve distributed computation problems, including being able to adapt their operation in open and dynamic environments in which the content and workload are continuously changing" [142].

Wooldridge summarizes the work of previous research regarding the main properties of an agent and comes to the following summary [143]:

- Autonomy: an agent acts autonomously.
- Social ability: an agent communicates with other agents.
- Reactivity: an agent perceives its environments and interacts with it.

- Pro-activeness: an agent does not only react but acts to achieve a goal.

In a more specific sense that generally stems from research in the area of Artificial Intelligence (AI), agents are characterized by these additional properties:

- Mobility: an agent travels in a network.
- Veracity: an agent does not knowingly communicate false information.
- Benevolence: an agent does not have conflicting goals.
- Rationality: an agent acts in order to achieve its goals.

Agent: "An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives." [143]

The idea of gathering information in complex and dynamic networks such as the WWW is not new at all and goes back into the 90's, where Nwana noted that mobile agents are "computational software processes capable of roaming wide area networks (WANs) such as the WWW, interacting with foreign hosts, gathering information on behalf of its owner and coming back home having performed the duties set by its user" [144]. The cloud is a distributed computing paradigm that at its core propose service delivery over a network.

5.1.2 Choosing an Architecture Style

Wooldridge defines four factors that indicate the appropriateness of using an agent-based approach[145]:

1. The environment is open or at least highly dynamic, uncertain, or complex: in such environments it is desirable to have components that act autonomously.

2. Agents are a natural metaphor: a typical metaphor for agents are entities that cooperate to achieve a goal or compete against each other.

3. Distribution of data, control or expertise: in some environments, a centralized solution is very difficult or impossible to achieve.

4. Legacy systems: legacy systems may be wrapped to adapt them for use in a modern system.

AAS is a system that operates in a cloud environment. As previously described in Section 2, a major characteristic of cloud environments is that they are dynamic and constantly changing. The dynamic nature of the cloud, where fundamental infrastructure changes happen on a regular basis, requires that a system that collects data inside it is at least as dynamic and flexible and therefore able to react to changes.

The second factor aims at problems that already make use of the agent metaphor, where multiple entities that normally act autonomously, interact with each other to achieve a common goal. In AAS there are always multiple entities involved in achieving a common goal: asserting the compliance with policies or violation thereof. There is always a group of at least three: a collector, an evaluator and a store. For the interaction, agents require a way of communicating with each other.

As discussed in Section 4.2, evidential data is scattered across many different sources at different architectural layers. Distributing collectors in the form of agents allows to consolidate that view of evidence (see Section 4.6.3).

Due to the distribution and heterogeneity of evidence sources, specialized collectors are needed. While this is not necessarily a fit in terms of legacy systems that need to be wrapped to be integratable in a newer one, the underlying idea is the same. In AAS, an evidence source is wrapped by an agent that enables the interaction of the system (AAS) with a source (i.e., collection of data). The collector is specific to the evidence source and for instance implements its API

and is aware of syntax and semantic of logs produced by the source on the one side and has means to communicate with AAS on the other side.

The implementation of AAS's architecture that is described in more detail in the remainder of this chapter, is based on the Java Agent DEvelopment Framework (JADE) [146]. JADE is a Foundation for Intelligent Physical Agents (FIPA) [147] compliant implementation implementation of a MAS framework written in Java. JADE provides the Agent Communication Language (ACL) for the interaction between agents (see [148] and [149] for a full specification). AAS uses that communication language and underlying protocol to enable command and control functionality for orchestrating agents.

The main reason for choosing an agent-based approach for AAS is the core attribute of requiring adaptability in a dynamically changing environment. A cloud infrastructure is constantly changing with for example VMs being added and removed, new physical and virtual network paths (e.g., dynamically changing routes in software-defined networking) being established and resources being reconfigured to achieve elasticity. A monolithic system does not provide the flexibility and adaptability that a MAS provides.

5.1.3 High-level Architecture Overview of the Audit Agent System

Based on the architectural considerations presented before and the choice of an agent-based approach, Figure 10 depicts a high-level overview of the AAS's architecture. There are several functional areas implemented as different kinds of agents. In chronological order from input to output of AAS, the major components are:

Policy: A machine-readable policy that defines requirements, obligations and controls which are being verified by the audit is the main input to the system (besides manual input provided by the auditor). See also Chapter 6.

Audit Policy Module (APM): This set of agents is responsible for creating and

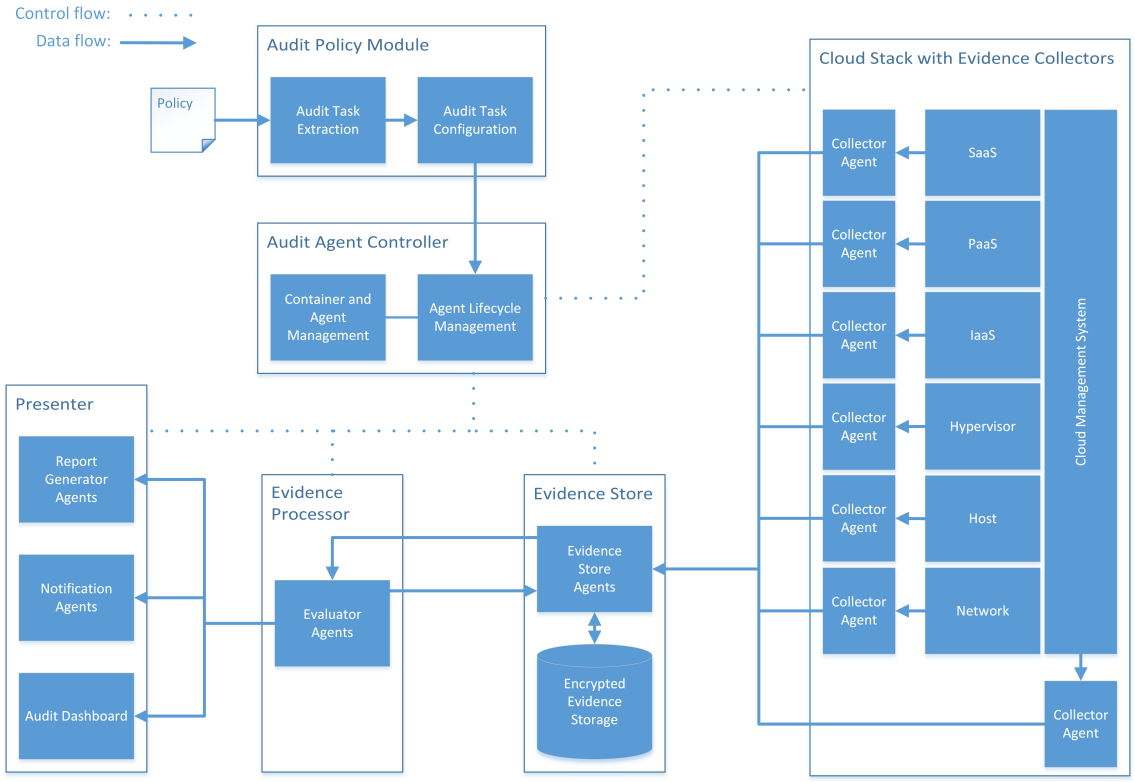

Figure 10: Audit Agent System Design – High-level Architecture

configuring audit task agents based on the provided input. See also Section 5.3.2.

Audit Agent Controller (AAC): This agent performs lifecycle control and monitoring functionality such as instantiating agents, controlling them via lifecycle messages and monitoring their health and correct execution in the environment continuously. This agent is considered a key agent for controlling AAS. See also Section 5.3.3.

Evidence Collectors: Evidence collectors are highly specialized gatherers of information from diverse sources (refer to Section 4.2) that are mobile in the environment. See also Section 5.3.1.

Evidence Storage: The evidence storage (for archival or later processing) is done by specialized agents that implement a suitable persistence mechanism that fulfill performance and security requirements. See also Section 5.3.5.

Evidence Processing: This component refers to agents that are specialized in using collected evidence in order to assess compliance with policies from the input side. See also Section 4.4.

Audit Result and Evidence Presentation: Different means of presentation (e.g., e-mail notification or presentation on a dashboard are addressed by presenter agents). See also Sections 5.3.1 and 5.6.

There are two different flows depicted in Figure 10: a data flow that indicates on the one hand the input of a machine-readable policy on the input side and on the other hand the flow of evidence from where it is generated and collected down to where it is stored, processed, evaluated and presented; the second flow relates to the AAC component that controls and monitors the other agents using ACL messages. The remainder of this chapter presents a much closer look on these components and how the aforementioned requirements are addressed.

5.2 System Actors

Based on the previous presentation of the usual actors that are involved in cloud computing in general, a more detailed look at the actors participating in cloud accountability audits are presented in the following.

5.2.1 Cloud Consumer

The cloud consumer in AAS is differentiated in two distinct variants:

- An **individual or organization** using a cloud service. This is the most common scenario where a service is consumed without adding further value to it. An individual cloud consumer is not considered to be a direct user of AAS. However, he is considered to be a recipient of (parts of) the audit results produced by AAS, since those provide assertions of data processing

happening according to what the individual expects. The same applies to organizations that consume the cloud service.

- A **cloud provider** combining services offered by third-party providers into a new service. This is a special case, where the cloud provider is itself a consumer of another cloud service by a different provider and uses it to enhance or complement its own offering.

5.2.2 Cloud Provider

The cloud provider in AAS follows closely the definition of a cloud provider as presented in Section 2.1. However, it is important to note that cloud providers can also assume the role of a cloud consumer as explained above. Cloud providers that act as auditors of another provider are considered to be users of AAS as stated in the next section. In any case, the cloud provider integrates AAS and offers it to the auditors.

5.2.3 Cloud Auditor

The main actor at which the AAS is geared towards is the *Cloud Auditor*. Based on the definition provided in Section 2.1, a cloud auditor examines information security controls put in place by cloud providers. This definition does not limit the cloud auditor to only be part of an independent third-party, but also allows cloud consumers and cloud providers to act as cloud auditors. There are differences to be considered, depending on who acts as an auditor.

Third-party as an auditor:

A third-party auditor is the most common form. In this case, the third-party is independent and capable of making an independent assessment of the audit object. A typical example for a third-party auditor is an external, independent security expert that assesses another company's security controls by trying to break in (e.g., penetration testing). This kind of auditor is

commonly seen in audit scenarios, where a company voluntarily commissions a third-party to conduct an audit as part of a certification process. In that case, the auditor needs to also be approved by the certification authority.

Data protection authority as an auditor:

The Data Protection Authority (DPA) is a special case of an independent third-party that can act as an auditor. A DPA is responsible for the oversight of the protection of data and privacy in the EU [36]. Typically, every member state in the EU has its own DPA. If they deem necessary, the DPA can conduct audits of companies to assess their compliance with data protection regulation. A popular example is the Facebook audit of 2011 by the Irish data protection commissioner [150].

Cloud provider as an auditor:

A cloud provider that is also the object of interest in an audit can at the same time act as an auditor. This case is typically known as self-auditing. During such a self-assessment, a member of the organization acts as an internal auditor. In general, this is the case when an organization (such as a cloud provider) is required to perform audits for instance by internal directives. This can be the case, when extra security assertions (by audits) are regarded as a competitive advantage or simply by a sensible management that proactively looks for potential for improvement in its privacy and security controls and internal IT processes.

Cloud customer as an auditor:

A special case of auditor in a cloud scenario is the cloud customer itself. Since the cloud customer is giving up control over the processing of its data, many cloud providers opt to provide at least some degree of transparency regarding what happens in their systems in return. Today, these audit capabilities come in the form of dedicated APIs that allow cloud customers to access data from the monitoring systems that are put in place by

the provider. It is important to note that these audit capabilities are typically not aimed at non-technical cloud customers (e.g., service consumers such as users of social networks), but at enterprise customers that have at least some degree of technical know-how (e.g., a business relying on cloud storage provided by a cloud provider).

In a cloud auditing scenario these kinds of auditors differ in the extent to which they have access to relevant evidence. A third-party auditor or a data protection authority may have a more extensive access to evidence. This is the case when a cloud provider is audited. A cloud consumer acting as an auditor however, must be confined to a limited set of data and isolated from other tenants to prevent potential data leakage and unauthorized access to other tenant's data. It becomes immediately obvious that the characteristics of the cloud impose distinct challenges regarding data protection that need to be addressed by any technical system that supports audits. This means, special care has to be taken to address multi-tenancy and different levels of control depending on the domain (e.g., in IaaS, PaaS and SaaS). Therefore, depending on the internal or external view of an organization, data protection is an issue to consider, when potential confidential information is processed during an audit.

Two variants of that case need to be differentiated:

- **Internal audit**:

  A cloud provider acting as an internal auditor is commonly associated with self-auditing and self-monitoring. Thereby, the auditor is not necessarily independent, as he is still part of the organization that is being audited. While this variant has its own set of problems with respect to auditor bias (the auditor may be required to review systems in which he has been involved during his normal work) or even lacking technical and organizational expertise (the auditor may not be familiar with common auditing techniques or lack special tracking to conduct his auditing functions properly).

However, the internal audit still remains an important function inside an organization to uncover potential risks, areas of non-compliance and areas of improvement. In a cloud scenario, the cloud provider conducts self-audits to assure compliance with the law, policies that were agreed upon with its customers and general best practices of information security and data processing.

- **External audit**:

  A cloud provider acting as an external auditor is associated with a service provision chain. Here, the cloud provider that consumes another service assumes the role of a cloud consumer and effectively becomes a cloud user. Technical expertise in information technology and conducting security analysis and audits can be assumed.

  It is therefore important that in this scenario sufficient audit support is being provided by the audited cloud provider. This is especially true for the scope and depth of evidence collection mechanisms. Information that is being collected in an external audit is used by the auditing cloud provider to extend and increase its own audits and thereby make its own audits more thorough.

5.3 Architecture Components

The architecture of AAS (see Figure 10) is based on using software agents to achieve flexibility, address requirements regarding the dynamics of cloud computing (e.g., rapid elasticity) and achieve the necessary extensibility. A typical way to approach heterogeneity in software systems is to use adapters. Software agents are used as adapters for collecting data from disparate sources in AAS such as those described in Section 4.2. Additionally, software agents can be deployed, moved and removed in distributed systems (as long as there is infrastructure support available, e.g., a runtime environment), which helps with

addressing the dynamic nature of clouds, where resources (i.e., potential evidence sources) are rapidly provisioned and released.

In the remainder of this section, particular aspects of the AAS architecture are presented. This includes the evidence format that is being used, a discussion of the different types of agents in AAS including their classification in AAS's components.

5.3.1 Types of Agents

In the following, the different types of agents that together form AAS are presented. In general, four different types of agents can be differentiated: collectors, evaluators, presenters and core agents.

Collector Agents

The most important agents in AAS are the agents responsible for collecting evidence. Their main functionality is to retrieve potentially evidential information of any kind from an evidence source and store it for evaluation in an audit, while making sure that the properties of evidence are being preserved (see Section 4.6). The second function that these agents fulfill is the transformation of the data into the evidence storage format (see Section 5.3.5 for more details). Additionally, the evidence records are enriched with meta information that is required to preserve the meaningfulness of the evidence. This includes a timestamp at the exact time that the record is made, an action and actor that can be associated with the record (if possible).

The list of collector agents that were designed and developed as part of this thesis includes:

- Apache Configuration Monitor: an agent that monitors Apache web server configuration files for changes on the filesystem level (i.e., write operations

to the file).

- Apache Log Collector: an agent that collects logs events from the Apache web server logging facility (i.e., *access.log* and *error.log* files).
- Accountability PrimeLife Policy Language Engine (A-PPL-E) Log Collector: an agent that collects logs from an access control and data handling enforcement engine by receiving and consuming log messages from that engine (see Section 6 for more details on the underlying A-PPL-E policy language).
- OpenStack Snapshot History Collector: an agent that collects snapshot event data that is generated in OpenStack by querying the Nova subsystem for that information.
- Data Transfer Monitor (DTM) Tool Collector: an agent that collects violations (i.e., illegal data transfers between geographical regions) reported by a DTM that monitors data location with respect to block storage volumes and VMs.
- File Change Monitor: an agent that monitors files (configurable) for changes by observing changes to its associated i-nodes.

Evaluator Agents

Evaluator agents take evidence records as input, build event trails, evaluate the evidence and generate a compliance statement with respect to the policy that served as the basis for the audit task. In the evaluation process, the evidence is analyzed with evaluation methods that depend on the complexity and overall goal of the audit task. However, the specifics of an evaluator agent are defined by a developer during audit task development. Evaluation methods range from simple keyword searches and calculation of metrics to more advanced and resource-intensive data mining methods using rule engines or artificial intelligence (e.g., ANNs). The generated output is the result of the audit process and

includes a compliance statement in the form of, i) *policy violation detected* or ii) the absence thereof, which is considered to be an indicator of compliance. Partial compliance may also be a potential result if the audit case allows it. However, in this work the binary choice between compliance and absence of evidence for compliance is considered. The absence of evidence can thereby include (partial) compliance or non-compliance, depending on a manual review of the collected evidence by an auditor. Furthermore, the audit report constitutes important evidence by itself and is therefore stored in the evidence store alongside the other evidence records.

The list of evaluator agents that were designed and developed as part of this thesis includes:

- Data Retention Evaluator: an evaluator that combines OpenStack snapshot history data and access control events from A-PPL-E in order to detect violations of data retention policies caused by non-deleted data in the snapshot.
- Apache Web Server Configuration Evaluator: an evaluator that audits Apache web server configuration files against an expected state (i.e., no unencrypted access available) on file-change.
- Intrusion Detector: an evaluator that audits access control events in order to detect potential intrusion attempts.
- Simple Evaluator: an evaluator that merely forwards any input as a violation (used if collected evidence always indicates a violation).
- Keyword search: an evaluator that parses evidence for the occurrence of specific keywords.

Since, depending on the amount evidence and computational requirements of the chosen analysis algorithm, resource demands can be quite high running the evaluator agents in the cloud seems like the obvious choice. However, if there

are concerns with cost, trust or with monitoring the evaluator runtime, an agent can always be moved off the cloud as long as connectivity with the evidence store is ensured.

Presenter Agents

Presenter agents in AAS are responsible for handling the output of the evaluation process (i.e., the audit report). The two major groups of presenter agents in AAS are Web-based presentation and Notifications.

The **web-based presentation** uses a dedicated gateway agent that bridges the software agent part of AAS to a web-application. The gateway agent thereby exposes a publicly available, authenticated ReSTful web interface that is used by a HTML5, CSS, Javascript (Node.js, Knockout.js) based client application.

The **notifications** approach allows for different active notification based approaches to be integrated as distinct agents that communicate evaluation events (i.e., compliance or absence of evidence for compliance). This mechanism is primarily intended to notify auditors, other relevant stakeholders or incident management systems in case a policy violation is detected. A notification includes a compliance statement and if necessary, supporting evidence that backs this claim. In any case, the notification mechanism is flexible enough to be tailored to the requirements of the audit task. As basic notification mechanisms, AAS implements:

- E-mail notification to a pre-defined group of recipients.
- Secure notifications based on Insynd [151].
- Notification to incident management tools that support automated incident notification via a public ReSTful interface.

It must be noted that notification agents are defined on a per audit-task basis. This means that different kinds of notification agents can easily coexist in the

same system and be shared by different tasks or be exclusively instantiated per task.

The list of presenter agents that were developed and implemented as part of this thesis includes:

- E-Mail Notifier: a notification agent that sends a (violation) report to predefined recipients via e-mail.
- REST Notifier: a notification agent that sends violation reports to an incident management tool or any other tool that provides a ReST interface for consuming such notifications.
- File Logger: an agent that writes (violation) reports to disk (mainly used for debugging and potentially archival on *Write Once Read Many (WORM)* media).
- Dashboard Presenter: the presentation of audit results in a web-based dashboard using a gateway agent.

Core Agents

Besides the previously described collector, evaluator and notification agents, there are several additional agents that provide core functionality such as storage, monitoring and lifecycle management in AAS. These core agents are grouped into modules. The modules that form the high-level architecture of AAS are called: *Audit Policy Module (APM)*, *Audit Agent Controller (AAC)*, *Evidence Processor and Presenter (EPP)* and *Evidence Store (ES)*. Agents implementing these modules are basically working the same as collector and evaluator agents and posses the same baseline capabilities (e.g., mobility) but fulfill different functions. In the remainder of this chapter, these modules are described in detail. All modules are implemented by strictly following the software agent development paradigm, where multiple agents interact with each other to accomplish a

task. By sticking to the agent paradigm for the core functions, advantages such as agent mobility are preserved. However, providing these functions as a set of core services could be an equally effective approach.

Additionally, due to the critical importance of the core agents' availability, they should be designed to allow for redundant operation. However, this requirement was out of scope of this thesis, since it was assumed that the underlying infrastructure (physical and virtual) was implemented highly available by allowing failover on the virtual resource layer.

5.3.2 Audit Policy Module

There are two types of input to the AAS: The main input to AAS are machine-readable policies that describe data handling obligations (e.g., access control), security controls (e.g., service configuration) and data protection mechanisms (e.g., encryption). A more detailed discussion of the integration of an exemplary policy language in AAS is presented in Chapter 6. From such policies, tasks for collecting evidence, and rules for evaluation of the evidence with the goal of producing a compliance statement are extracted. Additional input to the APM is provided by the auditor since there is always a need for at least some manual input for defining an automated audit.

Depending on the audit task, the input comprises of policies and auditor-supplied information:

1. Policies, which define obligations that have to be fulfilled by the cloud provider, such as data access restrictions and usage policies, requirements for the implementation of privacy controls, data retention requirements and general security requirements. A-PPL (see Section 6.2 for more details on the policy language) serves as the main input to the Audit Agent System and for defining audits. Integration aspects are discussed separately in Chapter 6.

2. It is possible that an input policy does not necessarily include all information required for mapping policy requirements to specific evidence sources, collectors (e.g., evidence source specific ReST client or log parser) and evaluators (e.g., API endpoints, access credentials). That information is provided by the auditor.

The APM uses both of the above-mentioned inputs to generate audit tasks. *Audit Task Extraction* is the process of identifying audit objectives and tasks from an input policy. The input policy is thereby a machine-readable document (e.g., XML or JSON).

Depending on the actual security control or data processing obligation, different parts of the input policy are extracted by the APM's language parser. The APM possesses a dedicated parser and extraction logic for each policy language, since different policy languages describe different areas of interest (e.g., description of an expected cloud deployment, data handling obligations or security controls). From the policy language representation, an object model of evidence collectors, evaluators and a container is derived depending on the obligation or rule. A Section that is dedicated to the analysis of potential input policy languages (focused on privacy, security and accountability policies) and how they integrate in the AAS as well as a description on how audit tasks are extracted from such policies in the prototype implementation is presented in Section 6.

In the second step *Audit Task Configuration*, the audit task is configured with parameters provided by the auditor. The input is collected using the AAS's web interface and mapped to the object model of audit task agents.

From the APM, the object model of agents is passed on to the AAC for instantiation and migration.

5.3.3 Audit Agent Controller

The AAC, can be regarded as the core component of the AAS. It is responsible for managing the life-cycle of evidence collection agents, controlling audit execution, storage of evidence and coordinating data flow between the components. For instance, the AAC deploys, according to what's specified in the audit task (as provided by the APM), evidence collector agents across the various architectural layers of a cloud infrastructure.

The *Container and Agent Management* part of the AAC is closely related to the core agent management functionality provided by the agent framework on which AAS is based upon (JADE). Container management is required for providing a distributed runtime environment for the agent, which enables them to be migrated or cloned from one container to another. A container is thereby part of a platform, i.e., a container is registered as such in the agent platform and thus extends the global platform-wide space of agents to run in.

In AAS, a container is used as an agent runtime environment that is positioned at or close to the evidence source from which an evidence collector gathers data. A dedicated streamlined deployment package was developed to be executed at or near the evidence source, register with the associated AAC and thus enable agent execution. The deployment package includes a pre-configured runtime environment for agents that can be executed as long as a Java virtual machine is available on the system. The agent management contains monitoring and basic agent control mechanisms, such as migration, cloning, pausing and persistence. This is also provided by the agent framework and extended by the AAC, with respect to rudimentary agent monitoring and health checks.

The *Agent Lifecycle Management* in combination with the container and agent management is responsible for managing groups of agents that together form audit tasks. A typical lifecycle of such a task is depicted in Figure 11. While the initiation part (*Preparation* and *Configuration* phases) are governed by the APM

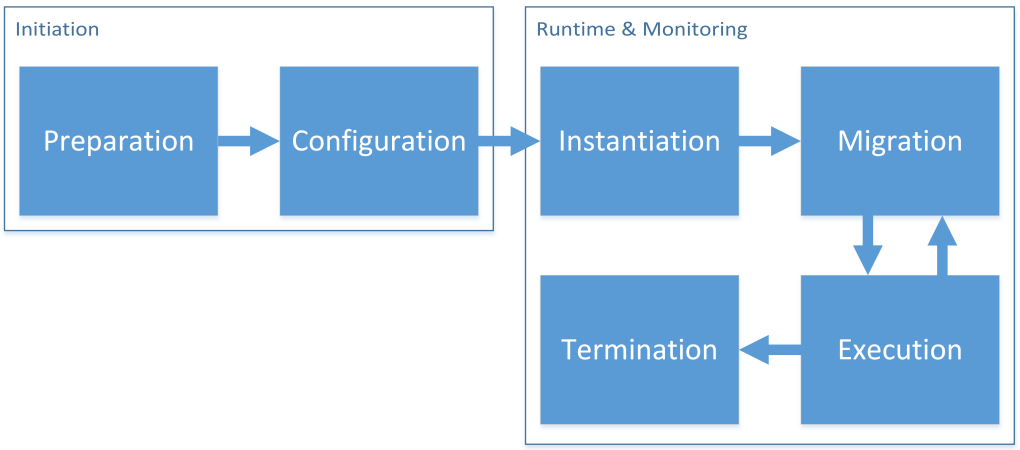

Figure 11: Audit Agent System Design – Audit Workflow

component, the execution and runtime parts are controlled by the AAC.

1. *Preparation*: The APM extracts audit task configurations from the policy, combines it with input provided by the auditor and passes it on to the AAC.
2. *Configuration*: According to the input provided by the APM, the AAC configures audit policies, its tasks and corresponding collection and evaluation agents.
3. *Instantiation*: the AAC instantiates the previously configured agents as well as the associated evidence store.
4. *Migration*: Agents are migrated from the core platform where the AAC is running to the target platforms (agent runtime environments at the evidence source).
5. *Execution*: This phase considers an agent's normal execution (e.g., a collector performing its job by gathering evidence).
6. *Termination*: The AAC disposes of the collector and evaluation agents when they are not needed anymore. It also handles archival and / or deletion of the corresponding evidence store in that case.

While most of the phase transitions are one way, an agent can transition back from execution to migration in order to travel in the environment (e.g., gathering

evidence at multiple sources). During the agents' lifetime, the AAC monitors registered platforms and agents, handles exceptions, and coordinates the creation, archival and deletion of evidence stores.

5.3.4 Evidence Processor and Presenter

The Evidence Processor and Presenter (EPP) component is responsible for evaluating policies based on the evidence gathered by the *Collector Agents*. This component is, similar to the AAC, logically composed of several agents; in this case *Evaluator Agents* are responsible for the evaluation of policies and *Presenter Agents* responsible for outputting results to the auditor. It is irrelevant, where these agents are executed as long as they are able to communicate via a network, which helps in balancing the load that is introduced by the audit process. The audit results are produced during the audit process and prepared by *Presenter Agents* according to the auditor's preferred display settings (e.g., a report document or a web-based dashboard).

After the collector agents have gathered evidence data and stored it in the evidence store, the evaluation agent(s) of an audit task retrieve that data and analyze it according to the rules that have been extracted from the policies in the preparation phase by the APM. Evaluation can happen continuously on the currently available evidence or in predefined intervals. The results (along with the evidence that caused the violation) that are produced by the evaluation agents are written back to the evidence store (i.e., as evidence that an audit happened). The result can be either positive (proven compliance or the absence of a violation) or negative (a violation is detected). Additionally, the result is passed on to presenter agents that inform the auditor and any other stakeholder (depending on the notification configuration) about the audit result.

Currently, the presenter agents can either display the audit result in a web-based dashboard, notify stakeholders via e-mail or pass on the violation in a machine-readable format to other tools or services via a ReST API.

5.3.5 Evidence Store

The Evidence Store (ES) is the central repository for securely storing evidence. An important characteristic of evidence stored in the ES is that it is always associated with a policy and was collected for the purpose of evaluating compliance with that policy (see also Section 4.6). For each cloud tenant, there can be at least one separate ES to ensure tenant isolation on a low level. Each evidence store and its associated storage space is implemented as an agent that provides an interface to the underlying data storage mechanism. In order to fulfill integrity and confidentiality-related requirements, AAS utilizes an encrypted evidence storage.

Evidence Record Format

As part of the Cloud Accountability Project (A4Cloud) project, a storage format for evidence in cloud ecosystems was developed collaboratively as part of the Stream C8 work package. While the author of this thesis was heavily involved in this work, the format specification is not to be considered a contribution (due to the collaborative nature of its development) of this thesis but merely a minor part that is being used in the evidence storage subsystem of AAS. There are several approaches to harmonizing the storage format for digital evidence that can be reused in the ES such as [48, 152, 153]. The goal of the format was to define a storage format for all types of evidence that is lightweight, simple and easy to extend. A basic example is depicted in Figure 12. For a complete example evidence record, please refer to Appendix B.2, where an example is depicted as it was collected by AAS from OpenStack in the demo scenario presented in Section 7.2.3.

Each record has the following characteristics:

- Record identifier: this uniquely identifies an evidence record.
- Action: this identifies an action (e.g., snapshot taken, or read access event

```
<record id="a_uuid_for_this_record">
    <action>an_action</action>
    <actor>an_actor</actor>
    <policyID>a_unique_policy_identifier</policyID>
    <supportingElements elementID="running_identifier">
        <signature>signature_of_element</signature>
        <element>the_actual_element</element>
    </supportingElements>
    <evidenceMetaData>
        <collectingInstance>the_collector</collectingInstance>
        <evidenceDetectionTime>time_of_collection</evidenceDetectionTime>
    </evidenceMetaData>
</record>
```

Figure 12: Audit Agent System Design – A Common Evidence Record Format

on data).

- Actor: if available, this contains a user identifier for the entity that performed the action.
- Policy Identifier: the policy that the evidence refers to.
- Supporting Elements: this contains references to or the actual evidence (e.g., parts of the log file that was collected from).
- Evidence Metadata: this provides additional metadata, such as the identifier of the collector agent or the time and date of the collection.

Each record possesses a unique identifier with which it can be addressed globally (AAS uses Universally Unique Identifiers (GUIDs)). Furthermore, if possible, meta information is captured from the evidence source (such as the collecting instance, agent or tool with a timestamp). An actor (e.g., a cloud user, a cloud subsystem or a process) and an action may be available depending on the evidence source (e.g., most likely available in an access control subsystem). Since an evidence record is always associated with a policy that is audited (without a policy it is not collected) this is put in the record as well. Supporting elements and evidence metadata provide the actual collected data (e.g., the original log message and a timestamp when the record was collected).

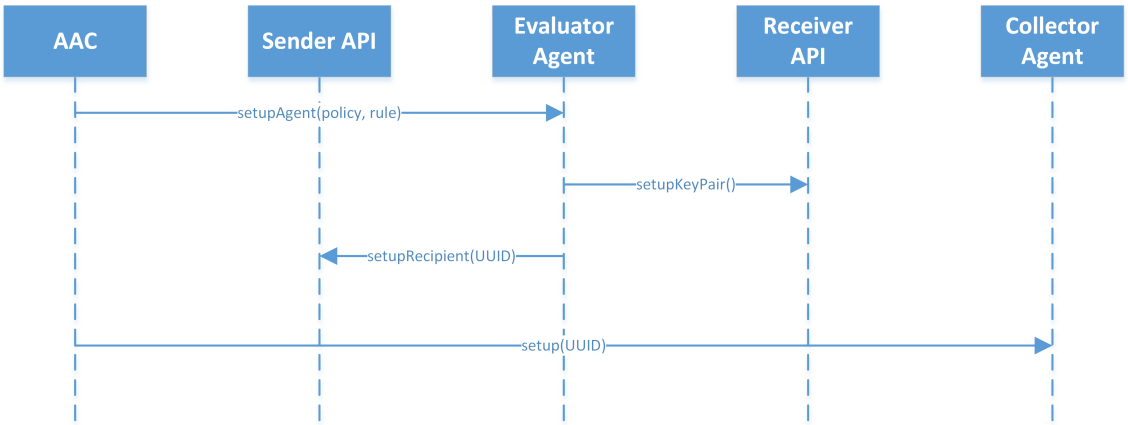

Figure 13: Evidence Collector and Evaluator Setup Sequence for Using an Encrypted Evidence Store

Interfacing with the Evidence Store

The evidence store as it was described previously is an abstract representation. The implementation of AAS uses a cryptographic scheme called Insynd [151, 154, 155] to enable protection of evidence (preserving integrity and confidentiality) in any phase of an audit (collection, storage, evaluation). Insynd provides a way of exchanging encrypted messages asynchronously between agents, while also providing persistence capabilities. In AAS, the automated setup of agents with key material (public, private keys) and registration with Insynd is particularly important. Figure 13 depicts the initialization sequence of collector and evaluator agents with a focus on key distribution.

The following steps have to be performed to setup the evidence collection and storage process for that particular rule:

1. In the first step, an evaluator agent is created and configured according to the input policy and rule respectively.

2. During the setup phase, the evaluator agent sets up a key pair at the Receiver API. The Receiver API is a ReSTful service that holds private key material and is therefore located at a trusted server.

3. After the key material has been generated, the evaluator agent registers

itself as a recipient at the Sender API. For this, it uses a GUID under which the receiving evaluator agent is uniquely identifiable.

4. In the last step, the AAC sets up the collector agents and connects them with the corresponding evaluator agents by using the unique recipient identifier.

Now, it is possible for the Collector Agents to send evidence records to their corresponding Evaluator Agents asynchronously. The messages will be encrypted at the Sender API service before storage, using the provided recipient's public key. The evaluator agent then pulls the evidence records from the ES using the Receiver API. The records are finally decrypted using the receiver's private key.

5.4 Interoperability in Inter-Cloud Scenarios

While a lot of today's cloud use cases only involve one service provider for service provision, there are also many cases where multiple providers are involved. A prominent example is Dropbox that heavily uses Amazon's S3 and EC2 services to provide its own SaaS offering [156]. This section discusses the integration of multi-provider evidence collection in the context of AAS.

For the remainder of this section, the concept of a provider chain is defined as follows:

1. At least two cloud providers (characterized by being either IaaS, PaaS or SaaS providers) are involved in the provision of a service to a cloud consumer (who can be an individual or business).

2. One of the cloud providers acts as a Primary Service Provider (PSP) to the cloud consumer.

3. Subsequent cloud providers do not have a direct relationship with the cloud consumer.

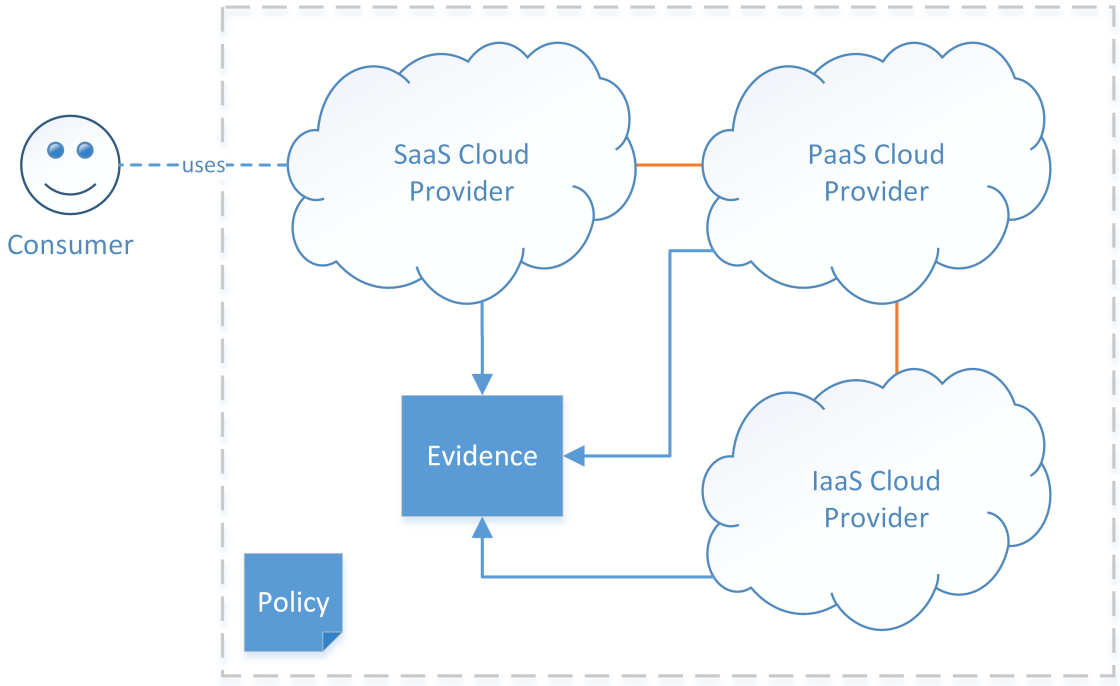

Figure 14: Provider Chains in Cloud Service Provision

4. The PSP *must* be and the subsequent providers *can* be cloud consumers themselves, if they use services provided by other cloud providers.

5. The data handling policies agreed between the cloud consumer and the PSP must not be relaxed if data is processed by a subsequent provider.

Figure 14 depicts a simplified general scenario where three cloud service providers are involved in provisioning of a seemingly single service to a cloud consumer. The SaaS provider acts as the primary service provider, whereas it uses the PaaS provider's platform for hosting its service. The PaaS provider in turn does not have its own data center but uses resources provided by an IaaS provider. The data handling policy that was agreed upon between the consumer and the PSP, applies to the whole chain (depicted by the dashed rectangle in Figure 14). All cloud providers produce evidence of their cloud operations. To demonstrate the compliance with policies, it is not sufficient to audit only the PSP but subsequent providers have to be taken into account as well, if sensitive data owned by the consumer is passed on beyond the PSP.

5.4.1 Evidence of Compliance in Cloud Provider Chains

At the core of any audit is evidence of compliance or non-compliance that is being analyzed. The types of evidence are closely linked to the type of audit (e.g., security audit, financial audit etc.) and are – from a technological perspective – especially diverse in the cloud due to the heterogeneity of its subsystems, architectures, layers and services. Evidence collection at a single cloud provider is already a complex task due to the diverse types of evidence sources and sheer amount of data that is being produced continuously (e.g., logs, traces and documents). In a provider chain, these problems are intensified by the introduction of administrative domains and the lack of transparency regarding the number of involved providers and their relationships.

Another problem that is introduced with the concept of provider chains are changing regulatory domains. In a simple single-provider scenario, there are typically only two regulatory domains to be considered: the one that applies to the cloud consumer and the one that applies to the cloud provider. Of course, this can be complicated if the provider operates in multiple domains simultaneously. With the addition of more cloud providers, the complexity of achieving regulatory compliance increases tremendously.

A simple example for such a case is the recent decision of the European Court in 2015 to declare the Safe Harbor agreement invalid, which leads to data transfers to the outside of the European Union based on Safe Harbour to be illegal [157, 158]. In a provider chain, where a European Cloud provider transfers data about European individuals to another provider in the US, regulatory compliance can be lost overnight. It can be seen that regulatory domains can have a tremendous impact on how a compliance audit may have to be performed, and on the type of evidence that may need to be collected at the different providers.

As previously suggested, the third major challenge for evidence collection in cloud provider chains is their inherent technological heterogeneity. APIs, pro-

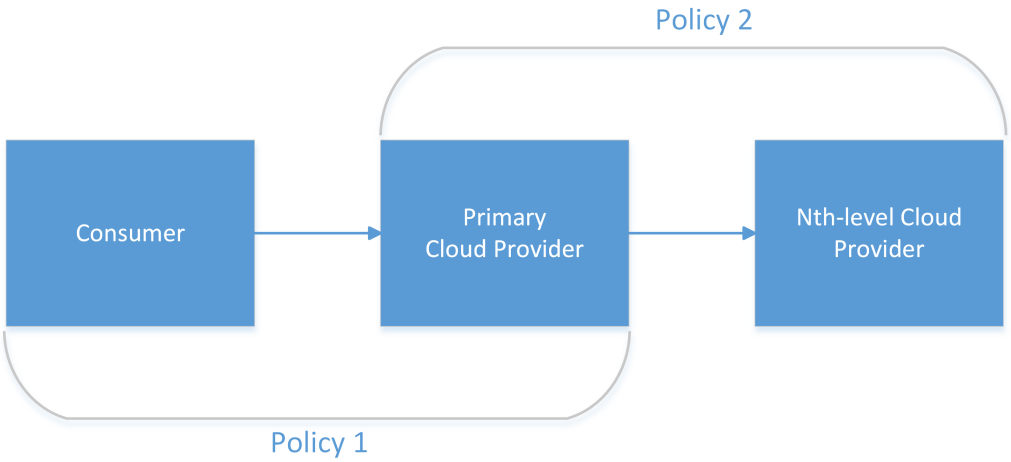

Figure 15: Scopes of Policy Applicability

tocols and data formats differ by provider and typically cannot be integrated easily (e.g., providers offering proprietary APIs). There are some approaches to homogenize some of the technologies, such as for example those presented in Section 3 that aim to provide well-defined APIs for enabling cloud providers to export transparency-enhancing information to auditors and cloud consumers.

In this approach, the technological heterogeneity at the architectural level of the system is addressed by ensuring flexibility, extensibility and enabling the easy development of adaptors for different evidence sources.

5.4.2 Scopes of Policy Applicability

In the following, three different scopes of policy applicability as depicted in Figure 15 are described. The same scenario as described in Section 5.4 is assumed. Also, the PSP is assumed to be a cloud provider. This does not necessarily have to be the case.

Scope A: Cloud Consumer / Primary Cloud Provider

In a typical cloud use case, a consumer uses the services provided by a single cloud service provider to accomplish a given task. The details of the service usage are governed by terms of service agreements, privacy polices etc. In this

most common scenario, the cloud consumer and the cloud provider agree on these terms before any service is provisioned. Typically, this happens during a registration or contract agreement phase. With respect to data flow between the consumer and the provider, this means that data processing is performed by the cloud provider in compliance with the agreed-upon policies (see *Policy 1* in Figure 15). Data that is disclosed by the cloud consumer to the cloud provider as part of regular service use, is processed by the cloud provider according to the limits defined in the policy. One size fits all agreements as they are common today may thereby be replaced by more fine-grained policies based on policy languages such as A-PPL in the future.

Scope B: Primary Cloud Provider / Nth-level Cloud Provider

Similar to the approach described in Scope A, there may be similar agreements between cloud providers. For instance, the primary cloud provider may require resources from the sub-provider, e.g., to extend its own service offering, to address peak loads in service usage or to outsource internal processes such as backups. In this case, the primary cloud provider (as depicted in Figure 15) becomes a cloud consumer itself. The integration of cloud services provided by a sub-provider in services of the primary cloud provider is governed by a contractual agreement between the two (see *Policy 2* in Figure 15).

Scope C: Cloud Consumer / Nth-level Cloud Provider

In case of a cloud scenario, where multiple service providers are involved in the provisioning of a single service, the cloud consumer may not necessarily be aware of this. The main difference to scope C is that since the cloud consumer has only contact with its immediate provider (the PSP in Figure 15), he might not necessarily be aware, that the primary cloud provider is using an additional external service. A typical example for such a scenario is a SaaS provider hosting its services on resources provided by an IaaS provider, or a SaaS provider that integrates another SaaS provider's service for data processing. Addition-

ally, a silent change of the supplementary service provider can happen, when the primary provider switches to another service (e.g., migrates to another infrastructure provider for cost or efficiency reasons). In this case, the restrictions that governs the policy agreement between the cloud consumer and the primary cloud provider (i.e., Policy 1) must also apply to the sub-provider, if data owned by the cloud consumer is transferred between them. This is the case, when either: similar policy rules exist in Policy 1 and 2, and the rules defined in Policy 2 are at least as strict as equivalent rules defined in Policy 1 (in this case, a matching of whether or not rules from policy 1 and 2 are compatible needs to be performed), or the downstream provider accepts rules from policy 1 directly.

5.4.3 Auditing Cloud Provider Chains

To assert compliance, the whole chain of providers including data flows that are governed by the policies have to be considered. This means that an audit with respect to a single policy rule may need to be split into several smaller evidence collection and evaluation tasks that are distributed among the providers.

The importance of widening the scope of audits in such scenarios is apparent, especially if at the same time the depth of analysis is widened beyond checking whether or not security and privacy controls are put in place to also checking for instance data location. In the following, two approaches are presented that can be taken towards auditing of provider chains: i) individually auditing each provider, and ii) delegating the audit of subsequent providers to the primary cloud provider.

Individual Provider Audits

Figure 16 describes the process of auditing individual providers in a service provision chain.

To audit the service as a whole, it is necessary to audit each provider separately

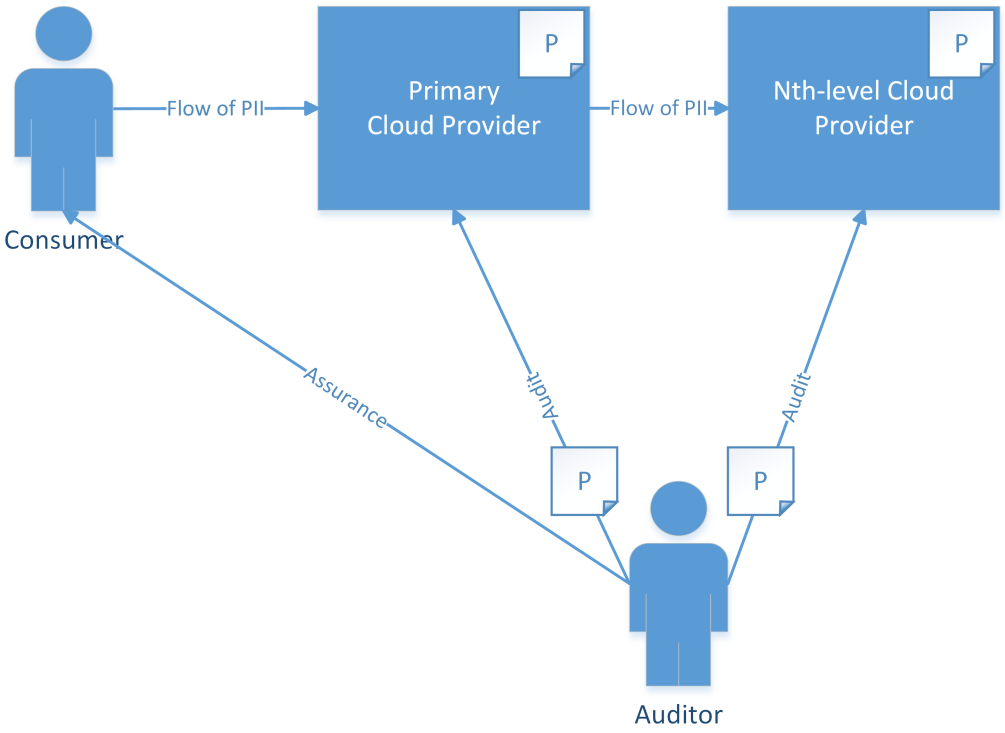

Figure 16: Individual Audit in Cloud Provider Chains

and then aggregate the results to form a complete picture of the service from an audit perspective. This means, that regarding data handling policies (e.g., location restrictions, access control etc.), each provider that holds data is audited. Obviously, the consumer-facing provider has to transparently disclose all his sub-providers and notify auditors about every sub-provider his data was stored at and where his data is currently stored.

The auditing process gains more complexity with an increasing number of Nth-level providers. Requests must be sent to each provider separately and each provider will deliver audit reports to the auditor, who then integrates the partial results to gain a compliance view of the whole chain. The individual audit scenario is an example of how chain audits (the scope of the audit is the whole provider chain instead of a single provider) can be performed by more influential stakeholders, such as data protection authorities.

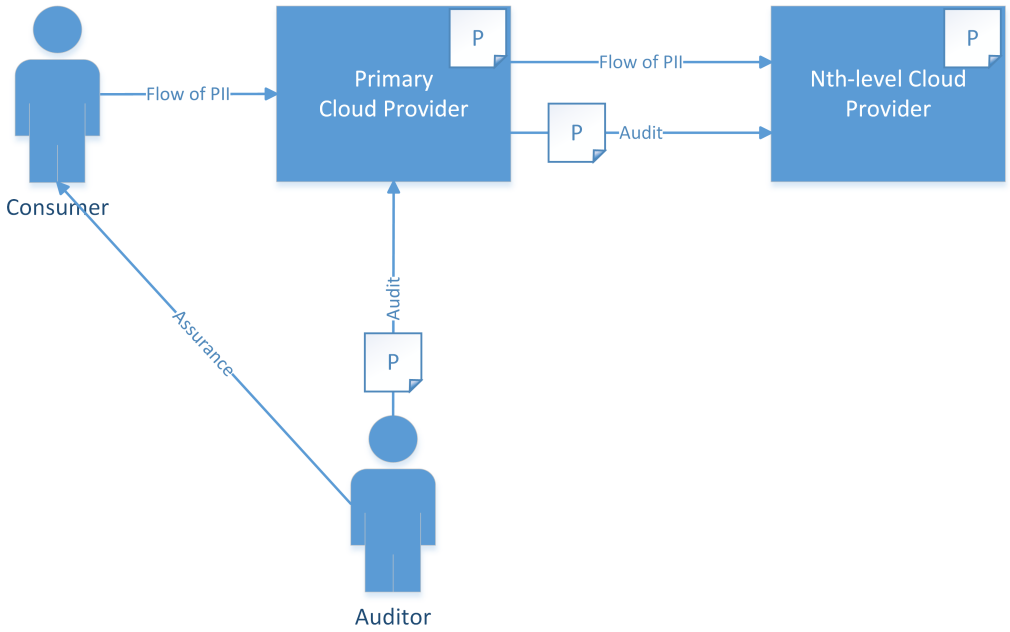

Figure 17: Delegated Audit in Cloud Provider Chains

Delegated Provider Audits

An alternative to individual audits are delegated audits (see Figure 17), where the auditor only interfaces with the PSP that in turn takes over the auditing of its sub-provider(s). This allows less influential stakeholders such as cloud consumers to act as an auditor towards the PSP (assuming required skills), while not having the same rights towards the Nth-level provider(s) and therefore missing the complete picture. Every audit request is sent to the PSP, who will then extract evidence requests for the subsequent providers.

5.4.4 Approaches for Collecting Evidence in Cloud Provider Chain Audits

There are several approaches available when it comes to collecting evidence for audit purposes in a service provider chain. These approaches differ in the following aspects:

1. The level of control an auditor has over the extent of the data that is being

published (i.e., whether the auditor is limited to information that a provider is already providing or if he has more fine-grained control and access to a provider's infrastructure).

2. Technical limitations imposed by the technological environment (i.e., the extent to which cloud providers have to implement additional evidence collection mechanisms).

3. The willingness or acceptance to provide such mechanisms by the publishing service provider (i.e. the potential disclosure of confidential provider information and required level of access to the provider's systems).

In the remainder of this section, three approaches are described and rated by the above-mentioned factors.

The first approach focuses on reusing already existing evidence sources by collecting via remote APIs of systems in a cloud environment. The second approach uses provider-provisioned evidence collectors and the third approach leverages the mobility of software agents for evidence collectors.

Remote API Evidence Collector

The first approach for collecting evidence that is relevant to automated auditing leverages existing APIs in cloud ecosystems. Several cloud providers such as Amazon or Rackspace already provide transparency over their cloud operations by providing their customers with access to proprietary monitoring and logging facilities (see [159, 160]). The extent to which data is shared is typically limited to information that is already produced by the cloud provider's system (e.g., events in the cloud management system) and restricted to information that immediately affects the cloud customer (e.g., events that are directly linked to a tenant). Data such as logs that are generated by the underlying systems are very important sources of evidence, since they expose a lot of information about the operation of cloud services, but are often out of scope in these systems.

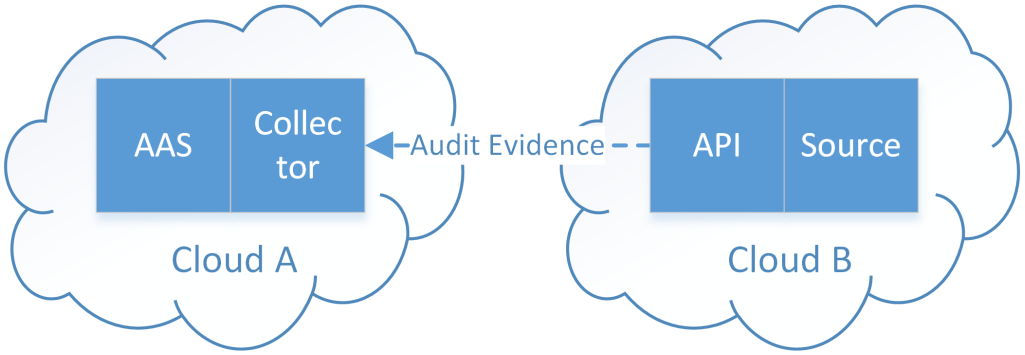

Figure 18: Evidence Collection in Cloud Provider Chains – Remote API Collector

The type of information is highly dependant on the actual system, the granularity of the produced logs and the scope of the provided APIs. For instance, on the infrastructure level, there are log events produced and shared that provide insight on virtual resource lifecycle (e.g., start/stop events of VMs).

Figure 18 depicts such a scenario. The AAS at Cloud A operates a collector that uses the API of the remote data source at Cloud B. It is configured with the access credentials of Cloud A, thus enabling the collector to request evidence from Cloud B. Since different services may provide different APIs (e.g., OpenStack vs. OpenNebula API), the collector is service-specific. For instance, a collector implements the data formats and protocols as defined in the OpenStack Nova API to collect evidence about the images that are owned or otherwise associated with Cloud A as a customer of Cloud B.

There are two general information exchange patterns that are typically used by such systems: a pull pattern and a push pattern. The CTP protocol is an approach by the CSA, which aims at harmonizing APIs in order to make the cloud more transparent. It provides both approaches in its specification and is used in the following discussion:

Figure 19 shows the individual evidence request sequence in a **pull** approach. When a request for evidence (or transparency elements using CTP's nomenclature) arrives at a provider the data is sent as a response to the requester. The evidence can either be collected on-demand or the request is fulfilled from the

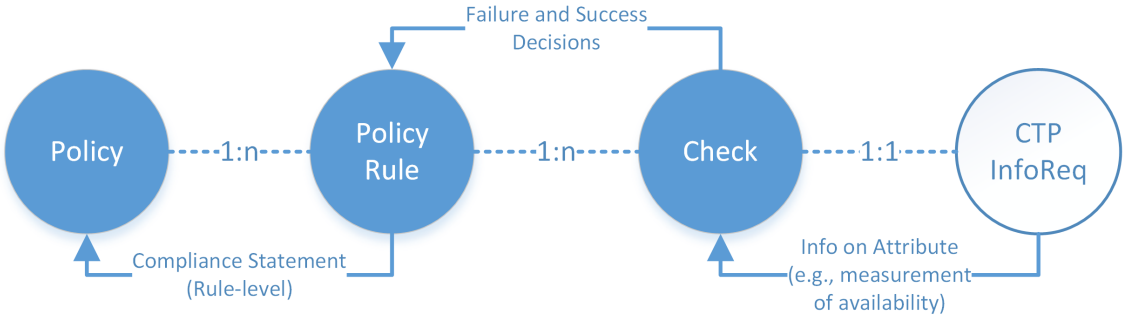

Figure 19: CTP Integration – Individual Data Requests

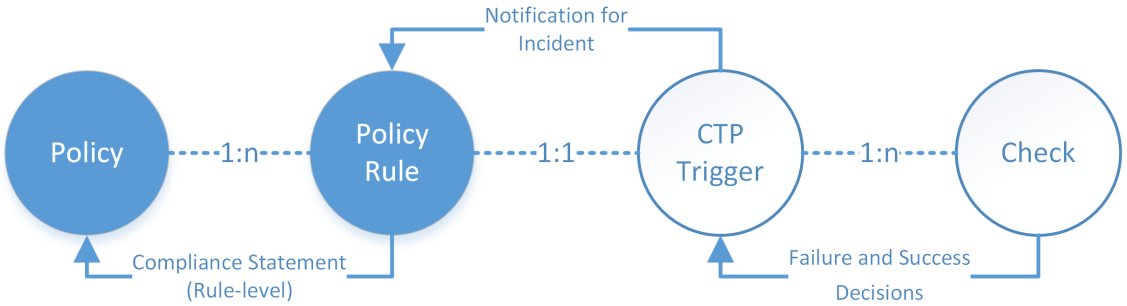

Figure 20: CTP Integration – Triggered Data Exchange

evidence store (e.g., from the target's AAS instance). It is then used by the requester check to ascertain if a policy is violated or not (i.e., the audit is performed). Polling for new data in relatively short intervals can introduce load problems at the auditing system. This issue is a common problem with pull / polling mechanisms. However, carefully choosing longer intervals such as hourly, daily, weekly and monthly reports help alleviating such problems.

Information that quickly needs to be processed, like security breaches or integrity violations, can be time-critical with respect to incident response and therefore should not rely on transport via pull mechanisms. Push mechanisms are typically associated with an event-driven approach, where an event is fired when a condition is met. Figure 20 (solid blue circle indicates the domain of the auditor whereas solid white indicates the provider) shows the **push** notification process. An auditor can specify the severity of an occurrence called trigger, to ensure that only significant information is pushed. A configured trigger will only fire when the condition set by the auditor is fulfilled.

Whether a push or pull approach is being used, the auditee retains full control over the published interface and therefore controls the exposure of evidence to an auditor. Considering a scenario, where multiple AAS instance are involved (e.g.,during an audit of a service provision chain with multiple providers), the exchange can of course also happen using an interface such as CTP between those instances.

- **Level of Auditor Control:** The amount of evidence that can be collected is severely limited by the actual APIs that are provided by a cloud provider. It is either: i) the evidence that an auditor is looking for is immediately available because the provider already monitors all relevant data sources and makes that data accessible via the API or ii) the data is not available. Since a lot of the cloud provider's systems expose remote APIs anyway, they have to be considered for evidence acquisition. However, the completeness of the exposed APIs and therefore the completeness of the collectable evidence is questionable due to the aforementioned reasons.

  If an auditee for some reason does not implement or provide access to AAS, an auditor may still collect evidence to a limited degree using this approach.

- **Technical limitations:** If lower-level access to the providers infrastructure is required to collect evidence (e.g., log events generated on the network layer or block storage-level access to data), an auditor might not be able to gain access to that information.

- **Acceptance:** This approach poses some challenges with respect to security, privacy and trust required by the auditee. Since the auditee is already exposing the APIs publicly, it can be expected that they will be used for auditing and monitoring purposes. The implementation of security and privacy-preserving mechanisms on the API-level is assumed. However, the extent to which such mechanisms are supported highly depends on the actual implementation of the APIs on the provider side.

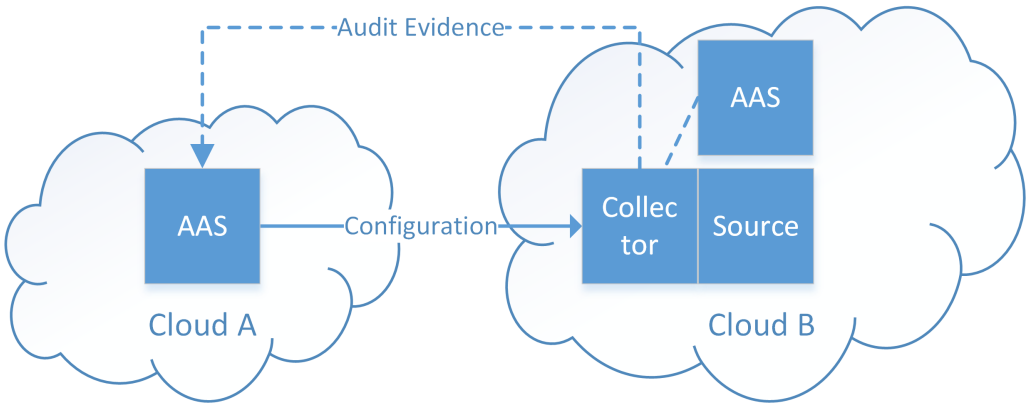

Figure 21: Evidence Collection in Cloud Provider Chains – Provider-provisioned Evidence Collector

While this way of providing evidence to auditors is likely to be accepted by cloud providers, it may be too limited with respect to the extent to which evidence can be collected at lower architectural levels.

Provider Provisioned Evidence Collector

In this approach, AAS still is the main component for evidence collection. All cloud providers that are part of the service provision chain are running a local AAS. However, the instantiation and configuration of the collector is delegated to the auditee. The auditee assumes full control over the collector and merely grants the auditor access to interact with the collector for evidence collection. The collector must be obviously be capable of providing the required interfaces to the auditor.

The auditee (see Cloud B in Figure 21) provisions evidence collectors and provides access to them to the auditor. The auditor (who is using AAS at Cloud A) configures evidence collection for the audit to connect to the collectors at Cloud B.

- **Level of Auditor Control:** The configuration of the evidence collector can be adjusted by the auditor to a degree that is controlled by the auditee (e.g., applying filters to logs) in that he limits means to configure a collector

and its ability to freely migrate in the auditee's infrastructure. At any time, the auditee can disconnect, change or otherwise control the collector. An auditor may be put off by the limitations posed by this approach since he is effectively giving up control over the central part of evidence collection and is relying solely on the cooperation of the auditee. For instance, simple tasks such as reconfiguring or restarting a collector may require extensive interaction between the two audit systems and potentially intervention by a human.

- **Technical limitations:** This approach is only limited by the availability of collectors for evidence sources and overhead associated with the provision, runtime management and termination of agents that is introduced at the provider.

- **Acceptance:** The auditee retains full control over the collector and the potential evidence that can be collected by it. The auditor can take some influence on the filtering of data that is collected from the evidence source and on general parameter such as whether evidence is pushed by or pulled from the collector. Most of the baseline configuration though, is performed by the auditee (such as access restrictions and deployment of the collector). The auditor's ability to influence the collector is severely limited by restricting the auditor's collector interactions to a well-defined set of configuration parameters and the evidence exchange protocol. This level of control that the auditee has over the evidence collection process may have positive influence on provider acceptance. However, due to the management overhead introduced, acceptance of this approach at the cloud provider is unlikely unless it is provided as a service to customers and auditors.

Mobile Evidence Collector

This approach is specific to a central characteristic of software agent systems, which is the ability to migrate over a network between runtime environments. In

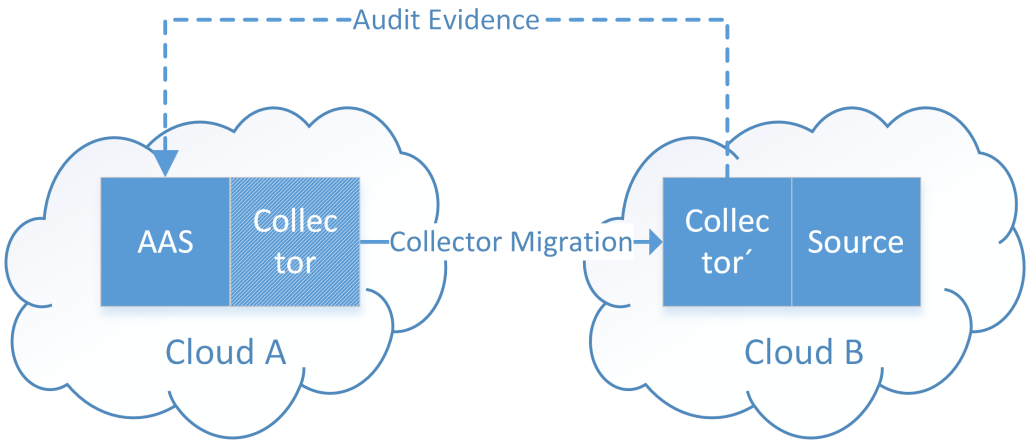

Figure 22: Evidence Collection in Cloud Provider Chains – Mobile Evidence Collector

this approach, the migration of evidence collectors between separate instances of AAS running at both Cloud A and B is demonstrated. The reason to use mobile collector in this approach is leverage the potential of agents travelling to various different data sources for evidence collection without the need of making these sources available in any other way than being accessible by agents (e.g., no additional public APIs). This matches well with some potential evidence sources not having a feasible way of remotely providing information without additional implementation effort. For instance, certain configuration file parameters (e.g., specific set of SSL cipher suites with which a web server might be setup at a particular point in time) may be considered evidence. However, configuration files are typically not publicly available. A mobile agent with sufficient authorization may easily read configuration files and collect certain parameters as evidence.

As depicted in Figure 22 the auditor prepares the required collector fully (i.e., agent instantiation and configuration) and then migrates (shaded box named *Collector*) the collector to the auditee (*Collector'*). There, the collector gathers evidence that is sent back to the auditor for evaluation. Generally however, agents rarely cross from one particular administrative domain to another. In this case, the collector crosses from Cloud A's administrative domain to Cloud B. This has significant impact on the acceptance of that approach.

- **Level of Auditor Control:** The auditor retains full control over the type of collector and its configuration. The auditee may not in any way change or otherwise influence the collector since this can be deemed a potentially malicious manipulation of the agent.

- **Technical limitations:** Since the auditor knows best about the actual configuration required for a collector, it is logical to take this approach and simply hand-over a fully prepared collector to the auditee to which the auditee only has to grant access to. However, this only works if both run the same audit system, or the auditee at the very least provides a runtime environment for the collector. This approach offers the most complete and most flexible way of collecting evidence at an auditee due to the comprehensive evidence collection capabilities.

- **Acceptance:** The main problem with this approach is required trust by the auditee. Since the collector that is being handed over to him by the auditor is in fact software that the auditee is supposed to run on its infrastructure, several security, privacy and trust-related issues associated with such cross-domain agent mobility need to be addressed. Several security controls need to be implemented to make cloud providers consider the implementation of AAS including the approach of using mobile collectors.

  The main security concerns of this approach stem from the fact that the auditee is expected to execute software on his infrastructure over which he does not have any control. He cannot tell for certain whether or not the agent is accessing only those evidence sources which he expects it to.

  Without any additional security measures, it cannot be expected that any cloud provider is willing to accept this approach. However, with the addition of security measures such as ensuring authenticity of the collector, which means the collector is in fact from the auditor it claims to be from and does nothing except what is described, this approach becomes more feasible. For example using collector source code reviews and code sign-

ing can help with that problem. Based on such extensive reviews, agents can be certified by a trusted third-party, which could improve acceptance of this approach. The discussion of such measures depends on the technology used by the implementation. Furthermore, these measure could be implemented by a trusted third party that conducts the audit, which would mitigate the risks associated with opening the cloud platform to customers on a low level. Without any additional measures, it can be assumed that this approach is only feasible if the auditor is completely trusted by the auditee. In that case, this approach is very powerful and flexible.

Round-up

All three approaches for evidence collection in provider chains have their very distinct advantages and disadvantages. Using remote API evidence collectors is simple, quickly implemented, securely and readily available, but severely limited regarding the scope of access to evidence sources. Using provider-provisioned evidence collectors is more powerful with respect to access to evidence sources but requires more effort in the configuration phase and leaves full control to the auditee. Using mobile evidence collectors is the most flexible approach that allows broad access to evidence sources at the auditee's infrastructure and leaves full control over the evidence collection to the auditor.

With respect to sharing responsibility, the remote API and provider-provisioned collector approaches are the best fits. They enable the interaction (i.e., transfer of evidence) between two distinct stakeholders in scenarios where the customer and provider run their own instances of AAS. In an IaaS model, this is more likely to be the case, since the cloud customer is responsible for the correct implementation of much more security controls than in any of the other service models. The implementation of an AAS instance on the customer side is thereby a natural fit to enable continuous compliance assessment. However, the interaction between multiple distinct AAS instances is considered future work

that will mostly revolve around problems stemming from defining appropriate contractual agreements and a suitable API.

In AAS, the use of remote APIs is integrated due to its simplicity and mobile collectors due to their flexibility and powerfulness as the main approaches for evidence collection.

5.5 Decentralization and Pre-processing of Audit Evidence Evaluation

Externally collecting evidence and merging it at a central evidence store that is only reachable via a network can easily become a bottle-neck in audit scenarios where either a lot of evidence records are produced externally or where the record size is big. This obviously has significant impact on the scalability of the whole system.

The problem can be addressed by making the evidence store (which is just a specialized form of an agent with a secure storage mechanism) distributable and also by de-centralizing parts of the evidence evaluation process. There are two concepts implemented in AAS:

1. Pre-processing: Pre-processing allows the evidence collector agent to apply evidence pre-processing (see Section 4.4). The goal is to reduce the amount of collected evidence to a manageable degree (without negatively impacting the completeness of the audit trail) and to reasonably reduce the amount of network operations by grouping evidence records and storing them in bulk. For example, by filtering the raw data at the evidence source for certain operations, subjects, tenants, or time frames data that is not immediately required for the audit the amount of evidence is reduced.

2. Intermediate Result Production: A second pre-processing strategy is to move (parts of) the evaluation process near the collector. This means that

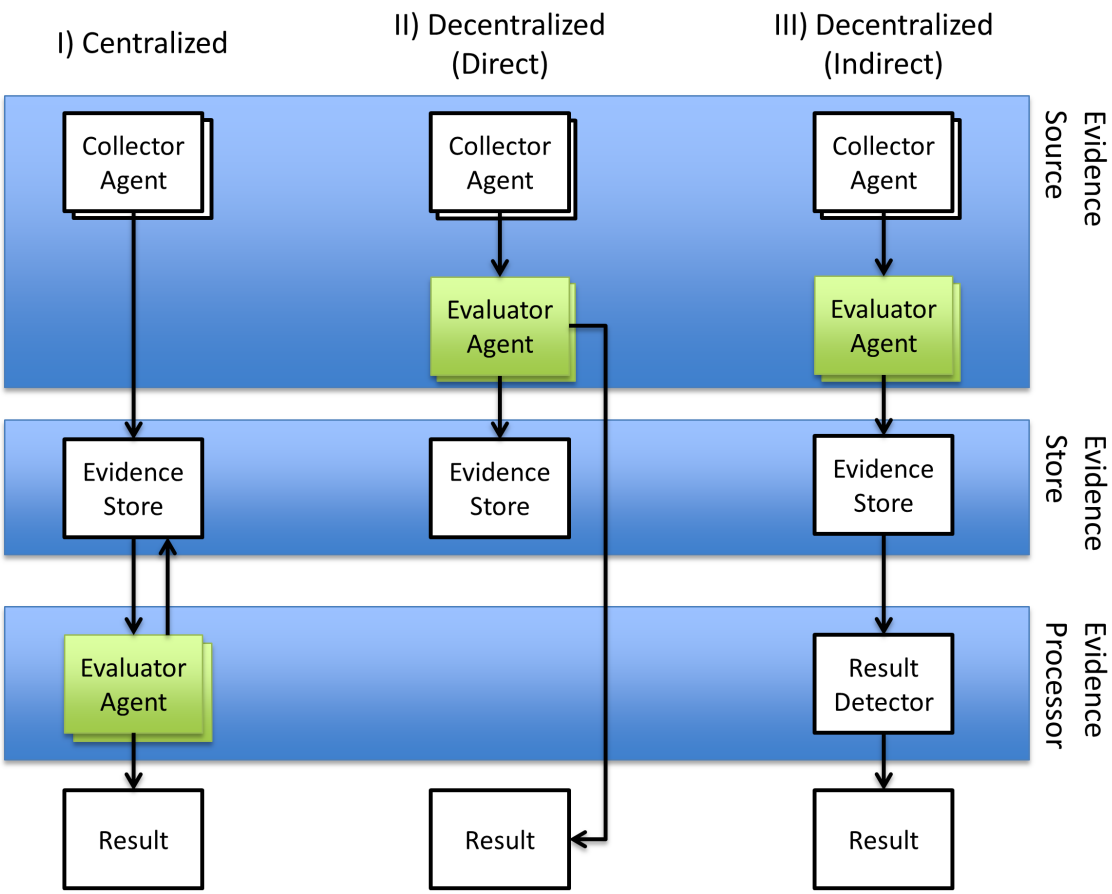

Figure 23: Pre-processing and Decentralization Approaches

the collected evidence is already reduced to the significant portions that indicate partial compliance or violation of policies.

The two approaches bring several implications with them with respect to privacy and security. Pre-processing can be considered a manipulation of evidence. Therefore, the unaltered source upon which the pre-processing happened should be protected to later be able to trace pre-processed evidence back to its unaltered form.

Immediate result production effectively moves the evaluation step of the audit into the domain of the auditee, where it is easy for him to manipulate the result. However, the same applies to the collection of evidence as well, where an auditor can intentionally manipulate the evidence source or the collector. This case is not considered in the current iteration of AAS but it is assumed that cloud

providers (auditees) are acting in good faith. This assumption can be justified by the potential increase in transparency and the associated strengthening of trust in the cloud provider that can mean a competitive advantage. On the other hand, intentional manipulation of evidence or intermediate results can have disastrous impact on a provider's credibility, reputation and trustworthiness upon detection.

Figure 23 presents a centralized and two de-centralized approaches:

1. Centralized: this approach is centralized in the way that all evaluator agents are executed on the same server. Collector store evidence in the ES and the evaluators pull it from there. Additionally, evaluators write back the results as evidence in the ES.

2. Decentralized (direct): this approach considers moving the evaluator to the runtime environment where the collector is executed. The collector directly feeds evidence to the evaluator, which then stores the evidence and the results in the ES.

3. Decentralized (indirect): this approach is a hybrid of the previous two, where the general flow of evidence and results is the same as in the direct approach but results are picked up from the ES by a simple agent that does nothing more than regularly trying to detect if new results are available.

5.6 Presentation of Audit Results

Considering the presentation of evidence and audit results is very important for several reasons: Scalability of audit and evidence presentation is important due to the massive scale of cloud infrastructures. Depending on the scope of an audit, massive amounts of evidence are produced and evaluated. Thus, mechanisms to manage complexity (such as filtering) and adequately presenting audit

results (such as heat maps), gain importance.

Audit result presentation in automatically generated natural language documents (e.g., using templates) can prove to be beneficial for less tech-savvy auditors such as an administrator at an Small or Medium-size Enterprise (SME) and/or for the management.

However, not only the presentation to human recipients is discussed in the following, but also the integration with other tools, such as incident response tools.

5.6.1 Dashboard for Audit Management and Reporting

The main interaction point with AAS for the auditor is its GUI. It allows the auditor to easily create audit tasks based on policies, monitor the current state of AAS's core components and agents, and most importantly review the result of audit tasks.

The GUI is implemented as a web-based dashboard using state of the art technologies and frameworks:

- Hypertext Markup Language (HTML) 5: as the main language for User Interface (UI) controls.
- Javascript: for interaction with the ReST server (mainly KnockoutJS [161] for templating audit tasks as described in Section 5.7 and jQuery [162]).
- Cascading Style Sheets (CSS): for defining the presentation aspects of the UI.
- JSON: as a compact data exchange format between dashboard and ReST service.
- Bootstrap: as a web framework that facilitate responsiveness of the frontend.

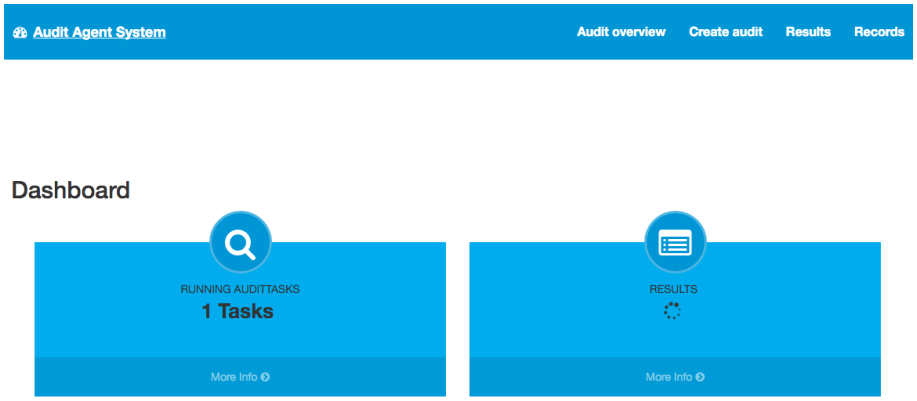

Figure 24: Audit Agent System User Interface – Landing Page

The different sections of the UI and the reasoning for their design and available controls is presented in the following sections.

User Interaction: Overview

The landing page for the auditor (i.e., the first page he sees of AAS) is depicted in Figure 24. The intention of this page is to provide a central hub for the auditor, where he can quickly gain an overview of the current audit state and has access to the most commonly used functions.

On the top, the auditor can quickly access the *Audit Overview*, *Create Audit*, *Results* and *Records* modules. On the bottom, the auditor is presented a very high-level overview of the current state of AAS. The left represents a count of all currently active audit tasks, whereas the right provides a summary of recently detected violations that might need to be reviewed by the auditor. Shortcuts are provided for each section to provide quick access to the relevant modules.

User Interaction: Audit Task Overview

During the execution of an audit task, the auditor can check its current state in the audit task overview module that is depicted in Figure 25. The presented information depends on the template that the audit task developer provides. This

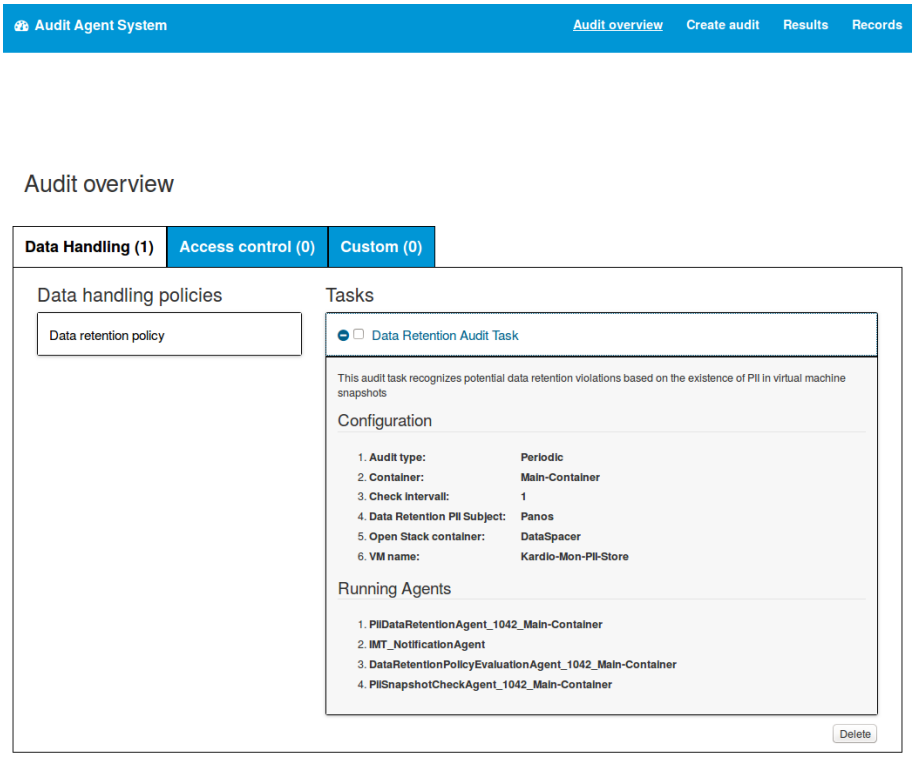

Figure 25: Audit Agent System User Interface – Audit Task Overview

is done to address the heterogeneity of audit tasks. The auditor is presented with an overview of the instantiated agents, their status, configuration and location in the environment. Furthermore, the auditor can take actions such as deleting an audit task, which results in the deletion of the associated evidence collector and evaluator agents as well as the disposal of any key material in order to *delete* the collected data in the evidence store. The deletion log itself should thereby be recorded as evidence of that action in the evidence store.

User Interaction: New Audit Task

The module for creating new audit tasks is depicted in Figure 26. The categories *Data Handling*, *Access Control* and *Custom* are selected dynamically based on the policy (i.e., type of rules and their association with aforementioned categories) that the auditor has provided. In this case, an A-PPL policy was used as input, which covers the aspects of data handling and access control. In the

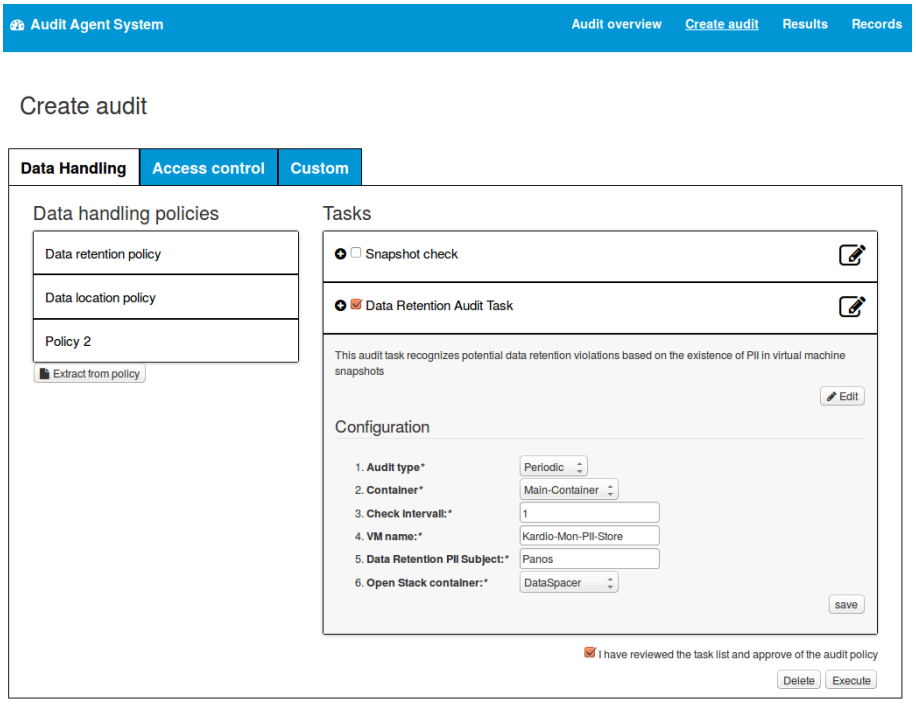

Figure 26: Audit Agent System User Interface – Task Creation

respective category (e.g., *Data handling policies*) the result of the input policy parsing process is presented (see Section 6.2) as a list of audit tasks that can be executed based on the rules that are defined in the input policy. For each audit task the auditor is given a partially pre-configured (on a best effort basis) configuration, which he is prompted to complete. With the *execute* command, the list of audit tasks and their configuration is sent to the AAS APM module, where the configuration parameter sanity is checked (i.e., input validation) and the instantiation process is triggered.

User Interaction: Result Presentation

In the results module (see Figure 27), violations that were detected by the audit tasks are presented to the auditor for review. The results are categorized in three different subsets:

- **Violation:** contains all violations detected by AAS that are with certainty identified as such.

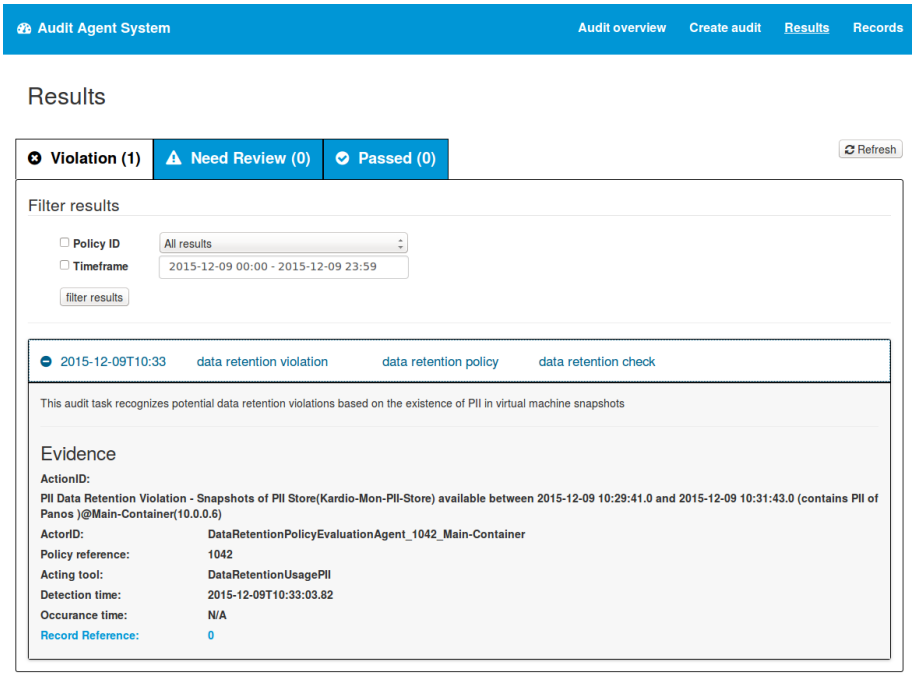

Figure 27: Audit Agent System User Interface – Violation Report

- **Needs Review:** contains all audit results that potentially represent a violation, but where AAS is unable to decide with certainty. A review of the collected evidence by the auditor is required.

- **Passed:** contains all results of audit tasks that directly check for compliance. In the demonstration cases (see Section 7.2), AAS checks for the occurrence of violations and potential violations and assumes the absence thereof to be a state of compliance at least with respect to the audit tasks.

The detection timestamp and the type of violation are highlighted in an overview list that provides additional information such as evidence that supports the violation claim. Additionally, the auditor can be provided with filtering capabilities that work on top of the result presentation list and enables him to limit the displayed results to a manageable amount, based on for instance the underlying policy or a specific period of time.

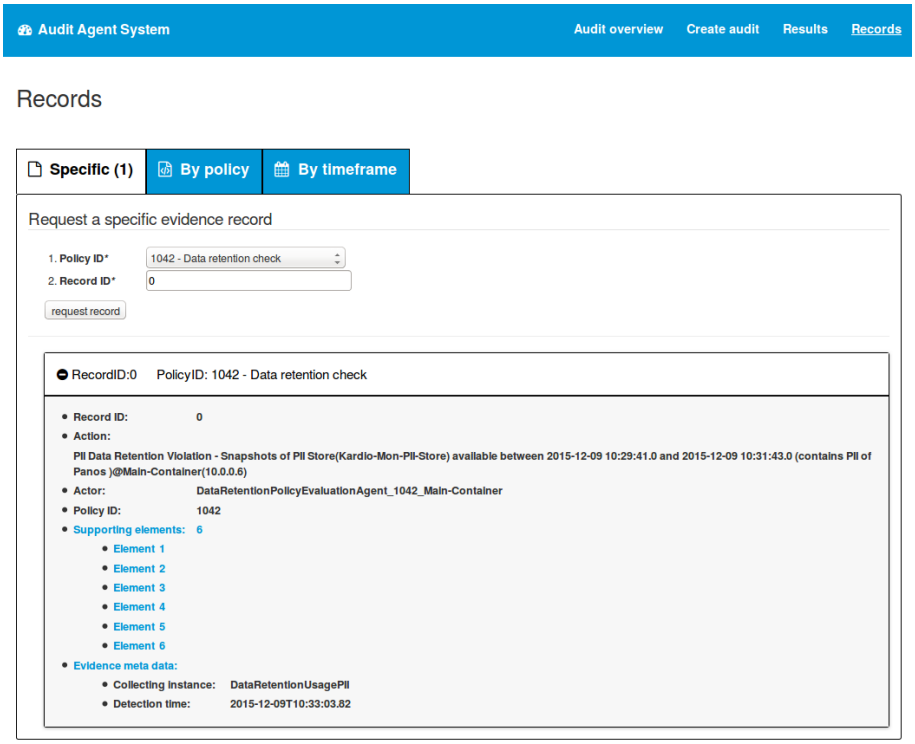

Figure 28: Audit Agent System User Interface – Evidence Record Representation of an Incident

User Interaction: Evidence Details

In the records module, the auditor can review the contents of evidence records that are stored in the evidence store (as depicted in Figure 28). Here, the auditor has access to the raw evidence that is stored in the record and on which AAS's audit tasks base their analysis.

5.6.2 Incident Notification

AAS can also be run without the GUI, simply by using its ReST interface. This facilitates the integration of AAS into other tools. For instance, this way audit tasks can be created and deleted automatically by another tool without the need for human interaction, under the one condition that all configuration parameters that would usually be supplied by the auditor are made available by the calling tool.

An important part of the reaction process after a policy violation has been detected, is the notification to relevant stakeholders (e.g., the auditor or the cloud provider in most cases, and the cloud consumer in very specific cases). AAS provides the ability to associate an appropriate notification mechanism with each audit task. From an architectural perspective the notification mechanisms are implemented as agents in the presentation component, which enables simple extension with new mechanisms and a one agent per tenant approach. Each agent provides a different kind of communication mechanism for violation reporting. In that sense, the notification agents function similarly to the collector agents but as adapters for the output of AAS to other systems. However, the notification part of an audit task is completely optional.

The following two sections are exemplary to show how notification is implemented in AAS. It is differentiated between notification to persons and notification to systems. In any case, the notification is generated from violations that are detected and reported by the evaluator agents. Of course, an audit task is not limited to a single notification agent but can make use of multiple.

Auditor Notification

The notification of the auditor (i.e., the person that actually created the audit task) is the most common notification mechanism. A simple way of notification can be implemented by using the *EMailNotificationAgent* that can be included in an audit task. In this case, the auditor is required to provide a valid e-mail address during task setup, which is later used for delivering violation notifications. However, with the addition of new notification agents that support different communication protocols (e.g., push notifications or instant messaging), there are virtually no limits to ways of getting urgent notifications to the auditor. Also, the notification process can become more flexible by not only allowing the notification to the auditor but also allowing automated notification to affected stakeholders (e.g., notifying consumers about an availability incident at their cloud resources). In any case, privacy issues need to be considered when

generating notifications both regarding the protection of other tenants privacy as well as protection from the auditor (who is usually bound by non-disclosure agreements).

Incident Management Reporting

The second way of notification functions almost identical but aims at sending notifications to other computer systems, such as incident management systems at a cloud provider. For this, AAS implements a *RESTNotificationAgent* that, depending on the other system's interface, is able to translate violation reports and their associated evidence records into the required output format. This form of incident notification is especially relevant for integrating AAS with incident management tools at the cloud provider, in order to allow a more fine grained management and reaction.

5.7 Modelling Audit Tasks

As described in Section 5.3 and Figure 10, the flow of evidence in the AAS can be quite complex. In this Section, three different layers that evidence data has to pass through from the collection up to the presentation of audit results are described. Controlling the flow of evidence from the source to the presentation is important for several reasons, such as scalability, privacy protection (correct recipients of evidence along the trail) and load balancing. An overview of the data flow between agents of an audit task is presented in Figure 29. The discussion in this section is aimed towards developers that extend AAS with new audit tasks, collectors and evaluator agents.

The design and development of a new audit task in AAS follows a set of phases. These phases are depicted in an overview in Figure 30, whereas the remainder of this chapter is used to describe them in more detail and provide the reader with a sense of the system extensibility that is gained by opting for an agent-based

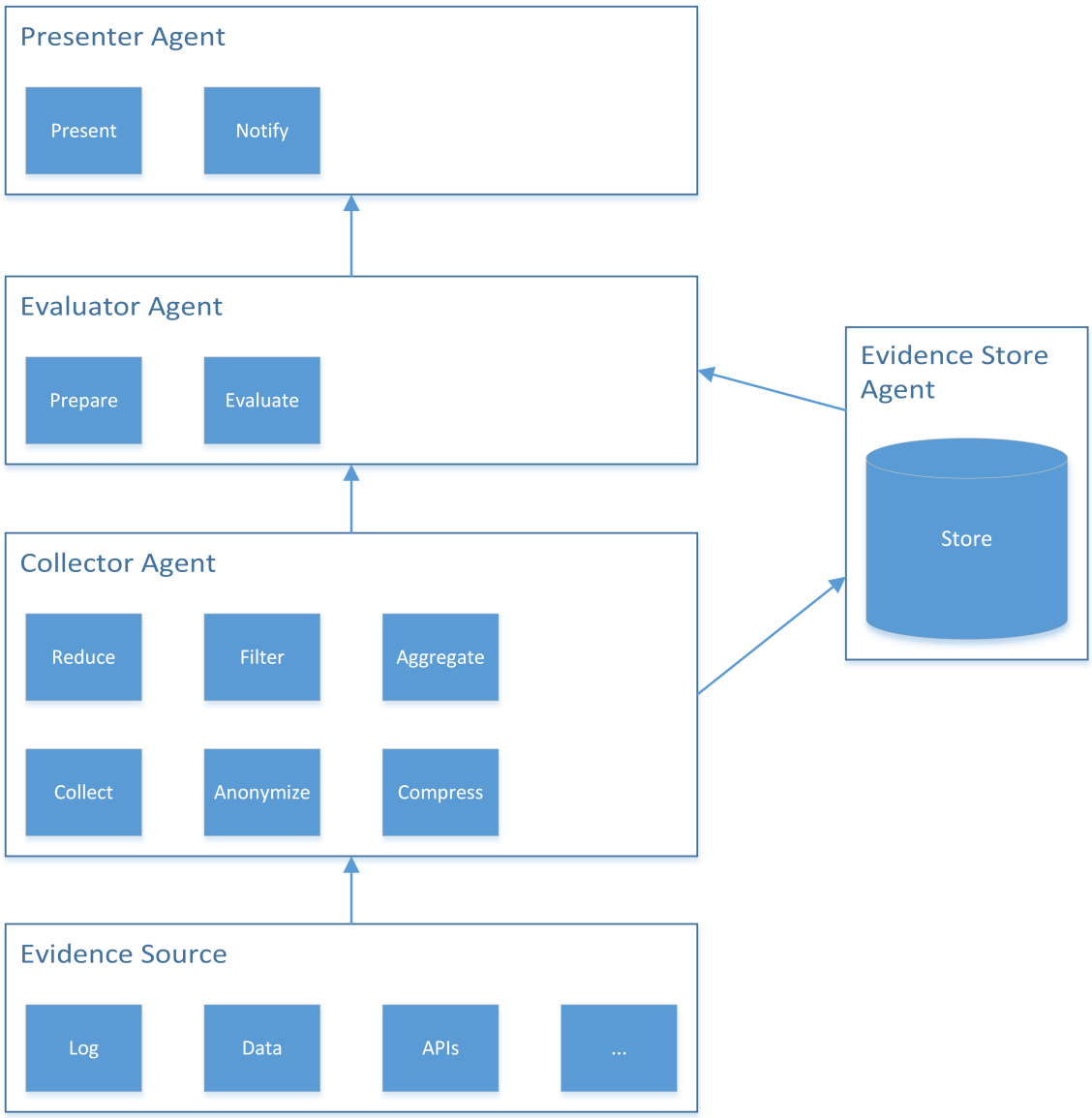

Figure 29: Agents of an Audit Task and Data Flows

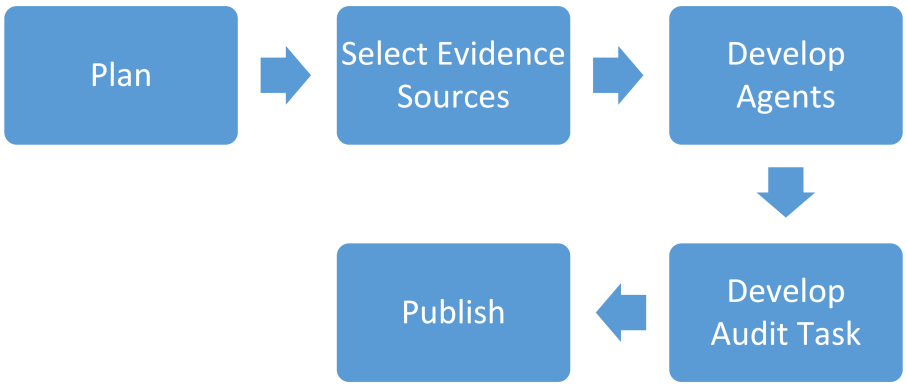

Figure 30: Phases of Developing an Audit Task

approach.

5.7.1 Planning Phase

In the first phase, the input policy language is analyzed by the developer. The main considerations are hereby:

- Review of policy rules: at foremost, an audit checks the current state against a desired state that is usually specified as rules. The developer considers the input policy and derives a potential audit task from it.

  For instance, if there is a rule in the policy that allows the governing of the location of VMs in a cloud, this means there is a potential audit task that is concerned with checking compliance of the cloud deployment with that policy. In many languages, the definition of what constitutes compliance and what a violation of a rule can be extracted directly, as long as the context of a complex cloud environment is not considered (e.g., as long as the complexity of provider chains is ignored and only a single provider considered). This can be the case when the language has limited applicability such as it is only concerned with the cloud from a SaaS point of view and omitting the context of other cloud layers beneath that, where failures can happen.

- Review of configuration parameters: The second consideration is based on

the mandatory and optional fields supplied in the input policy that can be used for audit task configuration. The goal is to always extract as much information as possible from the input policy, to reduce the amount of manual configuration that is required by the auditor. However, in some cases where the input policy is not expressive enough, deep knowledge of the cloud system may be required from the auditor in order to define tasks.

- Review of potential evidence sources: The final consideration is making a list of potential evidence sources that can be used to check the compliance with the aforementioned rule. This step requires a deep analysis of the potential compliance failures that can happen (i.e., looking at all the involved systems and their interactions and identifying potential failure scenarios based on this information). Also, a balance between a desirable level of certainty and effort that is required for the audit has to be defined (i.e., the depth of the audit in conventional offline audits) as the potential for failures is seemingly limitless in complex systems. This means, that the amount of evidence sources that is used in a single task and the evaluation mechanisms have to be chosen based on the defined balance between complexity of the task and required level of assurance.

- Definition of an audit task strategy: the choice of the actual audit type (see Section 4.5) needs to happen based on the policy rule that is being audited. The developer chooses a strategy for distribution and the audit interval. This information is important for the definition of the data flow in the audit task.

- Definition of an evaluation strategy: the choice of a suitable evidence evaluation mechanism (e.g., keyword search, feeding a rule engine or using artificial intelligence) is also prepared on the basis of the input policy. Potential evaluation strategies include simple and minimalistic approaches like keyword searches (e.g., in text such as logs, emails or documents) but can be as complex as using rule engines or ANN for analysis.

After these considerations, the evidence sources that have been identified are handled in the next step.

5.7.2 Evidence Source Selection Phase

In Section 4.2.2, it was stated that evidence can be produced by diverse sources in a cloud infrastructure. Here, the developer transforms the high-level consideration of the previous step into implementations, for example:

- Operations that are logged by the CMS are considered evidence ⇒ VM lifecycle events are collected from OpenStack's Nova service API.
- Access Control violations are considered evidence ⇒ Access control events that are emitted by the eXtensible Access Control Markup Language (XACML)-based enforcement tool are collected by pushing those events from the enforcement tool to a ReST-based event listener in the collector agent.

After this step is complete, the developer has defined specific evidence collection mechanisms and implementation details such as which protocols, data formats and interfaces are being used. With this information, the agent development phase can commence.

5.7.3 Agent Development Phase

This step covers the implementation of new and modification of existing agents with the goal of implementing them in an audit task. It includes at least the consideration of *collector agents*, *evaluator agents* and *presenter agents* (as depicted in Figure 29).

Collector Agent Development

Software agents collect evidence data from the evidence source, where it is produced. At the minimum the interfacing with the evidence source and the trans-

formation of raw data into evidence records has to be implemented. Optional functions include: reduction, filtering, aggregation, anonymization and compression of evidence as described in Section 4.4.

Every evidence collector agent in AAS is provided basic common functionality (e.g., behaviors, storing to the ES etc.) by extending the abstract class *EvidenceCollectingAgent*.

As new evidence sources are introduced into AAS audit tasks, new collector agents are required. Therefore, implementation effort for this kind of agents is expected to be high at the beginning when only a limited number of agents are available, but decreasing over time as collectors become available for reuse.

Evidence Evaluator Development

After the evidence data has been collected, it is passed to the evaluator agent and the evidence store. The evaluator agent compares the collected evidence against the rules defined in the policy using a suitable mechanism that was defined in the planning phase. The evaluation mechanism as well as the ruleset are then implemented by the developer as an agent.

Every evidence evaluator agent in AAS is provided basic common functionality (e.g., receiving from the ES or directly from collectors) by extending the abstract class *EvaluationAgent*.

As new evaluation mechanisms are introduced into AAS audit tasks, new evaluator agents are required. Therefore, implementation effort for these kinds of agents is expected to be high at the beginning when only a number amount of agents are available, but decreasing over time as evaluators become available for reuse.

Presenter Agent Development

Finally, there are presenter agents, that complete an audit task on the agent implementation level. It is very common to have an audit report as a document,

which includes the audit result (compliance statement with additional information and recommendations). Such documents can be generated automatically to some degree by using templates for Portable Document Format (PDF) generation (e.g., using LaTeX). This form of presentation is most useful, when audit intervals are quite long (for instance in a monthly audit).

There is also the presentation of the results in a web-based dashboard, as it is commonly done for monitoring and SIEM solutions. This approach is more useful if intervals are short or auditing is done continuously (i.e., as soon as a change event triggers a re-audit), because results can be displayed immediately. The dashboard presentation is part of AAS out of the box.

Additionally, there are notification mechanisms that can be implemented using a presenter agent (e.g., one for each protocol). A simple ReST client for dispatching violation notifications to other tools and an e-mail notification agent are already implemented to highlight the flexibility.

Most often, the presenter can be reused, since in most cases it is not dependent on the audit task objective. Also, with the provision of the two notification agents (tool and e-mail) as well as the dashboard, the effort for implementing additional presenter agents is expected to be minimal.

5.7.4 Audit Task Development Phase

In this phase, all individual agents are combined into an audit task. First a short overview of the general agent class hierarchy (see Figure 31) and the structure of a typical audit task (see Figure 32) are depicted using Unified Modeling Language (UML) class diagrams. These serve as a guide for implementation. The general approach to bundling agents in an audit task is presented afterwards.

In the second step, the audit task instantiation process in AAS is shown.

Finally, the approach to reducing effort in UI development and homogenizing the look and feel by using UI templating is described.

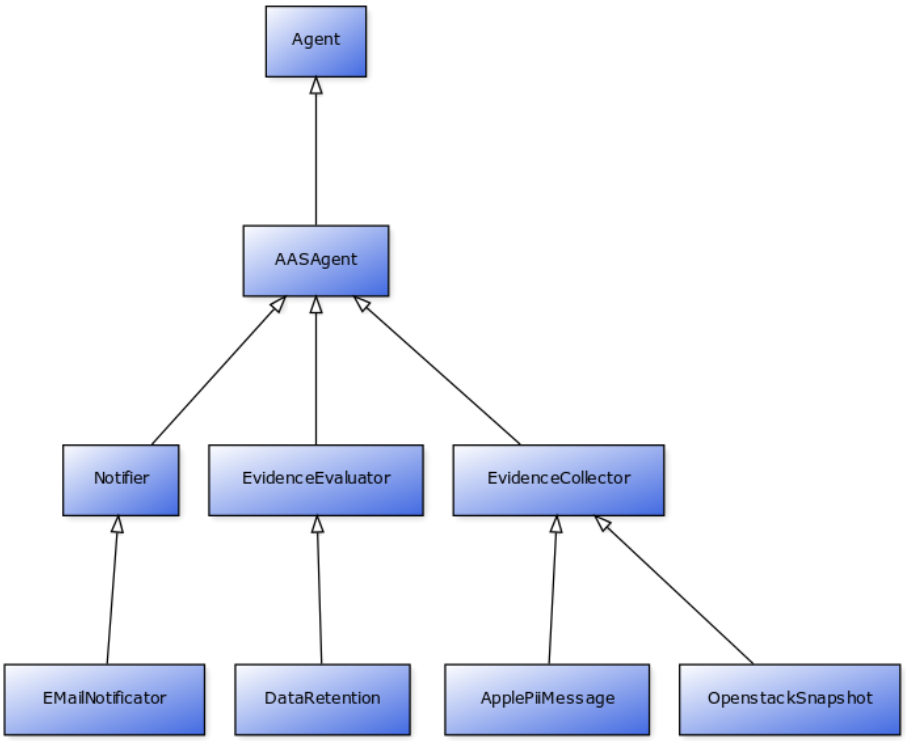

Figure 31: UML Diagram of Agent Class Hierarchy in the Audit Agent System

Audit Task Structure and Agent Bundling

Figure 31 depicts the general class hierarchy in AAS. An non-core agent in AAS is always associated with a abstract representation that identifies it as either a *Notifier* (with capabilities of sending messages), *EvidenceCollector* (with capabilities of receiving/gathering input and writing to the evidence store) or *EvidenceEvaluator* (with capabilities of reading from the evidence store and interacting with notifiers). These abstract representation combine shared functionality and enable more efficient management of agent groups by type in AAC. Every agent in AAS at some point inherits functionality from the type *AASAgent*, which itself is a specialization of the most general type *Agent*. At the most abstract level, everything is a JADE *Agent*.

The *AuditTask* class (see Figure 32) bundles a set of agents and configurations and is used as a control block in AAC during lifecycle management. It is a simple structure for representing an audit task with its associated 1-n collectors, 1-n evaluators and n notifiers. This way, the AAC can more easily perform

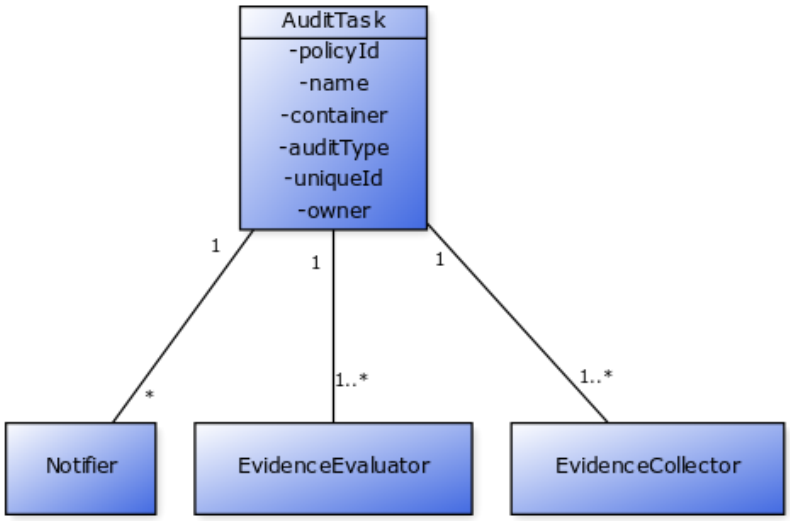

Figure 32: UML Diagram of Audit Task in the Audit Agent System

monitoring operations by simply iterating through the information contained in the control block.

Audit Task Instantiation

The runtime flow of audit task configuration and instantiation of agents is depicted in Figure 33 and executed as follows (Figure 33 is based on the A-PPL parser implemtation):

1. An auditor triggers the audit task creation process in the dashboard (action *Create Audit*).

2. The APM first searches the provided A-PPL policy (which is either uploaded or pre-configured on the AAS server-side) for possible audit tasks using the language parser. If possible, the audit task configuration is populated with general information such as available agent runtime environments to migrate collectors and evaluators to.

3. All potential audit task definitions are then sent back to the auditor for supplementation. This is carried out using JSON serialization of an audit task's object model in the JADE gateway, which acts as a ReST interface into the agent system.

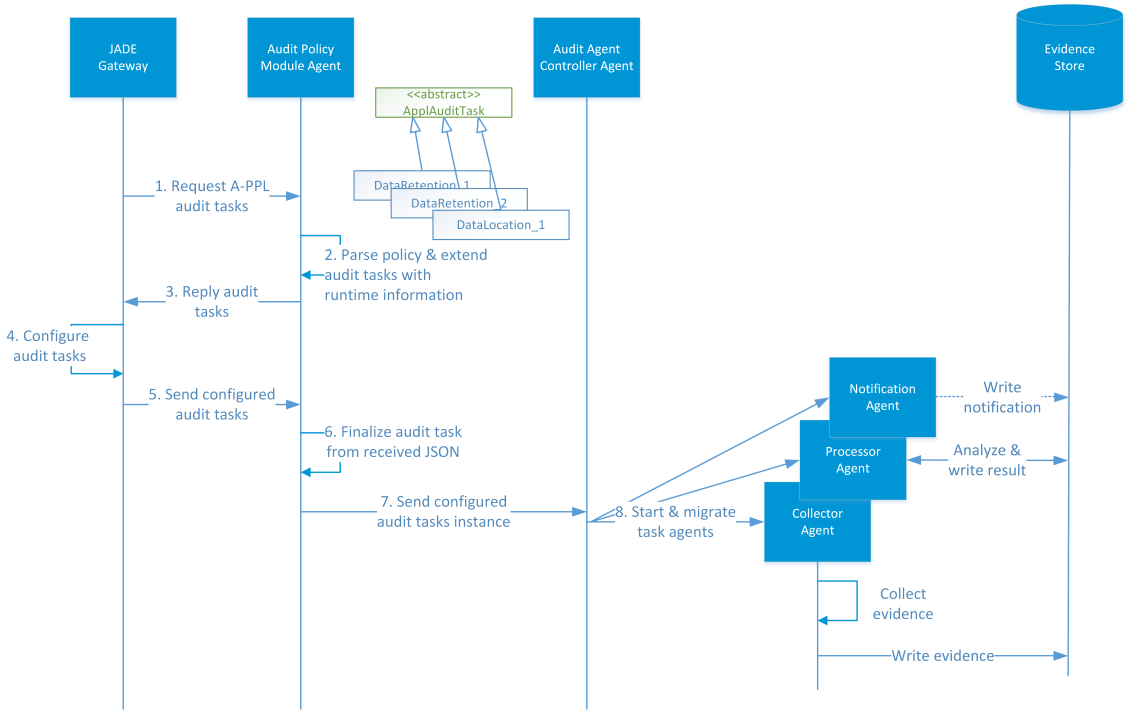

Figure 33: Technical Overview of Audit Task Initialization, Configuration and Agent Instantiation

4. The auditor is now prompted to select audit tasks based on what was detected in the policy, complete missing parameters and submit the final configuration for instantiation.

5. Once the list of finalized tasks arrives at the JADE gateway, they are forwarded to the APM.

6. The APM maps the received JSON structure to an audit task object model, generates a new GUID for it and passes the finalized task model to the AAC for instantiation and deployment.

7. The AAC starts and migrates all agents.

An audit task representation in JSON possesses (at least) the following fields:

- *auditTaskUUID*: The unique runtime identifier of an audit task.
- *type*: The actual audit task class identifier.

```
<h1>DataRetentionPolicyEvaluationAgent:</h1>

<label>Container:</label>
    <select data-bind="options: DataRetention_1.availableContainerHelper, value
        ↪ : DataRetention_1.evidenceAnalyzers[0].
        ↪ DataRetentionPolicyEvaluationAgent.
        ↪ PiiDataRetentionPolicyEvaluationAgent.container"/>

<label>Store VM name:</label>
    <input data-bind="value: DataRetention_1.evidenceAnalyzers[0].
        ↪ DataRetentionPolicyEvaluationAgent.
        ↪ PiiDataRetentionPolicyEvaluationAgent.piiStoreVM, valueUpdate: '
        ↪ afterkeydown'"/>
```

Figure 34: Example of a Template for Binding User Interface Elements to Audit Task Configuration Parameters

- *displayName*: The audit task name that is displayed by AAS in the dashboard.

- *description*: The audit task description that is displayed by AAS in the dashboard.

- *evidenceCollectors, evidenceEvaluators & notifiers*: Lists of every agent associated with the audit task.

Audit Task UI Templating

Audit tasks in AAS use Knockout.js [161] data binding to minimize effort for UI development. In the audit task's template folder, HTML files are used to define the data binding between audit task model (and its agents) and the UI controls. Figure 34 depicts a simple example, where different attributes of an agent are bound to different input fields in the UI. This effectively builds a binding between the UI elements and the more complex data model used in the background while enabling automated synchronization between the two.

This approach reduces development effort on the UI-side significantly.

5.7.5 Publishing Phase

The publishing phase is the last one of the agent development process and includes the integration of an audit task package into an AAS deployment. This phase can be extended by additional steps that assure the authenticity of the code that is provided, e.g., by code signing mechanisms or by putting the source code and accompanying audit task package up for a review by a third-party. Furthermore, introducing a public marketplace for pre-packaged audit tasks could address the problem of having only a limited set of audit tasks and agents available.

5.8 Summary

This chapter presented a broad overview of the design phase of AAS. The development methodology was described and an in-depth description of the architectural design was presented. It was thereby highlighted how the requirements with respect to functionality, security and privacy protection influenced the architectural choices. The main contributions of this chapter are the complete architecture design for automated, evidence-based audits in multi-provider scenarios, including development guidelines for new audit tasks that demonstrate AAS's extensibility.

Chapter 6

Audit Description using Software Agents

As described in the previous chapters, AAS is flexible enough to address different kinds of audits. As long as there is a machine-readable description of the policy (i.e., defining rules, conditions and requirements), the APM can be extended to incorporate parsers and adapters for these policy languages.

The incorporation of policy languages such as PrimeLife Policy Language (PPL) [163], A-PPL [74] or Cloud Audit Policy Language (CAPL) [97] enables AAS to cover different domains (e.g., data handling in A-PPL, infrastructure / software configuration as presented in [164] and security measures in CAPL).

In this chapter, the integration of security and accountability-related policy languages that can be considered input to the AAS to define an automated audit, are described. The focus is thereby put on A-PPL as a language for describing accountability policies, as this language was developed in the A4Cloud project (no direct involvement of the author of this thesis) and was integrated as a proof of concept for audit task extraction in AAS.

However, other policy languages and their potential for integration are discussed as an extension, in order to highlight gaps in A-PPL. Such gaps include infras-

tructure definition or the requirement for specific security and privacy protection measures. Both can be addressed by choosing an alternative policy language. For instance, Bosh Outer Shell (BOSH) is a language that enables the description of cloud service deployments from a PaaS perspective and thereby also covers details such as basic configuration parameters of involved VMs (i.e., VM Internet Protocol (IP) address, gateway, subnet, VLAN, operating system and application versions).

In general, evidence processing mechanisms highly depend on the audit task and the complexity of detecting potential violations. While the prototype implementation of AAS uses rather simple algorithms that depend on basic string search and counting, more complex variants can be implemented as well. For instance, a processor agent in AAS is only limited by the programming language and resources it may consume. Therefore, an agent could integrate a rule engine that uses collected evidence as input. There exist project that aim for integration of rule engines in an agent-based approach that could be used as a basis for a new AAS processor agent. Furthermore, processor agents that implement artificial neural networks for processing evidence (e.g., metrics from monitoring) could prove useful for certain audit goals as well.

While neither of these languages presented in the following was developed as a contribution of this thesis, they were chosen for evaluation due to their apparent suitability as policy languages from the cloud accountability, security audit and infrastructure declaration domains.

6.1 Audit Task Configuration

Figure 35 depicts the policy input as well as the process of deriving audit policies from input policies. The *Policy* is the collection of rules and obligations in a machine-readable format such as A-PPL.

Based on the input policy specification, the *Audit Policy* with its various *Audit*

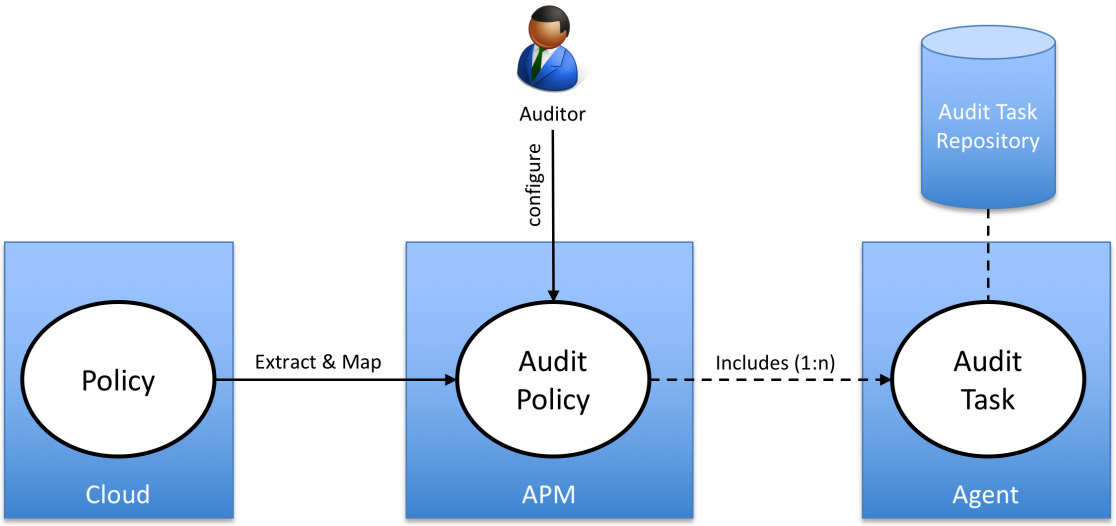

Figure 35: Audit Task Definition Process

Tasks are extracted and supplemented with knowledge input by the auditor if necessary. An audit policy always contains at least one audit task.

An audit task is a combination of evidence collector agents, evaluator agents, their associated evidence store and their configuration (which information to collect from a possibly very large pool) and thresholds (limits and conditions that constitute a policy violation). Audit tasks are prepared based on templates for a given rule for which compliance is to be audited. How an audit task is developed (concept and implementation) is described in Section 5.7 in more detail. The *Audit Task Repository* is an abstract term that describes the available set of audit task templates. In practice, it is realized by a set of Java packages that contain all required classes of the implementation.

For an audit target in the input policy a basic template is filled automatically on a best-effort basis by extracting relevant information from the policy. Missing information that is not available for extraction needs to be filled in manually. The APM integrates language parsers for different kinds of policy languages in order to enable automated configuration. The prototype implementation integrates a language parser for A-PPL that enables the extraction of configuration parameters. However, AAS's APM component can be easily extended to also include

parsers and task extractors for other languages as well.

Audit tasks are provided to the auditor in order to enable him to perform audits in an automated way that is based on evidence. The configuration of an audit task is completed by an auditor who provides additional information that is not contained in the policy. After the configuration is completed by the auditor, the audit task can be instantiated in the system.

6.2 Cloud Data Processing Description

One of the outcomes of the PrimeLife research project [165] was the PPL [163, 166]. The A4Cloud [9] research project developed a machine-readable policy language based on PPL called A-PPL [74, 167, 168]. A-PPL is capable of describing obligations providers have to adhere to in order to be considered accountable. It therefore allows to define how data should be handled by cloud providers in an machine-readable and enforceable way. This closes the gap, where previously there was no way of defining such policies other than in natural-language documents such as privacy policies and terms of service where data processing aspects such as retention periods, data sharing, usage control and data location (i.e., data center location) are usually described. In [168], the authors described several examples for obligations that can be represented in A-PPL. Those that go beyond the access control part of A-PPL that is provided by the underlying XACML layer, are concerning notification in case of data breaches, purpose binding of data and the obligation to collect evidence of data deletion.

To complement A-PPL as a language, the A-PPL-E enforces policies in Java-based applications. The engine is thereby an extension of the PPL engine [166]. Logs produced by the enforcement engine are considered an important source of evidence of data handling.

A-PPL	AAS	Mapping Description
Policy	-	Provides the root element and very general information like the *PolicyId*.
Target	-	Declaration of data types that A-PPL applies to.

Table 7: A-PPL to AAS Mapping for Audit Task Extraction – General

6.2.1 Mapping to Audit Tasks

While A-PPL can be used to describe data processing obligations to which a cloud provider should adhere to and A-PPL-E is used to enforce these obligations, audits are required to assure the compliance with and correct enforcement of such rules. As previously stated, data processing policies based on A-PPL are considered to be the most important input to AAS during the extraction and definition of audit tasks. AAS integrates a parser for A-PPL that allows for automated processing of policies.

The following Tables 7, 8 and 9 depict the mapping of A-PPL language constructs to AAS concepts illustrated in Figure 35. From an AAS perspective, A-PPL features three major sections:

1. General: see Table 7
2. Access control (XACML-based): see Table 8
3. Data handling (accountability extension): see Table 9

The *Policy* contains meta information about the policy document such as a description or namespace definitions which can be reused in AAS. The meta information contains properties such as the *PolicyId* that is used by the APM to link audit tasks to a specific policy upon which audits are performed. The tag *Target* contains resource definitions that directly correlate to data for which the policy applies.

While the general information provides little in terms of configuration parame-

A-PPL	AAS	Mapping Description
Group of all XACML rules from the access control section	AuditPolicy	Element for grouping AuditPolicies belonging to the same A-PPL policy.
Rule (XACML)	AuditTask	A single check that is (part of) the compliance check.

Table 8: A-PPL to AAS Mapping for Audit Task Extraction – Access Control

A-PPL	AAS	Mapping Description
DataHandlingPolicy	-	Container element
AuthorizationSet (group of authorizations)	AuditPolicy	Element for grouping AuditPolicies belonging to the same A-PPL policy.
Authorization	AuditTask	Describes a set of checks (AuditTasks) that need to be performed to evaluate compliance with an authorization rule.
ObligationSet (group of obligations)	AuditPolicy	Element for grouping AuditPolicies belonging to the same A-PPL policy.
Obligation	AuditTask	A single check that is (part of) the compliance check.

Table 9: A-PPL to AAS Mapping for Audit Task Extraction – Data Handling

ters for actual audit tasks, the two following sections are more important.

The access control part features common language elements from XACML such as *Rule*, *Target*, *Subject* and *Action*. All of these can be mapped to AAS's *AuditPolicy* and *AuditTask*. This is especially interesting, when access control logs are being audited. In such an audit, logged access control operations like *read* (Action) access to a *resource* (Target) was granted to a certain *entity* (Subject) are evaluated for failures.

A-PPL possesses a data handling section and the elements *ObligationsSet*, *Obligation*, *AuthorizationsSet* and *Authorization*. The tag *DataHandlingPolicy* thereby serves as a container element for the data handling sections of an A-PPL policy. Everything contained inside the section is considered a potential rule or obligation that is mapped onto an audit task.

A-PPL Element	**AAS Audit Task**
TriggerPersonalDataAccess-Permitted, TriggerPersonalData-AccessDenied	Addressed by the intrusion detection audit task described in Chapter 7.2.1.
ActionNotify	Addressed by the notification audit task described in Section 6.2.5.
Purpose	Addressed by the data retention audit task presented in Section 6.2.3 and Chapter 7.2.3.
AuthzUseForPurpose	Addressed by the location audit task described in Section 6.2.2 and Chapter 7.2.2.

Table 10: A-PPL to AAS Mapping for Audit Task Extraction – Audit Tasks

Each of those mappings contains further language elements that contain valuable information for the building of audit tasks. Table 10 highlights the connection between the language construct introduced by Azraoui et al. [169] and their audit task counterpart in AAS.

Additional details on specific properties of each of the constructs and how they add to an audit task are described in the remainder of this section as well as (from an implementation perspective) in the functional evaluation that is presented in Chapter 7.

In the following, A-PPL rules and obligations are used to illustrate how audit tasks are built. It is highlighted, which rules are being used to create audit tasks and which information for audit task configuration can be mapped and which has to be supplied by the auditor.

6.2.2 Example 1: Data Location

In A-PPL, the processing of data can be restricted to geographic regions. For AAS the geographic restriction means an audit task is required that verifies that the current location as well as the transfer history is in compliance with this rule.

Figure 36 depicts such an example, where PII processing according to that policy may only be happen in the EU and only in the context of finance.

```
<a-ppl:AuthzUseForPurpose>
    <a-ppl:Purpose location="Europe">http://www.w3.org/2002/01/p3pv1/financial</a-ppl:purpose>
</a-ppl:AuthzUseForPurpose>
```

Figure 36: Example for a Data Location Restriction in A-PPL

The evidence sources and respective evidence collectors in this audit task are as follows:

- Main source: data transfer monitoring. While there are rules on regional restrictions, there must also be an enforcement or monitoring tool. Logs that are produced by such tools and contain data provenance and location information are collected by an evidence collector and stored in the evidence store.
- Other sources: All system that give some indication of the current location of a particular data object serve as an evidence source. Transfer event logs are stored in the evidence store.

In Section 7.2.2 a demo audit is presented that is based on a data location rule from the data handling part of A-PPL and considers a data location monitoring tool as main evidence source.

6.2.3 Example 2: Retention Obligation

A similar example concerns the maximum data retention period for PII. An obligation as depicted in Figure 37 instructs the A-PPL-E to generate log events when PII is deleted. The rule reads as follows: every time a PII object is deleted from the storage (*TriggerPersonalDataDeleted*), log the current timestamp, action, purpose, subject and resource (*ActionLog*). In conjunction with the data retention rule depicted in Figure 38, each delete operation that is executed due to the maximum retention period (in this case 2 minutes) being reached for a particular data object generates a corresponding log event. The rule reads as

```
<Obligation>
    <TriggerPersonalDataDeleted/>
    <ActionLog>
        <Timestamp/>
        <Action/>
        <Purpose/>
        <Subject/>
        <Resource/>
    </ActionLog>
</Obligation>
```

Figure 37: Example for a Evidence Collection Obligation in A-PPL

```
<Obligation elementId="12374">
    <TriggersSet>
        <TriggerAtTime>
            <Start>
                <StartNow/>
            </Start>
            <MaxDelay>
                <Duration>P0Y0M0DT0H2M0S</Duration>
            </MaxDelay>
        </TriggerAtTime>
    </TriggersSet>
    <ActionDeletePersonalData/>
</Obligation>
```

Figure 38: Example for a Data Retention Obligation in A-PPL

follows: store the current time at the time of storage (*StartNow*) and delete (*ActionDeletePersonalData*) the object after a set maximum duration (*MaxDelay* and *Duration*) relative to the initial timestamp.

From the fact that there is a retention obligation that the cloud provider has to comply with, an audit task is derived. The evidence sources and respective evidence collectors in this audit task are as follows:

- Main source: delete events in A-PPL-E. Each time that a PII object is deleted, a corresponding delete event is logged by the A-PPL-E. The A-PPL-E evidence collector filters the A-PPL-E logs for delete events and records them in the evidence store.
- Other sources: Data duplication events (such as copy, snapshot, clone etc.) are generated at sources outside the scope of the enforcement engine such as OpenStack's Nova. At each evidence source there is an evidence

```
<Rule Effect="Permit" RuleId="120394">
    <Target>
        <Subject>
            <SubjectMatch MatchId="function:string-equal">
                <AttributeValue DataType="string">Administrator</AttributeValue
                    ↪ >
                <SubjectAttributeDesignator DataType="string" AttributeId="
                    ↪ subject:role-id"/>
            </SubjectMatch>
        </Subject>
            <Action>
                <ActionMatch MatchId="string-equal">
                    <AttributeValue DataType="string">read</AttributeValue>
                    <ActionAttributeDesignator DataType="string" AttributeId="
                        ↪ action-id"/>
                </ActionMatch>
        </Action>
    </Target>
</Rule>
```

Figure 39: Example for an Access Control Rule in A-PPL

collector that gathers log events that is associated with the aforementioned operations and stores them in the evidence store.

In Section 7.2.3 a demo audit is presented that is based on data retention rule from the data handling part of A-PPL and considers the enforcement engine's logs and the CMS history as main evidence sources.

6.2.4 Example 3: Access Control

In Figure 39 a rule is defined that describes an access control.

This rule defines the *read* access privilege for users with the role *Administrator*. The default decision of the access control system is thereby *deny* with this rule defining an exception. This rule is enforced but needs to be audited due to the scope of the enforcement engine. AAS takes this rule as input and forms the basis for automated auditing of access control decision that have been made based on that rule.

In Figure 40 an obligation is depicted that describes for which purpose a data set may be used. It describes a purpose *admin*, duration of *two years* and region

```
<a-ppl:AuthzUseForPurpose>
    <a-ppl:Purpose duration=2Y region=Europe>admin</a-ppl:Purpose>
</a-ppl:AuthzUseForPurpose>
```

Figure 40: Example for a Purpose Binding Obligation in A-PPL

Europe to which it applies.

The evidence sources and respective evidence collectors in this audit task are as follows:

- Main source: enforcement engine. Log events from the access control enforcement tool are of utmost importance. These decisions are compared against the policy in effect. While there should not be any violations, assuming the enforcement part is working correctly, validation and analysis that goes beyond decision making based on a single action is performed by AAS.

In Section 7.2.1 a demo audit is presented that is based on a rule from the access control part of A-PPL and takes the enforcement engine's input as main evidence source to detect intrusions.

6.2.5 Example 4: Notification Obligation

In Figure 41 a rule is defined that obligates a cloud provider to notify a certain stakeholder in case of a detected policy violation. In this case the trigger *TriggerOnPolicyViolation* is fired, when a violation is reported to the A-PPL-E. This causes *ActionNotify* to be executed as a reaction.

For AAS, this obligation is of particular importance, when assurance is required that all incidents (e.g., policy violations) have been notified to their according recipients in time. In this case, an audit task is extracted from *ActionNotify* that assures the correct notification and detects failures that might have been caused by for example bounced emails due to wrong email address.

```
<Obligation>
    <TriggersSet>
        <TriggerOnPolicyViolation/>
    </TriggersSet>
    <ActionNotify>
        <Media>e-mail</Media>
        <Address>stakeholder@example.com</Address>
        <Recipients>stakeholder</Recipients>
        <Type>Policy Violation</Type>
    </ActionNotify>
</Obligation>
```

Figure 41: Example for a Notification Obligation in A-PPL

The evidence sources and respective evidence collectors in this audit task are as follows:

- Violation reporting tool: The violation reporting tool, is where the violation is detected for the first time. This can be AAS itself. The type and date of the violation are most important. As previously stated, AAS stores violation reports as evidence in the evidence store.

- Violation reporting events in A-PPL-E: An evidence collector sits at the A-PPL-E engine and collects its internal logs as evidence. AAS reporting violations is one type of these log events. The A-PPL-E evidence collector filters A-PPL-E logs for violation reporting events and stores them in the evidence store.

- Notification mechanism: The notification event that is triggered by the violation (i.e., Mail Transport Agent (MTA) called to send an email) is recorded as evidence of the notification process. The mail evidence collector records logs of the emailing process (mail header, status code etc.) filtered by recipient (only recipient for whom there is also an obligation) and type (i.e., violation notification email).

6.3 Cloud Security and Deployment Description

As part of the SAaaS research project, CAPL [97] was developed. The main purpose of CAPL is to enable configuration of software agents to perform security checks primarily inside virtual machines in a cloud environment. The main security policies that were considered during the design of CAPL are:

- Malware, which includes triggers for malware scanning on VMs during an audit.
- Filesystem monitoring, which reports file changes during a continuous audit.
- Technical attribute modelling , which detects misconfigurations such as open network ports during an audit.
- VM content, which detects mistakes in software version and license management during an audit.
- VM scalability, which enforces limitations on automated scaling of resources depending on the continuous audit state.
- Data traces, detects potential data leaks by uncovering leftover keys, passwords etc. during an audit of publicly available resources.

Based on these requirements, CAPL was defined as an extension to Cloud Infrastructure Management Interface (CIMI) [170] in order to adopt it for SAaaS. CAPL's most important changes to CIMI are the introduction of several new classes that are required for conducting security audits.

6.3.1 Mapping to Audit Tasks

CAPL can be used as input to AAS in order to describe an audit based on what evidence has to be collected and where and also how this evidence has to be

CAPL Element	AAS Equivalent	Role in AAS
Group	-	No counterpart in AAS; language-based grouping of rules and resources in CAPL
Machine	Client bundle	Corresponds to AAS client bundle in case of servers and virtual machines
MachineTemplate	-	Not immediately available in AAS but can be retrieved from the CMS using the VM Globally Unique Identifier (GUID)
PolicySet	AuditPolicy	Describes a set of rule-based checks (AuditTasks) that need to be performed to evaluate access control compliance
Policy	AuditTask	A single rule-based check that is (part of) the compliance check
RuleType	AuditTask	RuleType information is implicitly included in an AAS AuditTask

Table 11: CAPL to AAS Mapping for Audit Task Extraction

processed. Due to CAPL's strong focus on security-related audits on the level of VMs and physical cloud hosts, it can be regarded as an important addition to AAS for describing security-focused policies that A-PPL, due to its focus on accountability issues, cannot address. In the following, a mapping between core elements of CAPL and correlating components in AAS is presented. It is also shown, how information about audits that is described in CAPL can be extracted and used in AAS to configure audit tasks.

By following this mapping in a CAPL extension of the APM, AAS can be enabled to extract audit tasks and perform compliance audits based on CAPL rules. Furthermore, due to the technological similarities between SAaaS and AAS it is conceivable to re-use and integrate specific agents of SAaaS in AAS. The following example elaborates on how the policy language integration could be achieved.

```
<group xmnls="http://research.cloud.hs-furtwangen.de/capl/">
    <id>https://aas.cloud.hs-furtwangen.de/groups/tenant-virtual-machines-1</id
        ↪ >
    <name>Virtual resources by tenant 1</name>
    <refName>tenant-virtual-machines-1</refName>
    <description>Group of virtual and physical reources that are provisioned
        ↪ for tenant with identifier 1</description>
    <enabled>true</enabled>
    <machines>
        <machine href="https://aas.cloud.hs-furtwangen.de/rest/machines/1"/>
    </machines>
</group>
```

Figure 42: Example for a Policy Governing Unencrypted Resource Access in CAPL – Resource Grouping

```
<RuleType>
    <resourceURI>https://research.cloud.hs-furtwangen.de/rest/ruleTypes/
        ↪ unencryptedResourceAccess</resourceURI>
    <category>security</category>
    <attributeKey>metric</attributeKey>
    <attributeKey>period</attributeKey>
    <attributeKey>threshold</attributeKey>
    <attributeKey>filter</attributeKey>
</RuleType>
```

Figure 43: Example for a Policy Governing Unencrypted Resource Access in CAPL – Rule Definition

6.3.2 Example: Unencrypted Resource Access

On a higher level of abstraction, a provider's security policy or an agreement with its customer may require that all data transfer that happens in a single cloud service may only happen in a sufficiently protected manner. This means, a suitable encryption scheme should be applied such as Hypertext Transfer Protocol Secure (HTTPS) using Transport Layer Security (TLS). While careful planning and service configuration can reduce the risk of accidental unencrypted data transfers (e.g., by prohibiting the use of plain Hypertext Transfer Protocol (HTTP) or weak encryption algorithms), an audit introduces assurance, that there are in fact no undetected misconfigurations that put data at risk of exposure. Such an audit can easily be described using CAPL.

Figures 42, 43 and 44 show a simplified example of how such an audit policy looks like. There are three distinct sections in this policy.

- **Resource group:** This section describes a group of resources to which the security requirement of exclusively encrypted data transfer applies. In this particular case, this resource group consists of only a single VM that hosts a web server (see *machine* tag for machine *1*). Furthermore, this group of resources is associated with a single tenant. This information is relevant for evidence collectors in AAS.

- **A custom audit description:** The rule type is described in a custom *RuleType*. The rule is identified by the complete Unique Resource Identifier (URI) for *unencryptedResourceAccess*. Furthermore, the custom rule requires the definition of the additional parameters *metric* (i.e., what and how should be counted), *period* (i.e., in what period should be counted), *threshold* (i.e., what is the threshold the indicates a violation) and *filter* (i.e., which events should be counted). This information is relevant for the evidence evaluators in AAS

- **The actual audit rule:** This section brings together the previously defined resource group and the rule type for the audit and combines them to a policy by associating them and providing the missing values for the abstract rule. Unencrypted resource accesses counts serve as the metric, a period of counting 60 seconds is relevant before the counter is reset for the next audit interval. A threshold of zero effectively leads to a policy violation if one or more unencrypted resource accesses are counted during a single period. To limit the amount of events (e.g., not all resources require the same level of protection), a filter is specified that in this case matches all resources that are exposed by the web server running on the VM. This information is relevant for the evidence evaluators in AAS

There are several approaches to declaratively defining cloud environments. Open Virtualization Format (OVF) [171] is a standardized format maintained by DMTF for building and describing distributable virtual appliances. It is agnostic to the type of hypervisor and is used in cloud as well as non-cloud environments.

```
<Policy xmnls="http://research.cloud.hs-furtwangen.de/capl/">
    <id>https://aas.cloud.hs-furtwangen.de/policies/unecryptedResourceAccess</
        ↪ id>
    <name>Disallow unencrypted resource access</name>
    <refName>encryptedacces1</refName>
    <description>There must be no more than X requests on a resource matched by
        ↪  FILTER via unencrypted HTTP in a given period of time</description>
    <enabled>true</enabled>
    <ruleType href="https://research.cloud.hs-furtwangen.de/rest/ruleTypes/
        ↪ unencryptedResourceAccess"/>
    <targetResource href="https://research.cloud.hs-furtwangen.de/rest/tenant-
        ↪ virtual-machines-1"/>
    <attribute key="metric">unencrypted-resource-accesses</attribute>
    <attribute key="period">60</attribute>
    <attribute key="threshold">0</attribute>
    <attribute key="filter">/resources/*</attribute>
</Policy>
```

Figure 44: Example for a Policy Governing Unencrypted Resource Access in CAPL – Policy Definition

Besides the disk image of an OVF package, the descriptor plays the most significant role in describing a virtual appliance. The descriptor is an XML-based (see [172] for the full schema definition) document that describes the appliance and the details of its deployment as VM such as: network configuration, disk images, allocation of virtual resources and more.

A language that enables the modelling of interactions between an IaaS cloud provider and its consumers is CIMI [170]. CIMI explicitly focuses only on IaaS aspects of the cloud and omits to consider PaaS and SaaS scenarios. It specifies a description language for basic resources in an IaaS cloud such as networks, machines, configurations and storage. Additionally, CIMI specifies a ReSTful protocol for exchanging such information between the cloud provider and the cloud consumer.

Another language that recently gained traction in the context of the PaaS cloud project Cloud Foundry [131] is BOSH [173]. BOSH is an open source tool that enables the provision of virtual environments in a cloud. It also supports software developers in packaging new releases and automating deployment and lifecycle management. BOSH's main area of application is in enabling the deployment of PaaS environments in Cloud Foundry-based clouds. However, BOSH

is flexible enough to be used in conjunction with many different IaaS providers and also outside of the context of cloud computing.

BOSH is built upon the concept of stemcells, which are similar to common virtual machine templates. A stemcell is a lightweight virtual machine that includes a BOSH agent. When a stemcell is instantiated (for instance by using the resources provided by an IaaS provider), BOSH interacts with the agent to install further software on the VM and perform configuration tasks. In that way, the simple stemcell that does not serve much purpose initially, can *grow* into a full application. Runnable cloud environments are called deployments in BOSH. A deployment is described by a deployment manifest, which is a configuration file written in JSON.

Deployment manifests in BOSH are suitable to describe an expected state of a virtual environment. This does not have to be always coupled with a full Cloud Foundry deployment. Because of BOSH's simplicity and extensibility, it is suitable to be parsed by APM to provide additional information for defining audits based on policies described in A-PPL or CAPL.

Additionally, BOSH and other infrastructure description languages can be suitable during audits where the objective is to assure that the actual cloud deployment (resources and software) is identical with the expected deployment as described in such language.

6.4 Audit Description

AAS provides the technological framework for auditing policy compliance by giving the auditor access to a repository of audit tasks and associated agents to choose from. They support the auditor by encapsulating repetitive tasks of setting up automated audits. However, an audit task cannot be fully captured by the policy language alone, due to the limitations of expressiveness and scope of the policies. This was demonstrated using languages from three different do-

mains. While A-PPL captures high-level aspects and obligations with respect to accountability, it does not define how a audit can be performed to assure compliance. The same holds true for CAPL and BOSH as well. This is why a semi-automated approach to audit task definition has been chosen in this project that incorporates a combination of agents (see Section 5.3.1 for list of developed agents) into a task that is associated with a specific type of rule.

6.5 Summary

In this chapter, three different policy languages were explored for their suitability as a baseline for describing audit tasks in AAS. As a proof of concept, the implementation of A-PPL-support in AAS was described using three different accountability rules. Furthermore, CAPL was discussed as a potential language for describing security focused audits. For both languages, a mapping of language constructs to concepts in AAS was presented that form the basis for the audit task extraction and configuration process in AAS's APM.

Additionally, BOSH was presented as a language that is used in PaaS cloud scenarios for describing deployments on virtual infrastructures. While it does not define rules similar to A-PPL or CAPL, it does provide a (should-be) declarative description of a cloud-based virtual environment based on resources such as VMs or networks and software configurations. These properties make BOSH a candidate, for both providing additional infrastructure information to the task parsing process that cannot be provided by either A-PPL or CAPL, and for auditing actual cloud deployments against an expected state that is defined in BOSH.

Chapter 7

Evaluation

In this Chapter, the AAS research results alongside its prototype implementation are evaluated and discussed from three different perspectives:

- Functional perspective
- Scalability perspective
- Privacy protection and security perspective

Based on this discussion, strengths and weaknesses of the AAS approach are presented with suggestions for future extensions that could address its shortcomings.

The demonstration scenario and deployment has served as a foundation for the evaluation of research results that have been published in the aforementioned papers associated with this thesis. This section contains, consolidates and extends these separate evaluations.

7.1 Deployment and Evaluation Scenario

For the development and evaluation of the AAS prototype, a demo deployment was built that matches the following story: the IaaS provider in this deployment

refers to CP2 in the scenario described in Section 4.1 and provides infrastructure level resources to its customers (i.e., the SaaS provider CP1) using OpenStack [14]. The deployment has the following properties:

- Two physical servers with dual 6-core Xeon X5660 processors and 40GB RAM, each.
- One of the servers acts as an OpenStack compute node that is only hosting VMs.
- The other server acts as an OpenStack controller and compute node hybrid, which hosts VMs as well as OpenStack's core services (such as Nova for computing resources) and optional services (such as the Horizon dashboard).
- Both servers provide AASs runtime environments for executing software agents on the hypervisor and host level.
- Two VMs simulate CP1's service. The VMs are distributed on the physical servers.

The definition of the scenario follows that provided in Section 4.1, with the limitation of the number of cloud providers that are involved in the service provision to two (i.e., CP1 and CP2) in order to simplify the functional demonstration.

The deployment is depicted in Figure 45. Blue elements are associated with the IaaS provider's (CP2) administrative domain, whereas orange elements are associated with the SaaS provider's (CP1) domain. Both providers run their own instance of AAS, which is depicted as the element *Core + UI*. Several *Runtime* elements provide the required agent runtime environments.

The evidence sources that are required for the audit cases demonstrated in the remainder of this chapter are:

- Host and IaaS level: the host level is depicted by CP2's AAS Runtime envi-

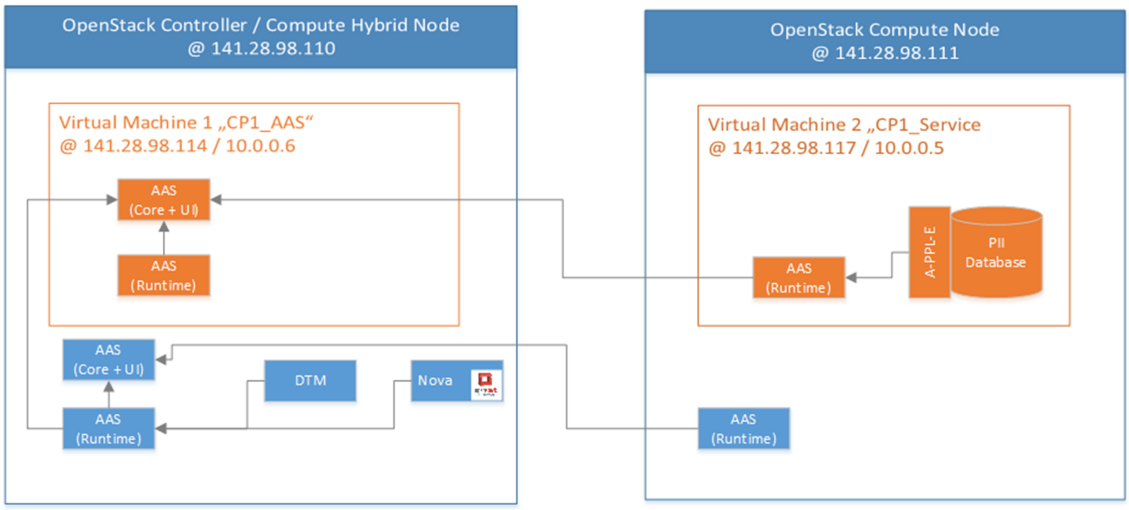

Figure 45: Audit Agent System Prototype Deployment for Evaluation

ronments on the actual physical servers and CP1's AAS Runtime environments inside its VMs.

- DTM: CP2 runs a dedicated tool for monitoring data transfers, which is used as an evidence source. This tool is external to AAS.

- Openstack's Nova service: This part of the CMS layer is made available to authorized users as an evidence source by CP2.

- Enforcement engine: the policy enforcement engine (A-PPL-E) of CP1's service is used as an evidence source as well.

The flow of evidence is depicted in the figure. Agents running at the *Runtime* component collect evidence and send it to evidence stores at the *Cores*. The border between the administrative domains of the two providers is crossed on the controller node, where a collector agent from CP1's core (orange) can be migrated to CP2's runtimes on both servers (blue) in order to collect data for example from the nova or DTM services.

7.2 Functional Evaluation

The functional perspective aims at evaluating AAS regarding its capability of enabling automated, evidence based and policy-driven audits of privacy, security and accountability policies (see objective 1 and 2 in Section 1). The evaluation methodology is based on a set of sample audit cases that focus on different interaction scenarios, evidence sources and evaluation mechanisms. The main reason for this approach is the ability to highlight AAS's flexibility as a framework for evidence-based audits.

The first audit case for evaluation deals with an approach to detecting potential breaches of a cloud service based on access patterns. The approach is lightweight (in terms of the complexity of the used evaluation mechanism to find violations) but provides added value due to its ability to uncover potentially unwanted data access.

The second approach shifts most of the evidence collection and violation detection work to another tool that is concerned with detecting potential data location violations. This scenario highlights AAS's capability to incorporate already existing monitoring and auditing tools that may be limited in their scope but complement AAS's evidence collection capabilities.

The final approach is used to show the integration of different evidence sources that are scattered vertically (on different architectural layers) and horizontally (at different providers) into a single audit case (refer also to objective 3 in Section 1). This shows the extensibility and complexity of audits that can be performed by AAS.

The steps for setting up the audit task and triggering the violation are executed automatically on the demo deployment using fully automated shell scripts.

The timing information is specific to the runs that were used to generate the data that is presented here. Due to the nature of the system and external factors such as host, network load and latency these are expected to differ slightly

between runs. The timing information are derived from AAS's internal logging mechanisms that is used to record events in the system during its runtime for debug purposes. The original logs used for this evaluation are available in the Appendix C in a mostly raw state as they have been produced by AAS during the tests. Some information has been cut in order to reduce the overall volume of the logs. However, these cuts are clearly indicated and do not influence the validity of the logs in any way.

The logs generally are formatted as follows: *timestamp SEP event_type SEP raw_data*

The *timestamp* indicates the time and date at which the message was recorded by AAS's logging facility. This is not necessarily the same time as the occurrence of the actual event in the system. The *SEP* character is simply a designated separator to enable automated parsing of the log since it is considered a tracing and debugging facility of AAS. The occurrence time is typically included in the field *raw_data*. The *event_type* is any of the constant event types defined in Table 12.

Type	**Description**	**Logger**
audit_task_-created	Indicates details about an audit task that was created in AAS either via the dashboard or the API. The message contains the audit task type and configuration.	APM
spawning_-task_agents	Indicates details about the preparation and migration of agents required for an audit task.	AAC
evidence_-record_created	Indicates the collection and storage of an evidence record in the ES.	All collector agents
received_-apple_log	Indicates the collection of certain messages from A-PPL-E as evidence. Contains sub types "PII store message" and "PII delete message"	A-PPL-E collector agent
snapshot_-detected	Indicates the collection of snapshot history for a VM from OpenStack.	OpenStack collector agent
policy_-violation_-detected	Indicates the detection of a policy violation. Contains sub types for "data retention", "data location" and "intrusion" for the audit tasks	All evaluator agents

evidence_-record_for_-violation_-created	Indicates that a detected policy violation was stored as evidence in the ES	All evaluator agents
notification_-email_sent	Indicates that a notification has been sent via email alongside the message's content and its recipient.	E-mail notification agent
notification_-sent_to_IMT	Indicates that a notification has been sent to an incident management tool alongside the message's content.	IMT notification agent

Table 12: Logging Event Types in the Audit Agent System Internal Logging

The field *raw_data* contains any kind of data that provides context to the event type. This can be an evidence record for the evidence related event types or for instance the complete notification for the notification-related event types.

7.2.1 Example Audit Case 1: Intrusion Detection

The first audit case for evaluation is concerned with a simple method of detecting potentially unwanted data access on the application level that is either caused by an insecure API or by a compromised user account that holds sufficient privileges to cause a severe data breach. It is assumed that there is no direct access to a lower level representation of data (e.g., on a database or VM level), but all accesses happen over a service's public interface. An unusual access pattern is considered an indicator of a potential violation. AAS's goal is to detect such patterns during regular audits.

This is a single-provider scenario with respect to evidence sources considered, where the service is offered by *CP1* to consumers (see Figure 46 for reference).

With respect to the agent placement in the demo deployment, the audit task looks as follows:

- The A-PPL-E log collector agent runs at the AAS runtime environment on VM 2 owned by CP1.

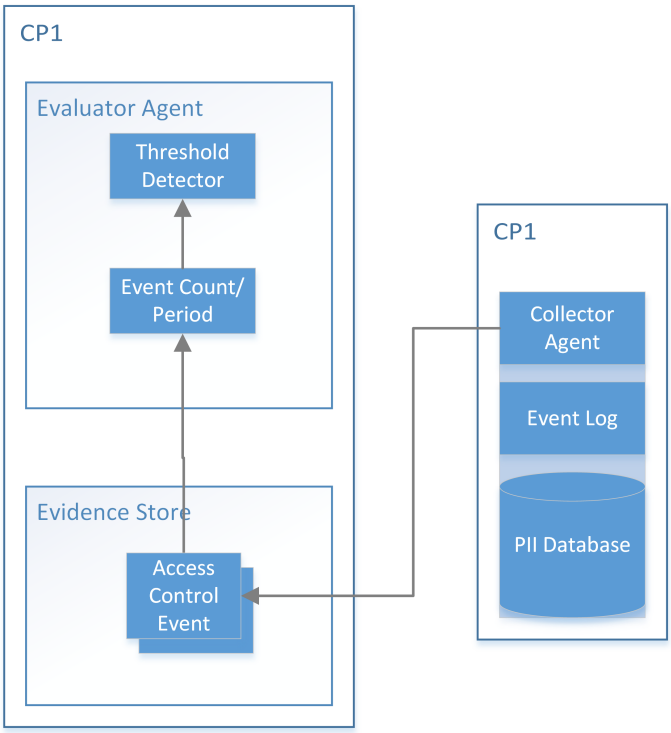

Figure 46: Intrusion Detection Audit Example

- The associated evaluator agent runs at the AAS runtime environment on VM 2 owned by CP1.
- The Incident Management Tool (IMT) notification agent runs at the AAS runtime environment on VM 2 owned by CP1.
- The evidence store runs at VM 1 owned by CP1.

Associated Policies

While none of the policy languages that AAS is aimed at supports defining rules such as "no data shall be leaked" explicitly, they can do this implicitly by defining authentication and authorization rules. However, there are cases, where an intrusion or data leak may happen although the accessing user was properly authenticated and authorized (e.g., when a user's access credentials were compromised).

This audit case is part of the custom audit tasks, that are originating from well-

known best practice catalogues (e.g., running a system for intrusion detection), such as those defined in IT-Grundschutz [174]. Language integration for access control rules and the integration in AAS was discussed in Section 6.2.4.

Evidence Sources

The sole evidence source in this scenario is the access control logging performed by the service that CP1 offers. In that service, access to PII is logged on the application level using A-PPL-E. This includes granted as well as denied access requests with additional event details such as timestamps, the object identifier and the accessing subject. The log events emitted by the access control system are picked-up by the evidence collector. A typical event log produced by A-PPL-E is depicted in Listing 7.1. It contains the owner of the PII object, the subject that is requesting access, the timestamp and the operation for which access is requested. The structure of the log is specific to A-PPL-E.

```
1 Message: Date:Thu Mar 03 15:39:39 CET 2016 Event associated with personal data
    ↪ 'Country' belonging to 'Panos', access attempt by subject 'Employee':
    ↪ Access permitted for, for action 'read'
2 piiAttributeName: Country
3 piiOwner: Panos
4 date: 2016-03-03 15:39:39.0
```

Listing 7.1: Intrusion Detection Audit Example – Access Control Log as collected from A-PPL-E by AAS

Evaluation Mechanism

The evaluation mechanism in this case can cover a wide range of complexity. In this demo scenario, a simple approach was chosen that merely counts the number of access control events that are issued by a single subject in a given period of time. Accordingly, a periodic audit was chosen that uses the same interval. Only *read* operations are considered and counted in *ReadOpCounter* (in a more complex case this could be extended for instance by *access denied* events,

but for simplicity reasons this was omitted in this example). If the counter exceeds the value of *Threshold* this means a policy violation. If a given threshold of events is passed in a period, AAS considers this to be a potential violation (e.g., a compromised account is used to "dump" the PII database).

Triggering Violations

To trigger a violation in the evaluation deployment, the following steps are executed:

1. The auditor creates an audit task that aims at detecting potential data breaches among events that, seen individually, do not indicate a violation.
2. The audit interval is set to two minutes by the auditor.
3. The audit task's filter is set to only consider read events on data sets owned by a data subject.
4. A threshold of *5 operations per period* is set.
5. 20 read operations on a data set owned by the data subject are triggered in rapid succession using the service's API (i.e., A-PPL-E's getPii() interface).
6. AAS detects the violation (*20 operations exceed allowed 5*) and presents it as an audit result in the dashboard.

Validation by an Automated Test Case

Figure 47 depicts the timeline of events in the automated test case. A log of this test case is available in the Appendix C, Listing C.1. The log is complete in the sense that all events are recorded that happen during the given time frame. However, some redundant information was removed for the sake of readability (this is pointed out explicitly where applicable).

In the following, the order of events is described. Events that are denoted above the timeline in Figure 47, relate to actions that are internal to AAS, whereas

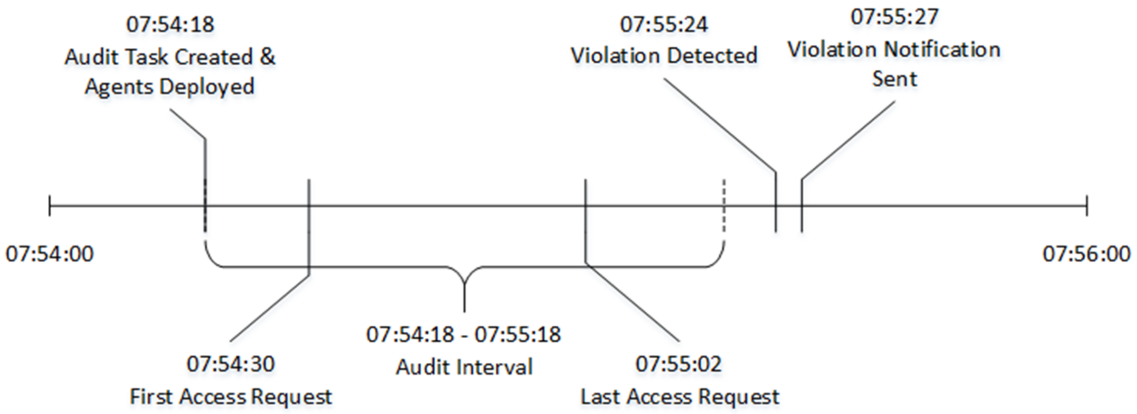

Figure 47: Intrusion Detection Audit Example – Timeline of Events

events that are denoted below the timeline indicate test case specific external events such as intentionally triggering a violation.

1. The relevant events in the first step are *audit_task_created* and *spawning_task_agents*. These events are artifacts of a new audit task being created in AAS. In this case, this happens via AAS's ReST interface[1]. Both events also reflect additional debugging information that was added to the task (e.g., the IP address of the target containers for the agents).

2. The audit task is setup to take into account access control events that occur in 1 minute intervals. Here the audit interval start at 07:54:18 and ends at 07:55:18 after which all events are analyzed.

3. Figure 47 marks the first and last access requests (at 07:54:30 and 07:55:02, respectively; 20 in total) that are being performed during the test case.

4. The *policy_violation_detected* and *evidence_record_created* events are produced by the evaluation agent that counts the access control events as previously described and generates an evidence record for the violation. An excerpt of the logged event (as collected from A-PPL-E) is presented in Listing 7.1.

[1] via HTTP GET: http://141.28.98.114/rest/setAuditTask?task=Intrusion%20attempt%20detection&container=1412898117&period=60000&threshold=5

5. The test case is concluded by the notification of another tool using their ReST interface and message format (as logged in event *notification_sent_to_IMT*).

It is important to note that there is a case where a violation may not be triggered when the access operations fall partially in two consecutive audit periods. However, this is a limitation of the weak detection algorithm. In a real-world scenario, this algorithm should be replaced with more sophisticated approaches that use more history data (i.e., events collected during previous periods) and more advanced evaluation algorithms to detect anomalies in data access patterns (e.g., ANN and more complex rule sets).

For this demo, the chosen algorithm was sufficient to demonstrate the feasibility of integrating access control logging as a vital source of evidence.

7.2.2 Example Audit Case 2: Data Location

The second evaluation audit case is concerned with data location restrictions put on the storage of PII. The focus is put on the underlying storage facility provided by CP2 (infrastructure) to CP1. The actual location of the virtual block storage device that is used for CP1's VM is transparent to CP1. Relocation of the device can happen at any time, if CP2 chooses to do so (e.g., based on load balancing algorithms that migrate VMs between data centers to achieve Quality of Service (QoS) goals). Transfer events can either be captured directly with a collector agent that monitor's OpenStack's internal message bus or by interfacing with another monitoring tool that provides similar functionality. It is assumed that migration events are detectable by those tools.

In the scenario depicted in Figure 48, the approach of interfacing with other, more specialized tools is presented to highlight the flexibility of AAS. The actual implementation of the scenario relies on the integration of a DTM [175] at the infrastructure provider that analyses transfer events according to the data location policy in effect and reports these events as well as detected violations to

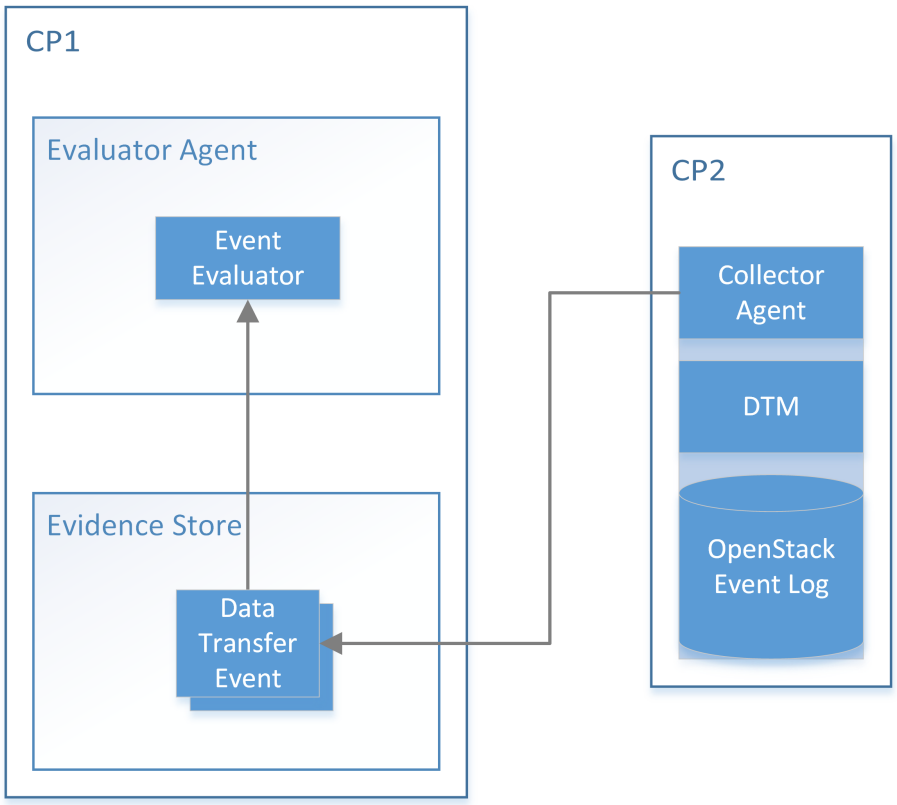

Figure 48: Data Location Audit Example – Overview

AAS. Data on the IaaS level can be "transferred within the infrastructure due to mechanisms such as snapshot, replication, instance migration etc." [175]. The DTM monitors OpenStack's API for these events, fills a data tracking knowledge base with them and in conjunction with a topology knowledge base detects data location violations caused by infrastructure level operations. Since this requires low-level access to the provider's infrastructure, this tool needs to be run by the provider and exposed to the customer by a secure interface.

The DTM used in this scenario is a tool that was developed independently from the work presented in this thesis and where no contribution other than the integration in AAS was made by the author. For the execution of this demo audit, static but realistic log data that was provided by the tool authors was used as evidence, since there was no access to a running instance of the DTM tool.

With respect to the agent placement in the demo deployment, the audit task

looks as follows:

- The DTM log collector agent runs at the AAS runtime environment on the OpenStack controller owned by CP2.
- The associated evaluator agent runs at the AAS runtime environment on the OpenStack controller owned by CP2.
- There is no notification agent.
- The evidence store runs at VM 1 owned by CP1.

Associated Policies

The most important input is the rule that defines allowed locations for the virtual machine and its storage. As previously described in Chapter 6, A-PPL is a policy language that supports the definition of data location rules. AAS provides an audit task based on the existence of a data location rule in the input policy. Language integration for access control rules and the integration in AAS was discussed in Section 6.2.2.

Evidence Sources

In this case, the DTM acts as the sole evidence source that generates and logs data transfer events as well as potential violations to AAS. However, in more complex cases, data transfer events are also generated on other layers, for example:

- Host layer: a virtual block devices is transferred to another location via S-FTP.
- PaaS layer: the application provides bulk transfer operations on the API level with logging of the requester's IP address (e.g., location identification via GeoIP).

- IaaS layer: same log information as on the PaaS layer but collected using web server logging (e.g., using Apache's access.log).

```
148712 INFO  com.sap.a4cloud.ruleengine.RuleEngine     - Potential violation
    ↪ detected: "Volume" name "EU Data vol.",  moved from original host in "
    ↪ Europe" to new host in "US"
```

Listing 7.2: Location Audit Example – Location Violation as Received from DTM by AAS

Listing 7.2 depicts an example of a violation that is reported by DTM. AAS acts as a receiver of such events using the DTM collector agent. DTM reports the transfer (detaching a volume from data center *Europe* to a VM in data center *US*). AAS picks up and stores these messages. Violation detection is performed by DTM.

Evaluation Mechanism

The evaluation mechanism in this example audit case is kept rather simple, due to the fact that the DTM tool already performs a verification of data transfers. All events that are collected from DTM are filtered for keywords that indicate a violation. If such an event is detected, the audit is failed and a violation produced. In the absence of such an event, the audit is passed.

Triggering Violations

The following steps are taken in AAS to setup an audit task and trigger a violation:

1. The auditor creates an audit task that aims at detecting potential data transfer violations by comparing the current location to the restriction defined in the A-PPL policy.

2. All checks are performed by the external tool DTM.

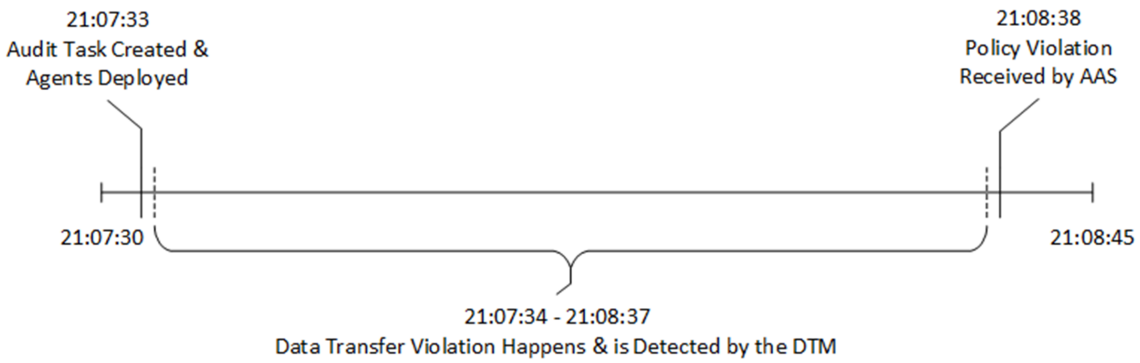

Figure 49: Data Location Audit Example – Timeline of Events

3. AAS continuously collects evidence from DTM (data transfer events and detected violations).

4. A data transfer event is triggered in OpenStack by either moving a VM to another zone or by detaching and attaching a block device to VMs in different zones.

5. AAS picks up violations (by keyword filtering) and presents them as an audit result in the dashboard.

Validation by an Automated Test Case

Figure 49 depicts the timeline of events in the test case. A log of this test case is available in the Appendix C, Listing C.2. The log is complete without any restrictions. No data was removed for the sake of readability due to the relative simplicity of the audit case and its data flows.

In the following, the order of events is described. Events that are denoted above the timeline in Figure 49 relate to actions that are internal to AAS, whereas events that are denoted below the timeline indicate test case specific external events such as intentionally triggering a violation.

1. The relevant events in the first step are *audit_task_created* and *spawning_task_agents*. These events are artifacts of a new audit task being cre-

ated in AAS. In this case, this happens via AAS's ReST interface[2]. Both events also reflect additional debugging information that was added to the task (e.g., the IP address of the target containers for the agents).

2. The audit task is setup to continuously take into account all incidents reported by the data monitoring tool. Therefore, an audit interval is omitted in Figure 47.

3. Figure 47 marks the beginning and end of a period, where a transfer violation is detected by the DTM tool.

4. AAS receives that violation notification from DTM via thespecialized DTM collection agent and stores it as evidence.

As long as no additional evidence sources are introduced in the task, AAS in this case mainly serves the purpose of documenting the timeline of events by archiving evidence and double-checking the data that is provided by DTM.

7.2.3 Example Audit Case 3: Data Retention

The third evaluation audit case, assumes the following scenario: In case of a breach of a service's database, sensitive information can be leaked. The database is hosted alongside CP1's service on a cloud infrastructure provided by CP2. Effectively, this means that the data is stored inside a virtual machine, to which CP1 has root / administrative access to. Since the cloud infrastructure provider personnel has administrative access to the underlying physical machine hosting the tenants virtual machines, copies of the data can be created. A data retention rule is applicable to the data stored in the PII database. This rule is enforced by CP1's service. However, there are many points where data can be duplicated that are beyond the direct control of CP1. The places where the data can duplicated are the following:

[2] via HTTP GET: http://141.28.98.114/rest/setAuditTask?task=DTMTAuditTask&container=1412898110&intervall=60000&dataTrackLocations=Europe,Africa

- On the host layer by the cloud infrastructure provider: due to the infrastructure provider being in full control over the physical computing resources, a low-level access to a customer's virtualized resources is always possible. A person with administration-level access rights to the underlying virtualization hosts or storage subsystem can always access and duplicate a virtual machine, regardless of the actual owner of the data that is contained inside it. Encryption of the virtual machine can be a suitable safeguard but is easily circumvented as the virtual machine must be decrypted for execution.

- On the virtual machine layer by the SaaS provider: the SaaS provider uses virtual resources (as in storage, network and machines) to host its service. Part of it is a database running on a virtual machine. A person with administration-level access rights to the underlying virtual machine (i.e., root credentials) can, at any time, copy the databases backing files (if file-level storage backend is used instead of a block-level backend).

- On the CMS layer by the SaaS provider: Similarly to the copying of the virtual machine image, clones and snapshots can duplicate data in a more controlled way (i.e., operations are logged, authentication and authorization is enforced on a higher level of abstraction than filesystems). A person with sufficient access rights to administration operations in the CMS can issue the creation of a clone or snapshot of a virtual disk image, which is the underlying or attached storage that is associated with a VM.

- On the application and database layer by the SaaS provider: On an even higher level of abstraction, data can also be copied by leveraging service-level interfaces or the database management system. A person with sufficient access rights to administration operations in the database management system can easily create a copy of the data by dumping (reading all records using SQL) the contents of tables or by exploiting web service APIs to extract data sets.

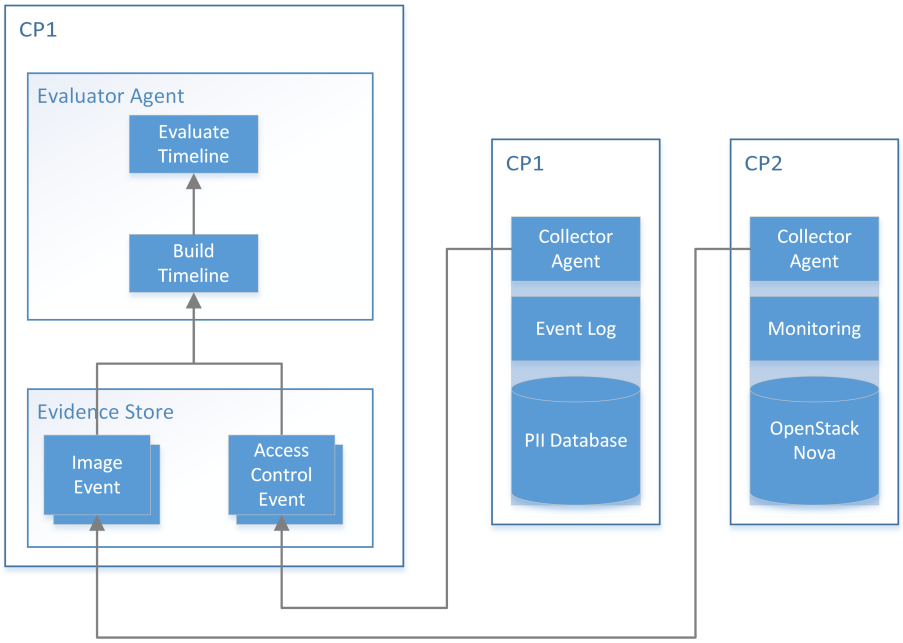

Figure 50: Data Retention Audit Example (Provider Chain) – Overview

Figure 50 depicts the implemented evaluation audit case, where evidence is collected from two different cloud providers and is combined to evaluate whether or not the data retention mechanisms that are incorporated into CP1's service are actually effective.

This case illustrates the combination of two different evidence sources at two different providers into a single audit task.

With respect to the agent placement in the demo deployment, the audit task looks as follows:

- The OpenStack Nova collector agent runs at the AAS runtime environment on the OpenStack controller owned by CP2.
- The A-PPL-E log collector agent runs at the AAS runtime environment on VM 2 owned by CP1.
- The associated evaluator agent runs at the AAS runtime environment on VM 2 owned by CP1.
- The IMT notification agent runs at the AAS runtime environment on VM 2

owned by CP1.

- The evidence store runs at VM 1 owned by CP1.

Associated Policies

The policy that is associated with the audit task is tightly coupled with data retention requirements. A maximum data retention period for a specific set of data can thereby stem from different origins. While there are legal obligations that dictate the deletion of data after a certain period of time, there may also be a need by the customer to protect its data by forcing the provider to verifiably delete data after a given period of time. Data retention obligations that are associated with a particular data owner can be represented in a machine-readable way using A-PPL. Language integration for data retention rules and the integration in AAS was discussed in Section 6.2.3.

Evidence Sources

The evidence sources that are relevant to this scenario are scattered across two different cloud providers. The information on whether or not a PII object was deleted by the service at CP1 due to data retention is provided by an operations log at that service. The information on lower-level duplicates (*VM snapshots* in particular) is provided by OpenStack owned by CP2. However, since CP1 is a tenant at CP2, it is provided access to that lower level information via the Nova service API as this is a public API that is commonly exposed. The evidence collection can therefore follow the remote API approach (as described in Section 5.4.4). However, the mobile collector approach works equally well.

To illustrate the potential complexity of this problem, Figure 51 shows an extension of this scenario, where additional evidence sources from both the host layer (i.e., the monitoring of low-level operations such as *copy* and *sftp transfer* on the VM backing file) and the IaaS layer (i.e., the monitoring of low-level operations

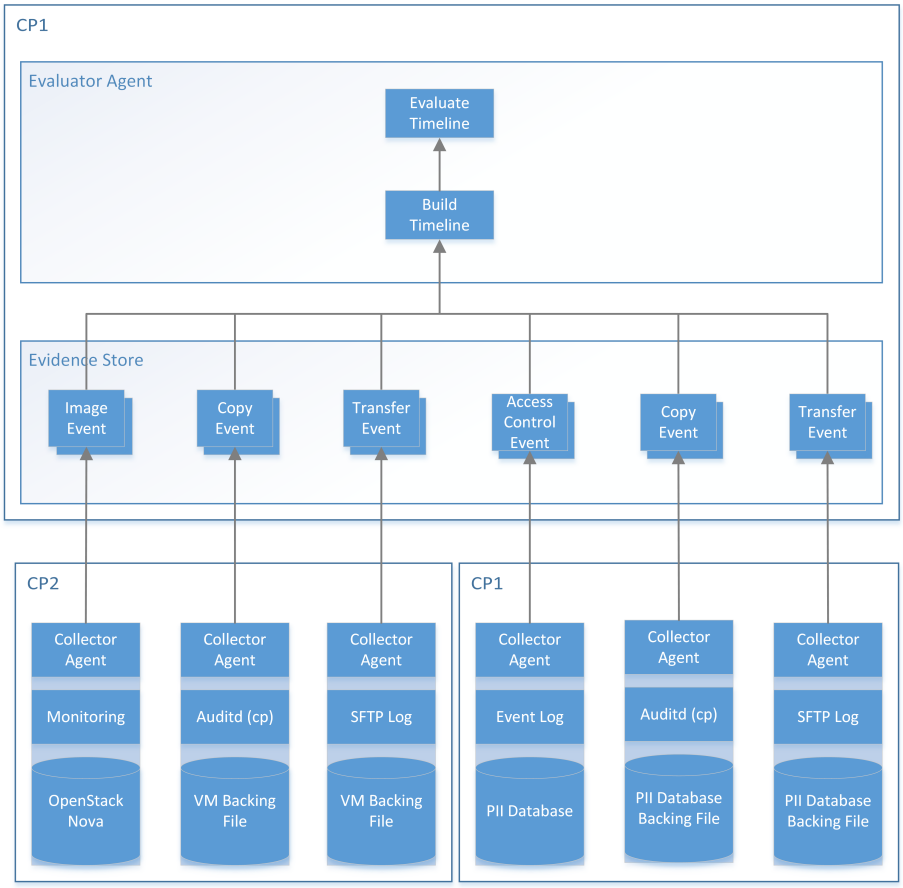

Figure 51: Data Retention Audit Example (Provider Chain) – Extension

such as *copy* and *sftp transfer* on the database (DB) backing file) are added.

Most of these additional evidence sources on the host level are not directly exposed to CP1. Therefore, CP2 has to provide the capability to deploy mobile collectors in their system. This can of course only be used if the cloud provider allows it.

Listing 7.3 depicts two events as they are collected by the A-PPL-E collector. In conjunction with the aforementioned snapshot information, data retention violations are detected.

```
1 Message: PII storedtype: policy admnistration
2 piiAttributeName: Country
3 piiOwner: Panos
4 date: 2016-03-02 12:04:27.0
5
```

```
Message: PII deleted type: policy enforcement
piiAttributeName: Country
piiOwner: Panos
date: 2016-03-02 12:06:28.0
```

Listing 7.3: Data Retention Audit Example – Logged Data Retention Enforcement Event

Evaluation Mechanism

The evaluation mechanism that is integrated in the evaluator agent of this audit task, follows two general steps:

1. Construct a timeline of events relevant to data retention.
2. Perform checks on timeline to uncover snapshots that hold copies of data.

The *timeline* of events is constructed based on events that are collected as evidence from the service and the CMS OpenStack. The service provides for each dataset the action, timestamp and owner, where an action follows CRUD (create, read, update and delete). For clarity of the demonstration, a filter was introduced to limit the number of events to only those belonging to a certain owner. Similar information is provided by the collector at OpenStack that regularly gathers information about the VM that holds the database with the datasets. This information includes the VM and snapshot identifiers, timestamp and action, where the only relevant action in this case is: snapshot. All events are ordered by time in order to construct a timeline of events for the analysis in the second step. The physical system clocks are synchronized using Network Time Protocol (NTP), while the hosted VMs share the system time with their physical hosts.

The *checks* that are performed in the second step follow the algorithm presented in Figure 1.

For a dataset the timestamps of the *create* and *delete* events are taken and the dataset lifetime is calculated. If the calculated value exceeds the maximum that

Algorithm 1 Evaluation Mechanism for Data Retention

$RetentionTime_{Dataset} \leftarrow Timestamp_{DatasetDelete} - Timestamp_{DatasetCreate}$
$RetentionTime_{Max} \leftarrow RetentionTime_{APPL}$
if $RetentionTime_{Dataset} > RetentionTime_{Max}$ **then**
  return Failed
end if
if $Timestamp_{SnapshotCreateEvent} \in RetentionTime_{Dataset}$ **then**
  if $Timestamp_{SnapshotDeleteEvent} \in RetentionTime_{Dataset}$ **then**
    return Passed
  else
    return Failed
  end if
else
  return Passed
end if

is defined in the policy, the test fails. Next, the information is combined with that of potential duplicates. If there is no snapshot event for the relevant VM during that period, the check is **passed**. Also, if during that lifetime a snapshot was created (i.e., there is a snapshot event in the timeline), it is checked whether or not that snapshot still exists. If the snapshot does not exist anymore, the check is **passed** else it is **failed**. A slightly condensed (due to space constraints non-essential information was removed) example of a violation evidence record, that includes all events that are associated with a retention violation is presented in the Appendix B.2.

Triggering Violations

The following steps are taken in AAS to setup an audit task and trigger a violation:

1. The auditor creates an audit task that aims at detecting potential data retention violations. The necessity of such an audit is derived from the A-PPL policy that defines a maximum data retention requirement.

2. The audit task is configured.

3. AAS continuously collects evidence from OpenStack and the PII enforce-

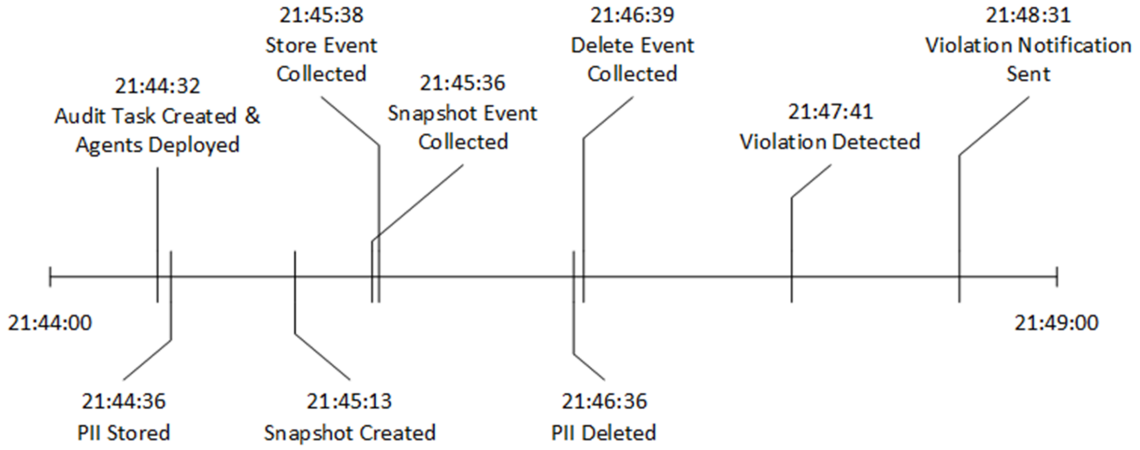

Figure 52: Data Retention Audit Example – Timeline of Events

ment tool (PII create and delete events) to construct trails of events during an audit.

4. A snapshot is created after a PII is created in the system but before its maximum retention period is reached.

5. AAS picks up violations and presents them as an audit result in the dashboard.

Validation by an Automated Test Case

Figure 52 depicts the timeline of events in the automated test case. A log of this test case is available in the Appendix C, Listing C.3. The log is complete in the sense that all events required for the depiction of the test case are recorded. However, the log was shortened and information was removed for the sake of readability (this is pointed out explicitly where required).

In the following, the order of events is described. Events that are denoted above the timeline in Figure 52 relate to actions that are internal to AAS, whereas events that are denoted below the timeline indicate test case specific external events such as intentionally triggering a violation. It needs to be noted, that evidence records do not necessarily need to be collected in a temporally ordered

way, but the order of events is reconstructed during the evaluation process.

1. The relevant events in the first step are *audit_task_created* and *spawning_task_agents*. These events are artifacts of a new audit task being created in AAS. In this case, this happens via AAS's ReST interface[3]. Both events also reflect additional debugging information that was added to the task (e.g., the IP address of the target containers for the agents).

2. The audit task is setup to continuously collect snapshot information for the given VM, see *snapshot_detected* event at 21:45:36 that contains the snapshot history provided by OpenStack Nova service and *received_apple_log_PII store message* at 21:45:38 (actual write event was at 21:44:36) and *received_apple_log_PII delete message* at 21:46:39 (actual delete event was at 21:46:36 i.e., after a retention period of 2 minutes) event pairs that are collected from the A-PPL-E instance running on that VM, which produces *store* and *delete* events for PII as depicted in Listing 7.3.

3. The policy violation is detected as *data_retention_policy_violation_detected* at 21:47:41, when all evidence is available at the evidence store and the evaluation is performed at the end of the audit interval.

4. Finally, an incident management tool is notified (see *notification_sent_to_IMT*).

This audit case highlights a more complex case, where evidence sources that are scattered across a cloud service provision chain are used to detect policy violations.

7.3 Scalability Evaluation

Since there are a vast number of evidence sources and therefore a potentially equal number of collectors, ensuring the scalability of the AAS collection and

[3] via HTTP GET:
http://141.28.98.114/rest/setAuditTask?task=dataRetentionPii&container=1412898117&vmName=CP1_Service&Username=Panos&osContainer=1412898110&intervall=60000

audit process, as well as the implementation is very important. Besides the obvious match of the adapter metaphor that enables the integration of heterogeneous evidence sources, the distributability property of MAS was a driving factor in the architecture design of this system. However, like any distributed system, MAS come with their own set of challenges regarding the scalability of systems that leverage that design. This has been discussed extensively in the literature before.

For instance, Rana et al. propose in [176] a categorization of performance metrics in *system parameters* and *coordination mechanisms*. Whereas system parameters are concerned with the agent management, their operations and agent transfer, the coordination mechanisms are concerned with parameters such as number of messages that agents exchange or the total number of agents involved. Many of the metrics of the system category, such as time to setup an agent or scheduling of agent activities on a host are more related to the implementation of the underlying agent framework.

The following discussion will however focus more on the scalability issues that are introduced by the evidence collection and auditing processes in AAS. The discussion is performed from three different perspectives that cover management aspects, performance aspects (that cover both system and coordination parameters), and presentation aspects.

7.3.1 Management Perspective

The management perspective focuses on organizational scalability aspects when performing an audit. For instance, in a cloud ecosystem, there are typically a lot of different tenants sharing the same resources. Also, the number of services provided by a cloud provider can be equally large. From an AAS point of view, this means that an auditor needs the tool to enable him to efficiently manage a large number of audit policies for large numbers of services and many customers. Furthermore, the preparation of AAS has to account for dynamically

changing infrastructure at the provider.

Risks

The scalability risks from management perspective in AAS are as follows:

- **Custom policies and agreements:** the audit tasks are generated on the basis of policies for which compliance should be checked. However, policy languages such as A-PPL intentionally move away from one-size fits all approaches towards customized policies, on which the service consuming party has some degree of choice or influence. This means that audit tasks may not necessarily be generalizable in a way that allows the check of a single rule, but variants of that rule for each active policy variant.
- **System preparation in a dynamic environment:** a central feature of the cloud is its dynamic nature in both the data center at the provider (e.g., extension storage, servers and networks) and the consumer (elasticity of virtualized resources). With this dynamic, evidence sources are continuously introduced and removed, which creates the need for AAS to grow or shrink with such changes.

Countermeasures

When considering custom policies, the need for customized evaluation parameters during an audit become apparent. The AAS system addresses this by offering the ability to extract tasks and parameters based on single policies. However, if policies differ between multiple parties, this approach is problematic since it requires the auditor to parse separate policy documents for each party and create widely similar audit tasks with minimal differences. This is not problematic, if the required audit tasks can be generated fully-automatically. But, as soon as manual input by the auditor is required, this approach fails. Therefore, additional tools are required that support the auditor during the extraction of audit

task from policies (i.e., making bulk operations more easy).

Due to the dynamic of the cloud, a manual approach of adding new evidence sources to the system is not feasible. Therefore, a lightweight runtime environment for the AAS agents was created to allow for easy deployment at new evidence sources. The runtime environment is packaged as a single compressed file that includes a simple configuration that has parameters for the core platform and a platform identifier. The platform identifier uniquely identifies the runtime environment at the evidence source, whereas the core platform identifier specifies to which AAS core instance it is supposed to connect to at initialization. This self-registration process allows for automated deployment of the required runtime environment alongside for example new virtualization servers using configuration management tools (e.g., Puppet [177]).

7.3.2 Performance Perspective

The performance perspective focuses on aspects that relate to the execution properties of AAS. For instance, with the growing number of audit tasks (e.g., due to more tenants or more complex policies to audit) the number of agents in the system increases. With that the number of evidence records that are collected and sent over the network, and the load produced by the evaluation of that evidence is increased as well.

Risks

The scalability risks from a performance perspective in AAS are as follows:

- **Data transfer volume:** amount of data being transferred over the network. This corresponds to the size of all messages that are being sent among agents over the network. This relates closely to the type of evidence source, the evidence requirements of the audit task and the granularity of single evidence data sets.

Case	Original Message	Evidence Record Size	Relative Overhead Introduced
1 Access Request	337 Byte	1826 Byte	82%
10 Access Requests	3370 Byte	7318 Byte	54%
100 Access Requests	33700 Byte	65181 Byte	48%

Table 13: Overhead Introduced by Formatting in Evidence Records

- **Message volume:** amount of evidence message transmissions over the network. This corresponds to the number of messages that are being exchanged among agents from their instantiation (migration-related transmissions), over control (for instance agent monitoring and control messages via ACL) and evidence collection (the transmission of evidence records and results).
- **Storage volume:** amount of storage required for evidence. This metric considers the size required for storing current as well as archived evidence in the evidence store.
- **Encryption overhead:** performance impact introduced by encryption and decryption. With an increasing number of messages, the encryption overhead increases as well.
- **Load increase:** amount of system load introduced by the agent management, evidence collection and evaluation.

For the evaluation with respect to those categories, three different cases have been considered that are all based on the aforementioned intrusion detection audit case (see Section 7.2.1). The actual detection of potential intrusions is not considered relevant but only the view on storage size and efficiency (associated overhead).

Three samples were collected based on the number of access requests that are logged and collected as evidence: case 1 collects a single access request, case 2

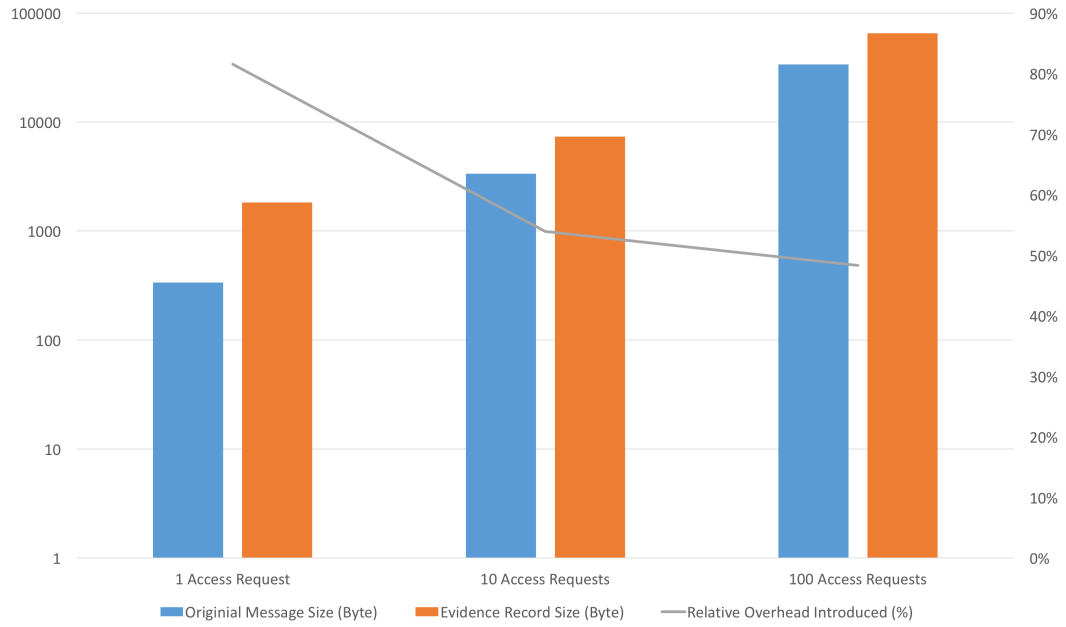

Figure 53: Overhead of Record Formatting in Evidence Storage

collects 10 and case 3 collects 100. The threshold limit is set to *1* and period to *1 minute*, which leads to a single violation evidence record being produced. Only the records produced by the evidence collector are considered for this analysis. However, the same conclusions apply to violations that are stored as evidence records by evaluator agents.

As can be seen in Figures 53 and 54, the overhead introduced into the system is quite significant. Table 13 depicts the actual sizes for the three different cases and highlights the amount of overhead introduced by using the XML-based evidence record format (see Section 12 for details). It compares the three cases based on the size of the original log message that is collected by the agent (blue), the same message but formatted in an evidence record (orange) and the relative overhead introduced by the amount of log messages contained in a single record. It is immediately obvious that an approach that buffers events on the input side and stores the whole package less frequently is more efficient that an approach where each event is stored as evidence separately.

Table 14 presents the amount of overhead introduced by using Insynd as an encryption scheme to securely store evidence records. It compares the size of the

Case	Encrypted Evidence Record Size	Total Size	Relative Overhead Introduced
1 Access Request	2058 Byte	5455 Byte	62%
10 Access Requests	9762 Byte	20863 Byte	53%
100 Access Requests	86910 Byte	175163 Byte	50%

Table 14: Overhead Introduced by Using Insynd

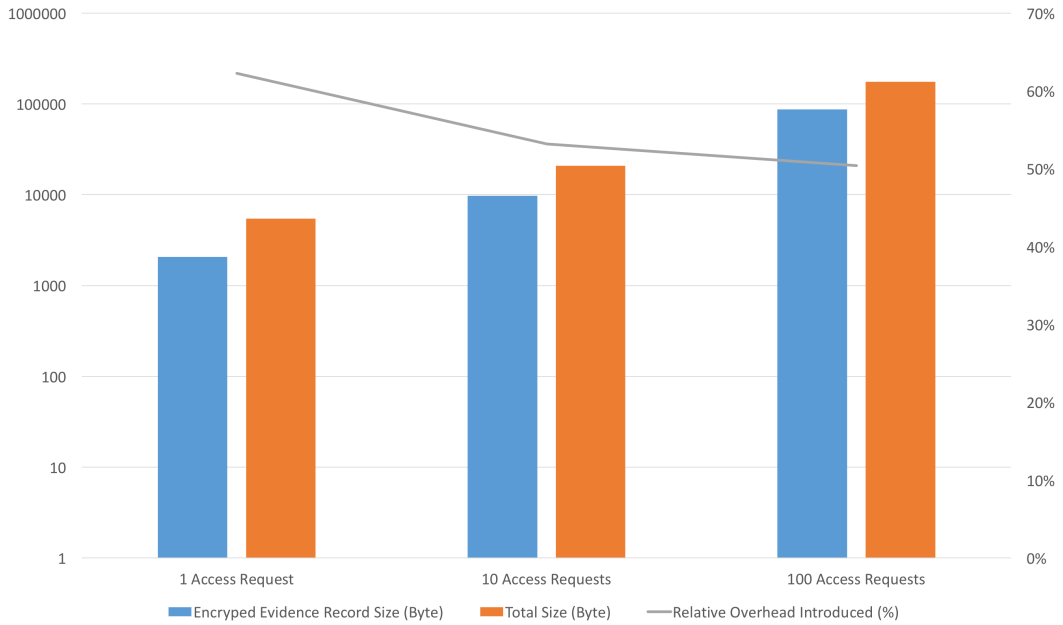

Figure 54: Overhead of Encryption in Evidence Storage

part of the Insynd data structure that contains the encrypted evidence record (blue), the total on disk size with additional fields (e.g., for key material, hashes etc.) and the relative overhead introduced by the scheme. While a similar conclusion can be made regarding the efficiency of buffering events and storing them in bulk, the effect is not quite as pronounced. However, when looking at absolute values such as the original message size from Table 13 and comparing it to the actual evidence on the output side (Total Size from Table 14) a vast gap can be observed: On the input side the log message is merely *337 Bytes* long, whereas the size of the message in the evidence store is *5455 Bytes*. An increase by a factor of more than *16*.

Countermeasures

To address the problem of congestion due to the number of messages that are being exchanged between the agents, central components have been removed as much as possible. All components of AAS are implemented as agents. While there are agents that are centralized (e.g., UI gateway, APM and AAC), agents that produce the most load on the network and servers are grouped into audit tasks and distributed (e.g., collector, evaluator and associated evidence store agents). This enables load balancing strategies for at least the evidence store agents and evaluators. A prerequisite is load monitoring that can be performed by those agents and reacted to accordingly.

Furthermore, the number of messages and their size is heavily influenced by the type of audit task (i.e., the evidence type that is required and the characteristics of the evidence source). As previously mentioned in Section 4.4, collectors can be developed and configured to perform filtering and aggregation on the evidence source, as long as this is acceptable with respect to the evidence properties and audit goals. Also, in some cases the evaluation can be moved to the *edge* of the network near the collector, which can be used to further distribute load (see Figure 23). This would help especially in cases where evaluation is very

compute intensive, for instance, where the amount of evidence sources that need to be combined in order to generate a complete statement on compliance need to considered, or where the evaluation algorithm itself is very resource intensive (e.g., in cases where artificial intelligence is used to detect potential violations). In cases where both compute and storage requirements are high, it may be required to move evidence storage and evaluation to dedicated zones in the cloud. This can go as far as deploying a separate, smaller *audit cloud* that facilitates the scalability of audit tasks. Such an audit cloud may as well by offered as a service by a third party, which can solve some of the trust issues previously described in this thesis.

7.3.3 Presentation Perspective

Presentation of audit state and results can quickly become challenging the more audit tasks collect evidence and produce results simultaneously.

Risks

The scalability risks from a presentation perspective in AAS are as follows:

- **Representation of compliance:** the presentation of audit results can quickly become overwhelming for an auditor considering the number of tasks that can run in parallel.
- **Working with evidence:** the number of evidence records associated with an audit task and result, is highly dependent on the nature of the task and grows with the number of active audit tasks.

Countermeasures

Basic countermeasures have been implemented in AAS to address both of these problems on a UI level. The auditor can filter violations in the dashboard by

periods and associated policies. This can be extended to provide even more fine-grained filtering capabilities to make the results more manageable.

The same filtering capabilities can be added to the presentation of evidence records. Furthermore, if an auditor reviews a violation and wants to access associated records, he can do this directly by following the links. More details on the UI implementation are presented in Section 5.6.

The potential countermeasures are very basic and are expected to not be sufficient in larger deployments with rapidly increasing numbers of audit tasks. New ways of visually presenting the state of compliance based on thousands of smaller checks need to be found. AAS's dashboard can be easily extended by additional visualization techniques such as for example heat maps.

7.4 Privacy Protection and Security Mechanisms Evaluation

In the following, security and privacy protection mechanisms are described according to the requirements presented in Section 4.6.2. Two perspectives on the system are utilized to categorize these mechanisms:

- Evidence perspective: this perspective covers all aspects that concern the handling of evidence from its collection at the data source, over the intermediate and long-term storage, to processing during an audit.

- Agent perspective: this perspective covers all aspects that concern security and trust issues that are introduced by the MAS-based architecture and indirectly affect the evidence handling.

7.4.1 Evidence Protection and Security

The protection of data can be split into three categories: *at-rest*, *in-transit* and *in-use*. The following views are presented alongside the audit process, where evidence is collected in the first step, stored in the second and finally processed and presented.

The example audit case 3 (see Section 7.2.3) is used to demonstrate the effectiveness of the integrated protection mechanisms (see also objective 4 in Section 1). The protection mechanisms used are thereby applicable to all other audit tasks in AAS as well.

The collector agent required for the above scenario communicates with our OpenStack CMS to gather evidence of the CMS behavior regarding virtual machine snapshots. The evaluator agent contains the logic for detecting snapshot violations. The collector agent is deployed at the CMS controller node and has access to OpenStack's ReSTful API. The evidence store is located on a separate, untrusted virtual machine. Now, the following steps are performed:

1. The collector agent opens a connection to the OpenStack ReST interface and requests a history of snapshot events for a tenant's virtual machine. HTTPS is used to secure the communication between the collector agent and the CMS. Since the policy only requires information about snapshots to be collected, the CMS agent limits evidence record generation to exactly that information, nothing more.

2. The collector agent sets up the receiver of the evidence according to the process depicted in Fig. 13 and sends the collected records to the evidence store. The communication channel is encrypted using HTTPS and the payload (evidence records) is encrypted with the receiving agent's public key.

3. The evaluator agent pulls records from the evidence store in regular intervals, analyses them and triggers a notification in case of a detected vio-

lation. The communication between the evaluator agent and the evidence store is again secured using HTTPS.

4. In the last step, evidence records are deleted because their retention limit has been reached. This is done by discarding the keys required for decryption.

A core technology for evidence protection in AAS is Insynd. In the following, the integration of Insynd is evaluated against the requirements described in Section 4.6. Furthermore, additional measures, that are not directly influenced by the integration of Insynd in AAS, are discussed.

Security Controls

At first, the evidence source needs to be considered. Depending on the type of evidence source, different protection measures and security controls need to be implemented to prevent the direct manipulation of the source itself. There is potential for manipulation of evidence at the earliest phase of evidence collection, by either manipulating the source to generate false data or by manipulating generated data, before it is picked-up by the collector agent (e.g., manipulating a log). AAS alone cannot solve that problem. Therefore additional security measures have to be implemented to protect evidence sources (e.g., system hardening, management network isolation, etc.). However, AAS works under the assumption that cloud providers (who effectively provide most of the evidence sources) act in good faith and do not intentionally manipulate AAS or its evidence sources, similarly to how it is not in their interest to manipulate already existing monitoring systems.

Addressed requirement: SP1 – Evidence source shall be protected.

Confidentiality

As soon as evidence is collected by an evidence collector agent (i.e., the data is still located at or near the evidence source) its confidentiality needs to be

ensured. It is thereby not sufficient to only consider the safe transfer (i.e., in-transit protection) of the evidence to the evidence store, but also while it is stored at it (i.e., at-rest protection).

For protecting both data in-transit and at-rest, a cryptographic scheme called Insynd [154, 155] is integrated into AAS. Insynd provides cryptographic properties to the underlying database that is used for storing records. A central property of Insynd is that it is always encrypting data using public-key cryptography. This enables AAS to generally store all collected evidence records encrypted. By encrypting the evidence store, compromising the privacy of cloud customer data that has been collected in the evidence collection processes, becomes almost impossible by attacking the evidence store directly. This goes as far as being able to safely outsource the evidence store to an untrusted third-party.

Additionally, it is good practice to also encrypt communication on the transport layer. Therefore, the communication between collector agent and evidence store, evidence store and evaluator agent, and also communication among core agents is performed over TLS-secured transport channels.

Addressed requirement: SP2 – Confidentiality of collected evidence should be provided.

Data Minimization

Data minimization as a guiding principle can be considered to be a prerequisite for privacy protection. AAS follows this principle in two ways. First, collector agents are always configured for a specific audit task, which is very limited in scope of what needs to be collected (i.e., the least amount of data that is required to audit a rule). Agents are never configured to arbitrarily collect data just in case it may prove useful later on, but are always limited to a specific source. This is in contrast to digital evidence collection that usually happens after the fact and considers also sources that *might* contain useful information. But, since in audits, the types of evidence sources that indicate compliance or violations are

usually known to the auditor, a more restrictive approach is chosen in AAS,in order to improve privacy protection by not being able to leak information that is not relevant to the audit task because it has never been collected in the first place.

Addressed requirement: SP3 – Data minimization regarding evidence collection should be considered.

Purpose Binding

Purpose binding of collected data is difficult to achieve and enforce. However, in AAS there is always one single goal that guides all data collection, which is the assertion of compliance with policies in a secure and automated way.

Agent collectors, after they have gathered data have only two options: send it to the evidence store or additionally send it to the evidence evaluator (as mentioned before in Section 5.5). Both recipients are fixed in the sense that all agents are part of a specific audit task. To enforce that evidence is only used for audit purposes in AAS, only the evaluator agent that is associated with the task is able to decrypt the collected evidence by making private key material owned by it (i.e., it is the sole *recipient* in terms of Insynd).

Addressed requirement: SP4 – Purpose binding of the collected evidence only to use in audits should be observed.

Data Retention

While today's storage isolation techniques can prevent some leakage risks of data from one tenant to another, secure deletion as it is common practice or even required in businesses, cannot be easily applied to the cloud. The reason for this is that in cloud computing, the precise location of a data object is usually not directly available, i.e., the actual storage medium used to store a particular block is unknown, making data deletion (e.g., secure wiping) hard. It becomes next to impossible to reliably delete a piece of information from all providers or

even tracking who is or was in possession of that data.

If data has been encrypted before storage, a reasonably safe way to ensure "deletion" is by discarding the key material required for decryption. Since evidence in AAS is encrypted on a per audit task basis at the collector and decrypted at an evaluator, this principle can be applied. That way, data retention rules can be applied to the evidence store in AAS by discarding the required key material for decryption of evidence records. Of course, this can only be done, if the data contained in the evidence store is not needed anymore (e.g., has been evaluated and is considered useless), or obviously if a maximum data retention time for that data has been reached.

Addressed requirement: SP5 – Observation of data retention requirements.

Integrity

The second evidence store property that is enabled by integrating Insynd is the detection of integrity failures. Insynd ensures that data cannot be tampered with, once it is stored in the evidence store. This property is guaranteed by Insynd's underlying data structure called Balloon [155]. More precisely, Insynd guarantees forward integrity as well as deletion detection. Both properties are highly desirable when handling evidence, since any manipulation of evidence, even its destruction, can be detected in the evidence store.

Addressed requirement: EH2 – Evidence must be tieable to the incident and may not be manipulated in any way and must be protected against any kind of tampering (intentionally and accidentally).

Completeness

The completeness of evidence is supported by the audit automation. A collector simply collects evidence according to its implementation and configuration. The implementation is performed by a programmer whereas the configuration is done by an auditor. If done correctly, the agent will collect evidence and feed it

into the evidence store. While Insynd as a cryptographic scheme cannot directly influence the evidence collection process, it can complement the evidence collection process by providing assurance that all data stored in the evidence store are made available as evidence, and not cherry-picked.

Addressed requirement: EH3 – Evidence must be viewpoint-agnostic and tell the whole story.

Reliability

The reliability of AAS is supported by the high degree of automation (i.e., implemented collection and analysis mechanisms can be reviewed in source code), integration of necessary mechanisms into the evidence collection process (such as Insynd) and the general operations logging as described in Section 7.4.2.

Addressed requirement: EH4 – There cannot be any doubts about the evidence collection process and its correctness.

7.4.2 Agent and Platform Protection and Security

From a MAS perspective, there are a couple of important privacy protection and security considerations to take into account. In this section, the main security and privacy protection problems and threats are described. The list of topics has been compiled based on literature review in the fields of agent security and distributed systems. The potential problems that come alongside MAS, have been considered in many publications such as by Wayner who stated back in 1995 that "... agents will not be able to work for very long in a serious environment if there can't be some way to extend trust across the network" [178]. For that problem, he proposed the introduction of Pretty Good Privacy (PGP)-based encryption into the system to allow encryption of data and verifiability of agent identities. Chess [179], Schelderup [180], Varadharajan [181] and Bürkle [182] worked on defining security models and identifying potential threats during the years. Additionally, Jansen et al. present in [183] and [184] a comprehensive re-

view of threats, security requirements and potential countermeasures for mobile agent systems. All models can be summarized and linked to the main categories that are presented in the following. Based on identification of potential security problems, suitable technical mechanisms were introduced in AAS. These are presented alongside the relevant categories.

Schelderup [180] mentions four elements of security in agent systems. The runtime environment, which is the Java Virtual Machine (JVM) in the case of AAS, acts as a protection layer between the host and JVM. Code signing can be used as an integrity-preserving mechanism that should reveal tampering with agent code. Host authentication is also required to identify hosts, when agents are mobile and migrate across the network. And lastly securing the channel over which agents travel across the network is required. As long as the underlying hosts are trusted, Schelderup considers these four mechanisms to be sufficient.

Authentication

The authenticity of mobile agents is of utmost importance in an open environment. Malicious agents that are injected into a system can potentially compromise the integrity and confidentiality of data in the system or the behaviour of the system itself. Since in AAS agents as data collectors play a central role, the importance of a reasonable security measure quickly becomes obvious. A MAS can be protected by protecting the environment it runs in for example by protecting a data center Local Area Network (LAN) with firewalls and therefore hindering external agents to join the platform. This approach breaks if the environment is opened or MASs that are operated by distinct independent organizations try to migrate agents between each other. In the latter case, the migration of agent from one environment to another without putting security controls into place or asserting the trust in the agent can be considered a code-injection that is potentially harmful for the receiving party. Such a code injection is generally not too different from how a computer virus acts [182].

Host authentication in AAS is achieved by the provision of the agent runtime environment by the provider. Most of the evidence sources in the cloud are under the control of the provider anyway (see a discussion on exceptions from that rule in Section 4.2).

With respect to agent authentication in JADE the Java Authentication and Authorization Service (JAAS)-based extension JADE Security AddOn [185] can be used. However, this extension does not authenticate components and agents on an individual level, but rather authenticates all of them en-block by associating them with the identity of the user who started the agent platform.

Authorization

The collection of evidence in AAS requires comprehensive access rights on the infrastructure. A malicious agent with comprehensive access rights may have access to unintended sources.

In AAS there are no restrictions on the access rights of the authenticated agent towards data that it collects. An agent is executed in the context of a JVM and is restricted to the same access rights. The access rights of the runtime environment should be configured to follow the principle of least privilege. Least privilege is thereby tightly coupled with the type of evidence source that a collector agent needs to gather from.

Since for many evidence sources, broad access rights (i.e., administrative level) are required to be able to collect data, no further restrictions are put into place. Authorization should therefore be limited on the runtime environment level to the lowest possible depending on the evidence source. For many cases, it is sufficient to grant read-only access (e.g., grant read access to system logs for the agent process).

Service Interference

The fact that AAS is running alongside the actual cloud service (and all its sub-

systems), faulty agents could have a negative impact on the service operation. This holds especially true for collector agents that are not strongly isolated from their evidence source. Considering a system log of a VM as an evidence source, the collector needs to operate on that same VM in order to fulfil its purpose.

AAS performs basic agent availability monitoring via the AAC. This could be extended to monitor the load that is introduced by the collectors in order to react before services are negatively impacted.

Denial of Service

Agent system disruption: as any networked computer system today, MASs in general and AAS in particular are susceptible to DoS attacks.

At this point, AAS does not implement specific countermeasures.

Traceability

Since AAS plays such an important role in the provision of evidence and the demonstration of a provider's policy compliance, traceability of events in the agent system itself becomes a desirable property.

In AAS traceability is achieved by internal logging of actions and states of the system (i.e., its agents). Tis is done by specialized internal logging agents, that interact with other agents and record their internal log messages. This approach improves transparency of AAS and its inner workings. This is a necessity for enabling audits and certification of AAS itself. Internal logging in AAS that enables traceability was introduced in Section 7.2. With this logging facility, the operator of the AAS can trace operations in AAS.

7.5 Adoption in Private and Public Cloud

Major obstacles can be expected when it comes to potential integration of this solution into public cloud services. As discussed, it is unlikely that cloud providers

adopt a system that provides so much insight into the inner workings of their service (as the AAS) unless there is a significant competetive advantage associated with it. As the cloud business matures and more workloads are moved to the cloud (for obvious reasons), customer requirements towards the cloud provider (especially regarding transparency and security) can be expected to rise. Cloud providers that proactively improve their services and provide means that make data processing practices more transparent could leverage this as an advantage. In the future, it would be a major achievement to have a public cloud provider integrate AAS's approaches and offer it as a service. As it stands now, AAS may be used as long as there full control over assets (VMs for executing AAS components and evidence sources), i.e., a private cloud. The AAS implementation presented in this scenario was developed in a private cloud scenario.

In private cloud scenarios, fewer obstacles are to be expected with respect to the adoption of a system like AAS. The main reason for this, is the level of control, the provider can exercise over the cloud infrastructure, platform and the auditing system, since he owns them all. Customers in a private cloud scenario are in a business environment typically part of the same organization (e.g., departments). Therefore, they are typically bound by the same internal directives and policies. Implementing a system that automatically and continuously inspects the compliance with those policies is just the next logical step to take.

7.6 Summary

In this chapter, the AAS was evaluated with respect to its functionality, scalability, and privacy and security protection mechanisms. The evaluation was performed based on a prototype implementation of several exemplary audit tasks. These audit tasks were created based on typical policy rules and compliance failure scenarios that are associated with those rules. Also, shortcomings in the current solution were highlighted.

It was shown, how AAS addresses security and privacy issues in a holistic way by leveraging encryption and integrity-preserving mechanisms end-to-end from the collection phase, throughout processing until presentation. The main contributions of this system are the automated collection and evaluation of evidences produced in a cloud service provision scenario. Other approaches from the forensics field lack in automation and usually rely on manual acquisition and evaluation of evidence. This is also true for conventional audits that rely on supporting tools but generally do not support continuous collection and evaluation but rely on periodic (e.g., yearly) repetition. AAS enables continuous auditing of cloud providers for compliance and accountability issues that may occur during operations. Furthermore, it was demonstrated how complex provider chain scenarios (a key problem of current cloud deployments) can be addressed by AAS. Other approaches for automation of audits or compliance checks do not support multi-provider scenarios, but are limited to smaller scopes (i.e., a provider's infrastructure or a tenant's virtual deployment).

Chapter 8

Conclusion and Future Work

CLOUD computing is an increasingly popular paradigm for outsourcing of service provision and data. The research project AAS focuses on privacy and accountability issues in today's cloud landscape and tries to address them by enabling privacy and accountability focused audits of cloud services. The following summarizes the major contributions and achievements of this project.

In Chapter 2, the current state of the cloud landscape, growth potential and privacy, security and accountability-related risks that are commonly associated with the adoption of cloud computing were presented. Additionally, fundamental concepts of cloud computing and accountability were discussed to provide a common understanding for the presentation of the rest of the thesis.

In Chapter 3, an overview of relevant literature and research projects was presented. Many of the core properties and principles of AAS originate from developments in the area of digital evidence and cloud forensics, especially with respect to evidence collection and processing, which is why a special focus has been put on the discussion of concepts and approaches related to these areas of research. Furthermore, related work in the areas of cloud audit and assurance were discussed. As a driving technology, AAS uses software agent technology, which is why a discussion of related work regarding the collection of data or

monitoring using software agents was included as well.

Chapter 4 started off with the introduction of a general scenario that is used throughout the remainder of the thesis to demonstrate examples. Based on the discussion of potential evidence sources that can be leveraged for compliance evaluation in an audit, the introduction and discussion of an adopted evidence collection process and the introduction of evidence processing, the general requirements of AAS were elicited. The most significant contribution of this chapter is the model of evidence sources specific to cloud computing as well as the application of evidence acquisition and processing techniques in the context of cloud audits. This allowed for building a more robust architecture that follows important data protection and security principles.

Based on the insights gained from Chapter 4 an architecture discussion of AAS is presented in Chapter 5. The architecture of AAS allows for the collection of evidence in dynamic, heterogeneous environments such as the cloud, where evidence is continuously produced at various sources. The reasoning behind choosing general characteristics of AAS, such as relying on software agent technology are discussed first. Following that, AAS is decomposed into modules that are described in detail. After the complete architecture was presented, AAS is extended to evidence collection in multi-provider scenarios with a discussion of several approaches which each have their own set of (dis-)advantages. Since scalability is an important property of cloud systems, different optimization strategies and approaches were discussed that improve scalability of AAS. The Chapter is closed with recommendations on audit result and evidence presentation aspects and a guide for extending AAS using its plugin-like approach. The major contribution of this chapter is the definition of a system that enables cloud providers to effectively and continuously audit their operations in order to assure compliance with accountability policies. This is a major step towards improving transparency of cloud services in order to improve trust in cloud computing on the consumer side. Furthermore, the contributions presented in this

chapter also consider cloud service provision scenarios to become increasingly complex by the aggregation and chaining of providers. An important contribution of the system was to consider security and privacy protection as early as possible in the design process.

At the start of any audit, there is a set of policies for which compliance is to be evaluated. Chapter 6 is focused on the integration of machine-readable languages that cover three different aspects: infrastructure, security controls and data handling / accountability. The integration of the A-PPL policy language that enables the definition of accountability rules and obligations is described, as well as the process of extracting audit tasks from such policies. Four different examples were thereby presented that relate to the concepts of notification, access control, data retention and data location.

The results of the AAS project are evaluated in Chapter 7. The evaluation thereby follows a scenario based approach covering three different perspectives: functionality, scalability, and privacy and security protection.

Based on the evaluation, shortcomings and achievements of AAS are discussed to finalize the thesis.

The research conducted as part of this thesis has met the objectives that have been defined in Chapter 1. The AAS implementation combines the research results in a system that enables cloud providers to continuously audit their operations for compliance with accountability audits. Furthermore, a wide range of dissemination activities has been performed, which includes multiple publications at well-recognized, international conferences, a journal publication as well as the presentations at workshops and project presentation sessions.

8.1 Limitations

While AAS has achieved most of the goals regarding enabling the automated, evidence-based auditing of cloud service chains against accountability-related

policies, there are still some open issues that require further research. In the following, a set of shortcomings in the current approach are presented alongside a discussion of suggested solutions.

8.1.1 Limitations of the Evidence Collection Approach

At the core of AAS is the evidence collection process. In order for AAS to be accepted as a useful tool and not being seen as yet another system that can potentially leak data if attacked, security must be thoroughly implemented. Section 7 presented a discussion of attack vectors against AAS and its components. Additionally, the problems of MAS systems with respect to security have been discussed. However, while AAS internally implements adequate protection mechanisms to secure evidence collection and handling, the resilience of the underlying agent system technology has to be improved.

Another important aspect is the trustworthiness of the data that is collected by AAS. In this research, the cloud provider is assumed to act in good faith and to not intentionally be manipulating evidence sources or the collector agents. This is reasonable under the assumption that the cloud provider uses AAS to enable auditors a deeper insight into the inner workings of its infrastructure to demonstrate that its acting according to the policy that were set. However, if that assumption changes and the cloud provider is seen as a potential adversary that wants to hide compliance failures, AAS cannot detect that. It is doubtful that this can be changed as long as the provider has full control over most of the evidence sources.

8.1.2 Limitations of the Evidence Processing and Evaluation Approach

In general, AAS supports an auditor by automating much of the monitoring and assurance process. However, this may lead to a false sense of security with re-

spect to being compliant. As demonstrated in the data retention example, there are cases where the number of potential policy compliance failures regarding a single rule can become enormously complex or even impossible to assess without doubt (e.g., proving that data is in fact unrecoverably deleted).

Therefore, the level of assurance is tightly coupled with the quality of the implemented audit tasks. The quality of the tasks is thereby dependent on the choice of evidence sources and evaluation mechanisms. This is a similar problem to the choice of depth in a conventional audit. A balance must be struck between level of assurance and effort. Effort in an AAS audit is thereby the complexity of the task and its implementation (e.g., number of evidence sources and choice of evaluation mechanism).

8.1.3 Limitation Regarding Applicability In Court

The applicability of evidence that has been collected by AAS in court is expected to be unlikely at first due to AAS not being a proven forensics tool. A system such as AAS has to undergo rigorous review, audits and certification or even standardization. Potential measures that could be taken are opening the source code of AAS's agents in order to improve transparency but also code certification (i.e., code signing of agents) are promising approaches.

8.2 Future Work

While this project presents important contributions especially with respect to automated acquisition of evidence for accountability audits of cloud providers, there are some areas that need to be addressed in future work:

Extension and Optimization

Future development based on AAS should focus on widening the scope of its applicability by supporting additional policy languages. Furthermore, a community-

driven approach towards supporting new incident detection scenarios and policy languages should be taken in order to further automate security, privacy and accountability audits.

Bigger Infrastructures

From a deployment perspective, AAS should be moved towards large cloud deployments in order to enable its evaluation in a more realistic environment.

Standardization and Certification

Facilitating the adoption of results that were presented in this thesis could be done by standardization. Formats, protocols and interfaces used in AAS and in the area of evidence acquisition, audits and accountability can only be successful if they are standardized. Furthermore, the believability and usefulness of evidence and audit results in court as they were presented in this thesis need to be improved.

Development of new forensic tools and techniques

While this work has advanced the field of evidence acquisition in cloud computing, further research should be conducted in this area in order to enable more thorough auditing and develop new techniques for evidence acquisition in cloud services that are becoming ever more complex.

Development of a Comprehensive Audit Description Language

Policy languages have been considered to be an important input to AAS but lack in comprehensiveness with respect to details on how to perform audits. A language that covers all aspects from the high-level rules to evidence sources and the required steps for evaluating evidence in an audit is needed. However, this is a difficult task since a variety of levels of abstraction of rules and obligations on the one hand need to be considered as well as supplementary information.

List of References

[1] Verizon Enterprise, *State of the market: enterprise cloud 2014*, http://cloud.verizon.com/enterprise-cloud-report, 2015. (visited on 05/23/2015).

[2] Forbes.com, LLC, *Market growth forecast for public it cloud services worldwide from 2011 to 2016*, http://www.statista.com/statistics/203578/global-forecast-of-cloud-computing-services-growth/, Mar. 2015.

[3] Gartner, *Gartner's 2014 hype cycle for emerging technologies maps the journey to digital business*, http://www.gartner.com/newsroom/id/2819918, 2014. (visited on 01/23/2016).

[4] Salesforce.com, Inc., *Salesforce.com*, https://www.salesforce.com/, 2016. (visited on 01/23/2016).

[5] Department for Business, Innovation & Skills and The Shareholder Executive, *2013 information security breaches survey*, https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/200455/bis-13-p184-2013-information-security-breaches-survey-technical-report.pdf, Apr. 2013.

[6] PwC, *Confidentiality of the data small businesses store on the internet in the united kingdom (uk) in 2014*, http://www.statista.com/statistics/318228/confidentiality-of-the-data-small-businesses-store-on-the-internet-in-the-uk/, 2014.

[7] ——, *Confidentiality of the data large organisations store on the internet in the united kingdom (uk) in 2014*, http://www.statista.com/statistics/318230/confidentiality-of-the-data-large-organisations-store-on-the-internet-in-the-uk/, 2014.

[8] ——, *Does your organisation have a security strategy in place for the following? (by region)*, http://www.statista.com/statistics/259103/information-security-strategies-in-place-in-companies/, 2013.

[9] *The cloud accountability project*, http://www.a4cloud.eu/, A4Cloud FP7 Project, 2016. (visited on 01/23/2016).

[10] S. Pearson and N. Wainwright, "An interdisciplinary approach to accountability for future internet service provision," in *International Journal of Trust Management in Computing and Communications*, S. M. Thampi, Ed., vol. 1, INDERScience Publishers, 2013, pp. 52–72.

[11] B. Dziminski, M. Felici, C. F. Gago, F. Gittler, T. Koulouris, R. Leenes, J. Luna, M. Niezen, D. Nunez, A. Pannetrat, S. Pearson, J.-C. Royer, D. Stefanatou, and V. Tountopoulos, *D:c2.1 - report detailing conceptual framework*, http://www.a4cloud.eu/sites/default/files/D32.1ConceptualFramework.pdf, 2014.

[12] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf, *Cloud computing reference architecture*, http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505, NIST, 2011.

[13] P. Mell and T. Grance, "The nist definition of cloud computing," National Institute of Standards and Technology, Information Technology Laboratory, Tech. Rep., 2011.

[14] OpenStack Project, *Openstack open source cloud computing software*, http://www.openstack.org/, 2016. (visited on 01/23/2016).

[15] OpenNebula Project, *Opennebula - flexible enterprise cloud made simple*, http://opennebula.org/, 2016.

[16] *Owncloud.org*, https://owncloud.org/, ownCloud Project, 2016. (visited on 01/23/2016).

[17] Amazon Web Services, Inc., *Amazon web services (aws)*, https://aws.amazon.com/de/, 2015.

[18] Google, *Google cloud platform - app engine*, https://cloud.google.com/appengine/, 2015.

[19] ——, *Google docs*, https://www.google.de/intl/en/docs/about/, 2016. (visited on 01/23/2016).

[20] RZV, *Studicloud project*, https://studicloud.hs-furtwangen.de/project/en/, 2016. (visited on 01/23/2016).

[21] RightScale, Inc., *State of the cloud report 2015*, http://www.rightscale.com/lp/2015-state-of-the-cloud-report?campaign=701700000012UP6. (visited on 01/23/2016).

[22] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010. [Online]. Available: http://doi.acm.org/10.1145/1721654.1721672.

[23] OASIS, *The charter for the oasis privacy management reference model technical committee*, https://www.oasis-open.org/committees/pmrm/charter.php, 2016. (visited on 01/23/2016).

[24] Cloud Security Alliance (CSA), *Top threats to cloud computing v1.0*, https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf, 2010. (visited on 01/23/2016).

[25] ——, *Top threats to cloud computing survey results update 2012*, https://downloads.cloudsecurityalliance.org/initiatives/top_threats/Top_Threats_Cloud_Computing_Survey_2012.pdf, 2013. (visited on 01/23/2016).

[26] ——, *The notorious nine - cloud computing top threats in 2013*, https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf, 2013. (visited on 01/23/2016).

[27] T. Haeberlen, L. Dupre, D. Catteddu, and G. Hogben, "Cloud computing: Benefits, risks and recommendations for information security," enisa - European Network and Information Security Agency, Tech. Rep., 2012.

[28] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS '09, Chicago, Illinois, USA: ACM, 2009, pp. 199–212. [Online]. Available: http://doi.acm.org/10.1145/1653662.1653687.

[29] H. Eken, "Security threats and solutions in cloud computing," *2013 World Congress on Internet Security, WorldCIS 2013*, pp. 139–143, 2013. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-84898493875%7B%5C&%7DpartnerID=40%7B%5C&%7Dmd5=66b5fc08c6113ab1e1e90f225b6e9e76.

[30] S. H. Albakri, B. Shanmgam, G. N. Samy, N. B. Idris, and A. Ahmed, "A case study for the cloud computing security threats in a governmental organization," *Computer, Communications, and Control Technology (I4CT), 2014 International Conference*, no. I4ct, pp. 452–457, 2014. [Online]. Available: http://ieeexplore.ieee.org.libproxy.utem.edu.my/xpls/icp.jsp?arnumber=6914225.

[31] ISO, *Iso 27005:2011 - information technology – security techniques – information security risk management*, http://www.iso.org/iso/catalogue_detail?csnumber=56742. (visited on 01/23/2016).

[32] OWASP, *The owasp top 10*, http://owasptop10.googlecode.com/files/OWASPTop10-2013.pdf. (visited on 01/23/2016).

[33] S. Pearson, "Toward accountability in the cloud," *Internet Computing, IEEE*, vol. 15, no. 4, pp. 64–69, Jul. 2011.

[34] W. Jansen and T. Grance, "Sp 800-144. guidelines on security and privacy in public cloud computing," Gaithersburg, MD, United States, Tech. Rep., 2011.

[35] Economic Cooperation and Development (OECD), *Privacy guidelines*, http://www.oecd.org/document/18/0,3746,en_2649_34255_1815186_1_1_1_1,00&&en-USS_01DBC.html. (visited on 01/23/2016).

[36] European Parliament and the Council of the European Union, "Directive 95/46/ec of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data," *Official Journal of the European Union*, vol. L 281, pp. 0031–0050, 1995.

[37] German Federal Data Protection Act, *German federal data protection act*, http://www.bfdi.bund.de/EN/DataProtectionActs/Artikel/BDSG_idFv01092009.pdf?__blob=publicationFile. (visited on 01/23/2016).

[38] European Commission, "Proposal for a regulation of the european parliament and of the council on the protection of individuals with regard to the processing of personal data and on the free movement of such data (general data protection regulation)," vol. 0011, 2012. [Online]. Available: http://ec.europa.eu/justice/data-protection/document/review2012/com_2012_11_en.pdf.

[39] D. J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigenbaum, J. Hendler, and G. J. Sussman, "Information accountability," *Commun. ACM*, vol. 51, no. 6, pp. 82–87, Jun. 2008. [Online]. Available: http://doi.acm.org/10.1145/1349026.1349043.

[40] A. Haeberlen, "A case for the accountable cloud," in *Proceedings of the 3rd ACM SIGOPS International Workshop on Large-Scale Distributed Systems and Middleware (LADIS'09)*, Big Sky, MT, Oct. 2009.

[41] Centre for Information Policy Leadership as Secretariat to the Galway Project, *Data protection accountability: The essential elements - a document for discussion*, http://www.informationpolicycentre.com/files/Uploads/Documents/Centre/Centre_Accountability_Compendium.pdf, 2009. (visited on 01/23/2016).

[42] S. Sundareswaran, A. C. Squicciarini, and D. Lin, "Ensuring distributed accountability for data sharing in the cloud," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 4, pp. 555–567, 2012.

[43] Scientific Working Groups on Digital Evidence and Imaging Technology, *Swgde and swgit digital & multimedia evidence glossary*, https://www.swgde.org/documents/Current%20Documents/2015-05-27%20SWGDE-SWGIT%20Glossary%20v2.8, 2015. (visited on 01/23/2016).

[44] U. D. of Justice - Technical Working Group for Electronic Crime Scene Investigation, "Electronic crime scene investigation: a guide for first responders," Tech. Rep., 2001, p. 93. [Online]. Available: http://www.iwar.org.uk/ecoespionage/resources/cybercrime/ecrime-scene-investigation.pdf.

[45] E. H. Holder, L. O. Robinson, and K. Rose, "Electronic crime scene investigation: an on-the-scene reference for first responders," National Institute of Justice, Tech. Rep., 2009.

[46] K. Kent, S. Chevalier, T. Grance, and H. Dang, "Guide to integrating forensic techniques into incident response," National Institute of Standards and Technology, Computer Security Division, Tech. Rep., 2006.

[47] C. M. Redfield and H. Date, "Gringotts: Securing data for digital evidence," in *Security and Privacy Workshops (SPW), 2014 IEEE*, May 2014, pp. 10–17.

[48] U. P. R. Brandner and T. Gondrom, *Evidence record syntax (ers)*, http://tools.ietf.org/html/rfc4998, 2007. (visited on 01/23/2016).

[49] R. Zhang, Z. Li, Y. Yang, and Z. Li, "An efficient massive evidence storage and retrieval scheme in encrypted database," in *Information and Network Security (ICINS 2013), 2013 International Conference on*, Nov. 2013, pp. 1–6.

[50] A. Gupta, "Privacy preserving efficient digital forensic investigation framework," in *Contemporary Computing (IC3), 2013 Sixth International Conference on*, Aug. 2013, pp. 387–392.

[51] S. Saleem, O. Popov, and R. Dahman, "Evaluation of security methods for ensuring the integrity of digital evidence," in *Innovations in Information Technology (IIT), 2011 International Conference on*, Apr. 2011, pp. 220–225.

[52] B. Schatz and A. J. Clark, "An open architecture for digital evidence integration," in *AusCERT Asia Pacific Information Technology Security Conference : Refereed R&D Stream*, A. J. Clark, M. McPherson, and G. M. Mohay, Eds., Gold Coast, Queensland: University of Queensland, May 2006, pp. 15–29. [Online]. Available: http://eprints.qut.edu.au/21119/.

[53] National Institute of Standards and Technology (NIST), *Nist cloud computing forensic science challenges*, http://csrc.nist.gov/publications/drafts/nistir-8006/draft_nistir_8006.pdf, 2014. (visited on 01/23/2016).

[54] G. Grispos, T. Storer, and W. B. Glisson, "Calm before the storm: the challenges of cloud computing in digital forensics," in *International Journal of Digital Crime and Forensics (IJDCF)*, vol. 4, Mar. 2012, pp. 28–48.

[55] T. Garfinkel and M. Rosenblum, "A virtual machine introspection based architecture for intrusion detection," in *Proc. Network and Distributed Systems Security Symposium*, Feb. 2003.

[56] A. Haeberlen, P. Aditya, R. Rodrigues, and P. Druschel, "Accountable virtual machines," in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, ser. OSDI'10, Vancouver, BC,

Canada: USENIX Association, 2010, pp. 1–16. [Online]. Available: http://dl.acm.org/citation.cfm?id=1924943.1924952.

[57] T. Y. Win, H. Tianfield, Q. Mair, T. A. Said, and O. F. Rana, "Virtual machine introspection," *Proceedings of the 7th International Conference on Security of Information and Networks*, 405:405–405:410, 2014. [Online]. Available: http://doi.acm.org/10.1145/2659651.2659710.

[58] G. W. Dunlap, S. T. King, S. Cinar, M. A. Basrai, and P. M. Chen, "Revirt: Enabling intrusion analysis through virtual-machine logging and replay," in *In Proceedings of the 2002 Symposium on Operating Systems Design and Implementation (OSDI)*, 2002, pp. 211–224.

[59] M. Carbone, M. Conover, B. Montague, and W. Lee, "Secure and robust monitoring of virtual machines through guest-assisted introspection," in *RAID*, 2012, pp. 22–41.

[60] LibVMI Project, *Libvmi - virtual machine introspection*, http://libvmi.com/, 2016. (visited on 01/23/2016).

[61] The Xen Project, *Xen hypervisor*, http://www.xenproject.org/, 2016. (visited on 01/23/2016).

[62] KVM Project, *Kvm hypervisor*, http://www.linux-kvm.org/, 2016. (visited on 01/23/2016).

[63] QEMU Project, *Qemu processor emulator*, http://www.qemu.org, 2016. (visited on 01/23/2016).

[64] The Volatility Foundation, *Volatility*, http://www.volatilityfoundation.org/, 2016. (visited on 01/23/2016).

[65] S. Zawoad, A. K. Dutta, and R. Hasan, "Seclaas: Secure logging-as-a-service for cloud forensics," in *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ser. ASIA CCS '13, Hangzhou, China: ACM, 2013, pp. 219–230. [Online]. Available: http://doi.acm.org/10.1145/2484313.2484342.

[66] J. Dykstra, "Seizing electronic evidence from cloud computing environments," in *Cybercrime and Cloud Forensics: Applications for Investigation Processes*, K. Ruan, Ed., Hershey, PA: Information Science, 2013, pp. 156–185.

[67] O. Q. Zhang, M. Kirchberg, R. K. L. Ko, and B. S. Lee, "How to track your data: The case for cloud computing provenance," HP Labs, Tech. Rep., 2012.

[68] R. Lu, X. Lin, X. Liang, and X. S. Shen, "Secure provenance: The essential of bread and butter of data forensics in cloud computing," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS '10)*, ACM, New York, NY, USA, 2010, pp. 282–292.

[69] A. Flaglien, A. Mallasvik, M. Mustorp, and A. Årnes, "Storage and exchange formats for digital evidence," *Digital Investigation*, vol. 8, no. 2, pp. 122–128, 2011.

[70] ISO, *Iso 27001:2013 - information technology – security techniques – information security management systems – requirements*, http://www.iso.org/iso/catalogue_detail?csnumber=42103. (visited on 01/23/2016).

[71] ——, *Iso 27002:2013 - information technology – security techniques – code of practice for information security controls*, http://www.iso.org/iso/catalogue_detail?csnumber=50297. (visited on 01/23/2016).

[72] Information Systems Audit and Control Association, *Control objectives for information and related technology (cobit)*, http://www.isaca.org/cobit/. (visited on 01/23/2016).

[73] *Payment card industry data security standard (pci-dss)*, https://www.pcisecuritystandards.org/. (visited on 01/23/2016).

[74] M. Azraoui, K. Elkhiyaoui, M. Önen, K. Bernsmed, A. Santana De Oliveira, and J. Sendor, "A-PPL: An accountability policy language," in *DPM 2014, 9th International Workshop on Data Privacy Management, September 10,*

2014, Wroclaw, Poland, W, Sep. 2014. [Online]. Available: http://www.eurecom.fr/publication/4381.

[75] *Federal risk and authorization program*, http://www.fedramp.gov. (visited on 01/23/2016).

[76] National Institute of Standards and Technology (NIST), *Guidelines on security and privacy in public cloud computing*, http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144.pdf, 2011.

[77] Cloud Security Alliance (CSA), *Cloud controls matrix*, https://cloudsecurityalliance.org/research/ccm/. (visited on 01/23/2016).

[78] ——, *Consensus assessments initiative questionnaire*, https://cloudsecurityalliance.org/research/cai/. (visited on 01/23/2016).

[79] ——, *Security, trust & assurance registry*, https://cloudsecurityalliance.org/star/. (visited on 01/23/2016).

[80] EuroCloud Deutschand_eco e.V., *Eurocloud star audit*, https://staraudit.org/. (visited on 01/23/2016).

[81] Distributed Management Task Force, Inc. (DMTF), *Cloud auditing data federation (cadf) - data format and interface definitions specification*, http://www.dmtf.org/sites/default/files/standards/documents/DSP0262_1.0.0.pdf, 2014. (visited on 01/23/2016).

[82] OpenStack Project, *Pycadf developer documentation*, http://docs.openstack.org/developer/pycadf/, 2016.

[83] Cloud Security Alliance (CSA), *Cloud trust protocol*, https://cloudsecurityalliance.org/research/ctp. (visited on 01/23/2016).

[84] Amazon, Inc., *Aws cloudtrail*, http://aws.amazon.com/cloudtrail/, 2016. (visited on 01/23/2016).

[85] Amazon, *Introduction to auditing the use of aws*, https://d0.awsstatic.com/whitepapers/compliance/AWS_Auditing_Security_Checklist.pdf, 2015. (visited on 01/23/2016).

[86] S. Worku, Z. Ting, and Q. Zhi-Guang, "Survey on cloud data integrity proof techniques," in *Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on*, 2012, pp. 85–91.

[87] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proceedings of the 29th conference on Information communications*, ser. INFOCOM'10, San Diego, California, USA: IEEE Press, 2010, pp. 525–533. [Online]. Available: http://dl.acm.org/citation.cfm?id=1833515.1833620.

[88] C. Wang, K. Ren, W. Lou, and J. Li, "Toward publicly auditable secure cloud data storage services," *Network, IEEE*, vol. 24, no. 4, pp. 19–24, 2010.

[89] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, *Provable data possession at untrusted stores*, Cryptology ePrint Archive, Report 2007/202, http://eprint.iacr.org/, 2007.

[90] A. Juels and B. S. Kaliski Jr., "Pors: Proofs of retrievability for large files," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS '07, Alexandria, Virginia, USA: ACM, 2007, pp. 584–597. [Online]. Available: http://doi.acm.org/10.1145/1315245.1315317.

[91] C. Stern, P. Adelt, V. Krummel, and M. Ackermann, "Reliable evidence of data integrity from an untrusted storage service," in *Networking and Services, 2008. ICNS 2008. Fourth International Conference on*, Mar. 2008, pp. 24–29.

[92] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: Theory and implementation," in *Proceedings of the 2009 ACM workshop on Cloud computing security*, ser. CCSW '09, Chicago, Illinois, USA: ACM, 2009, pp. 43–54.

[93] M. Azraoui, K. Elkhiyaoui, R. Molva, and M. Önen, "Stealthguard: Proofs of retrievability with hidden watchdogs," English, in *Computer Security -*

ESORICS 2014, ser. Lecture Notes in Computer Science, M. Kutyłowski and J. Vaidya, Eds., vol. 8712, Springer International Publishing, 2014, pp. 239–256.

[94] Y. Zhu, G.-J. Ahn, H. Hu, S. Yau, H. An, and C.-J. Hu, "Dynamic audit services for outsourced storages in clouds," *Services Computing, IEEE Transactions on*, vol. 6, no. 2, pp. 227–238, Apr. 2013.

[95] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, 2012, pp. 295–302.

[96] F. Dölitzscher, C. Reich, M. Knahl, A. Passfall, and N. Clarke, "An agent based business aware incident detection system for cloud environments," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 1, no. 1, p. 9, 2012.

[97] F. Dölitzscher, T. Rübsamen, T. Karbe, M. Knahl, and C. Reich, "Sun behind clouds - on automatic cloud security audits and a cloud audit policy language," *International Journal On Advances in Networks and Services*, vol. 6, no. 1, pp. 1–16, 2013.

[98] J. Ryoo, S. Rizvi, W. Aiken, and J. Kissell, "Cloud security auditing: Challenges and emerging approaches," *Security Privacy, IEEE*, vol. 12, no. 6, pp. 68–74, Nov. 2014.

[99] P. Massonet, S. Naqvi, C. Ponsard, J. Latanicki, B. Rochwerger, and M. Villari, "A monitoring and audit logging architecture for data location compliance in federated cloud infrastructures," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, May 2011, pp. 1510–1517.

[100] R. Marty, "Cloud application logging for forensics," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, ser. SAC '11, TaiChung, Taiwan: ACM, 2011, pp. 178–184. [Online]. Available: http://doi.acm.org/10.1145/1982185.1982226.

[101] J. S. Park, E. Spetka, H. Rasheed, P. Ratazzi, and K. J. Han, "Near-real-time cloud auditing for rapid response," in *Proceedings of the 2012 26th International Conference on Advanced Information Networking and Applications Workshops*, ser. WAINA '12, Washington, DC, USA: IEEE Computer Society, 2012, pp. 1252–1257.

[102] Z. Chen and J. Yoon, "It auditing to assure a secure cloud computing," in *Services (SERVICES-1), 2010 6th World Congress on*, 2010, pp. 253–259.

[103] N. Guts, C. Fournet, and F. Zappa Nardelli, "Reliable evidence: Auditability by typing," English, in *Computer Security – ESORICS 2009*, ser. Lecture Notes in Computer Science, M. Backes and P. Ning, Eds., vol. 5789, Springer Berlin Heidelberg, 2009, pp. 168–183. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04444-1_11.

[104] Nagios Enterprises, LLC, *Nagios*, http://www.nagios.org/, 2016. (visited on 01/23/2016).

[105] K. Kavanagh, M. Nicolett, and O. Rochford, *Magic quadrant for security information and event management*, https://scadahacker.com/library/Documents/White_Papers/Gartner-MagicQuadrantforSIEM.pdf, 2014. (visited on 01/23/2016).

[106] HP ArcSight, *Arcsight*, http://www8.hp.com/us/en/software-solutions/siem-security-information-event-management/index.html, 2016. (visited on 01/23/2016).

[107] IBM Corporation, *Security qradar siem*, http://www-03.ibm.com/software/products/en/qradar-siem, 2016. (visited on 01/23/2016).

[108] LogRhythm, Inc., *Logrhythm*, https://www.logrhythm.com/, 2016. (visited on 01/23/2016).

[109] McAfee, Inc., *Mcafee enterprise security manager*, http://www.mcafee.com/us/products/enterprise-security-manager.aspx, 2016. (visited on 01/23/2016).

[110] Splunk, Inc., *Splunk enterprise*, http://www.splunk.com/, 2016. (visited on 01/23/2016).

[111] Logentries.com, Inc., *Logentries*, https://logentries.com/, 2016. (visited on 01/23/2016).

[112] CloudPassage, Inc., *Cloudpassage halo*, https://www.cloudpassage.com, 2016. (visited on 01/23/2016).

[113] Amazon, Inc., *Aws cloudwatch*, http://aws.amazon.com/cloudwatch, 2016. (visited on 01/23/2016).

[114] Rackspace, Inc., *Rackspace monitoring 1.0*, https://developer.rackspace.com/docs/cloud-monitoring/v1/developer-guide/, 2016. (visited on 01/23/2016).

[115] Salesforce.com, Inc., *Force.com rest api developer guide*, https://developer.salesforce.com/docs/atlas.en-us.api_rest.meta/api_rest/, 2016. (visited on 01/23/2016).

[116] Alert Logic, Inc., *Alert logic*, https://www.alertlogic.com/, 2016. (visited on 06/23/2016).

[117] Y. Li and X. Gui, "An agent-based grid monitoring system featuring dynamic and on-demand sensor deployment and management," *2005 First International Conference on Semantics, Knowledge and Grid*, no. Skg 2005, pp. 56–56, 2005. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4125844.

[118] G. Sladic, M. Vidakovic, and Z. Konjovic, "Agent based system for network availability and vulnerability monitoring," in *Intelligent Systems and Informatics (SISY), 2011 IEEE 9th International Symposium on*, Sep. 2011, pp. 53–58.

[119] R. Pugazendi and K. Duraiswamy, "Mobile agents - a solution for network monitoring," in *Advances in Recent Technologies in Communication and*

Computing, 2009. ARTCom '09. International Conference on, Oct. 2009, pp. 579–584.

[120] X. Yi and C. K. Siew, "Software agent-mediated confidential information gathering system," in *Parallel and Distributed Systems, 2000. Proceedings. Seventh International Conference on*, 2000, pp. 523–528.

[121] R. Aversa, L. Tasquier, and S. Venticinque, "Management of cloud infrastructures through agents," in *Emerging Intelligent Data and Web Technologies (EIDWT), 2012 Third International Conference on*, Sep. 2012, pp. 46–53.

[122] T. Rübsamen and C. Reich, "Cloud audits and privacy risks," in *On the Move to Meaningful Internet Systems: OTM 2013 Conferences SE - 29*, ser. Lecture Notes in Computer Science, R. Meersman, H. Panetto, T. Dillon, J. Eder, Z. Bellahsene, N. Ritter, P. Leenheer, and D. Dou, Eds., vol. 8185, Springer Berlin Heidelberg, 2013, pp. 403–413. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-41030-7%5C_29.

[123] T. Rübsamen, C. Reich, T. Wlodarczyk, and C. Rong, "Evidence for accountable cloud computing services," Tech. Rep., 2013. [Online]. Available: http://dimacs.rutgers.edu/Workshops/TAFC/TAFC%5C_a4cloud.pdf.

[124] T. Rübsamen and C. Reich, "Supporting cloud accountability by collecting evidence using audit agents," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 1, Dec. 2013, pp. 185–190.

[125] *IEEE 802.1q: Vlan*, IEEE, http://www.ieee802.org/1/pages/802.1Q.html, 2005.

[126] M. Mahalingam and D. Dutt and K. Duda and P. Agarwal and L. Kreeger and T. Sridhar and M. Bursell and C. Wright, *Virtual extensible local area network (vxlan): a framework for overlaying virtualized layer 2 networks over layer 3 networks*, https://tools.ietf.org/html/rfc7348, 2014.

[127] Cisco Systems, Inc., *Cisco ios netflow*, http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html, 2016. (visited on 01/23/2016).

[128] Google, Inc., *Google apps activity api*, https://developers.google.com/google-apps/activity/, 2016. (visited on 01/23/2016).

[129] ——, *Reports api: Drive activity report*, https://developers.google.com/admin-sdk/reports/v1/guides/manage-audit-drive, 2016. (visited on 01/23/2016).

[130] Mattijs Perdeck, *Jsnlog*, http://jsnlog.com/, 2016. (visited on 01/23/2016).

[131] Cloud Foundry Foundation, *Cloud foundry project*, https://www.cloudfoundry.org/, 2016. (visited on 01/23/2016).

[132] G. M. Mohay, A. M. Anderson, B. Collie, O. de Vel, and R. D. McKemmish, *Computer and Intrusion Forensics*. Boston, MA, USA: Artech House, 2003, For more information about this book please refer to the publisher's website (see link) or contact the authors. [Online]. Available: http://eprints.qut.edu.au/10849/.

[133] D. Brezinski and T. Killalea, *Evidence collection and archiving*, http://tools.ietf.org/html/rfc3227, 2002. (visited on 01/23/2016).

[134] National Institute of Justice (U.S.), *Electronic crime scene investigation: An on-the-scene reference for first responders*. US Department of Justice, Office of Justice Programs, National Institute of Justice, 2009.

[135] E. E. Kenneally and C. L. Brown, "Risk sensitive digital evidence collection," *Digital Investigation*, vol. 2, no. 2, pp. 101–119, 2005.

[136] ISO, *Iso 19011:2011 - guidelines for auditing management systems*, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=50675. (visited on 01/23/2016).

[137] J. Park and R. Sandhu, "The uconabc usage control model," *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 1, pp. 128–174, Feb. 2004. [Online]. Available: http://doi.acm.org/10.1145/984334.984339.

[138] T. Rübsamen, T. Pulls, and C. Reich, "Secure evidence collection and storage for cloud accountability audits," in *CLOSER 2015 - Proceedings of the 5th International Conference on Cloud Computing and Services Science, Lisbon, Portugal, May 20 - 22, 2015.*, 2015, pp. 321–330.

[139] ——, "Cloud computing and services science: 5th international conference, closer 2015, lisbon, portugal, may 20-22, 2015, revised selected papers," in, M. Helfert, V. Méndez Muñoz, and D. Ferguson, Eds. Cham: Springer International Publishing, 2016, ch. Security and Privacy Preservation of Evidence in Cloud Accountability Audits, pp. 95–114. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-29582-4_6.

[140] T. Rübsamen, D. Hölscher, and C. Reich, "Towards auditing of cloud provider chains using cloudtrust protocol," in *CLOSER 2016 - Proceedings of the 6th International Conference on Cloud Computing and Services Science, Rome, Italy, April 23 - 25, 2016.*, vol. 1, 2016, pp. 83–94.

[141] T. Rübsamen, C. Reich, N. Clarke, and M. Knahl, "Evidence collection in cloud provider chains," in *CLOSER 2016 - Proceedings of the 6th International Conference on Cloud Computing and Services Science, Rome, Italy, April 23 - 25, 2016.*, vol. 1, 2016, pp. 59–70.

[142] S. Poslad, "Specifying protocols for multi-agent systems interaction," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 2, no. 4, 15–es, 2007. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1293731.1293735.

[143] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *Knowledge Engineering Review*, vol. 10, pp. 115–152, 1995.

[144] H. S. Nwana, "Software agents: An overview," *Knowledge Engineering Review*, vol. 11, pp. 205–244, 1996.

[145] M. Wooldridge, *An introduction to MultiAgent Systems*. West Sussex: John Wiles & Sons Ltd., 2002.

[146] JADE, *Java agent developement framework*, http://jade.tilab.com, 2016. (visited on 01/23/2016).

[147] IEEE Foundation for Intelligent Physical Agents (FIPA), *The foundation for intelligent physical agents*, http://www.fipa.org/, 2016. (visited on 01/23/2016).

[148] ——, *Fipa acl ontology service specification*, http://www.fipa.org/specs/fipa00086/XC00086D.pdf, 2002. (visited on 01/23/2016).

[149] ——, *Fipa acl message structure specification*, http://www.fipa.org/specs/fipa00061/SC00061G.pdf, 2002. (visited on 01/23/2016).

[150] Office of the Irish Data Protection Commissioner (DPC), *Facebook ireland ltd. - report of audit*, https://www.dataprotection.ie/documents/facebook%20report/final%20report/report.pdf, 2011.

[151] T. Pulls and R. Peeters, *Insynd: Secure one-way messaging through Balloons*, Cryptology ePrint Archive, Report 2015/150, 2015.

[152] A. Jerman Blaič, T. Klobučar, and B. D. Jerman, "Long-term trusted preservation service using service interaction protocol and evidence records," *Comput. Stand. Interfaces*, vol. 29, no. 3, pp. 398–412, Mar. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.csi.2006.06.004.

[153] P. Turner, "Unification of digital evidence from disparate sources (digital evidence bags)," *Digit. Investig.*, vol. 2, no. 3, pp. 223–228, Sep. 2005. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2005.07.001.

[154] T. Pulls, R. Peeters, and K. Wouters, "Distributed privacy-preserving transparency logging," in *WPES*, A.-R. Sadeghi and S. Foresti, Eds., ACM, 2013, pp. 83–94.

[155] T. Pulls and R. Peeters, "Computer security – esorics 2015: 20th european symposium on research in computer security, vienna, austria, september 21-25, 2015, proceedings, part ii," in, G. Pernul, P. Y A Ryan, and E. Weippl, Eds. Cham: Springer International Publishing, 2015, ch. Balloon: A Forward-Secure Append-Only Persistent Authenticated Data Structure, pp. 622–641. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-24177-7_31.

[156] Tom Cook, *Dropbox at aws re:invent 2014*, https://blogs.dropbox.com/tech/2014/12/aws-reinvent-2014/, 2014. (visited on 01/23/2016).

[157] InfoCuria - Case-law of the Court of Justice, *Judgment of the court (grand chamber) of 6 october 2015 (request for a preliminary ruling from the high court (ireland)) — maximillian schrems v data protection commissioner*, http://curia.europa.eu/juris/document/document.jsf?docid=172254&mode=req&pageIndex=1&dir=&occ=first&part=1&text=&doclang=EN&cid=981230, 2015. (visited on 01/23/2016).

[158] C. of Justice of the European Union, *Press release no 117/15: the court of justice declares that the commission's us safe harbour decision is invalid*, http://curia.europa.eu/jcms/upload/docs/application/pdf/2015-10/cp150117en.pdf, 2015.

[159] Amazon Web Services, *Amazon cloudwatch*, https://aws.amazon.com/de/cloudwatch/, 2016. (visited on 01/23/2016).

[160] Rackspace, *Rackspace cloud monitoring*, http://www.rackspace.com/cloud/monitoring, 2016. (visited on 01/23/2016).

[161] S. Sanderson, *Knockout.js*, http://knockoutjs.com/, 2016. (visited on 01/23/2016).

[162] jQuery Team, *Jquery library*, http://jquery.com/, 2016. (visited on 01/23/2016).

[163] C. A. Ardagna, L. Bussard, S. D. C. D. Vimercati, G. Neven, S. Paraboschi, E. Pedrini, S. Preiss, D. Raggett, P. Samarati, S. Trabelsi, and M. Verdicchio, *Primelife policy language*, http://www.w3.org/2009/policy-ws/papers/Trabelisi.pdf, 2009.

[164] A. Laube, G. Gagnerot, H. Plate, G. Hassenstein, M. Casalino, S. Paraboschi, C. Basile, T. Scholte, and H. Wenger, *D3.3 - configuration meta-model*, http://www.posecco.eu/fileadmin/POSECCO/user_upload/deliverables/D3.3_Configuration_Meta-Model.pdf, 2012.

[165] *Primelife project*, http://primelife.ercim.eu/, Primelife FP7 Project, 2011. (visited on 01/23/2016).

[166] S. Trabelsi, J. Sendor, and S. Reinicke, "Ppl: primelife privacy policy engine," *Proceedings - 2011 IEEE International Symposium on Policies for Distributed Systems and Networks, POLICY 2011*, pp. 184–185, 2011.

[167] W. Benghabrit, H. Grall, J.-C. Royer, M. Sellami, M. Azraoui, K. Elkhiyaoui, M. Onen, A. S. D. Olivera, and K. Bernsmed, "A cloud accountability policy representation framework," *CLOSER 2014 - 4th International Conference on Cloud Computing and Services Science*, pp. 489–498, 2014.

[168] W. Benghabrit, H. Grall, J.-C. Royer, M. Sellami, M. Azraoui, K. Elkhiyaoui, M. Önen, A. De Oliveira, and K. Bernsmed, "From regulatory obligations to enforceable accountability policies in the cloud," English, in *Cloud Computing and Services Sciences*, ser. Communications in Computer and Information Science, M. Helfert, F. Desprez, D. Ferguson, F. Leymann, and V. Méndez Muñoz, Eds., vol. 512, Springer International Publishing, 2015, pp. 134–150. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-25414-2_9.

[169] M. Azraoui, K. Elkhiyaoui, M. Önen, K. Bernsmed, A. De Oliveira, and J. Sendor, "A-ppl: An accountability policy language," English, in *Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance*, ser. Lecture Notes in Computer Science, J. Garcia-Alfaro, J.

Herrera-Joancomartí, E. Lupu, J. Posegga, A. Aldini, F. Martinelli, and N. Suri, Eds., vol. 8872, Springer International Publishing, 2015, pp. 319–326. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-17016-9_21.

[170] Distributed Management Task Force, Inc., "Cloud infrastructure management interface (cimi) model and restful http-based protocol - an interface for managing cloud infrastructure version 1.1.0," Distributed Management Task Force, Inc., Tech. Rep., 2013.

[171] ——, "Open virtualization format specification version 2.1.1," Distributed Management Task Force, Inc., Tech. Rep., 2015.

[172] ——, "Open virtualization format schema version 1.1.0," Distributed Management Task Force, Inc., Tech. Rep., 2009.

[173] *Bosh outer shell*, https://bosh.io, BOSH, 2016. (visited on 01/23/2016).

[174] Bundesamt für Sicherheit in der Informationstechnik, *It-grundschutz catalogues – 13th version*, https://gsb.download.bva.bund.de/BSI/ITGSKEN/IT-GSK-13-EL-en-all_v940.pdf, 2013.

[175] A. S. de Oliveira, J. Sendor, A. Garaga, and K. Jenatton, "Monitoring personal data transfers in the cloud," *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, vol. 1, pp. 347–354, 2013.

[176] O. F. Rana and K. Stout, "What is scalability in multi-agent systems?" In *Proceedings of the Fourth International Conference on Autonomous Agents*, ser. AGENTS '00, Barcelona, Spain: ACM, 2000, pp. 56–63. [Online]. Available: http://doi.acm.org/10.1145/336595.337033.

[177] P. Labs, *Puppet*, https://puppetlabs.com/, 2016. (visited on 01/23/2016).

[178] P. Wayner, *Agents Unleashed: A Public Domain Look at Agent Technology*. San Diego, CA, USA: Academic Press Professional, Inc., 1995.

[179] D. M. Chess, "Security issues in mobile code systems," in *Mobile Agents and Security*, London, UK, UK: Springer-Verlag, 1998, pp. 1–14. [Online]. Available: http://dl.acm.org/citation.cfm?id=648051.746186.

[180] K. Schelderup and J. Ølnes, "Intelligence in services and networks paving the way for an open service market: 6th international conference on intelligence and services in networks, is&n'99 barcelona, spain, april 27–29, 1999 proceedings," in, H. Zuidweg, M. Campolargo, and J. Delgado, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, ch. Mobile Agent Security — Issues and Directions, pp. 155–167. [Online]. Available: http://dx.doi.org/10.1007/3-540-48888-X_16.

[181] V. Varadharajan, "Security enhanced mobile agents," in *Proceedings of the 7th ACM Conference on Computer and Communications Security*, ser. CCS '00, Athens, Greece: ACM, 2000, pp. 200–209. [Online]. Available: http://doi.acm.org/10.1145/352600.352632.

[182] A. Bürkle, A. Hertel, W. Müller, and M. Wieser, "Evaluating the security of mobile agent platforms," *Autonomous Agents and Multi-Agent Systems*, vol. 18, no. 2, pp. 295–311, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10458-008-9043-z.

[183] W. Jansen and T. Karygiannis, *Nist special publication 800-19: Mobile agent security*, http://csrc.nist.gov/publications/nistpubs/800-19/sp800-19.pdf, 1999.

[184] W. Jansen, "Countermeasures for mobile agent security," *Comput. Commun.*, vol. 23, no. 17, pp. 1667–1676, Nov. 2000. [Online]. Available: http://dx.doi.org/10.1016/S0140-3664(00)00253-X.

[185] JADE, *Java agent developement framework security addon*, http://jade.tilab.com/dl.php?file=securityAddOn-3.9.zip, 2015. (visited on 01/23/2016).

[186] P. Ruf, T. Rübsamen, and C. Reich, "Accountability and security in the cloud: First summer school, cloud accountability project, a4cloud, malaga,

spain, june 2-6, 2014, revised selected papers and lectures," in, M. Felici and C. Fernández-Gago, Eds. Cham: Springer International Publishing, 2015, ch. Agent-Based Evidence Collection in Cloud Computing, pp. 185–198. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-17199-9_8.

[187] T. Rübsamen and C. Reich, "An architecture for cloud accountability audits," in *1. Baden-Württemberg Center of Applied Research Symposium on Information and Communication Systems SInCom 2014*, 2014.

[188] J. Lopez, T. Rübsamen, and D. Westhoff, "Privacy-friendly cloud audits with somewhat homomorphic and searchable encryption," in *Innovations for Community Services (I4CS), 2014 14th International Conference on*, Jun. 2014, pp. 95–103.

[189] N. Papanikolaou, T. Rübsamen, and C. Reich, "A simulation framework to model accountability controls for cloud computing," in *CLOUD COMPUTING 2014, The Fifth International Conference on Cloud Computing, GRIDs, and Virtualization*, 2014, pp. 12–19.

[190] T. Rübsamen and C. Reich, "Enhancing mobile device security by security level integration in a cloud proxy," *CLOUD COMPUTING 2012, The Third International Conference on Cloud Computing, GRIDs, and Virtualization*, 2012.

[191] ——, "Shibboleth web-proxy for single sign-on of cloud services," in *CLOSER 2012 - Proceedings of the 2nd International Conference on Cloud Computing and Services Science, Porto, Portugal, 18 - 21 April, 2012.*, SciTePress, 2012, pp. 89–95.

Appendix A

List of Publications

- T. Rübsamen, D. Hölscher, and C. Reich, "Towards auditing of cloud provider chains using cloudtrust protocol," in *CLOSER 2016 - Proceedings of the 6th International Conference on Cloud Computing and Services Science, Rome, Italy, April 23 - 25, 2016.*, vol. 1, 2016, pp. 83–94

- T. Rübsamen, C. Reich, N. Clarke, and M. Knahl, "Evidence collection in cloud provider chains," in *CLOSER 2016 - Proceedings of the 6th International Conference on Cloud Computing and Services Science, Rome, Italy, April 23 - 25, 2016.*, vol. 1, 2016, pp. 59–70

- T. Rübsamen, T. Pulls, and C. Reich, "Cloud computing and services science: 5th international conference, closer 2015, lisbon, portugal, may 20-22, 2015, revised selected papers," in, M. Helfert, V. Méndez Muñoz, and D. Ferguson, Eds. Cham: Springer International Publishing, 2016, ch. Security and Privacy Preservation of Evidence in Cloud Accountability Audits, pp. 95–114. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-29582-4_6

- T. Rübsamen, T. Pulls, and C. Reich, "Secure evidence collection and storage for cloud accountability audits," in *CLOSER 2015 - Proceedings of the 5th International Conference on Cloud Computing and Services Science, Lis-*

bon, Portugal, May 20 - 22, 2015., 2015, pp. 321–330

- P. Ruf, T. Rübsamen, and C. Reich, "Accountability and security in the cloud: First summer school, cloud accountability project, a4cloud, malaga, spain, june 2-6, 2014, revised selected papers and lectures," in, M. Felici and C. Fernández-Gago, Eds. Cham: Springer International Publishing, 2015, ch. Agent-Based Evidence Collection in Cloud Computing, pp. 185–198. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-17199-9_8

- T. Rübsamen and C. Reich, "An architecture for cloud accountability audits," in *1. Baden-Württemberg Center of Applied Research Symposium on Information and Communication Systems SInCom 2014*, 2014

- J. Lopez, T. Rübsamen, and D. Westhoff, "Privacy-friendly cloud audits with somewhat homomorphic and searchable encryption," in *Innovations for Community Services (I4CS), 2014 14th International Conference on*, Jun. 2014, pp. 95–103 ©2014 IEEE

- N. Papanikolaou, T. Rübsamen, and C. Reich, "A simulation framework to model accountability controls for cloud computing," in *CLOUD COMPUTING 2014, The Fifth International Conference on Cloud Computing, GRIDs, and Virtualization*, 2014, pp. 12–19

- T. Rübsamen and C. Reich, "Supporting cloud accountability by collecting evidence using audit agents," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 1, Dec. 2013, pp. 185–190 ©2013 IEEE

- T. Rübsamen and C. Reich, "Cloud audits and privacy risks," in *On the Move to Meaningful Internet Systems: OTM 2013 Conferences SE - 29*, ser. Lecture Notes in Computer Science, R. Meersman, H. Panetto, T. Dillon, J. Eder, Z. Bellahsene, N. Ritter, P. Leenheer, and D. Dou, Eds., vol. 8185,

Springer Berlin Heidelberg, 2013, pp. 403–413. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-41030-7%5C_29

- T. Rübsamen, C. Reich, T. Wlodarczyk, and C. Rong, "Evidence for accountable cloud computing services," Tech. Rep., 2013. [Online]. Available: http://dimacs.rutgers.edu/Workshops/TAFC/TAFC%5C_a4cloud.pdf

- F. Dölitzscher, T. Rübsamen, T. Karbe, M. Knahl, and C. Reich, "Sun behind clouds - on automatic cloud security audits and a cloud audit policy language," *International Journal On Advances in Networks and Services*, vol. 6, no. 1, pp. 1–16, 2013

- T. Rübsamen and C. Reich, "Enhancing mobile device security by security level integration in a cloud proxy," *CLOUD COMPUTING 2012, The Third International Conference on Cloud Computing, GRIDs, and Virtualization*, 2012

- T. Rübsamen and C. Reich, "Shibboleth web-proxy for single sign-on of cloud services," in *CLOSER 2012 - Proceedings of the 2nd International Conference on Cloud Computing and Services Science, Porto, Portugal, 18 - 21 April, 2012.*, SciTePress, 2012, pp. 89–95

Appendix B

AAS Prototype Evidence Record Listings

B.1 OpenStack Nova Evidence Record Example

Listing B.1 presents an evidence record as it was collected by an evidence collector in AAS and store in the evidence store. It depicts the existence of a snapshot and its history at a specific point in time.

```
1  <evidenceContainer>
2    <record id="17542194-9832-11e5-8994-feff819cdc9f">
3      <action>Snapshot 6b0a9a97-0ca7-4fa1-9d68-0714fe3b03c2</action>
4      <actor>OpenStack_UID d336b4929ef043c990755c67d695a1b9</actor>
5      <policyID>1042</policyID>
6      <supportingElements elementID="0">
7        <signature>m0Z9Pp2BGp+4/0fGl8mu2zM9Mrq4hWvDHAAczlW1ldI=</signature>
8        <element xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
             ↪  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
9          NovaImage{id=4844e48b-1d9d-447d-8b0d-adc08b5b1b9a,
10         name=PIISnap, status=ACTIVE, progress=100, size=1244200960, minRam
               ↪ =1024,
11         minDisk=40, created=Tue Oct 20 07:56:03 CEST 2015, updated=Tue Oct 20
               ↪ 07:56:48
```

```
12        CEST 2015, metadata={instance_uuid=6b0a9a97-0ca7-4fa1-9d68-0714fe3b03c2
              ↪ ,
13        image_location=snapshot, image_state=available, instance_type_memory_mb
              ↪ =4096,
14        user_id=d336b4929ef043c990755c67d695a1b9, image_type=snapshot,
15        instance_type_id=1, ramdisk_id=null, instance_type_name=m1.medium,
16        instance_type_ephemeral_gb=0, instance_type_rxtx_factor=1, kernel_id=
              ↪ null,
17        instance_type_flavorid=3, instance_type_vcpus=2, os_type=None,
18        instance_type_root_gb=40, base_image_ref=a86aa4ed-5df8-4452-bfd0-1
              ↪ d6e3d872839,
19        instance_type_swap=0, owner_id=93d3fa8a411e4cf0ad40889e153eba41},
20        links=[GenericLink{href=http://controller:8774/v2/655
              ↪ bf47adead485fa497d9a37b1060bd/images/4844e48b-1d9d-447d-8b0d-
              ↪ adc08b5b1b9a,
21        rel=self}, GenericLink{href=http://controller:8774/655
              ↪ bf47adead485fa497d9a37b1060bd/images/4844e48b-1d9d-447d-8b0d-
              ↪ adc08b5b1b9a,
22        rel=bookmark}, GenericLink{href=http://172.28.64.50:9292/655
              ↪ bf47adead485fa497d9a37b1060bd/images/4844e48b-1d9d-447d-8b0d-
              ↪ adc08b5b1b9a,
23        rel=alternate, type=application/vnd.openstack.image}],}
24      </element>
25    </supportingElements>
26    <evidenceMetaData>
27      <collectingInstance>PiiSnapshotEventCollector_ce7a6546-9831-11e5-8994-
            ↪ feff819cdc9f@Main-Container</collectingInstance>
28      <evidenceDetectionTime>2015-10-20T07:56:03+02:00</evidenceDetectionTime>
29    </evidenceMetaData>
30  </record>
31</evidenceContainer>
```

Listing B.1: AAS Evidence Record – Collected OpenStack Nova Snapshot Event

B.2 Violation Evidence Record Example

Listing B.2 presents an evidence record for a detected violation in the data retention audit case. It includes both the PII store and delete events that were collected from A-PPL-E as well as an associated VM snapshot event. Some further snapshot evidence records have been removed since they are deemed non-essential for this example (i.e., all records needed to detect a violation are present).

```
<evidenceContainer>
    <record id="0">
        <action>PII Data Retention Violation - Snapshots of PII Store(Kardio-
            ↪ Mon-PII-Store) available between 2015-12-09 10:29:41.0 and
            ↪ 2015-12-09 10:31:43.0 (contains  PII of Panos )@Main-Container
            ↪ (10.0.0.6)</action>
        <actor>DataRetentionPolicyEvaluationAgent_1042_Main-Container</actor>
        <policyID>1042</policyID>
        <supportingElements elementID="0"><signature>ycsqNPoCOX5ARMgGXWehyXSF+
            ↪ ZHwtfzEns6sBKUAluI=</signature>
            <element xsi:type="xs:string"><?xml version="1.0" encoding="UTF-8"
                ↪ standalone="yes"?>
                <evidenceContainer>
                    <record id="0">
                        <action>PII store message@Main-Container(10.0.0.6)</
                            ↪ action>
                        <actor>PiiDataRetentionAgent_1042_Main-Container</actor
                            ↪ >
                        <policyID>1042</policyID>
                        <supportingElements>
                            <signature>XE/
                                ↪ ZyplowLcEOB93vok80SW2XYhMDVKttHgLO85pQy8=</
                                ↪ signature>
                            <element xsi:type="xs:string" xmlns:xs="http://www.
                                ↪ w3.org/2001/XMLSchema" xmlns:xsi="http://www
                                ↪ .w3.org/2001/XMLSchema-instance">de.hfu.
                                ↪ A4Cloud.Agents.intrusionAttemptDetection.
```

```
                                ↪ ApplMessageWrapper@52188e99
16                              Message: PII storedtype: policy admnistration
17                              piiAttributeName: Country
18                              piiOwner: Panos
19                              date: 2015-12-09 10:29:41.0
20                          </element>
21                      </supportingElements>
22                      <evidenceMetaData>
23                          <collectingInstance>PiiDataRetentionAgent_1042_Main
                                ↪ -Container</collectingInstance>
24                          <evidenceDetectionTime>2015-12-09T10:31:01.703+01
                                ↪ :00</evidenceDetectionTime>
25                      </evidenceMetaData>
26                  </record>
27              </evidenceContainer>
28          </element>
29      </supportingElements>
30
31
32      <supportingElements elementID="1"><signature>
            ↪ OevSjwXB5vAtjMGs5WTcRlMCBmhYDc2V4+Q8PfNyMIA=</signature><element
            ↪  xsi:type="xs:string"><?xml version="1.0" encoding="UTF-8"
            ↪ standalone="yes"?>
33              <evidenceContainer>
34                  <record id="1">
35                      <action>PII delete message@Main-Container(10.0.0.6)</
                            ↪ action>
36                      <actor>PiiDataRetentionAgent_1042_Main-Container</actor
                            ↪ >
37                      <policyID>1042</policyID>
38                      <supportingElements>
39                          <signature>ASWyvEJSVfdMZGj0TaffeGu3SiI0B0bYUu+
                                ↪ L6loGw5k=</signature>
40                          <element xsi:type="xs:string" xmlns:xs="http://www.
                                ↪ w3.org/2001/XMLSchema" xmlns:xsi="http://www
                                ↪ .w3.org/2001/XMLSchema-instance">
41                              Message: PII deleted type: policy enforcement
```

```
42                                  piiAttributeName: Country
43                                  piiOwner: Panos
44                                  date: 2015-12-09 10:31:43.0
45                              </element>
46                          </supportingElements>
47                          <evidenceMetaData>
48                              <collectingInstance>PiiDataRetentionAgent_1042_Main
                                    ↪ -Container</collectingInstance>
49                              <evidenceDetectionTime>2015-12-09T10:32:02.101+01
                                    ↪ :00</evidenceDetectionTime>
50                          </evidenceMetaData>
51                      </record>
52                  </evidenceContainer>
53              </element>
54          </supportingElements>
55
56          <supportingElements elementID="2">
57              <signature>E7X+JPH4WyzU8E+z/mnx4R8BMEprX7Oej1wnVNtNsNY=</signature>
58              <element xsi:type="xs:string"><?xml version="1.0" encoding="UTF-8"
                    ↪ standalone="yes"?>
59                  <evidenceContainer>
60                      <record id="1">
61                          <action>SnapshotExists(1)_6b0a9a97-0ca7-4fa1-9d68-0714
                                ↪ fe3b03c2(Kardio-Mon-PII-Store)@null
                                ↪ (172.28.64.50,,,,,,,,,,,,,)</action>
62                          <actor>PiiSnapshotCheckAgent_1042_Main-
                                ↪ Container@AAS_Core_Container</actor>
63                          <policyID>1042</policyID>
64                          <supportingElements elementID="0">
65                              <signature>
                                    ↪ qtJtgU4OzdGkD53SqfjQzrr898fky0XiDbLpmUz9F8k=
                                    ↪ </signature>
66                              <element xsi:type="xs:string" xmlns:xs="http://www.
                                    ↪ w3.org/2001/XMLSchema" xmlns:xsi="http://www
                                    ↪ .w3.org/2001/XMLSchema-instance">NovaImage{
                                    ↪ id=54dd8d3e-f933-4df1-8853-f609b4bdddcc,
                                    ↪ name=PIISnap, status=SAVING, progress=25,
```

```
                                ↪ size=0, minRam=1024, minDisk=40, created=Wed
                                ↪  Dec 09 10:30:35 CET 2015, ... , type=
                                ↪ application/vnd.openstack.image}],
67                              }</element>
68                      </supportingElements>
69                      <evidenceMetaData>
70                          <collectingInstance>PiiSnapshotCheckAgent_1042_Main
                                ↪ -Container</collectingInstance>
71                          <evidenceDetectionTime>2015-12-09T10:31:01.532+01
                                ↪ :00</evidenceDetectionTime>
72                      </evidenceMetaData>
73                  </record>
74              </evidenceContainer>
75          </element>
76      </supportingElements>
77
78      <!-- more periodically collected snapshot information removed for
            ↪ clarity-->
79
80      <evidenceMetaData>
81          <collectingInstance>DataRetentionUsagePII</collectingInstance>
82          <evidenceDetectionTime>2015-12-09T10:33:03.820+01:00</
                ↪ evidenceDetectionTime>
83      </evidenceMetaData>
84  </record>
85 </evidenceContainer>
```

Listing B.2: AAS Evidence Record – Detected Data Retention Violation

Appendix C

AAS Prototype Logs

C.1 AAS Trace of Automated Test Case Intrusion Detection

Listing C.1 contains a log of events that are generated during the automated test case execution in AAS. This trace of events was used to build the timeline information and order of events demonstrated in Section 7.2.1. Some non-essential information was removed for clarity. This is indicated where required.

```
1 2016-03-24 07:54:17.939 task_received "IntrusionAttemptDetectionAuditTask [
    ↪ targetContainer=null, mailAddress=null, argsEmailNotificationAgent=null,
    ↪  argsSimplePolicyEvaluationAgent=null,
    ↪ argsIntrusionAttemptDetectionAgent=null, agents=null, intervall=60000,
    ↪ threshold=5]"
2
3 2016-03-24 07:54:18.02 audit_task_created IntrusionAttemptDetectionAuditTask [
    ↪ targetContainer=1412898117, mailAddress=null, argsEmailNotificationAgent
    ↪ =[null], argsSimplePolicyEvaluationAgent=[1412898117, 1004, admin, ABCD,
    ↪  https://141.28.98.114, https://141.28.98.114,
    ↪ IntrusionAttemptDetectionAuditTask, 8189, 8289],
    ↪ argsIntrusionAttemptDetectionAgent=[60000, 1412898117,
    ↪ IntrusionAttemptDetectionAuditTask, 5, 1004, admin, ABCD, https
    ↪ ://141.28.98.114, https://141.28.98.114, admin, ABCD, http
```

```
    ↪ ://141.28.98.117, http://141.28.98.117, apple, aas, 8181, 8282, 8189,
    ↪ 8289], agents=[ConfiguredAgent [agent=de.hfu.A4Cloud.Agents.
    ↪ intrusionAttemptDetection.IntrusionAttemptDetectionAgent, args=[60000,
    ↪ 1412898117, IntrusionAttemptDetectionAuditTask, 5, 1004, admin, ABCD,
    ↪ https://141.28.98.114, https://141.28.98.114, admin, ABCD, http
    ↪ ://141.28.98.117, http://141.28.98.117, apple, aas, 8181, 8282, 8189,
    ↪ 8289], name=IntrusionAttemptDetectionAgent_1004_1412898117],
    ↪ ConfiguredAgent [agent=de.hfu.A4Cloud.Agents.EmailNotificationAgent.
    ↪ EmailNotificationAgent, args=[null], name=
    ↪ EmailNotificationAgent_1004_1412898117], ConfiguredAgent [agent=de.hfu.
    ↪ A4Cloud.Agents.SimplePolicyEvaluationAgent.SimplePolicyEvaluationAgent,
    ↪ args=[1412898117, 1004, admin, ABCD, https://141.28.98.114, https
    ↪ ://141.28.98.114, IntrusionAttemptDetectionAuditTask, 8189, 8289], name=
    ↪ SimplePolicyEvaluationAgent_1004_1412898117], ConfiguredAgent [agent=de.
    ↪ hfu.A4Cloud.Agents.imtNotification.ImtNotificationAgent, args=[/opt/Core
    ↪ /AgentConfig.cfg], name=IMT_NotificationAgent_1004]], intervall=60000,
    ↪ threshold=5]
4
5 2016-03-24 07:54:18.022 spawning_task_agents
    ↪ spawning_agents_for_task_IntrusionAttemptDetectionAuditTask_"
    ↪ ConfiguredAgent [agent=de.hfu.A4Cloud.Agents.intrusionAttemptDetection.
    ↪ IntrusionAttemptDetectionAgent, args=[60000, 1412898117,
    ↪ IntrusionAttemptDetectionAuditTask, 5, 1004, admin, ABCD, https
    ↪ ://141.28.98.114, https://141.28.98.114, admin, ABCD, http
    ↪ ://141.28.98.117, http://141.28.98.117, apple, aas, 8181, 8282, 8189,
    ↪ 8289], name=IntrusionAttemptDetectionAgent_1004_1412898117]" "
    ↪ ConfiguredAgent [agent=de.hfu.A4Cloud.Agents.EmailNotificationAgent.
    ↪ EmailNotificationAgent, args=[null], name=
    ↪ EmailNotificationAgent_1004_1412898117]" "ConfiguredAgent [agent=de.hfu.
    ↪ A4Cloud.Agents.SimplePolicyEvaluationAgent.SimplePolicyEvaluationAgent,
    ↪ args=[1412898117, 1004, admin, ABCD, https://141.28.98.114, https
    ↪ ://141.28.98.114, IntrusionAttemptDetectionAuditTask, 8189, 8289], name=
    ↪ SimplePolicyEvaluationAgent_1004_1412898117]" "ConfiguredAgent [agent=de
    ↪ .hfu.A4Cloud.Agents.imtNotification.ImtNotificationAgent, args=[/opt/
    ↪ Core/AgentConfig.cfg], name=IMT_NotificationAgent_1004]"
6 2016-03-24 07:55:24.741 received_apple_log "de.hfu.A4Cloud.Agents.
    ↪ intrusionAttemptDetection.ApplMessageWrapper@2a6ba10d
```

```
7
8  2016-03-24 07:55:24.753 received_apple_log "de.hfu.A4Cloud.Agents.
       ↪ intrusionAttemptDetection.ApplMessageWrapper@790b91cf
9  Message: Date:Thu Mar 24 07:54:30 CET 2016 Event associated with personal data
       ↪ 'Country' belonging to 'Panos', access attempt by subject 'Employee':
       ↪ Access permitted for, for action 'read'
10 piiAttributeName: Country
11 piiOwner: Panos
12 date: 2016-03-24 07:54:30.0
13 "
14
15 ___SHORTENED (REPEATED CONSECUTIVE ACCESSES)___
16
17 2016-03-24 07:55:24.87 received_apple_log "de.hfu.A4Cloud.Agents.
       ↪ intrusionAttemptDetection.ApplMessageWrapper@46417f28
18 Message: Date:Thu Mar 24 07:55:02 CET 2016 Event associated with personal data
       ↪ 'Country' belonging to 'Panos', access attempt by subject 'Employee',
       ↪ for purpose http://www.w3.org/2002/01/P3Pv1/admin, for action 'read'
19 piiAttributeName: Country
20 piiOwner: Panos
21 date: 2016-03-24 07:55:02.0
22 "
23
24 2016-03-24 07:55:24.87 received_apple_log "de.hfu.A4Cloud.Agents.
       ↪ intrusionAttemptDetection.ApplMessageWrapper@46417f28
25 Message: Date:Thu Mar 24 07:55:02 CET 2016 Event associated with personal data
       ↪ 'Country' belonging to 'Panos', access attempt by subject 'Employee',
       ↪ for purpose http://www.w3.org/2002/01/P3Pv1/admin, for action 'read'
26 piiAttributeName: Country
27 piiOwner: Panos
28 date: 2016-03-24 07:55:02.0
29 "
30
31 2016-03-24 07:55:24.871 policy_violation_detected
       ↪ intrusion_detection_with_threshold_5_and_20_accesses_in_60000ns
32
33 2016-03-24 07:55:25.228 evidence_record_created <?xml version="1.0" encoding="
```

```
    ↪ UTF-8" standalone="yes"?>
<evidenceContainer>
    <record id="0">
        <action>read(Possible Intrusion Attempt - 20 accesses)@1412898117
            ↪ (10.0.0.5)</action>
        <actor>Data Subject</actor>
        <policyID>1004</policyID>
        <supportingElements elementID="0">
            <signature>1yvZkpyzGKcL1Lg77gjSKNyQ+pLvHM9n+3CYk+7w3VA=</signature>
            <element xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/
                ↪ XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                ↪ instance">de.hfu.A4Cloud.Agents.intrusionAttemptDetection.
                ↪ ApplMessageWrapper@790b91cf
                Message: Date:Thu Mar 24 07:54:30 CET 2016 Event associated
                    ↪ with personal data 'Country' belonging to 'Panos',
                    ↪ access attempt by subject 'Employee': Access permitted
                    ↪ for, for action 'read'
                piiAttributeName: Country
                piiOwner: Panos
                date: 2016-03-24 07:54:30.0
            </element>
        </supportingElements>

___SHORTENED (REPEATED CONSECUTIVE EVIDENCE RECORDS)___

        <supportingElements elementID="19">
            <signature>e1KOpo0YLIv+qULLIH/VqqgGlIvYb9YFvCHm0qyCa18=</signature>
            <element xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/
                ↪ XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                ↪ instance">de.hfu.A4Cloud.Agents.intrusionAttemptDetection.
                ↪ ApplMessageWrapper@6dd552d4
                Message: Date:Thu Mar 24 07:55:01 CET 2016 Event associated
                    ↪ with personal data 'Country' belonging to 'Panos',
                    ↪ access attempt by subject 'Employee': Access permitted
                    ↪ for, for action 'read'
                piiAttributeName: Country
                piiOwner: Panos
```

```
57                 date: 2016-03-24 07:55:02.0
58             </element>
59         </supportingElements>
60
61 2016-03-24 07:55:25.659 evidence_record_for_violation_created <?xml version
       ↪ ="1.0" encoding="UTF-8" standalone="yes"?>
62 <evidenceContainer>
63     <record id="1">
64         <action>read(Possible Intrusion Attempt - 20 accesses)@1412898117
               ↪ (10.0.0.5)@1412898117(10.0.0.5)</action>
65         <actor>SimplePolicyEvaluationAgent_1004_1412898117</actor>
66         <policyID>1004</policyID>
67         <supportingElements elementID="0">
68             <signature>wwDuBNiqLPCHisIcKzAEh6EiFOLQDVK01DiZxxBTUcQ=</signature>
69             <element xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/
                   ↪ XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                   ↪ instance"><?xml version="1.0" encoding="UTF-8" standalone="
                   ↪ yes"?>
70 <evidenceContainer>
71     <record id="0">
72         <action>read(Possible Intrusion Attempt - 20 accesses)@1412898117
               ↪ (10.0.0.5)</action>
73         <actor>Data Subject</actor>
74         <policyID>1004</policyID>
75         <supportingElements elementID="0">
76             <signature>1yvZkpyzGKcL1Lg77gjSKNyQ+pLvHM9n+3CYk+7w3VA=</signature>
77             <element xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/
                   ↪ XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                   ↪ instance">de.hfu.A4Cloud.Agents.intrusionAttemptDetection.
                   ↪ ApplMessageWrapper@790b91cf
78                 Message: Date:Thu Mar 24 07:54:30 CET 2016 Event associated
                       ↪ with personal data 'Country' belonging to 'Panos',
                       ↪ access attempt by subject 'Employee': Access permitted
                       ↪ for, for action 'read'
79                 piiAttributeName: Country
80                 piiOwner: Panos
81                 date: 2016-03-24 07:54:30.0
```

```
            </element>
        </supportingElements>

___SHORTENED (REPEATED CONSECUTIVE EVIDENCE RECORDS)___

        <supportingElements elementID="19">
            <signature>e1KOpo0YLIv+qULLIH/VqqgGlIvYb9YFvCHm0qyCa18=</signature>
            <element xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/
                ↪ XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                ↪ instance">de.hfu.A4Cloud.Agents.intrusionAttemptDetection.
                ↪ ApplMessageWrapper@6dd552d4
                Message: Date:Thu Mar 24 07:55:01 CET 2016 Event associated
                    ↪ with personal data 'Country' belonging to 'Panos',
                    ↪ access attempt by subject 'Employee': Access permitted
                    ↪ for, for action 'read'
                piiAttributeName: Country
                piiOwner: Panos
                date: 2016-03-24 07:55:02.0
            </element>
        </supportingElements>

2016-03-24 07:55:27.834 notification_sent_to_IMT "{
  "id" : "2ff32d23-405e-4897-ba74-01470e4d66fd",
  "type" : {
    "id" : "b2a086f8-a63b-40da-9fde-c52f5e1d84d3",
    "name" : "Access Control",
    "endpoint" : "http://141.28.98.125:8800",
    "incidents" : [ {
      "id" : "26c5eda5-d18a-4950-9154-4bfbc73104a5",
      "name" : "Intrusion Attempt Detection assumption",
      "type" : {
        "id" : "a4b4e021-2e17-4e4c-bf54-3f13239f0feb",
        "name" : "Intrusion Attempt",
        "description" : "This incident regards the right to know vs. need to
            ↪ know security scenario. Incidents of this type are possible
            ↪ detected intrusions analyzing the accesses in a time frame
            ↪ executed by a subject.",
```

```
110         "consequence" : 0.5
111
112 ___TRUNCATED (DUE TO MESSAGE SIZE OF NOTIFICATION)___
```

Listing C.1: Intrusion Detection Audit Example – Trace of Test Case (Internal AAS Logs)

C.2 AAS Trace of Automated Test Case for Data Location Audit

Listing C.2 contains a log of events that are generated during the automated test case execution in AAS. This trace of events was used to build the timeline information and order of events demonstrated in Section 7.2.2. Some non-essential information was removed for clarity. This is indicated where required.

```
1 2016-03-23 21:07:33.674 task_received "DTMTAuditTask [targetContainer
    ↪ =1412898110, mailAddress=null, argsEmailNotificationAgent=null,
    ↪ argsSimplePolicyEvaluationAgent=null, argsDTMTRecipientAgent=null,
    ↪ agents=null, intervall=60000, locations=Europe,Africa]"
2
3 2016-03-23 21:07:33.74 audit_task_created DTMTAuditTask [targetContainer
    ↪ =1412898110, mailAddress=null, argsEmailNotificationAgent=[null],
    ↪ argsSimplePolicyEvaluationAgent=[1412898110, 1005, admin, ABCD, https
    ↪ ://141.28.98.114, https://141.28.98.114, DTMT check, 8189, 8289],
    ↪ argsDTMTRecipientAgent=[60000, 1412898110, DTMT check, 1005, admin, ABCD
    ↪ , https://141.28.98.114, https://141.28.98.114, admin, ABCD, https
    ↪ ://141.28.98.110, https://141.28.98.110, dtmt, aas, Europe,Africa, 8189,
    ↪  8289, 8181, 8282], agents=[ConfiguredAgent [agent=de.hfu.A4Cloud.Agents
    ↪ .dtmtCommunication.DTMTRecipientAgent, args=[60000, 1412898110, DTMT
    ↪ check, 1005, admin, ABCD, https://141.28.98.114, https://141.28.98.114,
    ↪ admin, ABCD, https://141.28.98.110, https://141.28.98.110, dtmt, aas,
    ↪ Europe,Africa, 8189, 8289, 8181, 8282], name=
    ↪ DTMTRecipientAgent_1005_1412898110], ConfiguredAgent [agent=de.hfu.
    ↪ A4Cloud.Agents.SimplePolicyEvaluationAgent.SimplePolicyEvaluationAgent,
    ↪ args=[1412898110, 1005, admin, ABCD, https://141.28.98.114, https
```

```
    ↪ ://141.28.98.114, DTMT check, 8189, 8289], name=
    ↪ SimplePolicyEvaluationAgent_1005_1412898110]], intervall=60000,
    ↪ locations=Europe,Africa]
4
5 2016-03-23 21:07:33.749 spawning_task_agents spawning_agents_for_task_DTMT
    ↪ check_"ConfiguredAgent [agent=de.hfu.A4Cloud.Agents.dtmtCommunication.
    ↪ DTMTRecipientAgent, args=[60000, 1412898110, DTMT check, 1005, admin,
    ↪ ABCD, https://141.28.98.114, https://141.28.98.114, admin, ABCD, https
    ↪ ://141.28.98.110, https://141.28.98.110, dtmt, aas, Europe,Africa, 8189,
    ↪  8289, 8181, 8282], name=DTMTRecipientAgent_1005_1412898110]" "
    ↪ ConfiguredAgent [agent=de.hfu.A4Cloud.Agents.SimplePolicyEvaluationAgent
    ↪ .SimplePolicyEvaluationAgent, args=[1412898110, 1005, admin, ABCD, https
    ↪ ://141.28.98.114, https://141.28.98.114, DTMT check, 8189, 8289], name=
    ↪ SimplePolicyEvaluationAgent_1005_1412898110]"
6
7 2016-03-23 21:08:38.892 evidence_record_for_violation_created <?xml version
    ↪ ="1.0" encoding="UTF-8" standalone="yes"?>
8 <evidenceContainer>
9     <record id="1">
10         <action>DTMT Location violation@1412898110(10.0.0.1)@1412898110
               ↪ (10.0.0.1)</action>
11         <actor>SimplePolicyEvaluationAgent_1005_1412898110</actor>
12         <policyID>1005</policyID>
13         <supportingElements elementID="0">
14             <signature>BJAMX0EqFABskzUqydPes5D+9ZcWHBO5F1SqLcYivSI=</signature>
15             <element xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/
                   ↪ XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                   ↪ instance"><?xml version="1.0" encoding="UTF-8" standalone="
                   ↪ yes"?>
16                 <evidenceContainer>
17                     <record id="0">
18                         <action>DTMT Location violation@1412898110(10.0.0.1)</
                               ↪ action>
19                         <actor>DTMTRecipientAgent_1005_1412898110@1412898110</
                               ↪ actor>
20                         <policyID>1005</policyID>
21                         <supportingElements elementID="0">
```

```
22                          <signature>+BIFwjDbLavZC3UrE5wglhIS+
                                ↪ wlZlk9KWPhPFZ0bR/M=</signature>
23                          <element xsi:type="xs:string" xmlns:xs="http://www.
                                ↪ w3.org/2001/XMLSchema" xmlns:xsi="http://www
                                ↪ .w3.org/2001/XMLSchema-instance">148712 INFO
                                ↪   com.sap.a4cloud.ruleengine.RuleEngine
                                ↪ - Potential violation detected: "Volume"
                                ↪ name "EU Data vol.",  moved from original
                                ↪ host in "Europe" to new host in "US" </
                                ↪ element>
24                      </supportingElements>
25                      <supportingElements elementID="1">
26                          <signature>
                                ↪ gJrlCmdRQveKlpeib1RI7qkA5FoAW7iopxEYFTo1ltY
                                ↪ =</signature>
27                          <element xsi:type="xs:string" xmlns:xs="http://www.
                                ↪ w3.org/2001/XMLSchema" xmlns:xsi="http://www
                                ↪ .w3.org/2001/XMLSchema-instance">Record UUID
                                ↪ : 927b325f-b687-4b08-8992-bd28c23253bc</
                                ↪ element>
28                      </supportingElements>
29                      <evidenceMetaData>
30                          <collectingInstance>
                                ↪ DTMTRecipientAgent_1005_1412898110</
                                ↪ collectingInstance>
31                          <evidenceDetectionTime>2016-03-23T21
                                ↪ :08:37.860+01:00</evidenceDetectionTime>
32                      </evidenceMetaData>
33                  </record>
34              </evidenceContainer>
35          </element>
36      </supportingElements>
37      <supportingElements elementID="1">
38          <signature>caAjbe76RuiwivubduGHdwKiNFAN74O2m7Jt+X4mJqc=</signature>
39          <element xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/
                ↪ XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                ↪ instance">Record UUID: 3b065c8d-d3c1-46c5-97e7-c5a2cb5dbdb0
```

```
                ↪ </element>
40          </supportingElements>
41          <evidenceMetaData>
42              <collectingInstance>DTMT check</collectingInstance>
43              <evidenceDetectionTime>2016-03-23T21:08:38.755+01:00</
                    ↪ evidenceDetectionTime>
44          </evidenceMetaData>
45      </record>
46  </evidenceContainer>
```

Listing C.2: Data Location Audit Example – Trace of Test Case (Internal AAS Logs)

C.3 AAS Trace of Automated Test Case for Data Retention Audit

Listing C.3 contains a log of events that are generated during the automated test case execution in AAS. This trace of events was used to build the timeline information and order of events demonstrated in Section 7.2.3. Some non-essential information was removed for clarity. This is indicated where required.

```
1 2016-03-23 21:44:32.209 task_received "DataRetentionAuditTask [targetContainer
    ↪ =1412898117, agents=null, argsOpenStackRESTAgent=null,
    ↪ argsEmailNotificationAgent=null, argsDataRetentionPolicyEvaluationAgent=
    ↪ null, argsPiiSnapshotAgent=null, argsPiiApplMessageCheckAgent=null,
    ↪ policyID=1002, ruleID=null, recipient=1002, username=Panos, password=
    ↪ null, endpoint=null, port=null, tenant=null, intervall=60000, vmName=
    ↪ CP1_Service, existenceTime=null, directory=null, mailAddress=null,
    ↪ duration=null, configPath=null, osContainer=1412898110, taskContainer
    ↪ =1412898117]"
2
3 2016-03-23 21:44:32.237 audit_task_created DataRetentionAuditTask [
    ↪ targetContainer=1412898117, agents=[ConfiguredAgent [agent=de.hfu.
    ↪ A4Cloud.Agents.imtNotification.ImtNotificationAgent, args=[/opt/Core/
    ↪ AgentConfig.cfg], name=IMT_NotificationAgent_1042], ConfiguredAgent [
    ↪ agent=de.hfu.A4Cloud.Agents.dataRetention.PiiSnapshotCheckAgent, args
```

```
    ↪ =[1412898110, 1412898117, 1042, DataRetentionUsagePII, CP1_Service, /opt
    ↪ /Core/AgentConfig.cfg, 60000], name=
    ↪ PiiSnapshotCheckAgent_1042_1412898110], ConfiguredAgent [agent=de.hfu.
    ↪ A4Cloud.Agents.dataRetention.PiiApplMessageCheckAgent, args=[1412898117,
    ↪  DataRetentionUsagePII, CP1_Service, /opt/Core/AgentConfig.cfg, 1042,
    ↪ 60000, /opt/Core/policy/policy1.xml], name=
    ↪ PiiDataRetentionAgent_1042_1412898117], ConfiguredAgent [agent=de.hfu.
    ↪ A4Cloud.Agents.EmailNotificationAgent.EmailNotificationAgent, args=[null
    ↪ ], name=EmailNotificationAgent_1042_1412898117], ConfiguredAgent [agent=
    ↪ de.hfu.A4Cloud.Agents.DataRetentionPolicyEvaluationAgent.
    ↪ PiiDataRetentionPolicyEvaluationAgent, args=[1002, 1412898117, admin,
    ↪ ABCD, https://141.28.98.114, https://141.28.98.114,
    ↪ DataRetentionUsagePII, 1042, 1042, CP1_Service, /opt/Core/policy/policy1
    ↪ .xml, 8189, 8289], name=
    ↪ DataRetentionPolicyEvaluationAgent_1042_1412898117]],
    ↪ argsOpenStackRESTAgent=null, argsEmailNotificationAgent=[null],
    ↪ argsDataRetentionPolicyEvaluationAgent=[1002, 1412898117, admin, ABCD,
    ↪ https://141.28.98.114, https://141.28.98.114, DataRetentionUsagePII,
    ↪ 1042, 1042, CP1_Service, /opt/Core/policy/policy1.xml, 8189, 8289],
    ↪ argsPiiSnapshotAgent=[1412898110, 1412898117, 1042,
    ↪ DataRetentionUsagePII, CP1_Service, /opt/Core/AgentConfig.cfg, 60000],
    ↪ argsPiiApplMessageCheckAgent=[1412898117, DataRetentionUsagePII,
    ↪ CP1_Service, /opt/Core/AgentConfig.cfg, 1042, 60000, /opt/Core/policy/
    ↪ policy1.xml], policyID=1042, ruleID=1042, recipient=1002, username=Panos
    ↪ , password=null, endpoint=null, port=null, tenant=null, intervall=60000,
    ↪  vmName=CP1_Service, existenceTime=null, directory=null, mailAddress=
    ↪ null, duration=null, configPath=/opt/Core/AgentConfig.cfg, osContainer
    ↪ =1412898110, taskContainer=1412898117]
4
5 2016-03-23 21:44:32.274 spawning_task_agents spawning_agents_for_task_Data
    ↪ Retention PII Task_"ConfiguredAgent [agent=de.hfu.A4Cloud.Agents.
    ↪ imtNotification.ImtNotificationAgent, args=[/opt/Core/AgentConfig.cfg],
    ↪ name=IMT_NotificationAgent_1042]" "ConfiguredAgent [agent=de.hfu.A4Cloud
    ↪ .Agents.dataRetention.PiiSnapshotCheckAgent, args=[1412898110,
    ↪ 1412898117, 1042, DataRetentionUsagePII, CP1_Service, /opt/Core/
    ↪ AgentConfig.cfg, 60000], name=PiiSnapshotCheckAgent_1042_1412898110]" "
    ↪ ConfiguredAgent [agent=de.hfu.A4Cloud.Agents.dataRetention.
```

```
    ↪ PiiApplMessageCheckAgent, args=[1412898117, DataRetentionUsagePII,
    ↪ CP1_Service, /opt/Core/AgentConfig.cfg, 1042, 60000, /opt/Core/policy/
    ↪ policy1.xml], name=PiiDataRetentionAgent_1042_1412898117]" "
    ↪ ConfiguredAgent [agent=de.hfu.A4Cloud.Agents.EmailNotificationAgent.
    ↪ EmailNotificationAgent, args=[null], name=
    ↪ EmailNotificationAgent_1042_1412898117]" "ConfiguredAgent [agent=de.hfu.
    ↪ A4Cloud.Agents.DataRetentionPolicyEvaluationAgent.
    ↪ PiiDataRetentionPolicyEvaluationAgent, args=[1002, 1412898117, admin,
    ↪ ABCD, https://141.28.98.114, https://141.28.98.114,
    ↪ DataRetentionUsagePII, 1042, 1042, CP1_Service, /opt/Core/policy/policy1
    ↪ .xml, 8189, 8289], name=
    ↪ DataRetentionPolicyEvaluationAgent_1042_1412898117]"
6
7 2016-03-23 21:45:36.212 snapshot_detected "<?xml version="1.0" encoding="UTF-8"
    ↪  standalone="yes"?>
8 <evidenceContainer>
9     <record id="1">
10         <action>SnapshotExists(1)_a04437e1-2cb4-47a5-b2f5-0ba73cbe7bf3(
               ↪ CP1_Service)@1412898110(172.28.64.50,,,,,,,,,,,,)</action>
11         <actor>PiiSnapshotCheckAgent_1042_1412898110@1412898110</actor>
12         <policyID>1042</policyID>
13         <supportingElements elementID="0">
14             <signature>T1hiNg89Q/jUVCHfQzpG8GKSOltMyQFe44NVH2ymbBM=</signature>
15             <element xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/
                   ↪ XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                   ↪ instance">NovaImage{id=d840f4cb-11f1-432f-bc1f-3a7b922c1cce,
                   ↪  name=DemoSnap, status=SAVING, progress=25, size=0, minRam
                   ↪ =1024, minDisk=40, created=Wed Mar 23 21:45:13 CET 2016,
                   ↪ updated=Wed Mar 23 21:45:13 CET 2016, metadata={
                   ↪ instance_uuid=a04437e1-2cb4-47a5-b2f5-0ba73cbe7bf3,
                   ↪ instance_type_memory_mb=4096, user_id=
                   ↪ d336b4929ef043c990755c67d695a1b9, image_type=snapshot,
                   ↪ instance_type_id=1, instance_type_name=m1.medium,
                   ↪ instance_type_ephemeral_gb=0, instance_type_rxtx_factor=1,
                   ↪ instance_type_root_gb=40, instance_type_flavorid=3,
                   ↪ instance_type_vcpus=2, instance_type_swap=0, base_image_ref=
                   ↪ a86aa4ed-5df8-4452-bfd0-1d6e3d872839}, links=[GenericLink{
```

```
                ↪ href=http://141.28.98.110:8774/v2/655
                ↪ bf47adead485fa497d9a37b1060bd/images/d840f4cb-11f1-432f-bc1f
                ↪ -3a7b922c1cce, rel=self}, GenericLink{href=http
                ↪ ://141.28.98.110:8774/655bf47adead485fa497d9a37b1060bd/
                ↪ images/d840f4cb-11f1-432f-bc1f-3a7b922c1cce, rel=bookmark},
                ↪ GenericLink{href=http://172.28.64.50:9292/655
                ↪ bf47adead485fa497d9a37b1060bd/images/d840f4cb-11f1-432f-bc1f
                ↪ -3a7b922c1cce, rel=alternate, type=application/vnd.openstack
                ↪ .image}],
}</element>
        </supportingElements>
        <supportingElements elementID="1">
            <signature>gTQQPM/CSkaqoD6todgFEEYfQNKjJW1G/i/gVuCZVQo=</signature>
            <element xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/
                ↪ XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                ↪ instance">Record UUID: c4efe83c-856f-47a8-8db8-b998dab96616
                ↪ </element>
        </supportingElements>
        <evidenceMetaData>
            <collectingInstance>PiiSnapshotCheckAgent_1042_1412898110</
                ↪ collectingInstance>
            <evidenceDetectionTime>2016-03-23T21:45:36.081+01:00</
                ↪ evidenceDetectionTime>
        </evidenceMetaData>
    </record>
</evidenceContainer>
"

2016-03-23 21:45:38.571 received_apple_log_"PII store message" "<?xml version
    ↪ ="1.0" encoding="UTF-8" standalone="yes"?>
<evidenceContainer>
    <record id="0">
        <action>PII store message@1412898117(10.0.0.5)</action>
        <actor>PiiDataRetentionAgent_1042_1412898117</actor>
        <policyID>1042</policyID>
        <supportingElements>
            <signature>ZKR9xs8d1XhyrHM+iaLXmBz3na+vZeV4HKNvVKQlZ2c=</signature>
```

```
38            <element xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/
                  ↪ XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                  ↪ instance">de.hfu.A4Cloud.Agents.intrusionAttemptDetection.
                  ↪ ApplMessageWrapper@cb1ae5d
39                Message: PII storedtype: policy admnistration
40                piiAttributeName: Country
41                piiOwner: Panos
42                date: 2016-03-23 21:44:36.0
43            </element>
44        </supportingElements>
45        <supportingElements elementID="1">
46            <signature>i3U+vhHrtnDseVzk7vnm6zk1FZa2dkFmd8ODfqu66KA=</signature>
47            <element xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/
                  ↪ XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                  ↪ instance">Record UUID: 7099e7fb-ea90-44b6-bd73-66c4fc722c8a
                  ↪ </element>
48        </supportingElements>
49        <evidenceMetaData>
50            <collectingInstance>PiiDataRetentionAgent_1042_1412898117</
                  ↪ collectingInstance>
51            <evidenceDetectionTime>2016-03-23T21:45:38.106+01:00</
                  ↪ evidenceDetectionTime>
52        </evidenceMetaData>
53    </record>
54</evidenceContainer>
55"
56
57 2016-03-23 21:46:39.217 received_apple_log_"PII delete message" "<?xml version
    ↪ ="1.0" encoding="UTF-8" standalone="yes"?>
58<evidenceContainer>
59    <record id="1">
60        <action>PII delete message@1412898117(10.0.0.5)</action>
61        <actor>PiiDataRetentionAgent_1042_1412898117</actor>
62        <policyID>1042</policyID>
63        <supportingElements>
64            <signature>cO4ypyc4FA6Mox+PTIfydcOKHgqt9u01uNEvS2Gxxis=</signature>
65            <element xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/
```

```
                ↪ XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                ↪ instance">de.hfu.A4Cloud.Agents.intrusionAttemptDetection.
                ↪ ApplMessageWrapper@72da2335
                Message: PII deleted type: policy enforcement
                piiAttributeName: Country
                piiOwner: Panos
                date: 2016-03-23 21:46:36.0
            </element>
        </supportingElements>
        <supportingElements elementID="1">
            <signature>6MbfJukIPJhpq5FWm4HzllxxjR6lOcVEvubxkA4JWkE=</signature>
            <element xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/
                ↪ XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                ↪ instance">Record UUID: ce6f116b-06f2-477e-baad-9039facfd243
                ↪ </element>
        </supportingElements>
        <evidenceMetaData>
            <collectingInstance>PiiDataRetentionAgent_1042_1412898117</
                ↪ collectingInstance>
            <evidenceDetectionTime>2016-03-23T21:46:39.060+01:00</
                ↪ evidenceDetectionTime>
        </evidenceMetaData>
    </record>
</evidenceContainer>
"

2016-03-23 21:47:41.862 data_retention_policy_violation_detected "<?xml version
    ↪ ="1.0" encoding="UTF-8" standalone="yes"?>
<evidenceContainer>
<record id="0">
<action>PII Data Retention Violation - Snapshots of PII Store(CP1_Service)
    ↪ available between 2016-03-23 21:44:36.0 and 2016-03-23 21:46:36.0 (
    ↪ contains  PII of Panos )@1412898117(10.0.0.5)</action>
<actor>DataRetentionPolicyEvaluationAgent_1042_1412898117</actor>
<policyID>1042</policyID>
<supportingElements elementID="0">
___TRUNCATED (DUE TO MESSAGE SIZE OF VIOLATION - HAS PREVIOUSLY SHOWN EVIDENCE
```

```
    ↪ RECORDS ATTACHED)___

2016-03-23 21:48:31.576 notification_sent_to_IMT "{
  "id" : "4d226def-5c82-4edd-acad-b8f0599a1a75",
  "type" : {
    "id" : "e169c243-d60d-4c39-b3ce-4c7abd4c04e1",
    "name" : "PII snapshot exists notification",
    "endpoint" : "http://141.28.98.125:8800",
    "incidents" : [ {
      "id" : "1b007583-3df9-4c06-98e6-47fe89c68e6f",
      "name" : "PII Data Retention Violation",
      "type" : {
        "id" : "9749f483-0ba9-4bc0-8a1f-956140965a5f",
        "name" : "PII Data Retention Snapshot Violation",
        "description" : "PII should be completely deleted but still exists in
            ↪ snapshot",
        "consequence" : 0.5
      },
___TRUNCATED (DUE TO MESSAGE SIZE OF NOTIFICATION - HAS PREVIOUSLY SHOWN
    ↪ EVIDENCE RECORDS ATTACHED)___
```

Listing C.3: Data Retention Audit Example – Trace of Test Case (Internal AAS Logs)

List of Acronyms

A

A4Cloud Cloud Accountability Project.
AAC Audit Agent Controller.
AAS Audit Agent System.
ACL Agent Communication Language.
AI Artificial Intelligence.
ANN Artificial Neural Network.
API Application Programming Interface.
APM Audit Policy Module.
A-PPL Accountability PrimeLife Policy Language.
A-PPL-E Accountability PrimeLife Policy Language Engine.
AWS Amazon Web Services.

B

BDSG Bundesdatenschutzgesetz.
BOSH Bosh Outer Shell.

C

CADF Cloud Auditing Data Federation.
CAIQ Consensus Assessments Initiative Questionnaire.
CAPL Cloud Audit Policy Language.
CCM Cloud Controls Matrix.
CDESF Common Digital Evidence Storage Format.
CIMI Cloud Infrastructure Management Interface.
CMS Cloud Management System.
CRM Customer Relationship Management.
CSA Cloud Security Alliance.
CSS Cascading Style Sheets.
CTP CSA Cloud Trust Protocol.

D

DB database.
DEB Digital Evidence Bag.
DMTF Distributed Management Task Force.
DoS Denial of Service.
DPA Data Protection Authority.
DTM Data Transfer Monitor.

E

ENISA European Network and Information Security Agency.
EPP Evidence Processor and Presenter.
ERS Evidence Record Syntax.
ES Evidence Store.
EU European Union.

F

FedRAMP Federal Risk and Authorization Program.

FIPA Foundation for Intelligent Physical Agents.

G

GUI Graphical User Interface.
GUID Universally Unique Identifier.

H

HTML Hypertext Markup Language.
HTTP Hypertext Transfer Protocol.
HTTPS Hypertext Transfer Protocol Secure.

I

IaaS Infrastructure as a Service.
ICT Information and Communication Technology.
IGA Information Gathering Agent.
IMT Incident Management Tool.
IP Internet Protocol.
IS Information Security.
IT Information Technology.

J

JAAS Java Authentication and Authorization Service.
JADE Java Agent DEvelopment Framework.
JSON Javascript Object Notation.
JVM Java Virtual Machine.

K

KVM Kernel-based Virtual Machine.

L

LAN Local Area Network.

M

MAS Multi-agent System.
MTA Mail Transport Agent.

N

NIST National Institute for Standards and Technology.
NTP Network Time Protocol.

O

OECD Organisation for Economic Cooperation and Development.
OVF Open Virtualization Format.
OWASP Open Web Application Security Project.

P

PaaS Platform as a Service.
PDF Portable Document Format.
PDP Proof of Data Possession.
PGP Pretty Good Privacy.
PII Personal Identifiable Information.
PoR Proof of Retrievability.
PPEDFI Privacy Preserving Efficient Digital Forensic Investigation.
PPL PrimeLife Policy Language.
PSP Primary Service Provider.

Q

QoS Quality of Service.

R

RAM Random Access Memory.
ReST Representational State Transfer.
RZV Regionales Zentrum für Virtualisierung.

S

SAaaS Security Audit as a Service.
SaaS Software as a Service.
SDN Software-defined Networking.
SIEM Security Information and Event Management.

SLA Service Level Agreement.
SME Small or Medium-size Enterprise.
SNMP Simple Network Management Protocol.
STAR Security, Trust & Assurance Registry.

T

TLS Transport Layer Security.
TOS Terms of Service.

U

UCON Usage Control.
UI User Interface.
UML Unified Modeling Language.
URI Unique Resource Identifier.

V

VLAN Virtual Local Area Network.
VM Virtual Machine.
VMI Virtual Machine Introspection.
VPN Virtual Private Network.
VXLAN Virtual Extensible Local Area Network.

W

WORM Write Once Read Many.
WWW World Wide Web.

X

XACML eXtensible Access Control Markup Language.
XML eXtensible Markup Language.

Published Papers

Evidence Collection in Cloud Provider Chains

Thomas Rübsamen[1], Christoph Reich[1], Nathan Clarke[2] and Martin Knahl[3]
[1]*Institute for Cloud Computing and IT Security, Furtwangen University, Robert-Gerwig-Platz 1, Furtwangen, Germany*
[2]*Centre for Security, Communications and Network Research, Plymouth University, Portland Square, Plymouth, United Kingdom*
[3]*Furtwangen University, Robert-Gerwig-Platz 1, Furtwangen, Germany*
{*thomas.ruebsamen, christoph.reich, martin.knahl*}*@hs-furtwangen.de, nathan.clarke@plymouth.ac.uk*

Keywords: Cloud Computing, Audit, Federated Cloud, Security, Digital Evidence

Abstract: With the increasing importance of cloud computing, compliance concerns get into the focus of businesses more often. Furthermore, businesses still consider security and privacy related issues to be the most prominent inhibitors for an even more widespread adoption of cloud computing services. Several frameworks try to address these concerns by building comprehensive guidelines for security controls for the use of cloud services. However, assurance of the correct and effective implementation of such controls is required by businesses to attenuate the loss of control that is inherently associated with using cloud services. Giving this kind of assurance is traditionally the task of audits and certification. Cloud auditing becomes increasingly challenging for the auditor the more complex the cloud service provision chain becomes. There are many examples for Software as a Service (SaaS) providers that do not own dedicated hardware anymore for operating their services, but rely solely on other cloud providers of the lower layers, such as platform as a service (PaaS) or infrastructure as a service (IaaS) providers. The collection of data (evidence) for the assessment of policy compliance during a technical audit is aggravated the more complex the combination of cloud providers becomes. Nevertheless, the collection at all participating providers is required to assess policy compliance in the whole chain. The main contribution of this paper is an analysis of potential ways of collecting evidence in an automated way across cloud provider boundaries to facilitate cloud audits. Furthermore, a way of integrating the most suitable approaches in the system for automated evidence collection and auditing is proposed.

1 INTRODUCTION

As cloud computing becomes more accepted by mainstream businesses and replaces more and more on-premise IT installations, compliance with regulation, industry best-practices and customer requirements becomes increasingly important. The main inhibitor for even more widespread adoption of cloud services still remain security and privacy concerns of cloud customers (Cloud Security Alliance, 2013). In Germany, a preference for cloud providers that fall under German jurisdiction and also run their own data centers in Germany or at least inside the European Union can be observed recently (Bitkom Research GmbH, 2015). This comes as no surprise when privacy violations that have become known to the general population in recent years are considered (e.g., NSA and Snowden revelations). A feasible way to assess and ensure compliance of cloud services regularly is by using audits. For any technical audit, information has to be collected in order to assess compliance. This automated process is called evidence collection in our system. In our previous work on cloud auditing, the focus was put on automating the three major parts of an audit system, i) evidence collection and handling, ii) evaluation against machine-readable policies and iii) presentation of audit results (Rübsamen and Reich, 2013; Rübsamen et al., 2013; Rübsamen and Reich, 2014; Rübsamen et al., 2015).

Today, it is common to not only have a single cloud provider to provision a service to its customers, but multiple. The composition of multiple services provided by different providers can already be observed where Software as a Service (SaaS) providers host their offering on top of the computing resources provided by an Infrastructure as a Service (IaaS) provider. For instance, Dropbox and Netflix both host their services using Amazon's infrastructure. These composed services - they can be considered to form a chain of cloud providers, therefore cloud provider chain - can become very complex and opaque with respect to the flow of data between providers. Several

new challenges for the auditing of such cloud provider chains can be identified, which will be discussed in this paper. The other major contribution of this paper is a proposed solution to auditing of cloud provider chains, which is an extension to our previous work in this area.

This paper is structured as follows: in Section 2, related research projects and industrial approaches are discussed. Following that, in Section 3 the authors elaborate on the definition and properties of cloud provider chains and auditing. Afterwards, a discussion of three different approaches to evidence collection for provider chain auditing in Section 4 is presented. In Section 5, the architectural integration of the approaches in a system for automating cloud audits is presented and evaluated for their effectiveness using a fictitious scenario. Section 6 concludes this paper.

2 RELATED WORK

Standards and catalogues such as ISO27001 (ISO, 2005), Control Objectives for Information and Related Technology (COBIT) (Information Systems Audit and Control Association, 2012) or NIST 800-53 (National Institute of Standards and Technology), 2013) define information security controls. A major part of these frameworks is auditing, both in regular auditing as an control itself and by using audits to ensure the correct and effective implementation of the controls. They are typically generic and target information systems in general and do not address the specifics of cloud computing.

There are some extensions to the previous frameworks such as the Cloud Controls Matrix (Cloud Security Alliance, 2014). It aims at the integration of aspects from ISO and COBIT, and NIST's more cloud-focused security and privacy protection recommendations 800-144 (National Institute of Standards and Technology, 2011), as well as domain-specific frameworks such as PCI-DSS (PCI Security Standards Council, 2015) or FedRAMP (U.S. General Services Administration, 2014), into a common controls framework for cloud computing that facilitates the risk assessment of using cloud services. CSA's Security, Trust & Assurance Registry (Cloud Security Alliance, 2015) enables comparison of cloud providers based on self-certification of cloud providers using the Cloud Controls Matrix. However, conducting audits based on these standards is mostly a manual process, still. Our proposed approach supports the automatic collection and evaluation of evidence based on policies that may stem from these frameworks and therefore could enable continuous certification.

Monitoring systems provide similar functionality to audit systems with respect to the collection of data and synthesizing metrics that are compared against defined thresholds. There are several solutions for IT monitoring such as Nagios (Nagios Enterprises, LLC, 2014) or Ganglia (Ganglia, 2015) and several big commercial solutions. However, they often have a very distinct heritage in data center, cluster and grid monitoring and are therefore not necessarily suitable for the cloud due its dynamic infrastructure and potential chaining of cloud providers. More specialized monitoring systems such as Amazon's CloudWatch (Amazon Web Services, 2016) or Rackspace's monitoring (Rackspace, 2016) are naturally proprietary and do not support chaining outside of the providers own set of services. The integration of an evidence collection system with such widely used monitoring systems is of great importance, since they provide deep insight into cloud services and therefore are considered valuable sources of evidence.

Auditing and monitoring in cloud computing has gained more momentum in recent years and a growing number of research projects is addressing their unique challenges. Povedano-Molina et al. (2013) propose Distributed Architecture for Resource manaGement and mOnitoring in cloudS (DARGOS) that enables efficient distributed monitoring of virtual resources based on the publish/subscribe paradigm. They utilize monitor agents to gather information for their centralized collector node. Katsaros et al. (2012) describe a similar approach to cloud monitoring with virtual machine units (VMU) that contain data collectors (scripts). Their focus is on self-adaptation of the monitoring system by adjusting monitoring intervals and other parameters. While they introduce isolation of tenants in cloud environments, they do not at this stage show how their system would work in a multi-provider scenario.

Massonet et al. (2011) propose an approach to monitoring data location compliance in federated cloud scenarios, where an infrastructure provider is chained with a service provider (i.e., the service provider uses resource provided by the infrastructure provider). A key requirement of their approach is the collaboration of both providers with respect to collecting monitoring data. Infrastructure monitoring data (from the IaaS provider) is shared with the service provider (SaaS provider) that uses it to generate audit trails. However, their main focus is to monitor virtual execution environments (VEE) that "are fully isolated runtime modules that abstract away the physical characteristics of the resource", which roughly

translates to virtual machines. The actual infrastructure layer is out of scope. Also, opposed to our approach, monitoring probes (data collectors) do not have a way to be dynamically deployed where needed, but rather are included in the VEE on deployment time.

Kertesz et al. (2013) follow the idea of tightly integrating monitoring into their management system for federated clouds, inorder to facilitate provider selection on the basis of availability and reliability metrics. They introduce service monitoring by reusing SALMonADA (Muller et al., 2012). Their approach is geared towards provider decision making for stateless services based on performance metrics and does neither include protection mechanisms and dynamic collector distribution that are required in a system for evidence collection in the cloud.

Montes et al. (2013) introduce an important aspect to cloud monitoring by also including the client-side in the data collection in addition to the cloud provider. However, they do not consider integrating third-party cloud providers as well.

Xie and Gamble (2012, 2013); Xie et al. (2014) describe an approach to inter-cloud auditing on the web service level, where audit assets are requested from a federated service.

3 COMPLEX CLOUD PROVIDER CHAINS FOR SERVICE PROVISION

While a lot of today's cloud use cases only involve one service provider for service provision, there are also many cases where multiple providers are involved. A prominent example is Dropbox that heavily uses Amazon's S3 and EC2 services to provide its own SaaS offering (Tom Cook, 2015).

There are several terms for the concept of provider chains such as *federated cloud*, *inter-cloud* and *cloud service composition*. In this work these terms are used synonymously. The concept of a provider chain is defined as follows:

1. At least two cloud providers (characterized by being either IaaS, PaaS or SaaS providers) are involved in the provision of a service to a cloud consumer (who can be an individual or business).
2. One of the cloud providers acts as a primary service provider to the cloud consumer.
3. Subsequent cloud providers do not have a direct relationship with the cloud consumer.
4. The primary service provider *must* be and the subsequent providers *can* be cloud consumers themselves, if they use services provided by other cloud providers.
5. The data handling policies agreed between the cloud consumer and the primary service provider must not be relaxed if data is processed by a subsequent provider.

The terms *cloud consumer* and *cloud customer* are used synonymously as well, while relying on the definition of a cloud consumer and auditor provided by NIST (Liu et al., 2011).

Figure 1 depicts a simplified scenario where three cloud service providers are involved in provisioning of a seemingly single service to a cloud consumer. The SaaS provider acts as the primary service provider, while it uses the PaaS provider's platform for hosting its service. The PaaS provider in turn does not have its own data center but uses resources provided by an IaaS provider.

The data handling policy applies to the whole chain (depicted by the dashed rectangle in Figure 1). Data handling policies thereby govern the treatment of data such as data retention (the deletion of data after a certain time), location (geographical restrictions) and security requirements (access control rules and protection of systems that handle the data).

All cloud provider produce evidence of their cloud operations.

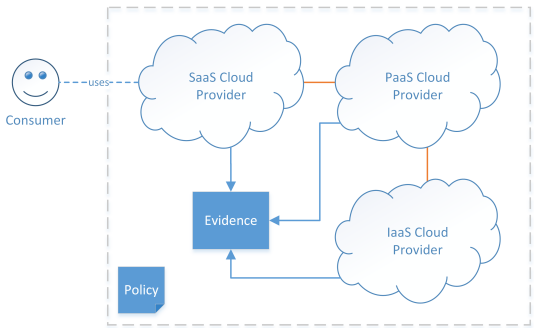

Figure 1: Cloud Provider Chains for Service Provision.

3.1 Evidence of Compliance in Cloud Provider Chains

At the core of any audit is evidence of compliance or non-compliance that is being analyzed. The types of evidence are closely linked to the type of audit (e.g., security audit, financial audit etc.) and are - from a technological perspective - especially diverse in the cloud due to the heterogeneity of its subsystems, architectures, layers and services. The notion of evi-

dence for cloud audits was discussed in our previous work in more detail (Rübsamen and Reich, 2013).

In general, we follow the definition of digital evidence that is "information of probative value that is stored or transmitted in binary form" (Scientific Working Groups on Digital Evidence and Imaging Technology, 2015). This means, that the types of evidence are diverse and include for example logs, traces, files, monitoring and history data from cloud management system like OpenStack's Nova service.

Evidence collection at a single cloud provider is already a complex task due to the diverse types of evidence sources and sheer amount of potentially required data that is being produced continuously. In a provider chain, these problems are intensified by the introduction of administrative domains and the lack of transparency regarding the number of involved providers and their relationships.

Another problem that is introduced with the concept of provider chains are changing regulatory domains. In a single-provider scenario, there are typically only two regulatory domains to be considered: i) the one that applies to the cloud consumer and ii) the one that applies to the cloud provider. With the addition of more cloud providers, the complexity of achieving regulatory compliance increases tremendously.

A simple example for such a case is the recent decision of the European Court in 2015 to declare Safe Harbor invalid, which leads to data transfers outside the European Union that are only governed by Safe Harbor to be invalid. In a provider chain, where a European Cloud provider transfers data about European individuals to another provider in the US, regulatory compliance could have been lost overnight. Here, it can be seen that regulatory domains can have a tremendous impact on how a compliance audit may have to look like, and on the type of evidence that may need to be collected at the different providers.

As previously suggested, the third major challenge for evidence collection in cloud provider chains is their inherent technological heterogeneity. APIs, protocols and data formats differ by provider and typically cannot be integrated easily (e.g., providers offering proprietary APIs). There are some approaches to homogenize some of the technologies, such as for example CSA CloudTrust Protocol (Cloud Security Alliance, 2016) that aims to provide a well-defined API that enables cloud providers to export transparency-enhancing information to auditors and cloud consumers. In this approach, the technological heterogeneity on the architectural level of the system is addressed by ensuring flexibility and extensibility and enabling the easy development of adapters for different evidence sources.

3.2 Audit Frameworks

Policy compliance assessment and validation is the main goal of our audit system. Policies can be of various kinds, for instance, a data protection policy is a typical tool used by cloud providers to frame their data protection and handling practices. In such policies, limits and obligations that a provider has to fulfill are defined. Typically, these documents are not machine-readable and are geared towards limiting liability of the provider.

Additionally, there are well-known standards, frameworks and industry best practices, which define various aspects of how data handling and protection should be implemented in practice. Such frameworks are for instance the well-known ISO27001 for information security management in general, COBIT for IT governance and CSA's Cloud Controls Matrix (CCM) (Cloud Security Alliance, 2014) for cloud-specific risk assessment. However, requirements and obligations stated by these frameworks are typically not available in a machine-readable format. There are approaches to making these requirements and obligations explicit in a machine-readable way, for example Accountability Primelife Policy Language (Azraoui et al., 2014) for defining data protection and data handling-related obligations for data processing in the cloud.

Traditionally, policy compliance is evaluated using audits and asserted with a certification of compliance (e.g., ISO27001 compliance certification). Typically, the intervals in which an audit is repeated are quite long (often yearly or longer). In the meantime, policy violations can potentially remain undetected for extended periods of time. One of our main goals is to address these periods of uncertainty by enabling the continuous assessment of cloud operations with respect to policy compliance. This is an important step towards continuous certification.

3.3 Auditing Cloud Provider Chains

According to NIST, a cloud auditor is defined as "A party that can conduct independent assessment of cloud services, information system operations, performance and security of the cloud implementation" (Liu et al., 2011). In our proposed system, the auditor is supported by a system for automated evidence collection and assessment. Evidence in the audit system is any kind of information that is indicative of compliance with policies or a violation of those. Typically, evidence is collected at the auditee. In general, an au-

ditee is an organization that is being audited, which in this paper, is always a cloud provider.

Complex cloud service provision scenarios introduce new challenges with respect to auditing. While in a typical scenario, where there is one cloud provider and one cloud consumer, policies can be agreed upon relatively easily between the two, this is not as easy in a provider chain. In fact, the cloud consumer might not be aware of or even interested in the fact that there is an unknown third-party that might have access to his data as long as his expectations regarding the protection and processing of his data are fulfilled. However, to assert compliance, the whole chain of providers, including data flows that are governed by the previously mentioned policies, have to be considered. This means that an audit with respect to a single policy rule may need to be split into several smaller evidence collection and evaluation tasks that are distributed among the providers.

For instance: assuming there is a restriction on data retention put in place that states that certain types of data (e.g., Personal Identifiable Information PII) has to deleted by the provider after a certain fixed period of time and no copies may be left over. This restriction can stem from regulatory framework such as the European Data Protection Directive or simply preferences that were stated by the data subject, whose data is being processed in the cloud. Such requirements can be formulated and enforced in for example the Accountability PrimeLife Policy Language (A-PPL) and its enforcement engine (A-PPL-E) Azraoui et al. (2015).

Auditing for compliance with such a policy requires, on a higher level, the check for the implementation of appropriate mechanisms and controls at each provider where the data itself or a copy thereof could have been stored. On a lower-level, the correct enforcement of the data retention rule could be evaluated in an audit by using evidence of data deletion that is being collected from all the cloud providers. In the overview depicted in Figure 1, that evidence could comprise of:

- Data deletion enforcement events generated by the service at the primary service provider as a reaction to the retention period being reached,
- Database delete log events produced by a database management system at the PaaS provider,
- And scan results on the IaaS level for data that may be still available outside of the running service in a backup subsystem provided by the IaaS provider.

The importance of widening the scope of audits in such dynamic scenarios is apparent, especially if at the same time the depth of analysis is widened beyond checking whether or not security and privacy controls are put in place.

4 APPROACHES FOR COLLECTING EVIDENCE IN CLOUD PROVIDER CHAIN AUDITS

There are several approaches available when it comes to collecting evidence for audit purposes in a service provider chain. These approaches differ in the following aspects:

1. The level of control an auditor has over the extent of the data that is being published, i.e. whether the auditor is limited to information that a provider is already providing or if he has more fine-grained control and access to a provider's infrastructure.
2. Technical limitations imposed by the technological environment, i.e. the extent to which cloud providers have to implement additional evidence collection mechanisms.
3. The expected willingness or acceptance to provide such mechanisms by the publishing service provider, i.e. the potential disclosure of confidential provider information and required level of access to the provider's systems.

In the remainder of this chapter, three approaches are described and rated by the above-mentioned factors.

The focus lies thereby on inspecting common components at two exemplary cloud providers that form a provider chain for the provision of a service. These components are:

AuditSys An audit system that enables automated, policy-based collection of evidence as well as the continuous and periodic evaluation of said evidence during audits.

Collector A component that enables the collection of evidentiary data such as logs at various architectural layers of the cloud, while addressing the heterogeneous nature of said evidence sources by acting as an adapter.

Source A location where evidence of cloud operations is generated.

Implementation details of these components are discussed in our previous work. The following discussion focuses on the different approaches to extend the system for cloud provider chains.

The first approach focuses on reusing already existing evidence sources by collecting via remote APIs of a cloud system. The second approach uses provider-provisioned evidence collectors and the third approach leverages the mobility of software agents (as used in the prototype implementation of our system) for evidence collection.

4.1 Remote API Evidence Collector

The first approach for collecting evidence that is relevant to automated auditing, leverages already existing APIs in cloud ecosystems. Several cloud providers, such as Amazon or Rackspace, already provide improved transparency over their cloud operations by providing their customers with access to proprietary monitoring and logging APIs (see (Amazon Web Services, 2016; Rackspace, 2016)). The extent to which data is shared is typically limited to information that is already produced by the clod provider's system (e.g., events in the cloud management system) and restricted to information that immediately affects the cloud customer (e.g., events that are directly linked to a tenant).

Data such as logs that are generated by the underlying systems are very important sources of evidence, since they expose a lot of information about the operation of cloud services. A specific example of such evidence are for instance: VM lifecycle events (created, suspended, snapshotted etc.) including timestamp of the operation and who performed. This can be requested from OpenStack's Nova service via its REST interface. The type of information is highly dependent on the actual system, the granularity of the produced logs and the scope of the provided APIs. For instance, on the infrastructure level, there are log events produced and shared that provide insight on virtual resource lifecycle (e.g., start/stop events of virtual machine).

Figure 2 depicts such a scenario. The AuditSys at Cloud A operates a collector that implements the API of the remote data source at Cloud B. It is configured with the access credentials of Cloud A, thus enabling the collector to request evidence from Cloud B. Furthermore, since different services may provide different APIs (e.g., OpenStack vs. OpenNebula API), the collector is service-specific. For instance, a collector implements the data formats and protocols as defined in the OpenStack Nova API to collect evidence about the images that are owned or otherwise associated with Cloud A as a customer of Cloud B.

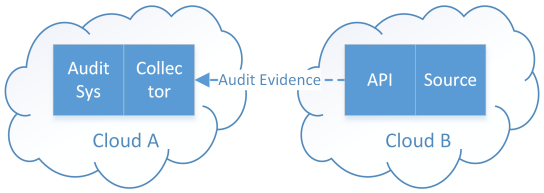

Figure 2: Remote API Evidence Collector.

4.1.1 Level of Auditor Control

The amount of evidence that can be collected is severely limited by the actual APIs that are provided by a cloud provider. It is either: i) the evidence that an auditor is looking for is immediately available because the provider already monitors all relevant data sources and makes that data accessible via the API or ii) the data is not available. Since a lot of the cloud provider's systems expose remote APIs anyway, they have to be considered. However, the completeness of the exposed APIs and therefore the completeness of the collected evidence is questionable due to the aforementioned reasons.

If an auditee for some reason does not implement or provide access to the audit system, an auditor may still collect evidence to a limited degree using this approach.

4.1.2 Technical limitations

If lower-level access to the providers infrastructure is required to collect evidence (e.g., log events generated on the network layer or block storage-level access to data), an auditor might not be able to gain access to that information.

4.1.3 Acceptance

This approach poses some challenges with respect to security, privacy and trust required by the auditee. Since the auditee is already exposing the APIs publicly, it can be expected that they will be used for auditing and monitoring purposes. The implementation of security and privacy-preserving mechanisms on the API-level is therefore assumed. However, the extent to which such mechanisms are implemented highly depends on the actual implementation of the APIs on the provider side.

While this way of providing evidence to auditors is likely to be accepted by cloud providers, it may be too limited with respect to the extent to which evidence can be collected at lower architectural levels.

4.2 Provider Provisioned Evidence Collector

In this approach, the audit system still is the main component for evidence collection. Here, all cloud providers that are part of the service provision chain are running a dedicated system for auditing. However, the instantiation and configuration of the collector is delegated to the auditee. The auditee assumes full control over the collector and merely grants the auditor access to interact with the collector for evidence collection.

The auditee (see Cloud B in Figure 3) provisions evidence collectors and provides access to them to the auditor. The auditor (who is using AuditSys at Cloud A) configures evidence collection for the audit to connect to the collectors at Cloud B.

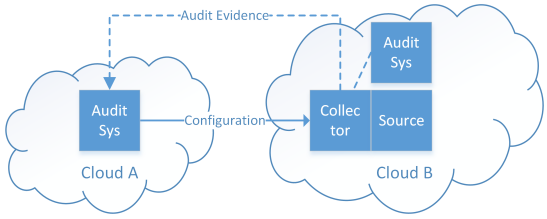

Figure 3: Provider-provisioned Evidence Collector.

4.2.1 Level of Auditor Control

The configuration of the evidence collector can be adjusted by the auditor to a degree that is controlled by the auditee (e.g., applying filters to logs). He is provided limited means to configure a collector but no direct, low-level access such as freely migrating the collector in the auditee's infrastructure. At any time, the auditee can disconnect, change or otherwise control the collector. An auditor may be put off by the limitations posed by this approach, since he is effectively giving up control over the central part of evidence collection and is relying solely on the cooperation of the auditee. For instance, simple tasks such as reconfiguring or restarting a collector may require extensive interaction between the two audit systems and potentially intervention by a human (e.g., an administrator).

4.2.2 Technical limitations

This approach is only limited by the availability of collectors for evidence sources.

4.2.3 Acceptance

In this approach, the auditee retains full control over the collector and the potential evidence that can be collected by it. The auditor can take some influence on the filtering of data that is collected from the evidence source and on general parameters, such as whether evidence is pushed by or pulled from the collector. Most of the baseline configuration though, is performed by the auditee (such as access restrictions and deployment of the collector). The auditor's ability to influence the collector is severely limited by the restriction of interactions to a well-defined set of configuration parameters and the evidence exchange protocol. This level of control that the auditee has over the evidence collection process may have positive influence on provider acceptance.

4.3 Mobile Evidence Collector

This approach is specific to a central characteristic of software agent systems, which is the ability to migrate over a network between runtime environments. In this approach, the migration of evidence collectors between separate instances of the audit system running at both Cloud A and B is proposed.

In our implementation, we opted for the well-known Java Agent Development framework (JADE) JADE (2014) for implementing collectors. The migration of collectors between providers is thereby performed by using JADE's mobile agent capabilities.

As depicted in Figure 4, the auditor prepares the required collector fully (i.e., agent instantiation and configuration) and then migrates the collector (shaded box named *Collector*) to the auditee (*Collector'*). There, the collector gathers evidence that is sent back to the auditor for evaluation. Generally however, agents do not cross from one particular administrative domain to another, but remain at one. In this case, the collector crosses from Cloud A's administrative domain to Cloud B. This may have significant impact on the acceptance of the approach.

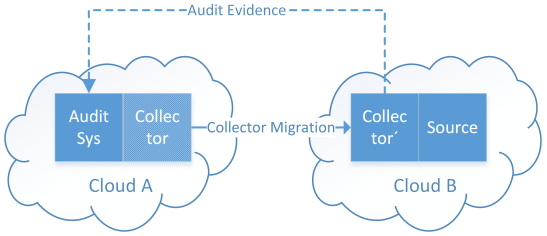

Figure 4: Mobile Evidence Collector.

4.3.1 Level of Auditor Control

The auditor retains full control over the type of collector and its configuration. The auditee may not in any

way change or otherwise influence the collector since this could be deemed a potentially malicious manipulation.

4.3.2 Technical limitations

Since the auditor knows most about the actual configuration required for a collector, it is logical to take this approach and simply hand-over a fully prepared collector to the auditee. However, this only works if both run the same audit system, or the auditee at the very least provides a runtime environment for the collector. In any case, this approach offers the most complete and most flexible way of collecting evidence at an auditee due to the comprehensive evidence collection capabilities.

4.3.3 Acceptance

The main problem with this approach is required trust by the auditee. Since the collector that is being handed over to him by the auditor is in fact software that the auditee is supposed to run on its infrastructure, several security, privacy and trust-related issues associated with such cross-domain agent mobility need to be addressed. Several security controls need to be implemented in order to make cloud providers consider the implementation of an audit system including the proposed approach of using mobile collectors.

The main security concerns of this approach stem from the fact that the auditee is expected to execute software on his infrastructure over which he does not have any control. He cannot tell for certain whether or not the agent is accessing only those evidence sources which he expects it to.

Without any additional security measures, it cannot be expected that any cloud provider is willing to accept this approach. However, with the addition of security measures such as ensuring authenticity of the collector (e.g., using collector code reviews and code signing) this approach becomes more feasible. The discussion of such measures depends on the technology used by the implementation and is out of scope of this paper. Without any additional measures, it can be assumed that this approach is only feasible, if the auditor is completely trusted by the auditee. In that case, this approach is very powerful and flexible.

4.4 Round-up

All three approaches for evidence collection in provider chains have their distinct advantages and disadvantages. Using remote API evidence collectors is simple, quickly implemented, securely and readily available, but severely limited regarding access to evidence sources. Using provider-provisioned evidence collectors is more powerful with respect to access to evidence sources, but requires more effort in the configuration phase and leaves full control to the auditee. Using mobile evidence collectors is the most flexible approach that allows broad access to evidence sources at the auditee's infrastructure and leaves full control over the evidence collection to the auditor. Therefore, a balance has to be struck between broad access to evidence sources when using mobile collectors (effectively having low-level access to logs and other files for evidence collection) and more limited access when using remote APIs (evidence limited to what the system that exposes the API provides).

In the audit system, the use of remote APIs is integrated due to its simplicity and mobile collectors due to their flexibility and powerfulness as the main approaches to evidence collection.

5 SCENARIO-BASED PROVIDER CHAIN AUDITING EVALUATION

In the previous Section 4, the approaches that can be taken when collecting evidence for auditing purposes in cloud provider chains were described. In this section, it is demonstrated how to incorporate the feasible approaches into an extension of the proposed audit system to enable automated, policy-driven auditing of cloud provider chains. The focus is put on the *Remote API Evidence Collector* and *Mobile Evidence Collector* approaches (see Section 4.1 and 4.3 respectively). The approach is validated by discussing a fictitious use case.

5.1 Audit Agent System

In Figure 5, an example deployment of the automated audit system is depicted. This deployment is not necessarily representative of real-world cloud environments but used to highlight possible combinations of services and data flows that can happen in a multi-cloud scenario. There are four cloud providers, which are directly or indirectly involved in the service provision. The SaaS provider A1 uses the platform provided by a PaaS provider B1, who does not have its own data center but uses computing resources provided by yet another IaaS provider C1. The IaaS provider C2 provides a low-level backup as a service solution that is used by provider C1. To enable auditing of the whole provider chain, each provider is

running its own instance of the audit system (AuditSys, as described in Section 4).

5.2 Provider Chain Auditing Extension

The auditor that uses AuditSys at the primary service provider A1 defines and configures continuous audits based on data protection and handling policy statements. Since these policy statements do not include any information about the service architecture, the auditor introduces his knowledge about the cloud deployment into the audit task, by defining evidence collection tasks that gather data on the PaaS and IaaS layer and also at the primary service provider. An audit task consists of collector, evaluator and notification agents. The type of evidence collection approach that has to be taken (as described in Section 4) is also defined by the auditor.

In this scenario it is assumed that all providers allow the auditor at A1 to collect evidence using the mobile evidence collectors and that the infrastructure providers also provide the auditor with access to their management system's APIs. As previously mentioned, the auditor is assumed trustworthy by all parties, which enables broad access to all cloud providers. Additionally, it is assumed that all cloud providers are acting in good faith and see the audit process as an opportunity to transparently demonstrate that they are acting in compliance with data handling policies.

As depicted in Figure 5, the auditor uses A1's AuditSys to define and audit task based on the data handling policy that is in effect. That task refers to the data retention obligation that was described earlier in Section 3.3. The retention time is defined as 6 months for every PII data record that is gathered about the users of provider A1. If the retention time is reached, the following delete process is executed as part of the normal operation of the service A1 provides:

1. The delete event fires at A1 due to max retention time being reached and the event is propagated to B1.
2. The data record is deleted from the database at B1.
3. The database is hosted on virtual machines provided by C1 and therefore does not require any delete actions.
4. A backup of the B1's database is available in C2's backup system and the delete action was not triggered in C2.

As part of the delete event, the following evidence is collected by the mobile evidence collectors as part of building an evidence trail for compliance evaluation at A1.

1. The data retention event is recorded as evidence by the collector running at A1.
2. The delete action of the database is recorded as evidence by the collector running at B1.
3. No evidence is recorded by the collector at C1 since there are no leftover copies such as virtual machine snapshots available.
4. The backup's meta-data such as creation timestamps are recorded as evidence by the collector at C2.

The evidence from all collectors (A1, B1, C2) is sent to the AuditSys at A1, where it is evaluated and a policy compliance statement is generated for the auditor. In this particular case, a policy violation is detected, because the audit trail shows that the record that should have been deleted is still available in a backup at C2. Provider A1, and B1 acted compliant by deleting the data, whereas C1 never stored a copy outside of B1's database.

5.3 Pre-processing and Intermediate Results of Audit Evidence Evaluation

The audit system uses a component at the AuditSys that is responsible for storing evidence records that are collected by the collector agents. Externally collecting evidence and merging it at a central evidence store that is only reachable via the network, can easily become a bottle-neck in audit scenarios where either a lot of evidence records are produced externally or where the record size is big. This obviously has significant impact on the scalability of the whole system.

The problem can be addressed by making the evidence store (which is just a specialized form of an agent with a secure storage mechanism) distributable and also by de-centralizing parts of the evidence evaluation process. There are generally two concepts:

1. Pre-processing: Pre-processing allows the evidence collector agent to apply filtering and other types of evidence pre-processing. The goal is to reduce the amount of collected evidence to a manageable degree (without negatively impacting the completeness of the audit trail) and to reasonably reduce the amount of network operations by grouping evidence records and storing them in bulk. For example, by filtering the raw data at the evidence source for certain operations, subjects, tenants, or time frames. Data that is not immediately required for the audit is filtered out.
2. Intermediate Result Production: A second preprocessing strategy is to move (parts of) the eval-

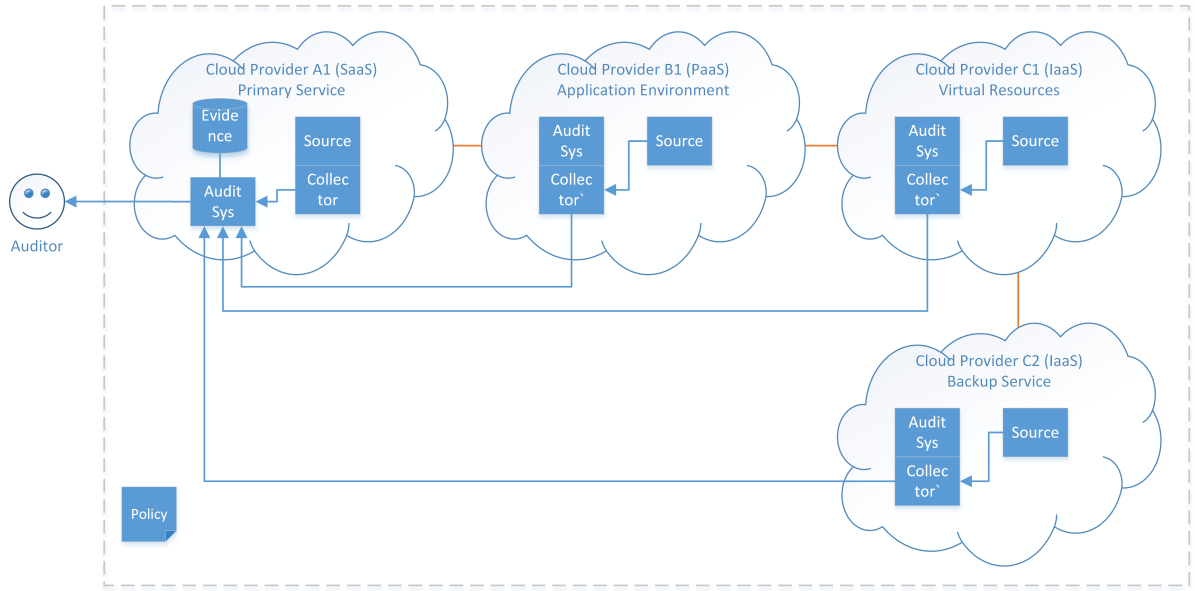

Figure 5: Provider Chain Auditing Architecture.

uation process near the collector. This means that the collected evidence is already reduced to the significant portions that indicate partial compliance or violation of policies. However, this strategy requires specific audit task types (where an audit result can be produced by combining several intermediate results).

The three concepts bring several implications with them with respect to privacy and security.

Pre-processing can be considered a manipulation of evidence. Therefore, the unaltered source upon which the pre-processing happened should be protected to later be able to trace pre-processed evidence back to its unaltered form.

Immediate result production effectively moves the evaluation of evidence step of the audit into the domain of the auditee, where it would be easy for him to manipulate the result. However, the same applies to the collection of evidence as well where an auditor can intentionally manipulate the evidence source or the collector.

This case is not considered in the current iteration of the system but it is assumed that cloud providers (auditees) are acting in good faith. This assumption can be justified by the potential increase in transparency and the associated strengthening of trust in the cloud provider that can mean a competitive advantage. On the other hand, intentional manipulation of evidence or intermediate results can have disastrous impact on a provider‘s credibility, reputation and trustworthiness upon detection.

6 CONCLUSIONS

Cloud auditing is becoming increasingly important as cloud adoption increases and compliance of data processing is put into focus of the cloud consumer. The key to making cloud audits a useful tool is the effectiveness of collection process that is used to build the basis for the evaluation of policy compliance or lack thereof.

While there are many systems for monitoring cloud providers (with varying level of completeness), there are fewer systems that automate audit tasks and even fewer still that enable continuous auditing, which is a key enabler of continuous certification. As long as there is only one cloud provider involved in service provisioning to the cloud consumer, monitoring and auditing is relatively simple (with the above mentioned restrictions). However, in more complex scenarios where there are chains of providers (or federations of cloud providers), current approaches are severely limited.

In this paper an extension to our previous work on automating continuous cloud audits that enables the collection of evidence across the boundaries of multiple cloud providers in a cloud provider chain was presented. The concept of cloud provider chains and three different approaches to evidence collection with their advantages and disadvantages were discussed. Furthermore, their implementation in an audit system was presented and validated using a scenario-based approach. It was shown how automated cloud audits can be extended to scenarios, where more than one

cloud provider is involved in the service provision.

In the future, the analysis of the different approaches and their integration in our system will be expanded in two main areas: i) expanding the security mechanisms that are already present to account for the notion of provider chains and ii) demonstrating the scalability and efficiency of the system.

ACKNOWLEDGEMENTS

This work has been partly funded from the European Commission's Seventh Framework Programme (FP7/2007-2013), grant agreement 317550, Cloud Accountability Project - http://www.a4cloud.eu/ - (A4CLOUD).

REFERENCES

Amazon Web Services (2016). Amazon cloudwatch. https://aws.amazon.com/de/cloudwatch/.

Azraoui, M., Elkhiyaoui, K., Önen, M., Bernsmed, K., Santana De Oliveira, A., and Sendor, J. (2014). A-PPL: An accountability policy language. In *DPM 2014, 9th International Workshop on Data Privacy Management, September 10, 2014, Wroclaw, Poland*, Wroclaw, POLAND.

Azraoui, M., Elkhiyaoui, K., Önen, M., Bernsmed, K., De Oliveira, A., and Sendor, J. (2015). A-ppl: An accountability policy language. In Garcia-Alfaro, J., Herrera-Joancomartí, J., Lupu, E., Posegga, J., Aldini, A., Martinelli, F., and Suri, N., editors, *Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance*, volume 8872 of *Lecture Notes in Computer Science*, pages 319–326. Springer International Publishing.

Bitkom Research GmbH (2015). Cloud Monitor 2015. https://www.kpmg.com/DE/de/Documents/cloudmonitor%202015_copyright%20_sec_neu.pdf.

Cloud Security Alliance (2013). Top threats to cloud computing survey results update 2012. https://downloads.cloudsecurityalliance.org/initiatives/top_threats/Top_Threats_Cloud_Computing_Survey_2012.pdf.

Cloud Security Alliance (2014). Cloud Controls Matrix. https://cloudsecurityalliance.org/research/ccm/.

Cloud Security Alliance (2015). Security, Trust & Assurance Registry. https://cloudsecurityalliance.org/star/.

Cloud Security Alliance (2016). Cloud Trust Protocol. https://cloudsecurityalliance.org/research/ctp.

Ganglia (2015). Ganglia. http://ganglia.sourceforge.net/.

Information Systems Audit and Control Association (2012). Control Objectives for Information and Related Technology (COBIT) 5. http://www.isaca.org/cobit/.

ISO (2005). ISO27001:2005. http://www.iso.org/iso/catalogue_detail?csnumber=42103.

JADE (2014). Java Agent DEvelopement framework. http://jade.tilab.com.

Katsaros, G., Kousiouris, G., Gogouvitis, S. V., Kyriazis, D., Menychtas, A., and Varvarigou, T. (2012). A self-adaptive hierarchical monitoring mechanism for clouds. *Journal of Systems and Software*, 85(5):1029 – 1041.

Kertesz, A., Kecskemeti, G., Oriol, M., Kotcauer, P., Acs, S., Rodríguez, M., Mercè, O., Marosi, A., Marco, J., and Franch, X. (2013). Enhancing federated cloud management with an integrated service monitoring approach. *Journal of Grid Computing*, 11(4):699–720.

Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., and Leaf, D. (2011). Nist cloud computing reference architecture. http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505.

Massonet, P., Naqvi, S., Ponsard, C., Latanicki, J., Rochwerger, B., and Villari, M. (2011). A monitoring and audit logging architecture for data location compliance in federated cloud infrastructures. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 1510–1517.

Montes, J., Sánchez, A., Memishi, B., Pérez, M. S., and Antoniu, G. (2013). Gmone: A complete approach to cloud monitoring. *Future Generation Computer Systems*, 29(8):2026 – 2040.

Muller, C., Oriol, M., Rodriguez, M., Franch, X., Marco, J., Resinas, M., and Ruiz-Cortes, A. (2012). Salmonada: A platform for monitoring and explaining violations of ws-agreement-compliant documents. In *Principles of Engineering Service Oriented Systems (PESOS), 2012 ICSE Workshop on*, pages 43–49.

Nagios Enterprises, LLC (2014). Nagios. http://www.nagios.org/.

National Institute of Standards and Technology (2011). Guidelines on security and privacy in public cloud computing. http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144.pdf.

National Institute of Standards and Technology) (2013). Security and privacy controls for federal information systems and organizations. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf.

PCI Security Standards Council (2015). Payment Card Industry Data Security Standard (PCI-DSS). https://www.pcisecuritystandards.org/.

Povedano-Molina, J., Lopez-Vega, J. M., Lopez-Soler, J. M., Corradi, A., and Foschini, L. (2013). Dargos: A highly adaptable and scalable monitoring architecture for multi-tenant clouds. *Future Generation Computer Systems*, 29(8):2041 – 2056.

Rackspace (2016). Rackspace cloud monitoring. http://www.rackspace.com/cloud/monitoring.

Rübsamen, T., Pulls, T., and Reich, C. (2015). Secure Evidence Collection and Storage for Cloud Accountability Audits. In *CLOSER 2015 - Proceedings of the 5th International Conference on Cloud Computing and Services Science, Lisbon, Portugal, May 20 - 22, 2015*. SciTePress.

Rübsamen, T. and Reich, C. (2013). Supporting cloud accountability by collecting evidence using audit agents. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 1, pages 185–190.

Rübsamen, T. and Reich, C. (2014). An Architecture for Cloud Accountability Audits. In *1. Baden-Württemberg Center of Applied Research Symposium on Information and Communication Systems SInCom 2014*.

Rübsamen, T., Reich, C., Wlodarczyk, T., and Rong, C. (2013). Evidence for accountable cloud computing services. http://dimacs.rutgers.edu/Workshops/TAFC/TAFC_a4cloud.pdf.

Scientific Working Groups on Digital Evidence and Imaging Technology (2015). SWGDE and SWGIT Digital & Multimedia Evidence Glossary. https://www.swgde.org/documents/Current%20Documents/2015-05-27%20SWGDE-SWGIT%20Glossary%20v2.8.

Tom Cook (2015). Dropbox at AWS re:Invent 2014. https://blogs.dropbox.com/tech/2014/12/aws-reinvent-2014/.

U.S. General Services Administration (2014). Federal Risk and Authorization Program. http://www.fedramp.gov.

Xie, R. and Gamble, R. (2012). A tiered strategy for auditing in the cloud. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 945–946.

Xie, R. and Gamble, R. (2013). An architecture for cross-cloud auditing. In *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, CSIIRW '13, pages 4:1–4:4, New York, NY, USA. ACM.

Xie, R., Gamble, R., and Ahmed, N. (2014). Diagnosing vulnerability patterns in cloud audit logs. In Han, K. J., Choi, B.-Y., and Song, S., editors, *High Performance Cloud Auditing and Applications*, pages 119–146. Springer New York.

Towards Auditing of Cloud Provider Chains Using CloudTrust Protocol

Thomas Rübsamen[1], Dirk Hölscher[1] and Christoph Reich[1]
[1]*Institute for Cloud Computing and IT Security, Furtwangen University, Robert-Gerwig-Platz 1, Furtwangen, Germany*
{*thomas.ruebsamen, dirk.hoelscher, christoph.reich*}*@hs-furtwangen.de*

Keywords: Cloud Computing, Audit, Federated Cloud, Security, Digital Evidence

Abstract: Although cloud computing can be considered mainstream today, there is still a lack of trust in cloud providers, when it comes to the processing of private or sensitive data. This lack of trust is rooted in the lack of transparency of the provider's data handling practices, security controls and their technical infrastructures. This problem worsens when cloud services are not only provisioned by a single cloud provider, but a combination of several independent providers. The main contributions of this paper are: we propose an approach to automated auditing of cloud provider chains with the goal of providing evidence-based assurance about the correct handling of data according to pre-defined policies. We also introduce the concepts of individual and delegated audits, discuss policy distribution and applicability aspects and propose a lifecycle model. Our previous work on automated cloud auditing and Cloud Security Alliance's (CSA) CloudTrust Protocol form the basis for the proposed system for provider chain auditing.

1 INTRODUCTION

An important problem that is commonly associated with the use of cloud services is the loss of control about who is processing data, how it is used and whether or not it is shared with third parties. A possible approach to mitigate this problem is to provide additional information to stakeholders (cloud users, cloud auditors). That information can be obtained as a part of regular cloud audits where evidentiary information about data processing is collected and compared against agreed-upon policies (such as terms of service or privacy policies). This way, what happens in the cloud becomes more transparent to the user, which could lead to improved trust in the cloud by providing additional information on data processing in a cloud service. Therefore, a system for automated audits is needed that provides meaningful evidence and shows how and where data is processed.

With lacking transparency comes low trust (Knode, 2009). Low trust in the cloud (especially in the security and privacy of data) hinders the growth of cloud computing as a business (Cloud Security Alliance, 2013). Customers are less likely willing to move their business applications into the cloud when they have no chance evaluating, whether or not their data is processed according to company policies. Transparency is not just showing cloud consumers how and where data is processed, it is also important to know by whom the consumer's data is processed. However, transparently showing by whom the data might be processed is not common practice today. Furthermore, cloud providers can incorporate services provided by Nth-level providers into their own, which makes it even harder for the auditor to follow where the consumer's data is currently being stored.

It makes sense for Software as a Service (SaaS) providers to utilize a third-party Infrastructure as a Service (IaaS) provider for service hosting. We consider such cloud provider chains to become increasingly important to look at, when a complete picture about data processing in a particular service should be provided. Most importantly, the compliance with data processing and privacy policies of all involved parties needs to be assessed. For instance, in some situations a cloud provider might be forced to transfer consumer data from one cloud provider to another, or from one geographic location to another for load-balancing or cost optimization. In such a case, the new provider has to ensure that all policies are covered just as much as the others in the chain have to.

Cloud auditing is becoming increasingly important for certification (e.g., FedRAMP (FedRAMP, 2015), ISO27001 (ISO, 2013) with the proposed ISO27017 (ISO, 2015) and ISO27018 (ISO, 2014)). However, the required audit is still largely a manual process. In this paper, we propose an extension to

our previous work see (Rübsamen and Reich, 2013; Rübsamen et al., 2015)) on automated, evidence-based cloud auditing, that provides improved transparency to cloud stakeholders about data processing in the cloud. The extension introduces the concept of cloud provider chains, data processing and privacy policies with an extended scope on all involved providers, and audit evidence exchange. The main contribution of this paper introduces the concept adapting CSA CloudTrust Protocol (CTP) (Cloud Security Alliance, 2015) for the use in inter-provider exchange of evidence during cloud audits.

Using the introduced CTP extension, additionally generated evidence (i.e., information not specified in the original CTP) will be used to enhance CTP reports. Furthermore the extension enables auditing of third-party contractors within provider chains to show that each statement made by the provider, with respect to the users established policies (e.g., data location, service availability), is fulfilled.

This paper is organized as follows: In the next Section 2 related work is presented. Section 3 introduces the concept of service provision chains and discusses the scopes and applicability of policies. In Section 4 two approaches towards cloud auditing are introduced. After that, we propose a lifecycle model for delegated auditing of cloud provider chains in Section 5. Next, we focus on data exchange patterns (Section 6) that are used in Section 7, where we present our proposed system. Following that, section 8 evaluates the presented results using a scenario description and a discussion of the threat model. In Section 9 we conclude this paper.

2 RELATED WORK

Security and privacy auditing are increasingly important topics in cloud auditing. They demonstrate that security controls are put in place by the provider and also that they are functioning correctly (i.e., data protection mechanisms are working correctly and effectively). There are some projects working on the architectural and interface level regarding the automation of security audits, such as the Security Audit as a Service (SAaaS) project (Doelitzscher et al., 2012; Doelitzscher et al., 2013). SAaaS is specifically designed to detect incidents in the cloud and thereby consider the dynamic nature of such ecosystems, where resources are rapidly provisioned and removed. However, SAaaS does not address provider chain setups or treat gathered data as evidence.

ABTiCI (Agent-Based Trust in Cloud Infrastructure) describes a system used for monitoring (Saleh, 2014). All relevant parts of a cloud infrastructure are monitored to be able to detect and verify unauthorized access. Integrity checks are done at boot-time, using Trusted Platform Module (TPM) boot or at runtime. Monitoring hardware and software configurations allow the system to detect changes at runtime. The aforementioned system is similar to our approach. Instead of using agents we utilize CTP. Furthermore, our approach relies on evidence collection through audits with pull and trigger mechanisms.

A centralized trust model is introduced by Rizvi et al. (Rizvi et al., 2014). Trust between consumer and provider is established by using an independent third-party auditor. With the adoption of a third-party auditing system, consumers are able to create baseline evaluation for providers they have never worked with to generate initial trust. The model acts as a feedback mechanism providing valuable insight into the providers processes. After initial trust was generated the third-party auditor continues to obtain trust values for the consumer. We see initial trust in the provider as a given factor and focus on obtaining trust values based on evidence within a multi-provider scenario.

A completely different approach is proposed by Gonzales et al., where the authors introduce an architecture for the measurement of integrity and confidentiality of a cloud provider (Gonzales et al., 2015). Their approach is based on best practices and security metrics. It uses trust zones to delineate resources (physical, logical or virtual) within multi-tenant IaaS infrastructures. Such a zone is used to separate interests. Sensitive business data is located in a Gold Zone, non-business partners are located in a less privileged zone and can't access the Gold Zone. The focus in this work lies in the separation of concerns. Trust is generated using best practices and security metrics. There is no provider auditing involved, but everything is estimated based on metric values. Whereas, in our approach metrics can be used to collect additional information but the focus lies in evidence collection.

The DMTF is also working on cloud auditing with the Cloud Audit Data Federation (CADF) working group. They focus on developing standardized interfaces and data formats to enable cloud security auditing (Distributed Management Task Force, Inc. (DMTF), 2014). A similar project is the Cloud Security Alliance's Cloud Trust Protocol (CTP), which defines an interface for enabling cloud users to "generate confidence that everything that is claimed to be happening in the cloud is indeed happening as described, ..., and nothing else" (Cloud Security Alliance, 2015), which indicates an additional focus on providing additional transparency of cloud services. The latter two projects, however, do not elaborate on

how the interfaces should be implemented nor do they describe explicitly focus on privacy and accountability. We use CTP as a basis and propose its extension and use in our proposed auditing system to enable automated auditing of cloud provider chains.

There are approaches that deal with checking compliance with data location policies. The principle of location transparency of data in the cloud (i.e., a user does not know in which server, data center or even country a specific data object is stored) is contrary to data locality requirements some cloud consumers have to fulfill (e.g., a legal obligation to ensure a certain geographic storage location). Massonet et al. propose a system that exposes infrastructure-level location monitoring information to the cloud consumer (Massonet et al., 2011). We use data location as an use case for the demonstration of our system.

A crucial part of cloud auditing is the collection of data on which an audit can be based upon. That data can be produced on all architectural layers of the cloud (e.g., on the bare-metal, in a VM, in a subsystem). Several approaches to addressing the unique requirements of cloud logging have been proposed. For instance, Marty presents a logging framework and guidelines for IaaS and SaaS logging (Marty, 2011). We have also previously discussed the different sources of data that can be used as evidence during audits. (Rübsamen and Reich, 2013). We demonstrate, how such data can be collected and more importantly used for auditing in a multi-provider scenario.

The CloudTrust Protocol (Cloud Security Alliance, 2015) establishes a mechanism that allows users to audit a CSP. An auditor can choose from a set of transparency elements for instance: geographic location of data objects and affirmation or results of latest vulnerability assessments. CTP has 23 pre-defined transparency elements and supports user-specified elements on which cloud consumer and provider agreed on. The purpose of the CTP is to transparently provide the user with important information about the cloud to show that processing is done as promised. By providing information about the inner-workings of the cloud service (with respect to the transparency elements), trust between the cloud provider and the consumer is supposed to be strengthened. If a consumer can trust his provider he is more likely willing to move sensible business processes into the cloud.

There are two main problems the protocol tries to solve:

- Restoration of control and freedom of choice at the cloud consumer by enabling him to specifically request information on configurations, vulnerabilities and data integrity.
- The provision of a standardized process, which enables providers to generate and expose additional information with respect to the transparency elements.

CTP needs to be beneficial for both cloud providers and consumers. Providers won't invest into structural changes of their services if the expected payoff is small. For this reason the protocol can be adjusted to the trust needs of consumers as well as operational circumstances of the provider. Only the request/response process and the associated data formats are specified, whereas there are no additional restrictions put on the actual implementation of the information gathering process.

Communication handling in CTP is done by two managers. The auditor is using CTP's *Request Manager* whereas the provider is using the *Response Manager*. These two architectural components are responsible for communication, tracking pending requests, CTP translation into service specific API calls and data conversion into CTP format. The data format for reporting is based on XML (used in an extended form in the proposed system) for the 2.0 version, respectively JSON for the currently proposed version 3.0 of CTP. In general CTP's protocol design follows the RESTful paradigm.

3 CLOUD SERVICE PROVISION CHAINS

In this chapter, we describe complex cloud service provision scenarios. We thereby focus on use cases where multiple cloud service providers are involved in the provision of a single cloud service (as perceived by the cloud consumer). In these use cases, the agreement on data processing and privacy policies, that apply on the whole service provider chain, can quickly become a difficult problem. Therefore, the auditing of compliance with such policies along the chain, both increases in complexity and difficulty.

In the following discussion of the different scopes of policy applicability, we assume the following definitions:

Cloud Consumer
We use the definition provided by NIST where a cloud consumer is "a person or organization that maintains a business relationship with, and uses service from, Cloud Providers" (Liu et al., 2011).

Cloud Service Provider
We use to the definition provided by NIST where a cloud provider is "a person, organization, or entity

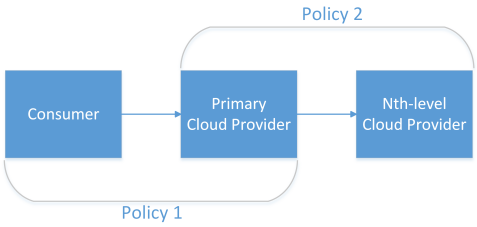

Figure 1: Audit Policy Scopes.

responsible for making a service available to interested parties" (Liu et al., 2011). Furthermore, we define the provider facing the consumer as the primary provider and each succeeding provider who interacts with the primary provider is defined as a Nth-level provider.

Cloud Service Provider Chain
A Cloud Service Provider Chain is characterized by at least two cloud service providers being part of providing a service by composing their individually offered services.

In the following, we describe three different scopes of policy applicability as depicted in Figure 1. In that scenario we assume a cloud service (provided by a primary cloud provider) that is provided to a service consumer, while utilizing an additional third-party service (provided by the sub-provider on the Nth level).

Scope A: Cloud Consumer / Primary Cloud Provider In a typical cloud use case, a consumer uses the services provided by a single cloud service provider to accomplish a given task. The details of the service usage are governed by terms of service agreements, privacy polices etc. In this most common scenario, the cloud consumer and the cloud provider agree on these terms before any service is provisioned. Typically, this happens during a registration or contract agreement phase. With respect to data flow between the consumer and the provider, this means that data processing is performed by the cloud provider in compliance with the agreed-upon policies (see *Policy 1* in Figure 1). Personal data that is disclosed by the cloud consumer to the cloud provider as part of regular service use is processed by the cloud provider according to the limits defined in the policy.

Scope B: Primary Cloud Provider / Nth-level Cloud Provider Similar to the approach described in Scope A, there may be similar agreements between cloud providers. For instance, the primary cloud provider may require resources from the sub-provider, e.g., to extend its own service offering, to address peak loads in service usage or to outsource internal processes such as backups. In this case, the primary cloud provider (as depicted in Figure 1) becomes a cloud consumer itself. The integration of cloud services provided by a sub-provider in cloud services provided by the primary cloud provider is governed by a contractual agreement between the two providers (see *Policy 2* in Figure 1).

Scope C: Cloud Consumer / Nth-level Cloud Provider In case of a cloud scenario, where multiple service providers are involved in the provisioning of a single service, the cloud consumer may not necessarily be aware of this. Since the cloud consumer has only contact with its immediate provider (primary cloud provider in Figure 1), he might not necessarily be aware, that the primary cloud provider is using an additional external service. A typical example for such a scenario is a SaaS provider hosting its services on resources provided by an infrastructure provider, or a SaaS provider that integrates another SaaS provider's service for data processing. Additionally, a silent change of the supplementary service provider can be imagined, when the primary provider switches to another service (e.g., uses another infrastructure provider for cost efficiency reasons). In this case, the restrictions that governs the policy agreement between the cloud consumer and the primary cloud provider (i.e., Policy 1) must also apply to the sub-provider, if data owned by the cloud consumer is transferred between them. This is the case, when either: i) similar policies policy rules exist in Policy 1 and 2, where the rules defined in Policy 2 are at least as strict as equivalent rules defined in Policy 1 (in this case, a matching of whether or not rules from policy 1 and 2 are compatible needs to be performed), or ii) the downstream provider accepts rules from policy 1 directly.

4 AUDITING CLOUD PROVIDER CHAINS

In this chapter, we illustrate different variations of auditing cloud provider chains. We thereby focus on traditional individual audits and delegated provider audits. Furthermore, we present several information delivery patterns.

4.1 Individual Provider Audits

Figure 2 describes the process of auditing individual providers in a service provision chain. All policies

will be distributed to each provider (as seen in Figure 2). Policy distribution can either be:

1. Manual policy evaluation: This approach is based on the specified policy documents (e.g., terms of service in human-readable form) given by the provider. The auditor manually maps statements of such documents to information requests for the providers (e.g., asking for specific process documentation or monitoring data and audit logs).
2. Deploying machine-readable policies: In this approach the auditor deploys a machine-readable policy document (XML, JSON) onto the provider. The provider will then automatically audit the tasks specified within the document. The auditor can request the results for the audited policy rules to verify if everything is fulfilled. The policy needs to be deployed to each involved provider. Within this approach new policies can easily be added and deployed for automated auditing.

The audit results are used to assure the consumer that policy and rule compliance is given or not. As previously described, a service provision chain contains at least one provider. In this case, two providers - a primary and a 2nd-level provider. To audit the service as a whole, it is necessary to audit each provider separately and then aggregate the results to form a complete picture of the service from an audit perspective. This means, that regarding data handling policies (e.g., location restrictions, access control etc.), each provider that holds data is audited. The same is true for the auditing of security and privacy controls that are put in place at the providers. Obviously, the consumer-facing provider has to transparently disclose all his sub-providers and notify auditors about every sub-provider his data was stored at and where his data is currently stored. Even though not every provider will get the consumer's data, the auditing process gains more complexity with an increasing number of Nth-level providers. Requests must be sent to each provider separately and each provider will deliver audit reports to the auditor.

4.2 Delegated Provider Audits

An alternative to individual audits are what we call delegated audits, where the auditor only interfaces with the primary service provider that in turn takes over the auditing of its sub-provider(s). Therefore the auditor only has to audit the primary service provider to obtain policy compliance results of all involved service providers. This allows less influential stakeholders such as the cloud consumer to act as an auditor towards the primary provider while not having the same

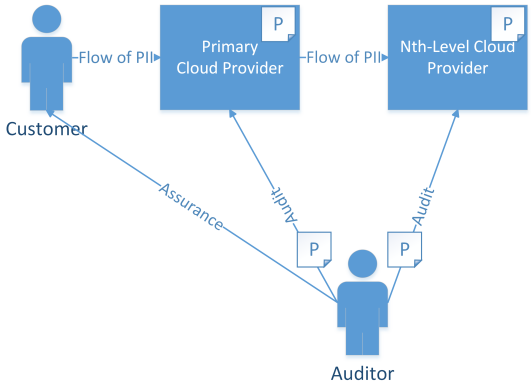

Figure 2: Individual Audit.

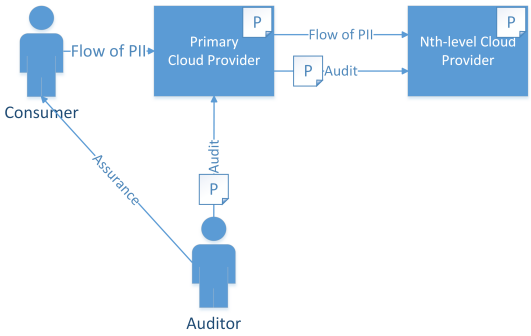

Figure 3: Delegated Audit.

rights towards the Nth-level provider(s). Whereas the individual audit scenario is an example of how chain audits could be performed with more influential stakeholders, such as data protection authorities. Figure 3 depicts the delegated provider audit scenario. Every audit request is sent to the primary provider who will then extract CTP calls from a previously deployed policy document (machine-readable document deployed by the auditor). Since the primary provider is acting as a mediator he has to delegate requests and communication. Existing problems regarding policy compliance is of major concern for the primary provider because complaints will always be addressed to him, even if he is not responsible for a failed audit. For the case a given audit response did not satisfy policy compliance the consumer will contact the primary provider with a complaint (e.g, data was transferred outside valid location). On the other hand the consumer's payoff can be much higher due to the centralized structure using a mediator ensuing low complexity for the auditor. Therefore, he can always rely on the data controller to forward his request to the data holding sub-provider without the need of adaption (send requests to different entities, use different API-calls).

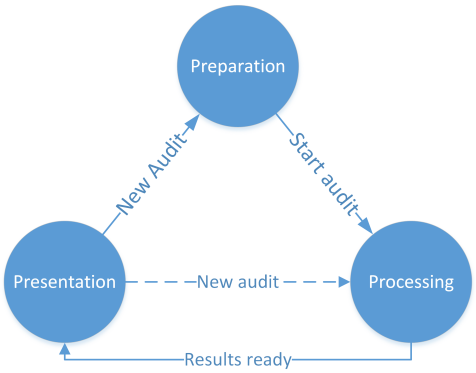

Figure 4: System Lifecycle.

5 Audit Lifecycle in Delegated Provider Audits

In the following section the audit system lifecycle is described. Figure 4 illustrates the three phases: i) Preparation, ii) Processing and iii) Presentation. In the following, we describe each phase in more detail:

5.1 Preparation Phase

The first phase of the lifecycle is the Preparation Phase in which the system is prepared. The most important task during the Preparation Phase is resource identifier distribution, which is required for request handling.

Request handling is done using unique resource identifiers (URI), which are used to identify any kind of resource that is part of an audit. A URI unambiguously identifies an object within a provider's domain. In our approach, each provider has its own namespace in which identifiers can be assigned arbitrarily.

The preparation process *Policy adding* allows the auditor to create new rules based on already existing policies. For instance, he can specify a new data location rule to ensure that his data will not leave his countries jurisdiction. Newly added rules are written into a machine-readable audit policy that describes evidence that is to be collected, and checks that are to be performed during the audit. From the new rule, auditable elements are derived, that an automated audit process provides all necessary information to enable the possibility of policy compliance assessing. Auditable elements include for example the location of data, logs and configurations.

The *Policy mapping* process, maps each new added rule or policy to transparency elements and associated requests. If a newly added rule cannot be mapped to an already existing transparency element a new element needs to be created. The mapping is done based on the specified policies. For this reason the policy adding process is limited to already defined policies and the associated rules within the contract. During the mapping each non-standard policy (i.e., a policy that requires a transparency element that is not part of CTP) will receive an URI and all necessary data sources needed to answer a request. The mapping process generates URIs and defines all auditable attributes for an element.

The preparation process *Policy distribution* propagates the resource identifiers throughout the system. Each sub-provider sends his resource identifiers to the primary provider. Afterwards, when all identifiers are known by the primary provider, he will forward them to the auditor.

5.2 Processing Phase

With the end of the Preparation Phase the second lifecycle phase starts. In Processing Phase all elements will be collected. For instance, all essential information for the inquired elements are retrieved from the evidence store and written into a CTP response. A policy evaluation is done to determine the policy compliance. All information, collected for one element are written into a response and sent back to the requesting entity.

5.3 Presentation

The last phase in the lifecycle is the Presentation Phase. Within this phase the auditor will be presented with the audit results. Thereby, each requested element will be presented to him containing the name of the policy rule as well as its achieved compliance state.

The lifecycle is complete, when the results were presented to the auditor. After this the lifecycle can continue with Preparation Phase again. Returning to the Preparation Phase is necessary if new policies/rules were added or in a continuous auditing scenario, where policy compliance is audited in short intervals or event-driven (e.g., on new or changed policy, on infrastructure change or on custom triggers defined by the auditor). During the new cycle only newly generated URIs will be distributed.

6 Audit Information Exchange

In this section two information exchange patterns are described that are used for different purposes in our proposed system. Obtaining information as well

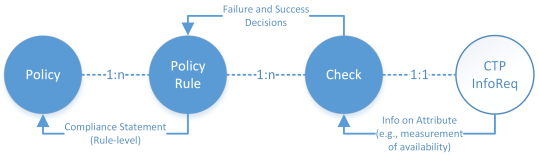

Figure 5: Individual Audit Data Requests.

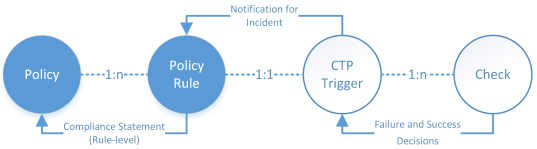

Figure 6: Triggered Audit Data Exchange.

as providing information can be a difficult task depending on how timely data is needed. Therefore, we specify a pull pattern (see Section 6.1) for non-critical information (e.g., evidence used during infrequent audits) and a push pattern (see Section 6.2) for critical information (evidence used during continuous audits).

6.1 Data Requests (Pull)

Within a pull scenario audit results (transparency elements request results) are only delivered, if there is an existing pull request for an element. Figure 5 shows the individual data request sequence. The shown figure does not imply the use of a pull mechanism, but can be used with one. For each pending request its corresponding data is pulled form a database. Pulled information will run through a compliance check afterwards, to verify if a requested policy with its corresponding rule is fulfilled. This means, that audit data request results are only delivered to the auditor if there is an existing request. When a request for transparency elements arrives the data is pulled from a database (evidence store) and will then go through a validity check to ascertain if a policy is fulfilled or not. The auditor will not necessarily receive critical information in a timely manner, unless he requested the data during the incidents occurrence, which is highly unlikely. Also, choosing the right request interval can quickly become a scalability issue. Polling for new data in relatively short intervals can introduce load problems at the auditing system. This issue is a common problem with pull / polling mechanisms. However, reasonable and usually quite common interval choices such as hourly, daily, weekly and monthly reports do not introduce such problems.

6.2 Triggered Notification (Push)

Critical information that quickly needs to be processed (e.g., forwarded to an analysis tool or presented to an auditor) like security breaches or integrity violations are time-critical and therefore cannot rely on transport via pull mechanism. Immediate notifications are necessary to avoid data control deprivation. Push mechanisms promise a more reactive and rapid way of transporting critical data. Push mechanisms are typically associated with an event-driven approach, where an event is fired when a condition is met. Such an event could be the occurrence of a data relocation. Such an event needs to be audited if there are policy rules limiting allowed geographical locations. Figure 6 shows the push notification process. An auditor can specify the severity of an occurrence called trigger, to ensure that only significant information are pushed. A configured trigger will only trigger when the condition set by the auditor is fulfilled. After the occurrence an incident notification will be pushed to the auditor. There is no need to define a trigger for all possible auditable elements. Therefore, it is not feasible to send a notification for every small change in the system. If a breach occurred or a vulnerability was found during the audit, a notification is pushed and countermeasures can be taken faster which will immensely reduce reaction time.

7 EXTENDING CLOUDTRUST PROTOCOL FOR PROVIDER CHAIN AUDITING

In our approach, we leverage CTP as a means for evidence exchange between cloud providers in complex cloud auditing scenarios. Additional functions and components are located above the protocol (as seen in Figure 7) and are responsible to exchange requests and responses with the CTP. This structure enables us to utilize the benefits of CTP out of the protocol's operational area without changing the protocol itself. Although the operational structure of CTP remains unchanged some optimisations for audit reports are required to be able to transfer additional information e.g. more detailed user access lists. In this case, the additional information would give the auditor not only authorized users but also since when they have authorization and who authorized user permissions.

Figure 7 illustrates the systems architecture in a two provider scenario. Within the figure it is assumed, that the Preparation Phase did end and all for the audit request necessary policies and rules were already mapped and distributed. Incoming transparency element requests will directly go to the Remote Evidence Collection component. In the following paragraph the system components of our proposed approach are de-

scribed:

Remote Evidence Collection

- **Request handling:** Every incoming audit request will arrive at the Request handler of the primary provider. A decision is made which resource identifier should be used based on current data location (Nth-level provider, primary provider). The resource identifiers are used to set up CTP-calls. Therefore, it will forward each request to the CTP Request Manager. Each request is processed separately to guarantee that context information or states do not get mixed up.
- **CTP Request Manager**: The Request Manager, sends each given request to the CTP Response Manager of a Nth-level provider(solid line in Figure 7 between both providers) using a pull pattern (see Section 6.1). Inter-provider communication is initiated by the Request Manager.
- **Context information extraction:** An incoming CTP-response contains the general response (specified in (Cloud Security Alliance, 2015)) as well as the corresponding context information and the compliance state for the requested element. The context information are extracted from the response and securely stored inside the evidence store. The remaining information which are used for report creation are stored as well for the audit report creation.
- **CTP Response Manager**: After receiving a request the Response Manger pulls data from the evidence store if the Nth-level provider is not able to determine the compliance state of his own or receives the data from the Evaluator. Obtained results are packed into a CTP-response and sent back to the primary providers CTP Request Manager. In case a trigger is fired the Response Manager will push the response immediately to the primary providers CTP Request Manager even in the absence of an audit request for the triggered element. A primary provider might be a Nth-level provider of another provider and thus needs a Response Manager. Requests for context information are send from the auditor to the primary providers Response Manager. Like a normal response the Manager pulls the context information for the requested object from the evidence store and writes them into a CTP-Response.

Evidence Store: The Evidence Store is a database containing all audit results (including context information) for the primary and its Nth-level providers. Each participating provider has its own evidence store where his achieved audit results are stored and can be pulled from by pending requests. The main purpose of the evidence store is to provide audit evidence and to make them accessible to the auditor. If no data was relocated to a Nth-level provider, a response will be generated from audit entries for the primary provider located in the evidence store.

Evaluation and Reporting

- **Evaluator**: The Evaluator runs policy compliance checks on all obtained results used for report creation. Achieved results can get one of three possible compliance states depending on the level of fulfillment:
  - **State 1 successful:** The results obtained from the database fulfills the policy.
  - **State 2 partially:** A policy is partially fulfilled.
  - **State 3 failed:** No records for this policy were found or the given results were unsatisfying.

  Configured triggers (see Section 6.2) are fired if a compliance check for a request failed or a deviation from the trigger specification is identified.
- **Report creation:** The stored content (state, CTP transparency elements results) is used to create the final report. The report can be of different types, for instance a representation of the results on a web dashboard or in a auto-generated document.
- **Notification:** An audit can take some time to finish. This largely depends on the size and scope of the audit. Therefore, asynchronous mechanisms are required to present audit results. An auditor can be notified via mail when his audit is finished and his audit report is available.

Implementation of each above described part is mandatory for every provider. It is possible that a primary provider is a Nth-level provider in a different audit-chain, whereas a Nth-level provider might be a primary provider in another audit chain.

8 EVALUATION

In the following Section, we evaluate our proposed approach towards auditing cloud provider chains. We split the evaluation in two parts: i) a functional evaluation using a fictitious cloud scenario with two providers involved in the service provision, and ii) a security analysis of the proposed approach.

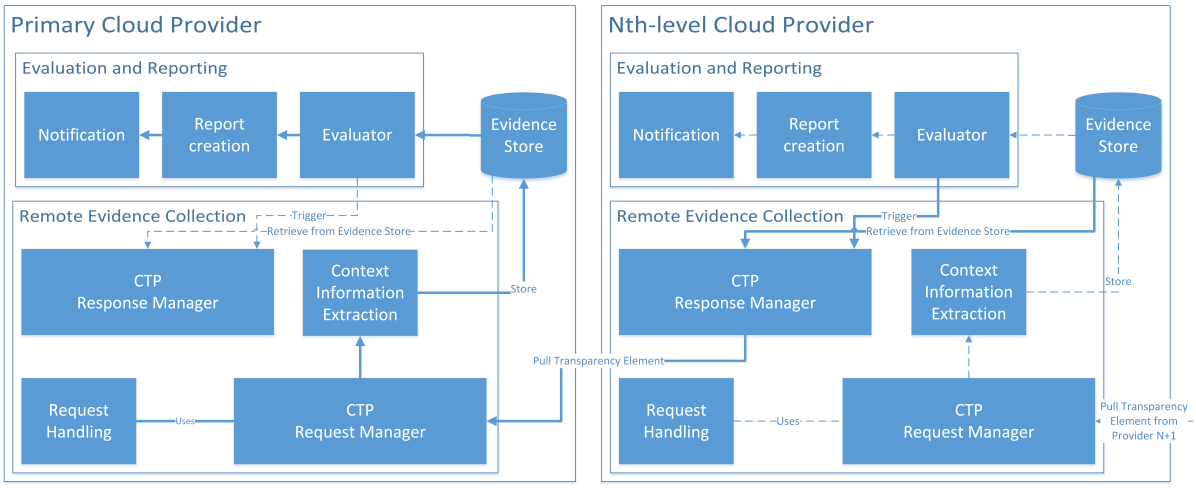

Figure 7: Multi-Provider Audit System Architecture.

8.1 Functional Analysis by Scenario

For the functional look at our proposed solution, we assume the following scenario (as depicted in Figure 1):

- Cloud provider 1 (CSP1) is a SaaS provider (and primary cloud provider) that hosts on the virtual resources provided by CSP2.
- Cloud provider 2 (CSP2) is an IaaS provider (and Nth-level cloud provider) with a data center in Germany and in Russia.
- A Cloud consumer (CC) uses the service provided by CSP1.
- CC and CSP1 agree that CC's data must not leave Germany.
- The auditor checks the compliance with the data location requirement on behalf of CC.

In this case the consumer may believe that his data is located within CSP1's datacenter in Germany. Due to CSP1 not having actual computing resources by its own, the data is actually located in CSP2's data centers, either in Germany, Russia or both. However, CSP1 is still obligated to adhere to the data locality restriction. CSP2 enables CSP1 to audit policy compliance by offering access to our tools for automated auditing. CSP1 establishes regular audits and evidence collection that is focused on data location. The auditor now audits the provider chain with CSP1 as a starting point. CSP1 also runs our audit tool. Its main user is the auditor acting on behalf of the consumer. The communication between the audit tools is implemented using CTP as described in Section 7. Both providers use the audit tool to collect information required for the audits. In the following, the request / response process for this scenario is described:

1. CSP1 is forwarding the audit to CSP2, requesting audit results regarding data location if the data is pulled or waiting till the data was moved which would cause the trigger to fire. For most elements it may be sufficient to list their state in a report but there are elements where immediate notification is indispensable. Such elements require CTPs trigger mechanism as described in Sections 6.2 and 7. In this case the data location request does not need a trigger.
2. At a later point, after the trigger has fired or the information was pulled, the response arrives at CSP1's Request Manager.
3. After receiving the response, CSP1 begins to extract all context information from the response and stores it in the evidence store. This step is necessary to ensure that the audit trail remains available and protected for either archival purposes (e.g., required by law) or re-use at a later point in time to claim remediation. This way, an auditor that feels the need to further investigate a statement made by any of the providers, can retrieve stored evidence from the evidence store.
4. The remaining information (state, CTP response) are used to create the final audit report. The final report for the element contains the policies name as well as the compliance state. The policy compliance check showed that the policy is partially fulfilled. Such an outcome would mean, that the data left Germany at one time.
5. Now the auditor has the possibility to request context information. To access the evidence store

CTP is used and each request requesting context information is sent directly to the Response Manager. The Evidence Store will then create a standardized report containing context information, about where the data is currently being stored (city, state, data center) and where it had been (country code), and send it back to the auditor. With the additional information the auditor can validate how severe the policy breach was. Therefore, network efficiency in multi-tenant environments is required to satisfy each tenants expectations.

8.2 Threat Model

It is important to consider the security of the proposed system to achieve confidence in the acquired audit results. Therefore we perform a security analysis of our proposed approach to cloud provider chain auditing. We follow a simple methodology of defining threat scenarios, categorizing them using the STRIDE (Microsoft Developer Network, 2014) threat model and proposing mitigation strategies for each of the identified threats. The mitigation of the threat categories will be discussed in more detail in section 8.3. STRIDE categorizes threats as follows:

- **S**poofing Identity
- **T**ampering with Data
- **R**epudiation
- **I**nformation disclosure
- **D**enial of Service
- **E**levation of Privilege

We have identified the following major threats to the evidence transfer and processing in multi-provider audits:

- *Unauthorized access (S,I)*: Using our system exposes valuable information such as internal logging, infrastructure design etc. to external entities in an automated way. A malicious external user may steal or otherwise illegitimately gain access to the API that is used for data exchange between the providers. While there is no direct access to consumer data provided by our system, transferred information usually contains metadata about a consumer's system/data properties. The given responses has the potential to expose potentially sensitive metadata. Such information may include but is not limited to data regarding configuration, access control lists and installed software from which vulnerabilities and attack vectors can be deferred. Another potential adversary is a malicious insider at a cloud provider. He can potentially gain access evidence data by directly attacking the evidence store or by intercepting communication between the system components on the internal network.
- *Data leakage (I)*: Audit trail data may intentionally or unintentionally become available. By collecting audit trails from the various evidence sources into the evidence store, a new data source becomes available. Security mechanisms of the evidence store may fail, which could lead to data leakage.
- *Eavesdropping, (I)*: A malicious external user may try to eavesdrop on audit information while it is being transmitted either to the auditor (audit result including audit trails), between cloud providers (information on transparency elements) or internally at a cloud provider (raw data flowing between evidence source and evidence store).
- *Denial of Service (D)*: Denial of Service attacks have unfortunately become a very common type of attack against networked computer systems, that's in many cases trivial to carry out. External adversaries attack either the system directly by exploiting flaws in the implementation or by generating bogus load with the goal of shutting the service down completely.
- *Audit trail manipulation (T,R,I)*: The data generated by our system is supposed to be used during automated audits. The results of these audits should be dependable and believable. An adversary may manipulate audit trail data at various points in the system. For instance, a malicious insider may manipulate the results that are returned by the API. Preserving the integrity of the audit data is therefore of utmost importance.

8.3 Security Analysis by Scenario

Some of the aforementioned threats can be mitigated by implementing appropriate security controls. In the following we present, how the threat categories are addressed in our system:

In order to mitigate the risk of spoofed identities and unauthorized access, we use strong authentication mechanisms based on user and system identification using certificates. This way simple brute force attacks against our publicly available API in order to guess access credentials can be prevented.

Risks of information disclosure and tampering/manipulation (i.e., data integrity) are typically addressed by introducing data encryption and hashing

schemes. We encrypt data at rest (e.g., while it is stored in the evidence store) and in transit (e.g., while it is transmitted between providers and/or the auditor or between system components). To have a full protection against data leakage, ideally there is also encryption of data during processing. While not the focus of this paper, more details on the data at rest and in transit encryption as well as the integrity protection implementation in our system can be found in (Rübsamen et al., 2015). However, this currently severely limits the usefulness, processing options and processing performance (Lopez et al., 2014).

Denial of Service risks can only be addressed by considering the software and the environment it runs in. Directed attacks on the application level can be mitigated using application level firewalls, code audits and security checks, whereas network-based attacks typically require the capability to filter malicious traffic upstream.

9 CONCLUSIONS

In this paper we introduced a system which allows automated auditing of provider chains. We discussed two different types of chain audits: i) individual provider audits where the auditor has to audit each Nth-level provider separately and ii) delegated provider audits where the primary cloud provider acts as an mediator. Our proposed system focuses on the latter approach. We consider the main goal of our cloud audit system to strengthen trust and transparency in cloud services. This could lead to an even better adoption of cloud computing. We also discussed the applicability of data handling and privacy policies and how they apply in complex scenarios where multiple providers share a cloud consumer's data. In the latter part of this paper, we focused on the architectural integration of the CloudTrust Protocol in the evidence collection and transport of our audit system. Finally, we concluded this paper with an evaluation of our proposed approach. We evaluated the functional soundness by demonstrating an audit scenario that involves a cloud consumer using a service that is intransparently provided by two different cloud providers. Additionally, we evaluated our approach by defining a threat model using threat scenarios and addressing those threats.

In our future work, we focus on even more complex service provision scenarios, where even more layers of service providers are involved. We will also put special focus on ensuring the scalability of our approach. Another interesting topic emerges, when any of the cloud providers is considered untrustworthy. This can be the case when a malicious insider tries to intrude in our system. We consider ensuring the integrity of evidence in the whole chain of providers to be a major challenge.

ACKNOWLEDGEMENTS

This work has been partly funded from the European Commission's Seventh Framework Programme (FP7/2007-2013), grant agreement 317550, Cloud Accountability Project - http://www.a4cloud.eu/ - (A4CLOUD).

REFERENCES

Cloud Security Alliance (2013). The notorious nine - cloud computing top threats in 2013. https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf.

Cloud Security Alliance (2015). Cloud Trust Protocol. https://cloudsecurityalliance.org/research/ctp.

Distributed Management Task Force, Inc. (DMTF) (2014). Cloud auditing data federation (cadf) - data format and interface definitions specification. http://www.dmtf.org/sites/default/files/standards/documents/DSP0262_1.0.0.pdf.

Doelitzscher, F., Reich, C., Knahl, M., Passfall, A., and Clarke, N. (2012). An Agent Based Business Aware Incident Detection System for Cloud Environments. *Journal of Cloud Computing: Advances, Systems and Applications*, 1(1):9.

Doelitzscher, F., Rübsamen, T., Karbe, T., Reich, C., and Clarke, N. (2013). Sun behind clouds - on automatic cloud security audits and a cloud audit policy language. *International Journal On Advances in Networks and Services*, 6(1 & 2).

FedRAMP (2015). Federal Risk and Authorization Program. http://www.fedramp.gov.

Gonzales, D., Kaplan, J., Saltzman, E., Winkelman, Z., and Woods, D. (2015). Cloud-trust - a security assessment model for infrastructure as a service (iaas) clouds. *Cloud Computing, IEEE Transactions on*, PP(99):1–1.

ISO (2013). ISO27001:2013 - Information technology – Security techniques – Information security management systems – Requirements. http://www.iso.org/iso/catalogue_detail?csnumber=54534.

ISO (2014). ISO/IEC FDIS 27018 - Information technology – Security techniques – Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processors. http://www.iso.org/iso/catalogue_detail.htm?csnumber=61498.

ISO (2015). ISO/IEC FDIS 27017 - Information technology – Security techniques – Code of practice for information security controls based on ISO/IEC 27002 for cloud services. http://www.iso.org/iso/catalogue_detail?csnumber=43757.

Knode, R. (2009). Digital trust in the cloud. http://assets1.csc.com/cloud/downloads/Digital_Trust_in_the_Cloud.pdf.

Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., and Leaf, D. (2011). Nist cloud computing reference architecture. http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505.

Lopez, J., Rübsamen, T., and Westhoff, D. (2014). Privacy-friendly cloud audits with somewhat homomorphic and searchable encryption. In *Innovations for Community Services (I4CS), 2014 14th International Conference on*, pages 95–103.

Marty, R. (2011). Cloud application logging for forensics. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 178–184, New York, NY, USA. ACM.

Massonet, P., Naqvi, S., Ponsard, C., Latanicki, J., Rochwerger, B., and Villari, M. (2011). A monitoring and audit logging architecture for data location compliance in federated cloud infrastructures. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 1510–1517.

Microsoft Developer Network (2014). The Stride Threat Model. https://msdn.microsoft.com/en-US/library/ee823878(v=cs.20).aspx.

Rizvi, S., Ryoo, J., Liu, Y., Zazworsky, D., and Cappeta, A. (2014). A centralized trust model approach for cloud computing. In *Wireless and Optical Communication Conference (WOCC), 2014 23rd*, pages 1–6.

Rübsamen, T., Pulls, T., and Reich, C. (2015). Secure Evidence Collection and Storage for Cloud Accountability Audits. In *CLOSER 2015 - Proceedings of the 5th International Conference on Cloud Computing and Services Science, Lisbon, Portugal, May 20 - 22, 2015*. to appear.

Rübsamen, T. and Reich, C. (2013). Supporting cloud accountability by collecting evidence using audit agents. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 1, pages 185–190.

Saleh, M. (2014). Construction of agent-based trust in cloud infrastructure. In *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on*, pages 941–946.

Security and Privacy Preservation of Evidence in Cloud Accountability Audits

Thomas Rübsamen[1], Tobias Pulls[2], and Christoph Reich[1]

[1] Furtwangen University,
Cloud Research Lab,
Furtwangen, Germany
{thomas.ruebsamen, christoph.reich}@hs-furtwangen.de
[2] Karlstad University,
Department of Mathematics and Computer Science,
Karlstad, Sweden
tobias.pulls@kau.se

Abstract. *Cloud accountability audits are promising to strengthen trust in cloud computing by providing reassurance about the processing data in the cloud according to data handling and privacy policies. To effectively automate cloud accountability audits, various distributed evidence sources need to be considered during evaluation. The types of information range from authentication and data access logging to location information, information on security controls and incident detection. Securing that information quickly becomes a challenge in the system design, when the evidence that is needed for the audit is deemed sensitive or confidential information. This means that securing the evidence at-rest as well as in-transit is of utmost importance. In this paper, we present a system that is based on distributed software agents which enables secure evidence collection with the purpose of automated evaluation during cloud accountability audits. We thereby present the integration of Insynd as a suitable cryptographic mechanism for securing evidence. We present our reasoning for choosing Insynd by showing a comparison of Insynd properties with requirements imposed by accountability evidence collection as well as an analysis how security threats are being mitigated by Insynd. We put special emphasis on security and privacy protection in our system analysis.*

1 INTRODUCTION

Cloud Computing is known for its on demand computing resource provisioning and has now become mainstream. Many businesses as well as private individuals are using cloud services on a daily basis. The nature of these services varies heavily in terms of what kind of information is being out-sourced to the cloud provider. Often, that data is sensitive, for instance when Personal Identifiable Information (PII) is being shared by an individual. Also, businesses that move (parts of) their processes to the cloud, for instance by using Customer Relationship Management Software as a Service, are actively participating in a major paradigm shift from having all data on-premise to moving data to the cloud.

However, many new challenges come along with this trend. Two of the most important issues are customer trust and compliance [14, 22]. These issues are closely tied to the loss of control over data. When moving to the cloud, direct control over i) where data is stored, ii) who has access to it and iii) how it is shared and processed is given up. Because of this loss of control, cloud customers have to trust cloud providers that they treat their data in an appropriate and responsible way. One way to enable that trust is by strengthening transparency and accountability [12, 30] of the cloud provider and services. This includes providing information about data locality, isolation, privacy controls and data processing in general.

Cloud audits can be used to check how data data has been processed in the cloud (i.e., by whom, for what purpose) and whether or not this happened in compliance with what has been defined in previously agreed-upon privacy and data handling policies. This way, a cloud customer can regain some of the information he has given up control of by moving to the cloud. A central responsibility of cloud audits is the collection of data that can be used as evidence. Depending on the data processing policies in place, various sources of evidence need to be considered. For instance, logs are a very important source of evidence, when it comes to auditing the cloud operation (e.g., access logs and error logs). However, other sources of information are also important, such as files (e.g., process documentation) or events registered in the cloud management system (e.g., access control decisions, infrastructure changes, data transfers).

To capture evidence from this variety of sources, centralized logging mechanisms are not enough. We therefore propose a system for accountability evidence collection and audit. With this system, cloud providers are enabled to demonstrate their compliance with data handling policies to their customers and to third-party auditors in an automated way.

In our previous work, we proposed a concept [28] for cloud accountability audits, that enables automated collection of evidential data in the cloud ecosystem with the goal of performing accountability audits. A key mechanism of this system is the secure and privacy-friendly collection and storage of evidence. In our previous work we also explored the use of a somewhat homomorphic encryption scheme to secure evidence collected in the evidence store [17], which has proven practical but very limited in terms of performance and functionality.

In this paper, we present a more practical alternative that imposes less restrictions on evidence collection.

The contributions of this paper are:

- An architecture for automated evidence collection for the purpose of cloud accountability audits
- A process for secure and privacy-protecting evidence collection and storage

The remainder of this paper is structured as follows: in Sect. 2 we present related work in the area of secure evidence collection and cloud auditing. The core principles of Insynd are introduced in Section 3. Section 4 introduces the Audit Agent System (AAS) and its architecture. Following that, we present in

Sect. 5 a mapping of typical characteristics of digital evidence and secure evidence collection in the cloud to how these are addressed by integrating Insynd in our audit agent system. In Sect. 6 we describe the architectural details of the Insynd integration. We also present a scenario-based evaluation of our system in Sect. 7 and conclude this paper in Sect. 8.

2 RELATED WORK

Redfield and Date propose a system called Gringotts [27] that enables secure evidence collection, where evidence data is signed at the system that produces it, before it is sent to a central server for archival using the Evidence Record Syntax. It is similar to our system with respect to the automatic collection of evidential data from multiple sources. However, their focus is on the archival of evidence, whereas we propose a system that also enables automated evidence processing for audits. Additionally, our system also addresses privacy concerns of evidence collection in a multi-tenant environment such as the cloud by introducing evidence encryption, whereas Redfield and Date focus on archival and preservation of evidence integrity.

Zhang et al. [31] identify potential problems when storing massive amounts of evidential data. They specifically address possible information leaks. To solve these issues, they propose an efficient encrypted database model that is supposed to minimize potential data leaks as well as data redundancy. However, they focus solely on the storage backend and do not provide a workflow that addresses secure evidence collection as a whole.

Gupta [11] identifies privacy issues in the digital forensics process, when it comes to data storage devices that typically do not only contain investigation related data, but may also hold sensitive information that may breach privacy. He also identifies a lack of automation in the digital investigation process. To address these issues, Gupta proposes the Privacy Preserving Efficient Digital Forensic Investigation (PPEDFI) framework. PPEDFI automates the investigation process by including knowledge about previous investigation cases, and which kinds of files were relevant then. With that additional information, evidence search on data storage devices is faster. However, while Gupta acknowledges privacy issues, the PPEDFI framework is focused on classic digital forensics and may not be applicable to a cloud ecosystem, where there is typically no way of mapping specific data objects to storage devices, in full.

The Security Audit as a Service (SAaaS) system proposed by Dölitzscher et al. [9, 10] is used to monitor cloud environments and to detect security incidents. SAaaS is specifically designed to detect incidents in the cloud and thereby consider the dynamic nature of such ecosystems, where resources are rapidly provisioned and removed. However, the main focus of SAaaS is not to provide auditors with a comprehensive way of auditing the cloud provider's compliance with accountability policies, which requires additional security and privacy measures to be considered in the data collection process.

3 INSYND

Insynd is a cryptographic scheme where a forward-secure *author* sends messages intended for *clients* through an untrusted *server* [24, 23]. The author is forward-secure in the sense that the author is initially trusted but assumed to turn into an active adversary at some point in time [5]. Insynd protects messages sent prior to author compromise. The server is untrusted, which is possible thanks to the use of Balloon, a forward-secure append-only persistent authenticated data structure [23]. This means that the server storing all messages can safely be outsourced, e.g., to traditional cloud services. Clients are assumed trusted to read messages sent to them by authors. Insynd contains support for clients to also be in the forward-security model, by discarding key-material as messages are read. For sake of ease of implementation, Insynd is designed around the use of NaCl [6], an easy-to-use high-speed cryptography software library.

Insynd provides the following properties:

Forward Integrity and Deletion Detection Nobody can modify or delete messages sent prior to author compromise, as defined by Pulls et al. [25]. This property holds independently for Balloon (the data structure) and the Insynd scheme. For Balloon, anyone can verify the consistency of the data structure, i.e., it is publicly verifiable [23].

Secrecy Insynd provides authenticated encryption [2].

Forward Unlinkability of Events For each run by the author of the protocol to send new messages, all the events sent in that run are unlinkable. This implies that, e.g., an attacker (or the server) cannot tell which events belong to which client [24]. When clients receive their events by querying the server, if they take appropriate actions including but not limited to accessing the server over an anonymity network like Tor [8], their events remain unlinkable.

Publicly Verifiable Proofs Both the author and client receiving a message can create publicly verifiable proofs of the message sender (the author), the receiving client (by registered identity), and the time the message was sent relative to e.g. a time-stamping authority [24]. The proof-of-concept implementation of Insynd uses Bitcoin transactions [20] as a distributed time-stamping server.

Distributed settings Insynd supports distributed authors, where one author can enable other authors to send messages to clients it knows of without requiring any interaction with clients. Client identifiers (public keys) are blinded in the protocol, ensuring forward-unlinkable client identifiers between different authors [24].

Pulls and Peters show that Insynd provide the above cryptographic properties under the assumptions of the decisional Diffie-Hellman (DDH) assumption on Curve25519, an unforgeable signature algorithm, an unforgeable one-time MAC, a collision and pre-image resistant hash function, a IND-CCA2 secure public-key encryption scheme, and the security of the time-stamping mechanism (in our case, the Bitcoin block-chain) [24]. The prototype implementation of Insynd shows performance comparable to state-of-the-art secure logging schemes,

like PillarBox [7], securing syslog-sized messages (max 1KiB) in the order of hundreds of microseconds on average on a commodity laptop. We stress that Insynd is subject to its own review and evaluation; in this paper, we use Insynd as a building block to facilitate secure evidence collection and storage for cloud accountability audits.

4 Audit Agent System

In the following, the main actors, components and the general flow of information from the evidence-producing source to the audit report in our Audit Agent System (AAS) are described.

4.1 Privacy and Accountability Cloud Audit System Actors

The main actor using the AAS is the *Cloud Auditor*. According to NIST, a cloud auditor is a "A party that can conduct independent assessment of cloud services, information system operations, performance and security of the cloud implementation." [16] In general, a cloud consumer, cloud provider or an independent third-party can act as a cloud auditor. Depending on the actual stakeholder that assumes the role of the auditor, isolation issues can arise:

- A *data protection authority (DPA)* typically acts in good faith as a third-party and assesses privacy policies. Therefore, they typically have broad access to a provider's internal documentation, infrastructure and potentially customer's data.
- A *commercial third-party auditor* is usually a specialized service provider (e.g., a penetration or security testing specialist) acting on behalf of the cloud provider. Their access to information is similar to that available to the DPA.
- A *customer* can also assume the role of an auditor, however with a much more limited scope of available information. We consider two major subtypes, businesses as customers and individuals as customers.

In our proposed system, we consider business customers (e.g., companies using cloud services to replace their IT) to be potential auditors but exclude private individuals. Additionally, providers use the AAS internally for self-auditing to regularly and continuously assess their policy compliance and detect potential violations in a timely manner. Depending on the view on an organization (i.e., depending on who assumes the role of cloud auditor), data protection is an issue to consider, when potential confidential information is processed during an audit. This means data confidentiality, integrity and isolation have to be preserved during an automated audit.

4.2 Architectural Components Audit Agent System

The architecture of the Audit Agent System (see Fig. 1) is based on the use of software agents. This allows for improved flexibility by allowing to rapidly react on infrastructure changes, and improved extensibility especially with respect to data collectors that are used to gather information that is evaluated during an audit. The collectors are adapters for the various heterogeneous sources of evidence in a cloud environment. In Sect. 6.1, we describe more details of how the collectors work. The architecture of the AAS comprises of the following components: *Audit Policy Module (APM)*, *Audit Agent Controller (AAC)*, *Evidence Processor and Presenter (EPP)* and *Evidence Store (ES)*. Especially the Evidence Store and the aforementioned collection agents make heavy use of Insynd to assure that data protection requirements are being met. To a lesser extent, the AAC and EPP also utilize Insynd for securely transporting evidence.

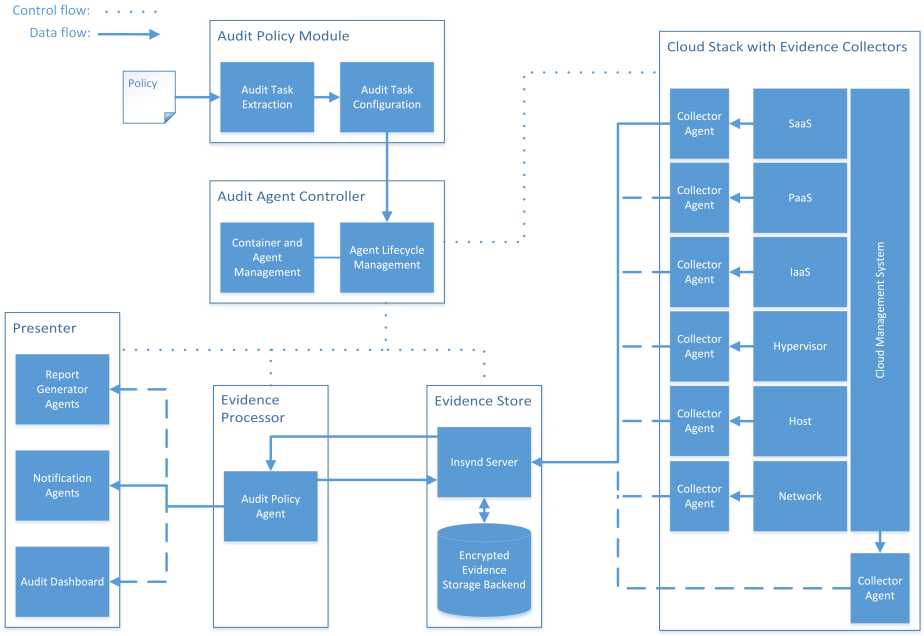

Fig. 1. Privacy and Accountability Cloud Audit System Architecture

All components are implemented as software agents based on the Java Agent DEveleopment framework (JADE) [13] and make heavy use of the JADE Agent Communication Language (ACL) for agent interaction. In the following, we describe the architecture components:

Audit Policy Module

The main input to the AAS are machine-readable policies that describe data

handling obligations (e.g., access control), security controls (e.g., service configuration) and data protection mechanisms (e.g., encryption). From such policies, tasks for collecting evidence, and rules for evaluation of the evidence with the goal of producing a compliance statement are extracted. Additional input to the APM is provided by the auditor. We assume, that there is always a need for at least some manual input for defining an automated audit because the input policy might not be complete with respect to all the parameters that are required for an automated audit. Such parameters include the audit type (periodic or event-driven), the frequency (e.g., daily, monthly. . .) but also more task-specific information that is not provided by the input policy. Depending on the actual audit task, the input comprises of policies and auditor-supplied information:

1. Policies, which define obligations that have to be fulfilled by the cloud provider, such as data access restrictions and usage policies, requirements for the implementation of privacy controls, data retention requirements and general security requirements. The A4Cloud [1] research project develops a machine-readable policy language based on the Primelife Policy Language [3] called Accountability PPL [4]. The A-PPL is capable of describing obligations providers have to adhere to, such access control rules and data handling (e.g., data location, purpose etc). A-PPL serves as the main input to the Audit Agent System and for defining audits.
2. It is possible that an input policy does not necessarily include all information required for mapping policy requirements to specific evidence sources, collectors (e.g., evidence source specific REST client or log parser) and evaluators (e.g., API endpoints, access credentials). That information is provided by the auditor.

With the above mentioned data, the APM builds audit tasks - a combination of evidence collector, processor and presenter agents - and passes that task on to the Audit Agent Controller for instantiation.

Audit Agent Controller
The AAC is the core component of the Audit Agent System. Its main responsibility is the management (i.e., instantiation, configuration, deployment) of any type of agent in the AAS. The main input comes from the APM, which effectively instructs the AAC on how to setup specific audit tasks. A typical audit task deployment in AAS is called an audit workflow. The typical audit workflow (depicted in Fig. 2) is as follows:

1. *Preparation*: The APM extracts audit task configuration from the policy, combines it with input provided by the auditor and passes it on to the AAC.
2. *Configuration*: According to the input provided by the APM, the AAC configures audit policies, its tasks and corresponding collection and evaluation agents.
3. *Instantiation*: the AAC instantiates the previously configured agents as well as the associated evidence store.

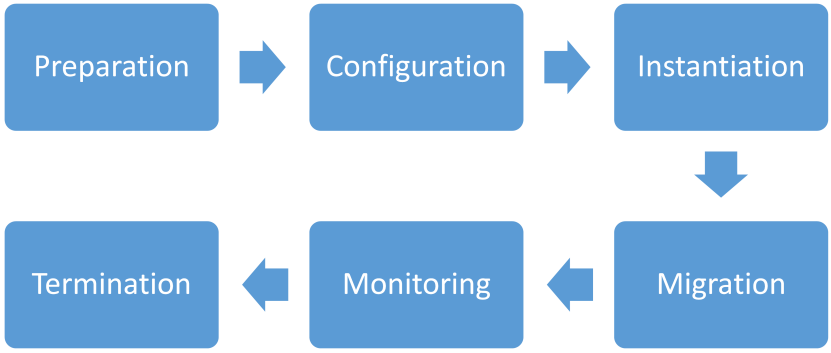

Fig. 2. Audit Agent System Architecture - Audit Workflow

4. *Migration*: Agents are migrated from the core platform where the AAC is running to the target platforms (agent runtime environments as close as possible to the evidence source).
5. *Monitoring*: During the agents' lifetime, the AAC monitors registered platforms and registered agents, handles exceptions, and manages the creation, archival and deletion of evidence stores
6. *Termination*: The AAC disposes of the collector and evaluation agents when they are not needed anymore. It also handles archival and / or deletion of the corresponding evidence store in that case.

Evidence Processor and Presenter
After the collector agents have gathered evidence data and stored it in the evidence store, the evaluation agent(s) of an audit task retrieve that data and analyze it according to the rules that have been extracted from the policies in the preparation phase by the APM. The results that are produced by the evaluation agents are written back to the evidence store. A result can either positive (e.g., a message of proven compliance or the absence of a violation) or negative (e.g., a violation that is detected by the evaluation agent). Additionally the result is passed on to presenter agents that inform the auditor about the audit results. Currently the presenter agents can either display the audit result in a web-based dashboard or pass on the violation in a machine-readable format to other tools or services via a REST API. The whole of processor and presenter agents logically forms the EPP component. It is thereby irrelevant, where these agents are running as long as they are able communicate via a network, which

helps in balancing the load that can be introduced with complex analysis mechanisms or the sheer amount of evidence data that needs to be analyzed. According to the complexity of task, due to the amount of obligations, or the volume of evidence to analyse, different verification processes may need to be considered for the evaluation agents, ranging from log mining, checking for predefined tokens or patterns, to automated analysers and automated reasoning upon the audit trail.

The processing or analysis of evidence consists of two steps:

1. Retrieve the appropriate information from Evidence store.
2. A verification process, which checks the correctness of recorded events according to defined obligations and authorizations.

Evidence Store
The ES is the central repository for storing evidence. Some of the more important characteristics of evidence are that they are associated with a policy for which they were collected and contain supporting information such as log entries collected by an agent, which points out a potential policy violation or incident. For each cloud tenant, there is a separate ES to ensure basic data protection principles are being adhered to by isolating tenants and their data. This addresses some of the confidentiality and privacy issues associated with a share data pool for potentially sensitive information.

There are several approaches to harmonizing the storage format for digital evidence that can be reused in the ES such as [26, 15, 29]. AAS uses a custom evidence format that is based on concepts described in [26] and [29].

Securing the transport and storage of evidence is a considerable challenge. The remainder of this paper focusses on how this is achieved in AAS by utilizing Insynd.

5 AUDIT EVIDENCE STORAGE REQUIREMENTS

In this Section, we present a comparison of general evidence attributes, how they apply in the context of evidence collection for cloud accountability audits and how the integration of Insynd solves key issues in evidence storage.

5.1 Requirements of Digital Evidence

In [19] the core principles of any evidence are described as:

Admissibility Evidence must conform to certain legal rules, before it can be put before a jury.
Authenticity Evidence must be tieable to the incident and may not be manipulated.
Completeness Evidence must be viewpoint agnostic and tell the whole story.
Reliability There cannot be any doubts about the evidence collection process and its correctness.

Believability Evidence must be understandable by a jury.

These principles apply to common evidence as well as digital evidence. Therefore, the evidence collection process for audits has to consider special requirements, which help in addressing these attributes and ensure best possible validity in audits and applicability in court.

In Table 1 we present a mapping of the previously described evidence attributes and how they are supported by the integration of Insynd as a means of storing evidence records. We thereby focus on the key properties of Insynd as described in Sect. 3.

Table 1. Mapping the Impact of Insynd Properties to Evidence Attributes.

		Insynd	
		Forward Integrity and Deletion Detection	Publicly Verifiable Proofs
Evidence Store	Admissibility		
	Authenticity	✓	✓
	Completeness	✓	✓
	Reliability	✓	✓
	Believability		

Admissibility of digital evidence is influenced by the transparency of the collection process and data protection regulation. Digital evidence can be any kind of data (e.g., e-mail messages, social network messages, files, logs etc.). Insynd does not have any direct influence on the admissibility of the evidence stored in it.

Authenticity of digital evidence before court is closely related to the integrity requirement put on evidence records. Evidence may not be manipulated in any way and must be protected against any kind of tampering (willingly and accidentally). Insynd ensures that data cannot be tampered with once it is stored.

Completeness is not directly ensured by Insynd, but rather needs to be ensured by the evidence collection process as a whole. Especially important are the definition of which evidence sources provide relevant evidence that need to be considered during the collection phase. Insynd can complement the evidence collection process by providing assurance of that all data stored in the evidence store are made available as evidence, and not cherry-picked.

Reliability is indirectly supported by integrating necessary mechanisms into the evidence collection process, such as Insynd.

Believability of the collected evidence is not influenced by implemented mechanisms, but rather by the interpretation and presentation by an expert in court. This is due to judges and juries usually being non-technical, which requires an abstracted presentation of evidence. Insynd does not influence the believability in that sense.

5.2 Privacy Requirements

Not all requirements that a secure evidence storage has to fulfill can be captured by analyzing the attributes of digital evidence. Other aspects have to be taken into account to address privacy concerns. Protecting privacy in the process of evidence collection is utmost importance, since the collected data is likely to contain personal data. For cloud computing, one limiting factor may be whether or not the cloud provider is willing to provide deep insight into its infrastructure. Table 2 presents a mapping of privacy principles and properties of our evidence process.

Table 2. Mapping of Insynd properties to Evidence Collection Requirements

		Insynd		
		Secrecy	Forward Unlinkability of Events	Forward Unlinkability of Recipients
Evidence Store	Confidentiality	✓	✓	✓
	Data Minimisation		✓	✓
	Purpose Binding			
	Data Retention	✓		

Below we summarise some key privacy principles:

Confidentiality of data evolves around mechanisms for the protection from unwanted and unauthorized access. Typically, cryptographic concepts, such as encryption, are use to ensure confidentiality of data.

Data Minimization states that the collection of personal data should be minimized and limited to only what is strictly necessary.

Purpose Binding of personal data entails that personal data should only be used for the purposes it was collected for.

Retention Time is concerned with how long personal data may be stored and used, before it needs to be deleted. These periods are usually defined by legal and business requirements.

Insynd and our evidence process provides various mechanisms that support these privacy principles.

Confidentiality A central property of Insynd is that it is always encrypting data using public-key cryptography. By encrypting the evidence store, compromising the privacy of cloud customer data that has been collected in the evidence collection processes becomes almost impossible by attacking the evidence store directly. This goes as far as being able to safely outsource the evidence store to an untrusted third-party, a key property of Insynd [24].

Data Minimisation Furthermore, Insynd provides forward unlinkability of events and client identifiers, as described in Sect. 3, which helps prevent several types of information leaks related to storing and accessing data. Collection agents are always configured for a specific audit task, which is very limited in scope of

what needs to be collected. Agents are never configured to arbitrarily collect data, but are alway limited to a specific source (e.g., a server log) and data objects (e.g., a type of log events).

Purpose Binding Neither Insynd nor our evidence process can directly influence the purpose for which collected data is used. Indirectly, the use of an evidence process like ours, incorporating secure evidence collection and storage, may serve to differentiate data collected for auditing purposes with other data collected e.g., for marketing purposes.

Retention time poses a real challenge. In cloud computing, the precise location of a data object is usually not directly available, i.e., the actual storage medium used to store a particular block is unknown, making data deletion hard. However, if data has been encrypted before storage, a reasonably safe way to ensure "deletion" is to discarding the key material required for decryption. Insynd supports forward-secure clients, where key material to decrypt messages are discarded as messages are read.

In Sect. 7, we also describe the threat model for the system described in this paper and present an evaluation of how Insynd is used to mitigate these threats.

6 SECURE EVIDENCE STORAGE ARCHITECTURE

In this Section, we provide an architectural overview of the integration of Insynd into a secure evidence collection and storage process. We describe the overall architecture and its components, how the components of Insynd are mapped into the Audit Agent System and which setup process is required to use Insynd for securing evidence collection and storage.

6.1 Architecture

In this Section we discuss the architectural integration of Insynd as an evidence store in our audit system. There are basically three different components required to perform secure evidence collection. Figure 3 shows an overview of these components - *Evidence Source*, *Evidence Store* and *Evidence Processing* (see Sect. 4 and Fig. 1 for reference) - as well as the flow of data between them. From the various sources of evidence in the cloud, evidence records are collected that will be stored in the evidence store on a per-tenant basis. The evidence store is thereby located on a separate server. As previously mentioned, the server may be an untrusted third-party cloud storage provider. This is important to ensure so that this approach scales well with a growing number of tenants, evidence sources and evidence records.

Evidence Collection
There are various evidence sources to be considered, such as logs, cryptographical proofs, documentation and many more. For each, there needs to be a suitable collection mechanism. For instance, a log parser for logs, a tool for cryptographical proofs or a file retriever for documentation. This is done by a software agent

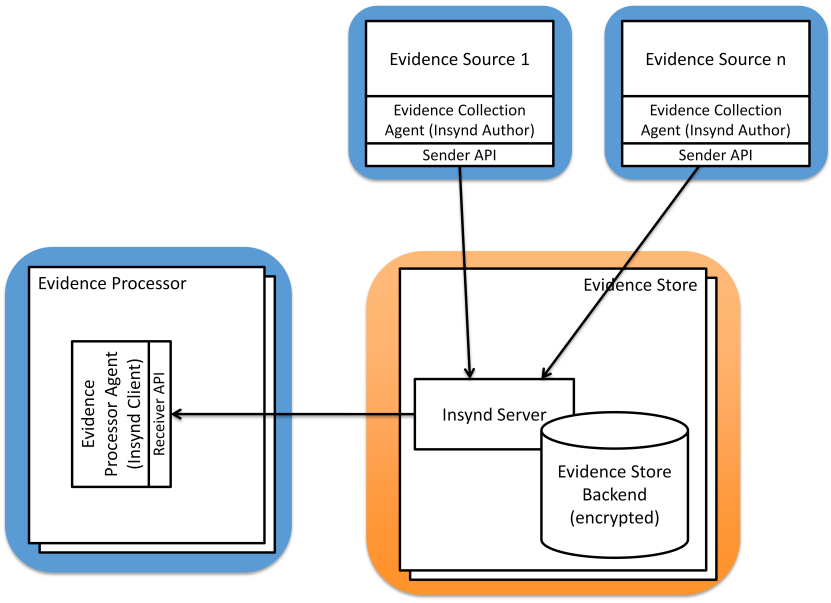

Fig. 3. Evidence Collection, Storage and Processing Workflow

called *Evidence Collection Agent* that is specifically developed for the data collection from the corresponding evidence source. The collection agent acts as an *Insynd Author* meaning it uses the *Sender API* to store evidence into the Evidence Store. The encryption happens in the Sender API. Typically, this agent incorporates or interfaces with a tool to collect evidential data, for instance forensic tools, such as file carvers, log parsers or simple search tools. Another type of collection agent have client APIs implemented to interface with more complex tools, such as Cloud Management Systems (CMS). Generally, these agents receive or collect information as input and translate that information into an evidence record, before storing it in the Evidence Store.

Evidence Storage

From the Evidence Collection Agent, evidence records are sent to the Evidence Store. The Evidence Store is implemented by the *Insynd Server*. Since Insynd functions as a key-value store for storing evidence records (encrypted messages identified by a key) NoSQL or RDBMS-based backend for persisting evidence records can be used. All data contained in the Evidence Store is encrypted. Each record is addressed to a specific receiver (e.g., an Evidence Processing Agent). The receiver's public key is used in the Sender API to encrypt the record on the Evidence Store. This means that only the receiver is able to access the evidence data from the Evidence Store. Isolation between tenants in a single Evidence Store is achieved by providing one container for each tenant where his evidence records are stored. However, even stronger isolation is also possible

by providing a separate Evidence Store hosted on a separate VM. Additionally, Evidence records require a unique identifier in the Evidence Store to enable selective retrieval of records. In our implementation, we use a combination of a policy identifier and a rule identifier (where a rule is part of a policy) to enable the receiver to reduce the amount of records to receive to a manageable size.

Evidence Processing
Evidence Processing components are located at the receiving end of this workflow. The Receiver API is used by the processing agent (Insynd Client) to retrieve evidence records from the Evidence Store. The receiver can request multiple records from a period of time at once. The Client is also in possession of the corresponding private key to decrypt evidence records, which means records can only be decrypted at the Client.

6.2 Identity Management and Key Distribution

Since asymmetric encryption is such an important part of our system, we describe the encryption key distribution sequence next. In this software agent-based system, the automated setup of key material and registration with Insynd is particularly important. Figure 4 depicts the initialization sequence of collection and processing agents with a focus on key distribution.

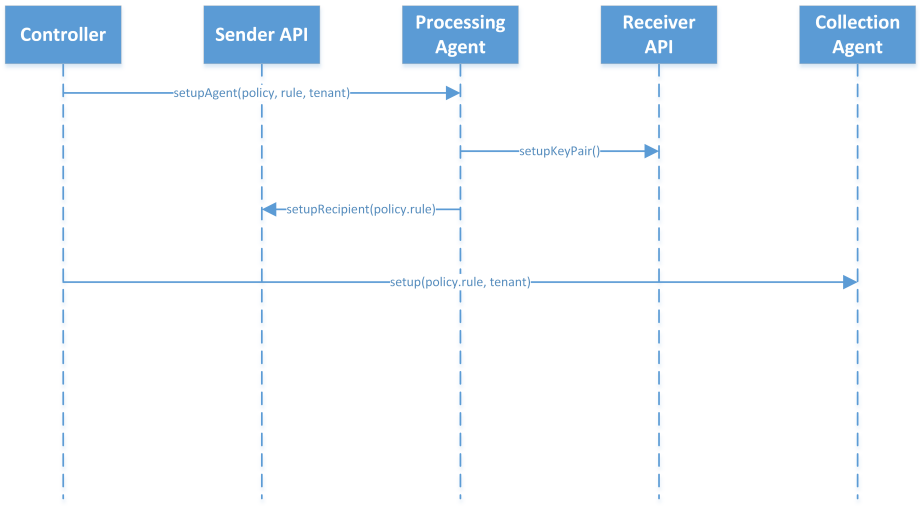

Fig. 4. Evidence Collection Setup Sequence

In Fig. 4, we introduce an additional component beyond those already described in the general architecture: the *Controller*. The Controller serves as an

entry point that controls the agent setup and distribution process in the audit system. It is an important part of the lifecycle management of the system's agents (e.g., creating and destroying of agents or migration between platforms).

In Fig. 4, we describe the initialization sequence for a simple scenario, where a particular tenant wishes to audit compliance with a policy and one rule included in that policy in particular. The following steps have to be performed to setup the evidence collection and storage process for that particular rule:

1. In the first step, a Processing Agent is created and configured according to the input policy and rule respectively for the tenant.
2. During the setup phase, the Processing Agent sets up a key pair at the Receiver API. The Receiver API is a RESTful service that holds private key material and is therefore located at the same servers hosting the Processing Agents (i.e., a trusted environment).
3. After the key material has been generated, the Processing Agent registers itself as a recipient at the Sender API. For this, it uses a unique identifier generated from the policy ID and the rule ID (i.e., *policyID.ruleID*).
4. In the last step, the Controller sets up the required Collection Agents and connects them with the corresponding Processing Agents by using the unique recipient identifier.

Now, it is possible for the Collection Agents to send evidence records to their corresponding Processing Agents. The messages will be encrypted at the Sender API service before storage, using the provided recipient's public key. The Processing Agent then pulls the evidence records from the Evidence Store using the Receiver API the records are decrypted using the receiver's private key.

7 EVALUATION

In this Section we present an informal security evaluation of the system we have implemented for secure evidence collection. We describe the evidence collection work flow using a fictitious scenario. By applying the evidence collection and storage process to the setting described in this scenario, we demonstrate how the requirements stated in Sect. 5 are addressed. Additionally, we provide a model that states threats and adversaries to the process as well as the mitigation functions introduced by Insynd.

In this scenario, the CCOMP company is a customer of the Infrastructure as a Service provider CloudIA. In particular, we analyze the security properties of the evidence collection process by looking at the data at rest as well as the data in transit protection at any time during the flow from the evidence source to its processor. We thereby assume that CloudIA is using OpenStack [21] as a its Cloud Management System (CMS), since this a widely popular open source CMS, which we use for developing our audit agent system. However, any other CMS could be used as well as long as it provides the needed monitoring interfaces.

7.1 Scenario

CloudIA is specialized in providing its customers with virtualized resources in the form of virtual machines, networks and storage. CCOMP has outsourced most of its IT services to CloudIA. Among them is a service that processes data of CCOMP's customers. For that data, CCOMP has to guarantee data retention. CCOMP has identified snapshots to be one major problem with respect to the data retention policy, since the virtual machine's storage is duplicated in the process. This means for CCOMP that in order to be compliant with the data retention policy, a snapshot of that virtual machine may have a maximum lifetime of one day, which limits its usefulness to e.g., backing up before patching. Now, we assume a trustworthy but sloppy administrator at CCOMP who creates a snapshot before patching software on the virtual machine, but then omits deleting the snapshot after he is done. However, an automated daily audit of its cloud resources was put in place by CCOMP to detect such compliance violations.

7.2 Implementation

The collection agent required for the above scenario communicates with our OpenStack CMS to gather evidence of the CMS behavior regarding virtual machine snapshots. The processing agent contains the logic for detecting snapshot violations (i.e., base virtual machine and a maximum age of the snapshot derived from the retention policy). The collection agent is deployed at the CMS controller node and has access to OpenStack's RESTful API. The processing agent is located on the same trusted host as the controller agent (see Fig. 3 for reference). The evidence store is located on a separate, untrusted virtual machine. Now, the following steps are performed:

1. The collection agent opens a connection to the OpenStack RESTful API on the same host and requests a history of snapshot events for CCOMP's virtual machine. Despite there being no communication over the network, HTTPS is used to secure the communication between the collection agent and the CMS. Since the policy only requires information about snapshots to be collected, the CMS agent limits evidence record generation to exactly that information, nothing more.
2. The collection agent sets up the receiver of the evidence according to the process depicted in Fig. 4 and sends the collected records to the evidence store (Insynd). The communication channel is encrypted using HTTPS and the payload (evidence records) is encrypted with the receiving agent's public key.
3. The processing agent pulls records from the evidence store in regular intervals (e.g., every 24 hours), analyses them and triggers a notification of a detected violation. The communication between the processing agent and the evidence store is secured using HTTPS.
4. In the last step, evidence records are deleted because their retention limit has been reached. This is done by discarding the keys required for decryption.

7.3 Threat Model

To demonstrate which security threats exist for the evidence collection process and Insynd is used to mitigate them, we describe the threat model for this system categorized according to the STRIDE [18] threat categorization:

- **S**poofing Identity
- **T**ampering with Data
- **R**epudiation
- **I**nformation disclosure
- **D**enial of Service
- **E**levation of Privilege

We have identified the following major threats to the evidence collection and storage process:

- *Unauthorized access to evidence (S,I)*: the protection of evidence from being accessed by unauthorized persons. Possible adversaries are a malicious third-party evidence storage provider (cloud service provider), another tenant (isolation failure) or an external attacker. Using Insynd for evidence collection and storage addresses this threat since recipients of messages are authenticated using appropriate mechanisms such as user credentials for API authentication and public keys for encryption.
- *Data leakage (S,I)*: the protection from unintentional data leakage. This could be caused by misconfiguration (e.g., unencrypted evidence being publicly available). Using Insynd for evidence collection and storage addresses this threat by encrypting data by default.
- *Eavesdropping, (T,I)*: the protection of evidence during the collection phase, especially in transit. Possibly adversaries are another tenant (isolation failure) or external attackers in case evidence is transported to an external storage provider or auditor. Using Insynd for evidence collection and storage addresses this threat by using transport layer as well as message encryption.
- *Denial of Service (D)*: the protection of the evidence collection and storage process from being attacked directly with the goal of disabling or shutting it down completely (e.g., to cover-up simultaneous attacks on another service). Possible adversaries are external attackers. This is a very generic threat that cannot be addressed by a single tool or control but rather requires a set a measures (on the network and application layer) to enhance denial of service resilience.
- *Evidence manipulation (T,R,I)*: the protection of evidence from intentional manipulation (e.g., deletion of records, changing of contents, manipulation of timestamps). Possible adversaries are malicious insiders and external attackers. Using Insynd for evidence collection and storage addresses this threat, since Insynd provides tampering and deletion detection.

Some of these threats can be mitigated by implementing appropriate security controls (i.e., using Insynd for evidence transport and storage). It provides effective protection by employing security techniques described in Sect. 3.

7.4 Requirements Evaluation

In this section, we evaluate the integration of Insynd against the requirements described in Sect. 5. In step 1 of the fictitious scenario, the data minimization principle is being followed because the specialized agent only collects evidence on the existence of snapshots.

This workflow is secure as soon as the collection agent inserts data into the evidence store in step 2. More precisely, evidence records are tamper-evident and encrypted. This is true, even though the evidence is actually stored on an untrusted virtual machine. The only way to compromise evidence now, is to attack the availability of the server hosting the Insynd server.

When the processing agent in step 3 retrieves records for evaluation, it can be assured of the authenticity of the data and that it has been provably collected by a collection agent. Since evidence records may be subject to maximum data retention regulation, records that are not needed anymore are deleted.

As previously mentioned in Sect. 6 we use JADE as an agent runtime. To secure our system against non-authorized agents, we use the TrustedAgents add-on for the JADE platform. This ensures that only validated agents are able to join our runtime environment. This effectively prevents agent injection attacks, where malicious agents could be inserted at either the collection or processing side to compromise our system.

As can be seen, the evidence records are protected all the way from the evidence source to the processing agent using only encrypted communication channels and having an additional layer of security (message encryption) provided by Insynd. Additionally, while the evidence is being stored, it remains encrypted.

7.5 Scalability

Obviously, since there is a vast amount of evidence sources and therefore a potentially equal number of collection agents, ensuring the scalability of the process and the implementation is very important. This has been considered very early in the design process by choosing an software agent-based approach for the system architecture. Software agents are inherently distributable and allow for complex message flow modeling in an infrastructure. Therefore, the core components evidence collection, storage and processing become distributable as well. In our future work, we'll focus on the scalability aspects. We will follow a methodology where we focus on the following technical key scalability indicators:

- Data transfer volume: amount of evidence data being transferred over the network
- Message volume: amount of evidence message transmissions over the network
- Storage volume: amount of storage required for evidence
- Encryption overhead: performance impact introduced by encryption and decryption

Based on the identified performance impact of each of these indicators, in the second step, we model different message flow optimization strategies to alleviate their impact and ensure scalability.

8 Conclusion and Future Work

In this paper, we presented our system design and implementation for secure evidence collection in cloud computing. The evidence provides the general basis for performing cloud accountability audits. Accountability audits take a large variety of evidence sources and data processing requirements into account.

We showed what the requirements for a secure evidence collection process are and demonstrated how these issues are addressed by incorporating Insynd into our system. We described how the core principles of digital evidence are addressed by our system. Additionally, we considered data protection principles for the evidence collection process, how they influence our approach and how they are addressed in our system by integrating Insynd. For this, we presented the relevant architectural parts of our prototype. Additionally, we provided an overview of how the evidence collection is integrated in our system for automated cloud audits.

In our future work, we will focus on the scalability of our audit system in general and the scalability of the components involved in evidence collection in particular. For that reason, we will focus on the distribution of the audit system and evidence collection not only in the same domain (i.e., in the same infrastructure), but also taking into account outsourcing and multi-provider collection scenarios.

Acknowledgments. This work has been partly funded from the European Commission's Seventh Framework Programme (FP7/2007-2013), grant agreement 317550, Cloud Accountability Project - `http://www.a4cloud.eu/` - (A4CLOUD).

References

1. A4Cloud FP7 Project. `http://www.a4cloud.eu/` (2015)
2. An, J.H.: Authenticated encryption in the public-key setting: Security notions and analyses. IACR Cryptology ePrint Archive 2001, 79 (2001), `http://eprint.iacr.org/2001/079`
3. Ardagna, C.A., Bussard, L., Vimercati, S.D.C.D., Neven, G., Paraboschi, S., Pedrini, E., Preiss, S., Raggett, D., Samarati, P., Trabelsi, S., Verdicchio, M.: Primelife policy language. `http://www.w3.org/2009/policy-ws/papers/Trabelisi.pdf` (2009)
4. Azraoui, M., Elkhiyaoui, K., Önen, M., Bernsmed, K., Santana De Oliveira, A., Sendor, J.: A-PPL: An accountability policy language. In: DPM 2014, 9th International Workshop on Data Privacy Management, September 10, 2014, Wroclaw, Poland. Wroclaw, POLAND (09 2014), `http://www.eurecom.fr/publication/4381`

5. Bellare, M., Yee, B.: Forward-security in private-key cryptography. In: Topics in Cryptology—CT-RSA 2003, pp. 1–18. Springer (2003)
6. Bernstein, D.J., Lange, T., Schwabe, P.: The security impact of a new cryptographic library. In: Hevia, A., Neven, G. (eds.) Progress in Cryptology - LATINCRYPT 2012 - 2nd International Conference on Cryptology and Information Security in Latin America, Santiago, Chile, October 7-10, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7533, pp. 159–176. Springer (2012), http://dx.doi.org/10.1007/978-3-642-33481-8_9
7. Bowers, K.D., Hart, C., Juels, A., Triandopoulos, N.: PillarBox: Combating Next-Generation Malware with Fast Forward-Secure Logging. In: Research in Attacks, Intrusions and Defenses Symposium. vol. 8688, pp. 46–67. Springer (2014), http://dx.doi.org/10.1007/978-3-319-11379-1_3
8. Dingledine, R., Mathewson, N., Syverson, P.F.: Tor: The second-generation onion router. In: Blaze, M. (ed.) Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA. pp. 303–320. USENIX (2004), http://www.usenix.org/publications/library/proceedings/sec04/tech/dingledine.html
9. Doelitzscher, F., Reich, C., Knahl, M., Passfall, A., Clarke, N.: An Agent Based Business Aware Incident Detection System for Cloud Environments. Journal of Cloud Computing: Advances, Systems and Applications 1(1), 9 (2012)
10. Doelitzscher, F., Ruebsamen, T., Karbe, T., Reich, C., Clarke, N.: Sun behind clouds - on automatic cloud security audits and a cloud audit policy language. International Journal On Advances in Networks and Services 6(1 & 2) (2013)
11. Gupta, A.: Privacy preserving efficient digital forensic investigation framework. In: Contemporary Computing (IC3), 2013 Sixth International Conference on. pp. 387–392 (Aug 2013)
12. Haeberlen, A.: A case for the accountable cloud. In: Proceedings of the 3rd ACM SIGOPS International Workshop on Large-Scale Distributed Systems and Middleware (LADIS'09) (Oct 2009)
13. JADE: Java Agent DEvelopement framework. http://jade.tilab.com (2015)
14. Jansen, W., Grance, T.: Sp 800-144. guidelines on security and privacy in public cloud computing. Tech. rep., National Institute of Standards & Technology, Gaithersburg, MD, United States (2011)
15. Jerman Blaič, A., Klobučar, T., Jerman, B.D.: Long-term trusted preservation service using service interaction protocol and evidence records. Comput. Stand. Interfaces 29(3), 398–412 (Mar 2007), http://dx.doi.org/10.1016/j.csi.2006.06.004
16. Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., Leaf, D.: Nist cloud computing reference architecture. http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505 (2011)
17. Lopez, J., Ruebsamen, T., Westhoff, D.: Privacy-friendly cloud audits with somewhat homomorphic and searchable encryption. In: Innovations for Community Services (I4CS), 2014 14th International Conference on. pp. 95–103 (June 2014)
18. Microsoft Developer Network: The Stride Threat Model. https://msdn.microsoft.com/en-US/library/ee823878(v=cs.20).aspx (2015)
19. Mohay, G.M., Anderson, A.M., Collie, B., de Vel, O., McKemmish, R.D.: Computer and Intrusion Forensics. Artech House, Boston, MA, USA (2003), http://eprints.qut.edu.au/10849/, for more information about this book please refer to the publisher's website (see link) or contact the authors.
20. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Consulted 1(2012), 28 (2008)

21. OpenStack: Openstack. http://www.openstack.org/ (2015)
22. Pearson, S.: Toward accountability in the cloud. Internet Computing, IEEE 15(4), 64–69 (July 2011)
23. Pulls, T., Peeters, R.: Balloon: A forward-secure append-only persistent authenticated data structure. In: Computer Security-ESORICS 2015. Springer (2015), to appear
24. Pulls, T., Peeters, R.: Insynd: Secure one-way messaging through Balloons. Cryptology ePrint Archive, Report 2015/150 (2015)
25. Pulls, T., Peeters, R., Wouters, K.: Distributed privacy-preserving transparency logging. In: Sadeghi, A.R., Foresti, S. (eds.) WPES. pp. 83–94. ACM (2013)
26. R. Brandner, U.P., Gondrom, T.: Evidence record syntax (ers). http://tools.ietf.org/html/rfc4998 (2014)
27. Redfield, C.M., Date, H.: Gringotts: Securing data for digital evidence. In: Security and Privacy Workshops (SPW), 2014 IEEE. pp. 10–17 (May 2014)
28. Ruebsamen, T., Reich, C.: Supporting cloud accountability by collecting evidence using audit agents. In: Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on. vol. 1, pp. 185–190 (Dec 2013)
29. Turner, P.: Unification of digital evidence from disparate sources (digital evidence bags). Digit. Investig. 2(3), 223–228 (Sep 2005), http://dx.doi.org/10.1016/j.diin.2005.07.001
30. Weitzner, D.J., Abelson, H., Berners-Lee, T., Feigenbaum, J., Hendler, J., Sussman, G.J.: Information accountability. Commun. ACM 51(6), 82–87 (Jun 2008), http://doi.acm.org/10.1145/1349026.1349043
31. Zhang, R., Li, Z., Yang, Y., Li, Z.: An efficient massive evidence storage and retrieval scheme in encrypted database. In: Information and Network Security (ICINS 2013), 2013 International Conference on. pp. 1–6 (Nov 2013)

Secure Evidence Collection and Storage for Cloud Accountability Audits

Thomas Ruebsamen[1], Tobias Pulls[2] Christoph Reich[1]
[1]*Cloud Research Lab, Furtwangen University, Furtwangen, Germany*
[2]*Department of Mathematics and Computer Science, Karlstad University, Karlstad, Sweden*
{*thomas.ruebsamen, christoph.reich*}*@hs-furtwangen.de, tobias.pulls@kau.se*

Keywords: Cloud Computing, Security, Accountability, Digital Evidence

Abstract: Cloud accountability audits can be used to strengthen trust of cloud service customers in cloud computing by providing reassurance regarding the correct processing of personal or confidential data in the cloud. However, such audits require various information to be collected. The types of information range from authentication and data access logging to location information, information on security controls and incident detection. Correct data processing has to be proven, which immediately shows the need for secure evidence record storage that also scales with the huge number of data sources as well as cloud customers. In this paper, we introduce Insynd as a suitable cryptographic mechanism for storing evidence for accountability audits in our previously proposed cloud accountability audits architecture. We present our reasoning for choosing Insynd by showing a comparison of Insynd properties with requirements imposed by accountability evidence collection as well as an analysis how security threats are being mitigated by Insynd. Additionally, we describe an agent-based evidence collection process with a special focus on security and privacy protection.

1 INTRODUCTION

Cloud Computing is known for its on demand computing resource provisioning and has now become mainstream. Many businesses as well as private individuals are using cloud services on a daily basis. The nature of these services varies heavily in terms of what kind of information is being out-sourced to the cloud provider. More often than not that data is sensitive, for instance when Personal Identifiable Information (PII) is being shared by an individual. Also, businesses that move (parts of) their processes to the cloud, for instance by using a Customer Relationship Management Software as a Service provider, are actively participating in a major paradigm shift from having all data on-premise to moving data to the cloud.

New challenges come along with this trend. Two of the most important issues are customer trust and compliance (Jansen and Grance, 2011; Pearson, 2011). These issues are closely tied to the loss of control over data. When moving to the cloud, direct control over i) where data is stored, ii) who has access to it and iii) how it is shared and processed is given up.

Because of this loss of control, cloud customers have to trust cloud providers that they treat their data in an appropriate and responsible way. This includes providing information about data locality, isolation, privacy controls and data processing in general. One way to enable that trust is by strengthening transparency and accountability (Haeberlen, 2009; Weitzner et al., 2008) of the cloud provider and services.

To regain information on the kind of data processing, cloud audits can be used to check how it has been done. An important part of cloud audits is evidence collection. Depending on the data processing policies in place, various sources of evidence need to be considered. Logs are a very important source of evidence, when it comes to auditing the cloud operation (e.g., access logs and error logs). However, other sources of information are also important, such as files or events registered in the cloud management system. To capture evidence from this variety of sources, centralized logging mechanisms are not enough. We therefore propose a system for accountability evidence collection and audit. With this system, cloud providers are enabled to demonstrate their compliance with data handling policies to their customer's and third-party auditors in an automated way.

In our previous work, we introduced a system (Ruebsamen and Reich, 2013) for cloud accountability audits, that enables automated collection of evidential data in the cloud ecosystem with the goal of

performing accountability audits. A key mechanism of this system is the secure and privacy-friendly collection and storage of evidence. In our previous work we also explored the use of a somewhat homomorphic encryption scheme to secure evidence collected in the evidence store (Lopez et al., 2014). In this paper, we present a more practical alternative that imposes less restrictions on evidence collection. The contributions of this paper are:

- An architecture for automated evidence collection for the purpose of cloud accountability audits
- A process for secure and privacy-protecting evidence collection and storage

The remainder of this follow-up paper is structured as follows: in Section 2 we present related work in the area of secure evidence collection and cloud auditing. The core principles of Insynd are introduced in Section 3. Following that, we present in Section 4 a mapping of typical characteristics of digital evidence and secure evidence collection in the cloud to how these are addressed by integrating Insynd in our audit agent system. In Section 5 we describe the architectural details of the Insynd integration. We present a scenario-based informal evaluation of our system in Section 6 and conclude this paper in Section 7.

2 RELATED WORK

Redfield and Date propose a system called Gringotts (Redfield and Date, 2014) that enables secure evidence collection, where evidence data is signed at the system that produces it, before it is sent to a central server for archival using the Evidence Record Syntax. It is similar to our system with respect to the automatic collection of evidential data from multiple sources. However, their focus is on the archival of evidence, whereas we propose a system that also enables automated evidence processing for audits. Additionally, our system also addresses privacy concerns of evidence collection in a multi-tenant environment such as the cloud by introducing evidence encryption, whereas Redfield and Date focus on archival and preservation of evidence integrity.

Zhang et al. (Zhang et al., 2013) identify potential problems when storing massive amounts of evidential data. They specifically address possible information leaks. To solve these issues, they propose an efficient encrypted database model that is supposed to minimize potential data leaks as well as data redundancy. However, they focus solely on the storage backend and do not provide a workflow that addresses secure evidence collection as a whole.

Gupta (Gupta, 2013) identifies privacy issues in the digital forensics process, when it comes to data storage devices that typically do not only contain investigation related data, but may also hold sensitive information that may breach privacy. He also identifies a lack of automation in the digital investigation process. To address these issues, Gupta proposes the Privacy Preserving Efficient Digital Forensic Investigation (PPEDFI) framework. PPEDFI automates the investigation process by including knowledge about previous investigation cases, and which kinds of files were relevant then. With that additional information, evidence search on data storage devices is faster. However, while Gupta acknowledges privacy issues, the PPEDFI framework is focused on classic digital forensics and may not be applicable to a cloud ecosystem, where there is typically no way of mapping specific data objects to storage devices, in full.

The Security Audit as a Service (SAaaS) system proposed by Doelitzscher et al. (Doelitzscher et al., 2012; Doelitzscher et al., 2013) is used to monitor cloud environments and to detect security incidents. SAaaS is specifically designed to detect incidents in the cloud and thereby consider the dynamic nature of such ecosystems, where resources are rapidly provisioned and removed. However, the main focus of SAaaS is not to provide auditors with a comprehensive way of auditing the cloud provider's compliance with accountability policies, which requires additional security and privacy measures to be considered in the data collection process.

3 INSYND

Insynd is a cryptographic scheme where a forward-secure *author* sends messages intended for *clients* through an untrusted *server* (Pulls and Peeters, 2015b; Pulls and Peeters, 2015a; Pulls et al., 2013). The author is forward-secure in the sense that the author is initially trusted but assumed to turn into an active adversary at some point in time (Bellare and Yee, 2003). Insynd protects messages sent prior to author compromise. The server is completely untrusted, which is possible thanks to the use of Balloon, a forward-secure append-only persistent authenticated data structure (Pulls and Peeters, 2015a). This means that the server storing all messages can safely be outsourced, e.g., to traditional cloud services. Clients are assumed trusted to read messages sent to them by authors. Insynd contains support for clients to also be in the forward-security model, by discarding key-

material as messages are read.

Insynd provides the following properties:

Forward Integrity and Deletion Detection Nobody can modify or delete messages sent prior to author compromise, as defined by Pulls et al. (Pulls et al., 2013). This property holds independently for Balloon (the data structure) and the Insynd scheme. For Balloon, anyone can verify the consistency of the data structure, i.e., it is publicly verifiable (Pulls and Peeters, 2015a).

Secrecy Insynd provides public-key authenticated encryption (An, 2001) thanks to the use of NaCl (Bernstein et al., 2012).

Forward Unlinkability of Events For each run by the author of the protocol to send new messages, all the events sent in that run are unlinkable. This implies that, e.g., an attacker (or the server) cannot tell which events belong to which client (Pulls and Peeters, 2015b). When clients receive their events by querying the server, if they take appropriate actions including but not limited to accessing the server over an anonymity network like Tor (Dingledine et al., 2004), their events remain unlinkable.

Publicly Verifiable Proofs Both the author and client receiving a message can create publicly verifiable proofs of the message sender (the author), the receiving client (by registered identity), and the time the message was sent relative to e.g. a time-stamping authority (Pulls and Peeters, 2015b). The proof-of-concept implementation of Insynd uses Bitcoin transactions (Nakamoto, 2008) as a distributed time-stamping server.

Distributed settings Insynd supports distributed authors, where one author can enable other authors to send messages to clients it knows of without requiring any interaction with clients. Client identifiers (public keys) are blinded in the protocol, ensuring forward-unlinkable client identifiers between different authors (Pulls and Peeters, 2015b).

Pulls and Peters show that Insynd provides forward integrity and deletion detection, secrecy, publicly verifiable proofs, and forward-unlinkability of client identifiers in the standard model under the assumptions of the decisional Diffie-Hellman (DDH) assumption on Curve25519, an unforgeable signature algorithm, an unforgeable MAC, a collision and pre-image resistant hash function, and the security of the time-stamping mechanism (in our case, the Bitcoin block-chain) (Pulls and Peeters, 2015b). Forward unlinkability of events is provided in the random oracle model under the DDH assumption on Curve25519 (Pulls and Peeters, 2015b). The prototype implementation of Insynd shows performance comparable to state-of-the-art secure logging schemes, like PillarBox (Bowers et al., 2014), securing syslog-sized messages (max 1KiB) in the order of hundreds of microseconds on average on a commodity laptop. We stress that Insynd is subject to its own review and evaluation; in this paper, we use Insynd as a building block to facilitate secure evidence collection and storage for cloud accountability audits.

4 AUDIT EVIDENCE STORAGE REQUIREMENTS

In this Section, we present a comparison of general evidence attributes, how they apply in the context of evidence collection for cloud accountability audits and how the integration of Insynd solves key issues in evidence storage.

4.1 Requirements of Digital Evidence

In (Mohay et al., 2003) the core principles of any evidence are described as:

Admissibility Evidence must conform to certain legal rules, before it can be put before a jury.

Authenticity Evidence must be tieable to the incident and may not be manipulated.

Completeness Evidence must be viewpoint agnostic and tell the whole story.

Reliability There cannot be any doubts about the evidence collection process and its correctness.

Believability Evidence must be understandable by a jury.

These principles apply to common evidence as well as digital evidence. Therefore, the evidence collection process for audits has to consider special requirements, which help in addressing these attributes and ensure best possible validity in audits and applicability in court.

In Table 1 we present a mapping of the previously described evidence attributes and how they are supported by the integration of Insynd as a means of storing evidence records. We thereby focus on the key properties of Insynd as described in Section 3.

Admissibility of digital evidence is influenced by the transparency of the collection process and data protection regulation. Digital evidence can be any kind of data (e.g., e-mail messages, social network messages, files, logs etc.). Insynd does not have any

direct influence on the admissibility of the evidence stored in it.

Authenticity of digital evidence before court is closely related to the integrity requirement put on evidence records. Evidence may not be manipulated in any way and must be protected against any kind of tampering (willingly and accidentally). Insynd ensures that data cannot be tampered with once it is stored.

Completeness is not directly ensured by Insynd, but rather needs to be ensured by the evidence collection process as a whole. Especially important are the definition of which evidence sources provide relevant evidence that need to be considered during the collection phase. Insynd can complement the evidence collection process by providing assurance of that all data stored in the evidence store are made available as evidence, and not cherry-picked.

Reliability is indirectly supported by integrating necessary mechanisms into the evidence collection process, such as Insynd.

Believability of the collected evidence is not influenced by implemented mechanisms, but rather by the interpretation and presentation by an expert in court. This is due to judges and juries usually being non-technical, which requires an abstracted presentation of evidence. Insynd does not influence the believability in that sense.

Table 1: Mapping the Impact of Insynd Properties to Evidence Attributes.

		Insynd	
		Forward Integrity and Deletion Detection	Publicly Verifiable Proofs
ES	Admissibility		
	Authenticity	✓	✓
	Completeness	✓	✓
	Reliability	✓	✓
	Believability		

4.2 Privacy Requirements

Not all requirements that a secure evidence storage has to fulfill can be captured by analyzing the attributes of digital evidence. Other aspects have to be taken into account to address privacy concerns. Protecting privacy in the process of evidence collection is utmost importance, since the collected data is likely to contain personal data. For cloud computing, one limiting factor may be whether or not the cloud provider is willing to provide deep insight into its infrastructure. Table 2 presents a mapping of privacy principles and properties of our evidence process.

Below we summarise some key privacy principles:

Confidentiality of data evolves around mechanisms for the protection from unwanted and unauthorized access. Typically, cryptographic concepts, such as encryption, are use to ensure confidentiality of data.

Data Minimization states that the collection of personal data should be minimized and limited to only what is strictly necessary.

Purpose Binding of personal data entails that personal data should only be used for the purposes it was collected for.

Retention Time is concerned with how long personal data may be stored and used, before it needs to be deleted. These periods are usually defined by legal and business requirements.

Insynd and our evidence process provides various mechanisms that support these privacy principles.

Confidentiality A central property of Insynd is that it is always encrypting data using public-key cryptography. By encrypting the evidence store, compromising the privacy of cloud customer data that has been collected in the evidence collection processes becomes almost impossible by attacking the evidence store directly. This goes as far as being able to safely outsource the evidence store to an untrusted third-party, a key property of Insynd (Pulls and Peeters, 2015b).

Data Minimisation Furthermore, Insynd provides forward unlinkability of events and client identifiers, as described in Section 3, which helps prevent several types of information leaks related to storing and accessing data. Collection agents are always configured for a specific audit task, which is very limited in scope of what needs to be collected. Agents are never configured to arbitrarily collect data, but are alway limited to a specific source (e.g., a server log) and data objects (e.g., a type of log events).

Purpose Binding Neither Insynd nor our evidence process can directly influence the purpose for which collected data is used. Indirectly, the use of an evidence process like ours, incorporating secure evidence collection and storage, may serve to differentiate data collected for auditing purposes with other data collected e.g., for marketing purposes.

Retention time poses a real challenge. In cloud computing, the precise location of a data object is usually not directly available, i.e., the actual storage medium used to store a particular block is unknown, making data deletion hard. However, if data has been

encrypted before storage, a reasonably safe way to ensure "deletion" is to discarding the key material required for decryption. Insynd supports forward-secure clients, where key material to decrypt messages are discarded as messages are read.

Table 2: Mapping of Insynd properties to Evidence Collection Requirements

		Insynd		
		Secrecy	Forward Unlinkability of Events	Forward Unlinkability of Recipients
ES	Confidentiality	✓	✓	✓
	Data Minimisation		✓	✓
	Purpose Binding			
	Data Retention	✓		

In Section 6, we also describe the threat model for the system described in this paper and present an evaluation of how Insynd is used to mitigate these threats.

5 SECURE EVIDENCE STORAGE ARCHITECTURE

In this Section, we provide an architectural overview of the integration of Insynd into a secure evidence collection and storage process. We describe the overall architecture and its components, how the components of Insynd are mapped into the audit agent system and which setup process is required to use Insynd for securing evidence collection and storage.

5.1 Architecture

In this Section we discuss the architectural integration of Insynd as an evidence store in our audit system. There are basically three different components required to perform secure evidence collection. Figure 1 shows an overview of these components - *Evidence Source*, *Evidence Store* and *Evidence Processing* - as well as the flow of data between them. From the various sources of evidence in the cloud, evidence records are collected that will be stored in the evidence store on a per-tenant basis. The evidence store is thereby located on a separate server. As previously mentioned, the server may be an untrusted third-party cloud storage provider. This is important to ensure so that this approach scales well with a growing number of tenants, evidence sources and evidence records.

Our architecture is built around using software agents for evidence collection, evidence evaluation and controlling the overall system. Agent technology helps with extensibility by allowing us to easily introduce new evidence sources and processors by building new agents. On top of that, it allows the audit system to address rapid infrastructure changes, which are very common in cloud infrastructures by easily deploying and destroying agents when needed. We base our system on the Java Agent DEvelopment Framework (JADE, 2015). This effectively means that anywhere, where a Java runtime environment is available, a collection agent can be deployed.

5.1.1 Evidence Collection

There are various evidence sources to be considered, such as logs, cryptographical proofs, documentation and many more. For each, there needs to be a suitable collection mechanism. For instance, a log parser for logs, a tool for cryptographical proofs or a file retriever for documentation. This is done by a software agent called *Evidence Collection Agent* that is specifically developed for the data collection from the corresponding evidence source. The collection agent acts as an *Insynd Author* meaning it uses the *Sender API* to store evidence into the Evidence Store. The encryption happens in the Sender API. Typically, this agent incorporates or interfaces with a tool to collect evidential data, for instance forensic tools, such as file carvers, log parsers or simple search tools. Another type of collection agent have client APIs implemented to interface with more complex tools, such as Cloud Management Systems (CMS). Generally, these agents receive or collect information as input and translate that information into an evidence record, before storing it in the Evidence Store.

5.1.2 Evidence Storage

From the Evidence Collection Agent, evidence records are sent to the Evidence Store. The Evidence Store is implemented by the *Insynd Server*. Since Insynd functions as a key-value store for storing evidence records (encrypted messages identified by a key) NoSQL or RDBMS-based backend for persisting evidence records can be used. All data contained in the Evidence Store is encrypted. Each record is addressed to a specific receiver (e.g., an Evidence Processing Agent). The receiver's public key is used in

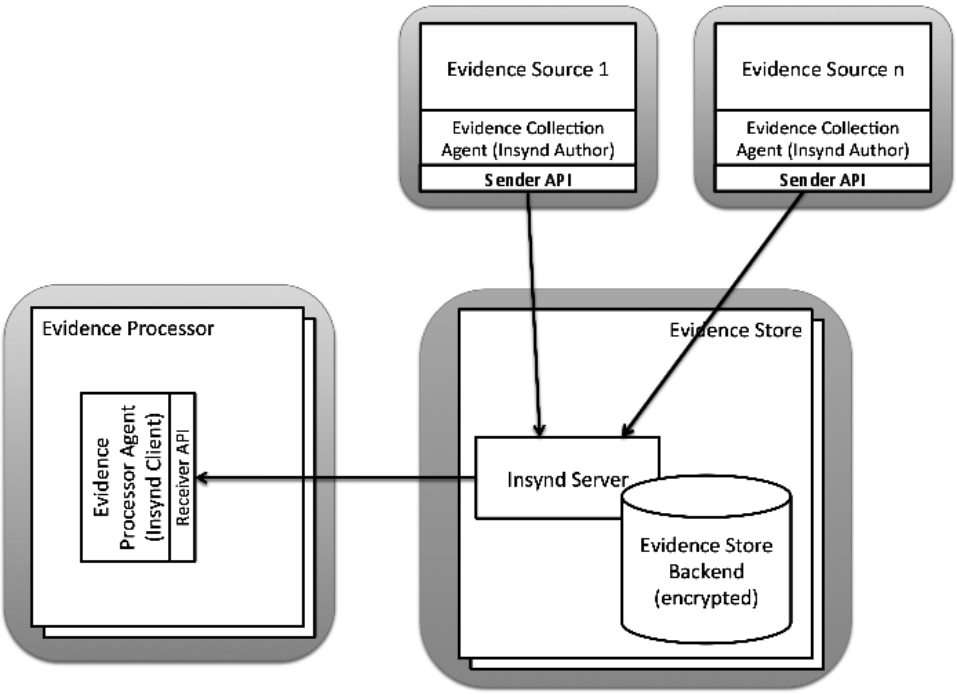

Figure 1: Evidence Collection, Storage and Processing Workflow

the Sender API to encrypt the record on the Evidence Store. This means that only the receiver is able to access the evidence data from the Evidence Store. Isolation between tenants in a single Evidence Store is achieved by providing one container for each tenant where his evidence records are stored. However, even stronger isolation is also possible by providing a separate Evidence Store hosted on a separate VM. Additionally, Evidence records require a unique identifier in the Evidence Store to enable selective retrieval of records. In our implementation, we use a combination of a policy identifier and a rule identifier (where a rule is part of a policy) to enable the receiver to reduce the amount of records to receive to a manageable size.

5.1.3 Evidence Processing

Evidence Processing components are located at the receiving end of this workflow. The Receiver API is used by the processing agent (Insynd Client) to retrieve evidence records from the Evidence Store. The receiver can request multiple records from a period of time at once. The Client is also in possession of the corresponding private key to decrypt evidence records, which means records can only be decrypted at the Client.

5.2 Identity Management and Key Distribution

Since asymmetric encryption is such an important part of our system, we describe the encryption key distribution sequence next. In this software agent-based system, the automated setup of key material and registration with Insynd is particularly important. Figure 2 depicts the initialization sequence of collection and processing agents with a focus on key distribution.

In Figure 2 we introduce an additional component beyond those already described in the general architecture: the *Controller*. The Controller serves as an entry point that controls the agent setup and distribution process in the audit system. It is an important part of the lifecycle management of the system's agents (e.g., creating and destroying of agents or migration between platforms).

In Figure 2 we describe the initialization sequence for a simple scenario, where a particular tenant wishes to audit compliance with a policy and one rule included in that policy in particular. The following steps have to be performed to setup the evidence collection and storage process for that particular rule:

1. In the first step, a Processing Agent is created and configured according to the input policy and rule respectively for the tenant.
2. During the setup phase, the Processing Agent

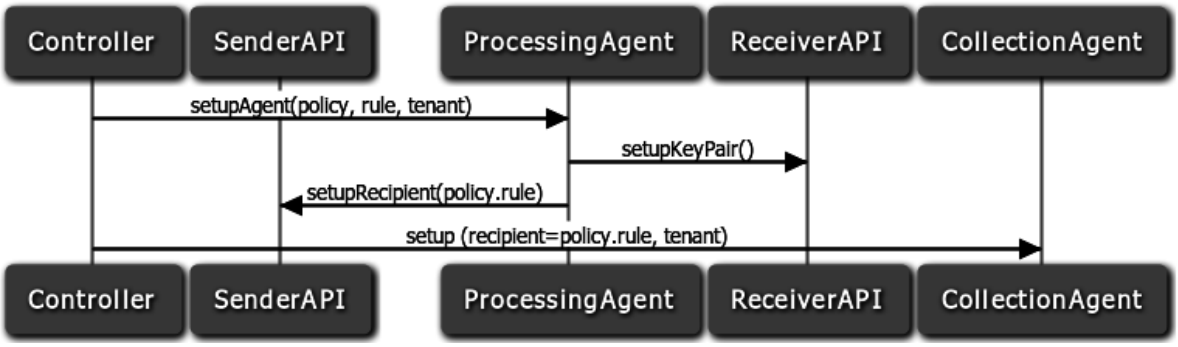

Figure 2: Evidence Collection Setup Sequence

sets up a keypair at the Receiver API. The Receiver API is a RESTful service that holds private key material and is therefore located at the same servers hosting the Processing Agents (i.e., a trusted environment).

3. After the key material has been generated, the Processing Agent registers itself as a recipient at the Sender API. For this, it uses a unique identifier generated from the policy ID and the rule ID (i.e., *policyID.ruleID*).
4. In the last step, the Controller sets up the required Collection Agents and connects them with the corresponding Processing Agents by using the unique recipient identifier.

Now, it is possible for the Collection Agents to send evidence records to their corresponding Processing Agents. The messages will be encrypted at the Sender API service before storage, using the provided recipient's public key. The Processing Agent then pulls the evidence records from the Evidence Store using the Receiver API the records are decrypted using the receiver's private key.

6 EVALUATION

In this Section we present an informal security evaluation of the system we have implemented for secure evidence collection. We describe the evidence collection work flow using a fictitious scenario. By applying the evidence collection and storage process to the setting described in this scenario, we demonstrate how the requirements stated in Section 4 are addressed. Additionally, we provide a model that states threats and adversaries to the process as well as the mitigation functions introduced by Insynd.

In this scenario, the CCOMP company is a customer of the Infrastructure as a Service provider CloudIA. In particular, we analyze the security properties of the evidence collection process by looking at the data at rest as well as the data in transit protection at any time during the flow from the evidence source to its processor. We thereby assume that CloudIA is using OpenStack (OpenStack, 2015) as a its Cloud Management System (CMS), since this a widely popular open source CMS, which we use for developing our audit agent system. However, any other CMS could be used as well as long as it provides the needed monitoring interfaces.

6.1 Scenario

CloudIA is specialized in providing its customers with virtualized resources in the form of virtual machines, networks and storage. CCOMP has outsourced most of its IT services to CloudIA. Among them is a service that processes data of CCOMP's customers. For that data, CCOMP has to guarantee data retention. CCOMP has identified snapshots to be one major problem with respect to the data retention policy, since the virtual machine's storage is duplicated in the process. This means for CCOMP that in order to be compliant with the data retention policy, a snapshot of that virtual machine may have a maximum lifetime of one day, which limits its usefulness to e.g., backing up before patching. Now, we assume a trustworthy but sloppy administrator at CCOMP who creates a snapshot before patching software on the virtual machine, but then omits deleting the snapshot after he is done. However, an automated daily audit of its cloud resources was put in place by CCOMP to detect such compliance violations.

6.2 Implementation

The collection agent required for the above scenario communicates with our OpenStack CMS to gather evidence of the CMS behavior regarding virtual machine snapshots. The processing agent contains the logic for detecting snapshot violations (i.e., base virtual machine and a maximum age of the snapshot de-

rived from the retention policy). The collection agent is deployed at the CMS controller node and has access to OpenStack's RESTful API. The processing agent is located on the same trusted host as the controller agent (see Figure 1 for reference). The evidence store is located on a separate, untrusted virtual machine. Now, the following steps are performed:

1. The collection agent opens a connection to the OpenStack RESTful API on the same host and requests a history of snapshot events for CCOMP's virtual machine. Despite there being no communication over the network, HTTPS is used to secure the communication between the collection agent and the CMS. Since the policy only requires information about snapshots to be collected, the CMS agent limits evidence record generation to exactly that information, nothing more.
2. The collection agent sets up the receiver of the evidence according to the process depicted in Figure 2 and sends the collected records to the evidence store (Insynd). The communication channel is encrypted using HTTPS and the payload (evidence records) is encrypted with the receiving agent's public key.
3. The processing agent pulls records from the evidence store in regular intervals (e.g., every 24 hours), analyses them and triggers a notification of a detected violation. The communication between the processing agent and the evidence store is secured using HTTPS.
4. In the last step, evidence records are deleted because their retention limit has been reached. This is done by discarding the keys required for decryption.

6.3 Threat Model

To demonstrate which security threats exist for the evidence collection process and Insynd is used to mitigate them, we describe the threat model for this system categorized according to the STRIDE(Microsoft Developer Network, 2015) threat categorization:

- **S**poofing Identity
- **T**ampering with Data
- **R**epudiation
- **I**nformation disclosure
- **D**enial of Service
- **E**levation of Privilege

We have identified the following major threats to the evidence collection and storage process:

- *Unauthorized access to evidence (S,I)*: the protection of evidence from being accessed by unauthorized persons. Possible adversaries are a malicious third-party evidence storage provider (cloud service provider), another tenant (isolation failure) or an external attacker. Using Insynd for evidence collection and storage addresses this threat since recipients of messages are authenticated using appropriate mechanisms such as user credentials for API authentication and public keys for encryption.
- *Data leakage (S,I)*: the protection from unintentional data leakage. This could be caused by misconfiguration (e.g., unencrypted evidence being publicly available). Using Insynd for evidence collection and storage addresses this threat by encrypting data by default.
- *Eavesdropping, (T,I)*: the protection of evidence during the collection phase, especially in transit. Possibly adversaries are another tenant (isolation failure) or external attackers in case evidence is transported to an external storage provider or auditor. Using Insynd for evidence collection and storage addresses this threat by using transport layer as well as message encryption.
- *Denial of Service (D)*: the protection of the evidence collection and storage process from being attacked directly with the goal of disabling or shutting it down completely (e.g., to cover-up simultaneous attacks on another service). Possible adversaries are external attackers. This is a very generic threat that cannot be addressed by a single tool or control but rather requires a set a measures (on the network and application layer) to enhance denial of service resilience.
- *Evidence manipulation (T,R,I)*: the protection of evidence from intentional manipulation (e.g., deletion of records, changing of contents, manipulation of timestamps). Possible adversaries are malicious insiders and external attackers. Using Insynd for evidence collection and storage addresses this threat, since Insynd provides tampering and deletion detection.

Some of these threats can be mitigated by implementing appropriate security controls (i.e., using Insynd for evidence transport and storage). It provides effective protection by employing security techniques described in Section 3.

6.4 Requirements Evaluation

In this section, we evaluate the integration of Insynd against the requirements described in Section 4. In

step 1 of the fictitious scenario, the data minimization principle is being followed because the specialized agent only collects evidence on the existence of snapshots.

This workflow is secure as soon as the collection agent inserts data into the evidence store in step 2. More precisely, evidence records are tamper-evident and encrypted. This is true, even though the evidence is actually stored on an untrusted virtual machine. The only way to compromise evidence now, is to attack the availability of the server hosting the Insynd server.

When the processing agent in step 3 retrieves records for evaluation, it can be assured of the authenticity of the data and that it has been provably collected by a collection agent. Since evidence records may be subject to maximum data retention regulation, records that are not needed anymore are deleted.

As previously mentioned in Section 5 we use JADE as an agent runtime. To secure our system against non-authorized agents, we use the TrustedAgents add-on for the JADE platform. This ensures that only validated agents are able to join our runtime environment. This effectively prevents agent injection attacks, where malicious agents could be inserted at either the collection or processing side to compromise our system.

As can be seen, the evidence records are protected all the way from the evidence source to the processing agent using only encrypted communication channels and having an additional layer of security (message encryption) provided by Insynd. Additionally, while the evidence is being stored, it remains encrypted.

6.5 Scalability

Obviously, since there is a vast amount of evidence sources and therefore a potentially equal number of collection agents, ensuring the scalability of the process and the implementation is very important. This has been considered very early in the design process by choosing an software agent-based approach for the system architecture. Software agents are inherently distributable and allow for complex message flow modeling in an infrastructure. Therefore, the core components evidence collection, storage and processing become distributable as well. In our future work, we'll focus on the scalability aspects. We will follow a methodology where we focus on the following technical key scalability indicators:

- Data transfer volume: amount of evidence data being transferred over the network
- Message volume: amount of evidence message transmissions over the network
- Storage volume: amount of storage required for evidence
- Encryption overhead: performance impact introduced by encryption and decryption

Based on the identified performance impact of each of these indicators, in the second step, we model different message flow optimization strategies to alleviate their impact and ensure scalability.

7 CONCLUSIONS

In this paper, we presented our system design and implementation for secure evidence collection in cloud computing. The evidence provides the general basis for performing cloud accountability audits. Accountability audits take a large variety of evidence sources and data processing requirements into account.

We showed what the requirements for a secure evidence collection process are and demonstrated how these issues are addressed by incorporating Insynd into our system. We described how the core principles of digital evidence are addressed by our system. Additionally, we considered data protection principles for the evidence collection process, how they influence our approach and how they are addressed in our system by integrating Insynd. For this, we presented the relevant architectural parts of our prototype.

In our future work, we will focus on the scalability of our audit system in general and the scalability of the components involved in evidence collection in particular. For that reason, we will focus on the distribution of the audit system and evidence collection not only in the same domain (i.e., in the same infrastructure), but also taking into account outsourcing and multi-provider collection scenarios.

ACKNOWLEDGEMENTS

This work has been partly funded from the European Commissions Seventh Framework Programme (FP7/2007-2013), grant agreement 317550, Cloud Accountability Project - http://www.a4cloud.eu/ - (A4CLOUD).

REFERENCES

An, J. H. (2001). Authenticated encryption in the public-key setting: Security notions and analyses. *IACR Cryptology ePrint Archive*, 2001:79.

Bellare, M. and Yee, B. (2003). Forward-security in private-key cryptography. In *Topics in Cryptology—CT-RSA 2003*, pages 1–18. Springer.

Bernstein, D. J., Lange, T., and Schwabe, P. (2012). The security impact of a new cryptographic library. In Hevia, A. and Neven, G., editors, *Progress in Cryptology - LATINCRYPT 2012 - 2nd International Conference on Cryptology and Information Security in Latin America, Santiago, Chile, October 7-10, 2012. Proceedings*, volume 7533 of *Lecture Notes in Computer Science*, pages 159–176. Springer.

Bowers, K. D., Hart, C., Juels, A., and Triandopoulos, N. (2014). PillarBox: Combating Next-Generation Malware with Fast Forward-Secure Logging. In *Research in Attacks, Intrusions and Defenses Symposium*, volume 8688, pages 46–67. Springer.

Dingledine, R., Mathewson, N., and Syverson, P. F. (2004). Tor: The second-generation onion router. In Blaze, M., editor, *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pages 303–320. USENIX.

Doelitzscher, F., Reich, C., Knahl, M., Passfall, A., and Clarke, N. (2012). An Agent Based Business Aware Incident Detection System for Cloud Environments. *Journal of Cloud Computing: Advances, Systems and Applications*, 1(1):9.

Doelitzscher, F., Ruebsamen, T., Karbe, T., Reich, C., and Clarke, N. (2013). Sun behind clouds - on automatic cloud security audits and a cloud audit policy language. *International Journal On Advances in Networks and Services*, 6(1 & 2).

Gupta, A. (2013). Privacy preserving efficient digital forensic investigation framework. In *Contemporary Computing (IC3), 2013 Sixth International Conference on*, pages 387–392.

Haeberlen, A. (2009). A case for the accountable cloud. In *Proceedings of the 3rd ACM SIGOPS International Workshop on Large-Scale Distributed Systems and Middleware (LADIS'09)*.

JADE (2015). Java Agent DEvelopement framework. `http://jade.tilab.com`.

Jansen, W. and Grance, T. (2011). Sp 800-144. guidelines on security and privacy in public cloud computing. Technical report, Gaithersburg, MD, United States.

Lopez, J., Ruebsamen, T., and Westhoff, D. (2014). Privacy-friendly cloud audits with somewhat homomorphic and searchable encryption. In *Innovations for Community Services (I4CS), 2014 14th International Conference on*, pages 95–103.

Microsoft Developer Network (2015). The Stride Threat Model. `https://msdn.microsoft.com/en-US/library/ee823878(v=cs.20).aspx`.

Mohay, G. M., Anderson, A. M., Collie, B., de Vel, O., and McKemmish, R. D. (2003). *Computer and Intrusion Forensics*. Artech House, Boston, MA, USA. For more information about this book please refer to the publisher's website (see link) or contact the authors.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1(2012):28.

OpenStack (2015). Openstack. `http://www.openstack.org/`.

Pearson, S. (2011). Toward accountability in the cloud. *Internet Computing, IEEE*, 15(4):64–69.

Pulls, T. and Peeters, R. (2015a). Balloon: A forward-secure append-only persistent authenticated data structure. Cryptology ePrint Archive, Report 2015/007.

Pulls, T. and Peeters, R. (2015b). Insynd: Secure one-way messaging through Balloons. Cryptology ePrint Archive, Report 2015/150.

Pulls, T., Peeters, R., and Wouters, K. (2013). Distributed privacy-preserving transparency logging. In Sadeghi, A.-R. and Foresti, S., editors, *WPES*, pages 83–94. ACM.

Redfield, C. M. and Date, H. (2014). Gringotts: Securing data for digital evidence. In *Security and Privacy Workshops (SPW), 2014 IEEE*, pages 10–17.

Ruebsamen, T. and Reich, C. (2013). Supporting cloud accountability by collecting evidence using audit agents. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 1, pages 185–190.

Weitzner, D. J., Abelson, H., Berners-Lee, T., Feigenbaum, J., Hendler, J., and Sussman, G. J. (2008). Information accountability. *Commun. ACM*, 51(6):82–87.

Zhang, R., Li, Z., Yang, Y., and Li, Z. (2013). An efficient massive evidence storage and retrieval scheme in encrypted database. In *Information and Network Security (ICINS 2013), 2013 International Conference on*, pages 1–6.

An Architecture for Cloud Accountability Audits

Thomas Rübsamen, Christoph Reich, Martin Knahl
Cloud Research Lab
Furtwangen University
Furtwangen, Germany
Email: {ruet, rch, knahl}@hs-furtwangen.de

Nathan Clarke
Centre for Security, Communications and Network Research
Plymouth University
Plymouth, United Kingdom
Email: N.Clarke@plymouth.ac.uk

***Abstract*—As more and more sensitive data is stored in the cloud, privacy and security become more important. Today's cloud services, their internal processes and details about how and by whom data is processed is opaque to the cloud user. To address the trust issues stemming from this loss of control, we propose a software agent-based system for auditing accountability policies, to increase transparency of cloud services.**

***Keywords*—Accountability, Audit, Cloud Computing, Digital Evidence**

I. Introduction

Cloud Computing is an increasingly popular paradigm for service delivery in today's Internet [1] and may lead to significant advantages such as reduced upfront investments [2], rapid provisioning and automatic scaling of resources [3]. However, the adoption of cloud computing is accompanied with several security and privacy problems. For instance, data breaches and data loss are amongst the major threats in cloud computing [4]. Therefore, two of the key issues are customer trust and compliance [2], [5]. Because of the loss of control, cloud customers have to trust cloud providers to handle their data appropriately and that sufficient data protection mechanisms are in place. Cloud Providers use terms of service (TOS), which are generally non-negotiable [5], and privacy agreements to describe how data will be handled in their services. However, beyond such documents, there is usually a lack of transparency regarding details about security processes and controls. Trust is also an issue when service providers use additional services provided by third-parties, because trust will not necessarily be transitive in such complex scenarios [2]. This lack of trust can be addressed by strengthening transparency and accountability [6], [7] on the cloud provider side.

The Audit Agent System proposed in this paper, strives to enable automated cloud accountability audits by addressing transparency and privacy protection issues associated with cloud computing. Accountability is regarded as a means to strengthen customer trust in cloud services. By auditing compliance with data policies, transparency and privacy of the cloud shall be improved. This includes the secure and privacy-aware collection of evidence supporting claims made in audit reports. In this paper, we describe an architecture for the Audit Agent System.

This paper is structured as follows: in Section II a brief overview about current research projects and industrial approaches is given. In Section III, we describe the proposed system architecture for the Audit Agent System. We close this paper with a conclusion in Section IV.

II. Related Work

There are several academic approaches to various aspects of cloud auditing. For instance, security auditing is a very important part of accountability auditing of a cloud provider, since it demonstrates that required security controls are put in place and are functioning correctly. There are some projects working on the architectural and interface level regarding the automation of security audits such as the Security Audit as a Service (SAaaS) project [8]. The Distributed Management Task Force (DMTF) is also working on cloud auditing with the Cloud Auditing Data Federation (CADF) working group. They are focusing mostly on developing standardized interfaces and data formats to enable cloud security auditing [9]. A similar project is the Cloud Security Alliance's (CSA) Cloud Trust Protocol (CTP), which defines an interface for enabling cloud users to "generate confidence that everything that is claimed to be happening in the cloud is indeed happening as described, ..., and nothing else" [10], which indicates an additional focus on providing additional transparency of cloud services. The latter two projects, however, do neither detail an actual architecture and how the interfaces shall be implemented nor do they describe explicitly focus on accountability.

A lot of current research is not focused on the overall automation of cloud accountability audits, but rather on aspects, that would be part of such an audit (i.e., may be implemented as part of the system described in this paper). Such approaches are for example concerned with the provenance of data in the cloud [11], proof of retrievability and provable data possession [12], virtual machine introspection [13] and replay as an advanced monitoring and forensic analysis technique.

When looking at cloud audits and the associated process of collecting evidence to assess policy compliance, it is important to look at industry practices regarding monitoring. Many such tools, such as the well-established Nagios [14] support agent-based data collection. New Relic [15], a Software as a Service (SaaS) software analytics solution enables the collection of data on various different scopes and devices. However, most of these tools are mainly concerned with performance monitoring and tracing, whereas our approach mainly considers the automation of security and accountability auditing. Security Information and Event Management (SIEM) systems are the main source of monitoring information in today's more complex IT infrastructures. They provide additional means of detecting security incidents by collecting information from various sources in the infrastructure. However, when it comes to auditing policies on the level data objects and regarding accountability requirements specific to individual customers,

there is still lacking functionality.

III. Architecture

In this Section, we describe the high-level architecture of the Audit Agent System and its components. We also describe the input to the Audit Agent System in the form of accountability policies and illustrate the data flow of evidence from its source to the processing components across the different architectural layers.

A. Audit Agent System Architecture Introduction

In Figure 1 the overall architecture of the Audit Agent System is depicted. In the following, we describe the tool's main actors, components and the general flow of information from the evidence-producing source to the audit report.

AAS Actors:
There is one actor using the Audit Agent System: the auditor. According to NIST, a cloud auditor is a "A party that can conduct independent assessment of cloud services, information system operations, performance and security of the cloud implementation." [16] Based on this, a cloud customer, cloud provider or any third-party can act as a cloud auditor. From this, requirements regarding depth and presentation of audit results can be derived. Also, since an auditor can be internal or external to an organization (i.e., a cloud service provider), data protection is an issue to consider, when potential confidential information is processed during an audit. These issues shall be addressed by the presentation and anonymization components described later in this section.

AAS Components:
The architecture of the Audit Agent System is based on using software agents to achieve flexibility, address requirements regarding the dynamics of cloud computing (e.g., rapid elasticity) and achieve the necessary extensibility required by the cloud, where evidence data may need to be gathered from highly diverse evidence sources. The proposed architecture comprises of four major functional components: *Audit Policy Module (APM)*, *Audit Agent Controller (AAC)*, *Evidence Processor and Presenter (EPP)* and *Evidence Store (ES)*. For a high-level overview of the system, refer to Figure 1.

Audit Policy Module: There are two types of input to the Audit Agent System:

1) Accountability policies, which define obligations that have to be fulfilled by the cloud provider, such as data access restrictions and usage policies, data retention requirements and general security requirements (e.g., use of encryption). The A4Cloud [17] research project develops a machine-readable policy language based on the Primelife Policy Language [18] called Accountability PPL [19], which will serve as input to the Audit Agent System.
2) Since the A-PPL does not address technical aspects, such as mapping policy requirements to specific tools to use for evidence collection and details of the processing of such evidence, additional manual input is required by the cloud auditor.

The Audit Policy Module (APM) uses both inputs to generate audit tasks. Audit tasks are managed by the Audit Agent Controller.

Audit Agent Controller: The Audit Agent Controller (AAC), can be regarded as the core component of the Audit Agent System. It is responsible for managing the life-cycle of evidence collection agents, controlling audit execution, storage of evidence records and managing data flow between the components. For instance, the Audit Agent Controller deploys, according to what's specified in the audit policy and audit task, evidence collection agents across the various architectural layers of a cloud infrastructure (i.e., in a virtual machine, on a virtualization host, in an application server). From there, data, such as logs, object storage information, block storage information and analysis application output is collected.

Evidence Processor and Presenter: The Evidence Processor and Presenter (EPP) component is responsible for evaluating policies based on the evidence gathered by the audit agents. This component is, similar to the Audit Agent Controller, logically formed by several agents; in this case Processing Agents are responsible for the evaluation of audit policies and Presentation Agents responsible for outputting results to the auditor. The audit results are produced by the audit process and prepared by Presentation Agents according to the auditor's preferred display settings (e.g., a report document or a web-based dashboard).

Evidence Store: The Evidence Store is the central repository for storing evidence records. Some of the more important characteristics of evidence records are, that they are associated with an accountability policy for which they were collected and contain supporting information such as important log entries collected by an agent, which points out a policy violation. For each cloud tenant, there is a separate Evidence Store. This addresses some of the confidentiality and privacy issues associated with a share data pool for potentially sensitive information. Only authorized persons in the role of an auditor may access the Evidence Store.

Multi-layer Evidence Collection:
Collecting evidence in a cloud infrastructure is a very complex process. The main problem lies in integrating a multitude of heterogeneous and distributed sources. As a basis for evidence source classification, we use a simple cloud architecture stack as depicted in Figure 1. There, we consider low-level evidence sources, such as data extracted from the network layer using NetFlow or SNMP, information collected on a virtualization host, information collected inside a customer's virtual machine and also information provided by the software layer (as in SaaS logging). Last, but not least, we consider the cloud management system (CMS), such as OpenStack or OpenNebula to be among the most important sources of evidence, since lots of information provided by CMS logging is directly relevant for auditing against accountability policies (e.g., virtual resource life-cycle and data transfer events for data provenance, and authentication and authorization logging for data security).

B. Policy Input and Audit Task Definition

In this Section, we describe the input to the Audit Agent System, which is derived from A-PPL policies. A-PPL policies capture accountability-related obligations in a policy language.

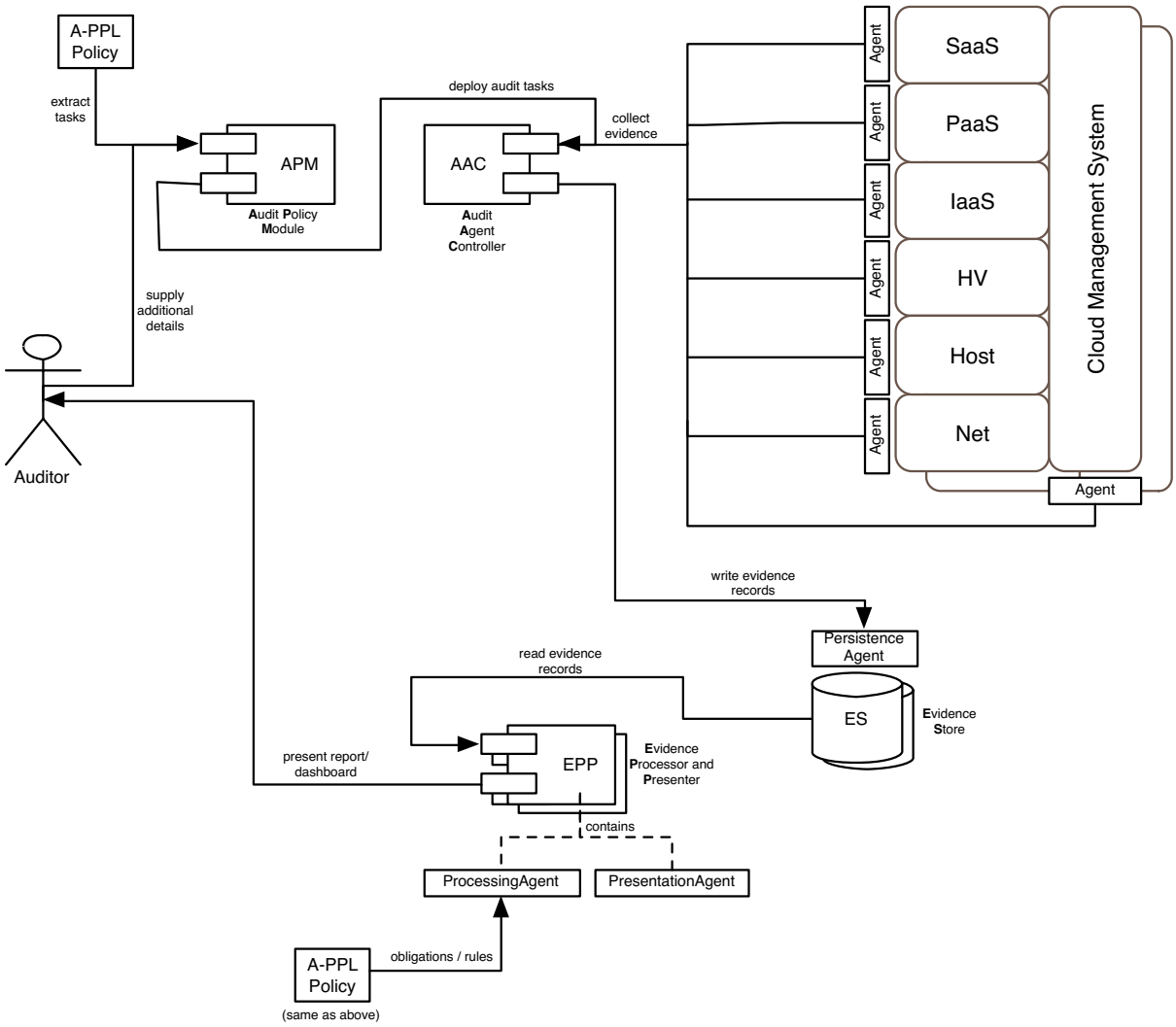

Fig. 1. Audit Agent System Architecture

A-PPL is not part of the Audit Agent System but rather serves as a means for describing what to audit and to decide which accountability requirements need to be fulfilled. A-PPL is developed as part of the A4Cloud research project. It is an extended version of the PrimeLife Policy Language (PPL), which itself is based on the well-established XACML access control management policy language. Therefore, XML as a defining technology, is a given.

Figure 2 depicts the policy-related input as well as the process of deriving audit policies from A-PPL policies. Based on the A-PPL policy input, Audit Tasks are extracted. An Audit Task is a combination of an evidence collection agent (describes where to collect information using which tool), its configuration (which information to collect from a possibly very large pool) and thresholds (limits and conditions that constitute a policy violation). Audit tasks are prepared somewhat similar to templates. For instance, a cloud management system agent is a program that is able to interface (e.g., via the logging and monitoring API) with the CMS and extract certain information. For this, it needs a basic configuration (e.g., how to connect to the CMS). The program and the basic configuration form a template. In the actual audit, the template is populated with all the required basic information (such as authentication credentials and IP address of the CMS), the actual information to collect (e.g., the agent is instructed to build a list of all life-cycle of a virtual machine in a specific time-frame) and possibly a failure condition (e.g., snapshot events are a policy violation).

An Audit Policy, similar to an A-PPL policy containing multiple rules and obligations, contains at least one Audit Task. Several Audit Tasks may need to be executed to be able to evaluate a policy. Performing the reasoning in case of multiple evidence items is part of the Evidence Processor and Presenter.

C. Audit Data Flows

As described in Section III-A, the actual flow of information in the Audit Agent System can be quite complex. In this Section, we describe three different layers that evidence data has to pass through from the collection up to the presentation

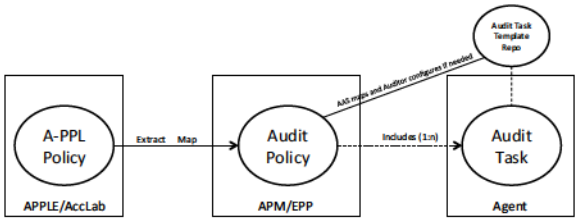

Fig. 2. Audit Agent System Policy Input

of audit results. An overview of this process is presented in Figure 3.

Raw Data: On this layer, information that can be used as evidence is generated. In Section III-A, we mentioned that evidence can be produced by diverse tools in a common cloud infrastructure. Typically, such evidence is generated in the form of logging information, cryptographic hashes (e.g., of files), configuration details or the output of analysis tools, such as from digital forensics tools. Logs, while being similar in general structure (e.g., typically one line per event, beginning with a time-stamp), they differ very much in the used syntax (e.g., time-stamp format, order of event elements, etc.). In our approach, this problem is addressed by interfacing with evidence sources on the Raw Data layer individually. More precisely, for every data source, there is a specialized agent, which is aware of the syntax, semantic and interfaces of the evidence source on one side and of the syntax, semantic and interfaces of the Audit Agent System on the other side. The method of interfacing with an evidence source can be diverse as well. For instance, the agent may (I) use an evidence-generating tool's API to collect information, (II) monitor log files or (III) parse the output of analysis tools.

Agent: Software agents collect evidence data from the Raw Data layer, where it is produced. An agent has two major components, a *Collector* and a *Minimizer*. The Collector interfaces with the evidence source and extracts evidence data. The Minimizer performs several pre-processing actions on the collected data. It removes unnecessary information to reduce the amount of data before transmission to the Audit Agent System core components. The anonymizer is an optional component that tries to remove sensitive information from the collected data in order to protect the confidentiality and privacy of affected persons. How data is to be anonymized and whether or not the anonymization/removal is actually feasible or negatively impacts the audit results has to be decided on a case-by-case basis by an auditor during preparation of the audit task. In any case, the principle of only collecting data that is absolutely required plays an important role at this point of evidence processing and should be observed during audit policy creation. The pseudonymizer works the same way as the anonymizer but allows reversal.

Evidence Processing & Presentation: After the evidence data has been collected and preprocessed by the agent, it is passed to the Evidence Processing & Presentation component. Here, the data passed by the agent is processed by the Evaluator, Aggregator and Presenter components. These components are software agents themselves, but together they logically form the EPP component. The Evaluator is used to

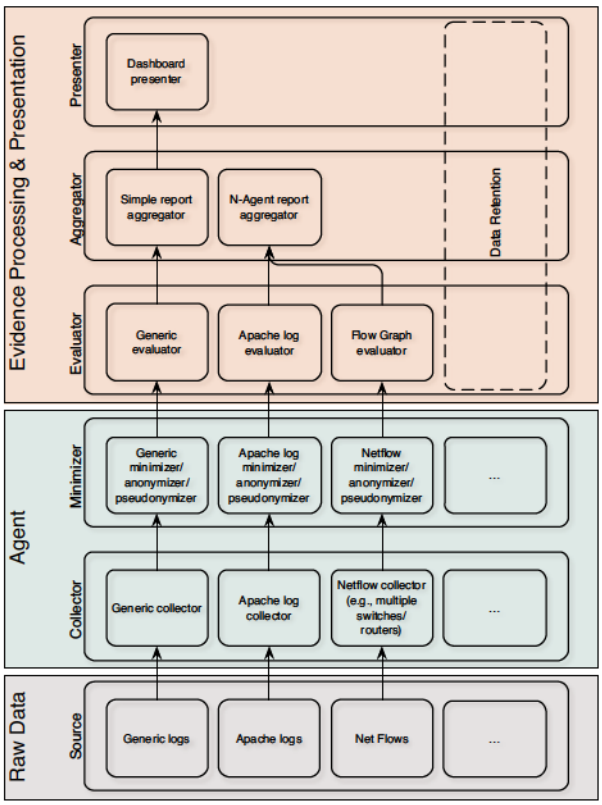

Fig. 3. Audit Agent System Data Flow

compare evidence collected by multiple agents against the policy. Since a single agent can only evaluate parts of the more complex audit policy, the Evaluator is required to put the individual results into context and generate an audit result for the whole audit policy. Evaluators are implemented as an additional agent type (similar to collectors, and aggregators). The evaluation function greatly depends on the input policy and can be as simple as keyword search in text files but also more complex when time lines from various log sources need to be constructed and analyzed. The Aggregator is used to combine the results of multiple audit policies into a single base for the Presenter. There are multiple Presenters, one for each method of presentation and also differing in level of detail depending on the technical knowledge of the auditor. It is very common to have an audit report as a document, which includes the audit result (compliance statement) and if necessary supporting evidence that has been collected. Such documents can be generated automatically to some degree. This form of presentation is most useful, when audit intervals are quite long (for instance in a monthly audit). There is also the presentation of the results in a web-based dashboard, as it is commonly done in monitoring solutions. This approach is more useful, if intervals are short or auditing is done continuously (i.e., as soon as a change event triggers a re-audit), because results can be displayed immediately.

IV. Conclusion

In this paper, we presented a software architecture for performing accountability audits on cloud ecosystems. We

based our approach on the use of software agents, to address problems arising from the wide range of data sources producing evidence and the dynamics of cloud infrastructures. The Audit Agent System is extensible, by allowing to easily develop new agents either on the collection, processing or presentation layer.

We also discussed the input and output interfaces of the Audit Agent System to demonstrate, how such a system can potentially be used by a cloud auditor to automate audit tasks and enable continuous auditing.

By providing cloud customers with such auditing functionality, transparency of cloud services as well as data processing in the cloud can be increased, which may have positive influence on the trust in such services. Additionally, the proposed system enables cloud providers to demonstrate, that they are acting according to the agreed upon policies (between them and their customers), which is a major part of demonstrating accountability.

REFERENCES

[1] IDC for the European Commission, "Quantitative estimates of the demand for cloud computing in europe and the likely barriers to take-up," http://cordis.europa.eu/fp7/ict/ssai/docs/study45-d2-interim-report.pdf, [retrieved: Sep 2014] 2012.

[2] S. Pearson, "Toward accountability in the cloud," *Internet Computing, IEEE*, vol. 15, no. 4, pp. 64–69, July 2011.

[3] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, Information Technology Laboratory, Tech. Rep., 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[4] Cloud Security Alliance (CSA), "The notorious nine - cloud computing top threats in 2013," https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf, [retrieved: Sep 2014] 2013.

[5] National Institute of Standards and Technology (NIST), "Guidelines on security and privacy in public cloud computing," http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144.pdf, 2011.

[6] A. Haeberlen, "A case for the accountable cloud," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, pp. 52–57, Apr. 2010. [Online]. Available: http://doi.acm.org/10.1145/1773912.1773926

[7] D. J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigenbaum, J. Hendler, and G. J. Sussman, "Information accountability," *Commun. ACM*, vol. 51, no. 6, pp. 82–87, Jun. 2008. [Online]. Available: http://doi.acm.org/10.1145/1349026.1349043

[8] F. Doelitzscher, T. Ruebsamen, T. Karbe, C. Reich, and N. Clarke, "Sun behind clouds - on automatic cloud security audits and a cloud audit policy language," *International Journal On Advances in Networks and Services*, vol. 6, no. 1 & 2, 2013.

[9] Distributed Management Task Force, Inc. (DMTF), "Cloud auditing data federation (cadf) - data format and interface definitions specification," http://www.dmtf.org/sites/default/files/standards/documents/DSP0262_1.0.0.pdf, 2014.

[10] Cloud Security Alliance (CSA), "Cloud Trust Protocol," https://cloudsecurityalliance.org/research/ctp, [retrieved: Sep 2014].

[11] O. Q. Zhang, M. Kirchberg, R. K. L. Ko, and B. S. Lee, "How to track your data: The case for cloud computing provenance," HP Labs, Tech. Rep., 2012.

[12] S. Worku, Z. Ting, and Q. Zhi-Guang, "Survey on cloud data integrity proof techniques," in *Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on*, 2012, pp. 85–91.

[13] T. Garfinkel and M. Rosenblum, "A virtual machine introspection based architecture for intrusion detection," in *In Proc. Network and Distributed Systems Security Symposium*, 2003, pp. 191–206.

[14] Nagios Enterprises, LLC, "Nagios," http://www.nagios.org/, [retrieved: Sep 2014] 2014.

[15] New Relic, Inc, "New relic," http://newrelic.com/, [retrieved: Sep 2014] 2014.

[16] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf, "Nist cloud computing reference architecture," http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505, 2011.

[17] "A4Cloud FP7 Project," http://www.a4cloud.eu/, [retrieved: Sep 2014] 2014.

[18] C. A. Ardagna, L. Bussard, S. D. C. D. Vimercati, G. Neven, S. Paraboschi, E. Pedrini, S. Preiss, D. Raggett, P. Samarati, S. Trabelsi, and M. Verdicchio, "Primelife policy language," http://www.w3.org/2009/policy-ws/papers/Trabelisi.pdf, [retrieved: Sep 2014] 2009.

[19] M. Azraoui, K. Elkhiyaoui, M. Önen, K. Bernsmed, A. Santana De Oliveira, and J. Sendor, "A-PPL: An accountability policy language," in *DPM 2014, 9th International Workshop on Data Privacy Management, September 10, 2014, Wroclaw, Poland*, Wroclaw, POLAND, 09 2014. [Online]. Available: http://www.eurecom.fr/publication/4381

A Simulation Framework to Model Accountability Controls for Cloud Computing

Nick Papanikolaou

European Patent Office
Rijswijk, Netherlands
Email: npapanikolaou@epo.org

Thomas Rübsamen, Christoph Reich

Cloud Research Lab
Hochschule Furtwangen University
Furtwangen, Germany
Email: {ruet, rch}@hs-furtwangen.de

***Abstract*—In this paper we present an implemented system to model and visually represent the functioning of accountability mechanisms for cloud computing (such as policy enforcement, monitoring, intrusion detection, logging, redress and remediation mechanisms) over provider boundaries along the supply chain of service providers. Service providers can use these mechanisms, among others, in a variety of combinations to address data protection problems in the cloud, such as compliance failures, losses of governance, lock-in hazards, isolation failures, and incomplete data deletion. The focus here is on technical mechanisms for the purposes of simulation (the currently implemented tool demonstrates policy enforcement, monitoring and logging); in general, an accountability approach requires a combination of technical measures and legal and regulatory support, of course. We survey existing work on accountability in the cloud and discuss ongoing research in the context of the Cloud Accountability project. We discuss modelling considerations that apply in this context namely, how accountability may be modelled statically and dynamically. Details of the current implementation of the Accountability Simulation Engine (ASE), and the first version of a graphical animation of data flows in the cloud, are described.**

Keywords–accountability; data protection; modelling language; simulation; visualisation; sticky policies; policy enforcement; logging; redress

I. INTRODUCTION

In this paper we present the background and modelling considerations associated with Accountability Simulation Engine (ASE), a simulation framework to model and visualize accountability mechanisms for cloud computing. We will discuss the motivation and objectives behind ASE, as well as the features that have been implemented so far. As this is still ongoing work, the primary purpose of the paper is to inform the community and to impart some structure on the development activities; a detailed discussion of future work has also been included.

The starting point for this work is the realization that both cloud computing service providers, as well as customers of cloud computing services, need to have a good understanding of the controls that may be used for managing data flows in the cloud while complying with prevailing data protection laws, rules and regulations, as well as industry standards, best practices, and corporate data handling guidelines in an efficient yet demonstrable manner. The massive scale of cloud computing infrastructures, as well as the enormous complexity of legal and regulatory compliance across multiple jurisdictions, makes this a significant and difficult challenge that service providers, customers, regulators and auditors need to meet on a continual basis.

Accountability for cloud computing service provision is emerging as a holistic approach to this set of issues, and is being actively developed in the context of the Cloud Accountability Project (A4Cloud). Drawing on a multitude of sources, including legal and regulatory frameworks for accountability, as well as technical solutions for achieving data protection compliance, this project aims to provide cloud service providers, auditors, regulators and others with a concrete set of tools for achieving accountability. The simulation framework described in this paper is a research tool whose purpose is to demonstrate how such tools might work, and what problems they are intended to address.

As seen in Section II, accountability encompasses a number of different controls that may be used by a cloud service provider to ensure appropriate governance of their customers data. By developing a simulation of how these controls might or should function, we can reason about their necessity and suitability to particular data handling scenarios. We are mainly interested in technical, automated means of achieving accountability; higher-level mechanisms (e.g. legal rulings or precedents, new regulations, ethical guidelines) are only implicitly modelled as rules incorporated in technical enforcement mechanisms (such as privacy or access control policies).

In order to better understand what form of simulation would be appropriate, we surveyed a number of existing simulation tools and frameworks (Section III). We identified two classes of tools discrete-event modelling formalisms with mostly textual output, as well as visual simulation tools, which permit rapid prototyping, and the creation of graphical animations.

An important part of this work has been identifying what components a suitable simulation model might include, as well as what use cases and scenarios might best illustrate the functionality of accountability mechanisms. These topics are discussed in Section IV. It is interesting to note that there are both static and dynamic aspects of accountability, and different types of simulation are suited to these aspects.

The next section details our model namely, what actors, behaviours and relationships we have chosen to include. The design choices are not definitive, and are likely to vary across use cases. However, this section establishes which kinds of issue could be demonstrated during a simulation, and which responses or mechanisms are appropriate when an accountability-based approach is taken. Section V describes our current implementation of an accountability simulation engine (ASE), which comprises (i) a domain-specific mod-

elling language for accountability scenarios, (ii) an actual simulator for accountability related events, (iii) a web-based user interface for inputting scenario descriptions and observing simulation output, and (iv) a web service which links (ii) and (iii) together. An example of ASEs functionality is given in Section VI, with a simulation of the dynamics of a simple cloud service provision chain. Finally, we conclude with a discussion of future work; this includes prototyping a visual animation of data flows using existing simulation tools and extending the current implementation of ASE with graphical output.

II. The Need for Accountability in the Cloud

As identified by Pearson [1], accountability "for complying with measures that give effect to practices articulated in given guidelines" has been present in many legal and regulatory frameworks for privacy protection; certainly the notion originates from the data protection context, and carries with it the idea of responsible data stewardship. The Galway project attempted to define accountability in this context as follows:

Accountability is the obligation to act as a responsible steward of the personal information of others, to take responsibility for the protection and appropriate use of that information beyond legal requirements, and to be accountable for any misuse of that information.

Pearson [1] observes that the key elements of notion of accountability implied by this definition are *transparency, responsibility, assurance and remediation*. In Pearson and Wainwright [2] it is argued that, to support these elements, it is possible to co-design legal and technical controls for cloud service providers belonging to three categories (i) preventive controls (e.g. risk analysis decision support tools, policy enforcement using machine-readable policies, privacy enhanced access control and obligations), (ii) detective controls (e.g. intrusion detection systems, policy-aware transaction logs, and reasoning tools, notification mechanisms), and (iii) corrective controls (e.g. liability attribution tools, incident management tools). Other categories of controls exist for different kinds of participants in a cloud service provision ecosystem, including end users and regulators.

Our interest is in creating a simulation framework, which enables us to address concerns such as the following:

- What problems can arise in a cloud service provision chain when controls such as those described above are absent from service providers infrastructures;
- What benefits the adoption of such controls can have on service providers operational responses to problems, such as data breaches;
- How accountability can be maintained along a supply chain of cloud service providers;
- What potential impact the introduction of a new control can have on a service providers operations.

While the goal of designing the simulation is to demonstrate the added benefits of adopting an accountability approach, in order to do so it is necessary first to identify the problems and events of interest that this approach provides responses for; both the events and the responses to these events can then be explicitly accounted for in the simulation model. Additionally, audit plans can be derived from this model more precisely and more fully.

A. Data Protection Problems

Based on the risk categorization presented in [3], we identify here five typical classes of data protection problems that cloud service providers need to mitigate:

- Compliance failures
- Losses of governance (e.g. data breaches)
- Lock-in hazards
- Isolation failures
- Incomplete data deletion

Compliance failures. As mentioned in the introduction, cloud service providers need to ensure compliance with prevailing laws and regulations (see [4] and [5]) in the jurisdictions where customers data are stored. This is a non-trivial matter, given that cloud data centres are located in multiple, different locations across the globe, and data often needs to be relocated from one data centre to another for efficiency, bandwidth or other considerations. To ensure compliance on an ongoing basis, applicable local rules need to be checked before, during and after data relocation and evidence has to be given by audits. What makes this particularly complex is that rules are not consistent everywhere, and often transformations need to be applied to the data itself (e.g. in the case of anonymisation of personal data) before a transfer can occur. Any failure to comply with laws and regulations carries significant consequences for the reputation and profits of a service provider; therefore it is of paramount importance to ensure immediate corrective action if any case of non-compliance is detected (e.g. by audits).

Compliance hazards are not confined to legal and regulatory requirements, of course; in order to maintain industrial certifications and badges, service providers need to ensure compliance with appropriate industry standards, whether specific to cloud computing practices, data handling practices, or quality control, among other things. These are typical tasks of a cloud audit system. Failure to maintain such compliance can result in loss of accreditation and, again, loss of reputation for a cloud service provider.

Losses of governance. As data flows from service provider to service provider and beyond, problems can occur at the boundaries: the controls employed by a service provider can only directly ensure appropriate governance of data within the boundaries of that providers infrastructure. The primary cloud service provider within a service provision chain namely, the main cloud service provider in a chain, interacting directly with an enterprise customer loses control over data as it is handed over to that customer. If an entity with malicious intent gains control at the cloud service provider customer interface, this loss of governance on the part of the cloud service provider can have serious consequences for the confidentiality, integrity and availability of the customers data. Data provenance mechanisms, which are not restricted to a single cloud service provider might help to mitigate these problems [6].

Lock-in Hazards. Cloud service providers can create vendor lock-in issues for customers by forcing them to use particular formats for data. If those formats are not widely accepted, it may be very difficult to extract and convert the data for use further down the cloud service provision chain. A hazard can occur during an attempted conversion of data to another format particularly if the format in which the data is stored is encrypted, as such encryption is necessarily lost during the process, thus revealing the data to a potential attacker.

Isolation failures. In a multi-tenanted cloud environment, multiple customers data are stored on the same infrastructure by a cloud service provider; a standard contractual requirement in such a scenario is that isolation of different customers data and operations is maintained; in the absence of such isolation, attacks and hazards affecting one customer can affect another, due to interactions occurring on the common underlying infrastructure. Isolation failures can cause rapid propagation of viruses, worms and similar infections, affecting multiple customers data and damaging the cloud service providers reputation.

Incomplete data deletions. Data retention laws typically require cloud service providers to maintain customer data for a certain period of time after service has terminated. After this period has lapsed, the data has to be deleted from the cloud service providers infrastructure and, depending on the contractual terms applicable for the particular customer, disposed of using particular technical means. Failure to delete data in accordance with the relevant contractual terms can have serious consequences, and could even cause integrity issues for new customers using the same infrastructure if only partially overwritten.

Data protection problems such as the above are illustrative of issues that we need to instantiate in a simulation framework for accountability in the cloud.

B. Addressing Data Protection Problems: Controls for Accountability

While an accountability-based approach to data governance combines a number of mechanisms, ranging from high-level, legal obligations, all the way down to technical controls, our interest is in demonstrating just the latter namely, how technical measures, particularly automated tools, can be introduced into a cloud service providers infrastructure to address issues such as those presented in the previous section. As we have seen, we can classify controls into three categories preventive, detective, corrective depending on whether they are intended as measures to be deployed prior to or after a problem occurs.

Next we describe the types of controls that we are modelling in the ASE framework.

Among preventive controls, we focus on **policy enforcement mechanisms**, in particular tools that allow organisations to ensure that pre-defined, machine-readable policies are enforced automatically within their IT infrastructures. For the purposes of simulation, we will define **accountability policies** and the types of rules that may be encountered in such policies.

Detective controls usually take the form of background processes or aspects in a system; such controls can be active or passive, or some combination of the two. Active controls such as **monitoring** or **intrusion detection** react to particular events and patterns of behaviour, such as threats or data breaches. Passive controls include, for example, **logging tools**, whose function is to record all events that take place (with the source of the event, type of event, and other details) so that a service provider can (i) trace particular activities and identify sources of problems (this relates to attribution capabilities needed for accountability), (ii) prove compliance (with rules, regulations, standards, best practices and more) to external parties such as auditors. An example for such passive controls is Amazon's AWS CloudTrail [7], which provides cloud customers with an API call history and logs.

For corrective controls there is a lack of previous work; mechanisms that are relevant are tools for providing **redress** to customers in cases where data protection problems have not been mitigated by preventive or detective controls. **Incident management tools** are relevant here, but exactly what remedies or responses are appropriate for different types of incidents remains an ongoing research challenge. For the purposes of simulation using ASE, we will assume that financial remedies (including **payment of fines** and other **penalties** for service providers) are suitable responses. The introduction of additional preventive measures, such as storage encryption depending on the type and sensitivity of stored data may also be a response. However, for the purpose of this paper, this is out of scope.

III. Review of Selected Current Simulation Tools and Platforms

Although we have developed the ASE simulation framework from the ground up, we have surveyed and experimented with a number of existing simulation tools; only discrete-event simulation tools have been considered, since our interest is in understanding behaviours and mechanisms that can be effectively modelled using this paradigm.

The tools of interest include software libraries providing dedicated simulation functionality, such as built-in data structures for event queues, random number generation using different probability distributions, timing information and more, as well as visual tools for designing simulations using predefined components.

Discrete-event simulation is detailed in the authoritative text by Law [8], which also includes a library for use in C programs, named *simlib*. This enables one to make use of commonly used data structures for simulation, as mentioned above. There are other libraries with similar capabilities, and indeed *simlib* has been rewritten and adapted for use in other programming languages (e.g. Brian J. Huffman has produced a Java version of the library [9]). We are also aware of the Java-based *Greenfoot* framework [10], which allows simulations to be prototyped easily; so far we have not found a way to turn *Greenfoot* code into web-based applications, which was desirable for our purposes.

An interesting, more recent Java-based simulation library that we experimented with is the agent-based simulation framework MASON [11], which also includes graphical animation capabilities. The distinctive feature of MASON is that it allows one to produce animated graphical user interfaces to demonstrate interactions in multi-agent systems. Since the

demonstrations we have been building are currently relatively small-scale, as opposed to its usual applications, we abandoned MASON early on. Nevertheless, its modular design and graphical capabilities may well be used in future versions of the accountability simulator.

Another approach that we considered included the use of industrial-strength visual simulation tools, such as MATLAB™-Simulink™ [12] and Simio™ [13]. Using the trial version of Simio™, we were able to produce a simple, 3D graphical animation of data flows between cloud service providers, as shown in Figure 1. We were not able to simulate accountability mechanisms using the trial version, as this would require building/coding a significant number of Simio™ processes, a feature that is limited. This will be included in our future work. However, we were able to gain visual insight into the nature and purpose of the simulation, which will be discussed in the next section.

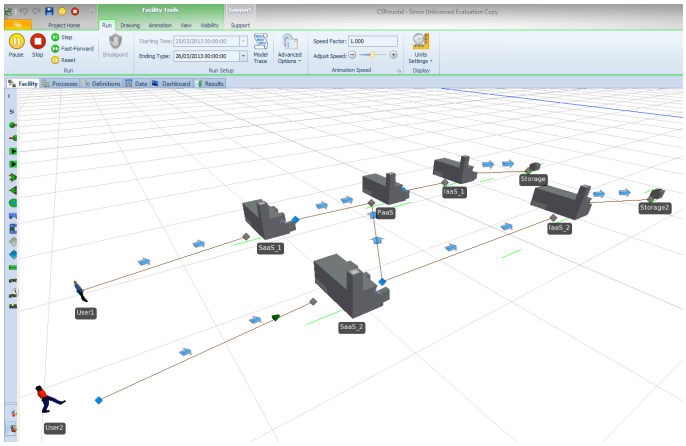

Figure 1. Screenshot of simulated data flows between cloud service providers

IV. Modelling Considerations

As we have seen in previous sections, in order to build a simulation of accountability in the cloud, we need to identify a way to show (i) data protection problems that arise in cloud ecosystems, and (ii) how accountability controls or mechanisms work to mitigate and respond to these problems. The objective of this work is to build a graphical simulation which can provide insight for a variety of stakeholders, including cloud service providers, regulators, auditors and even the general public interested in how accountability can be achieved in a complex chain of cloud service provision. But what should be the underlying conceptual model of the simulation? There are different aspects to consider here.

A. Static Modelling: Actors and Relationships

One aspect to consider is the set of relationships (and the properties of these relationships) between different cloud service providers in a service provision chain. From the data protection point of view, there are different roles for cloud service providers when it comes to handling personal data terms used in the European Data Protection Directive 95/46/EC [5] for these roles are *data controller* and *data processor*. Depending on the service offering, providers may take one or both of these roles, with complex and ambiguous cases arising frequently. Modelling what this implies in terms of concrete obligations for cloud service providers is what we will refer to as static *modelling of accountability*.

TABLE I. The possible roles that the different kinds of actors can take on in a given scenario as per our model.

Actor Type	Possible Roles
Individual	Data subject
Cloud service provider	Data controller Data processor
Third party	Data controller Data processor Accountability Agent
Auditor	(Auditor) Accountability Agent
Regulator	(Regulator) Accountability Agent

The static modelling of a cloud service provision chain involves identified actors, roles and responsibilities and the relationships between them.

1) Actors, Roles and Responsibilities: In our model, in a cloud ecosystem there are five different kinds of actors **individuals, cloud service providers, third parties, auditors** and **regulators**. We classify the different types of roles that these actors may take on in a particular scenario into six kinds **data subject, data controller, data processor, accountability agent, auditor** and **regulator**. A particular scenario is defined as a specified set of roles for a specified set of actors.

The possible roles that the different kinds of actors can take on in a given scenario as per our model are defined in Table I.

The roles that we have included take into account the static modelling discussion in Section IV-A. *Accountability agent* represents a role that is intended to encompass internal oversight activities within an organization (e.g., self-auditing), as opposed to the roles of *auditor* (an external entity performing an audit on behalf of enterprise) and *regulator* (typically a government entity responsible for setting, implementing and monitoring standards), which by definition correspond to oversight external to an organization; note that we model two classes of organizations here cloud service providers and third parties, the latter being providers of non-cloud services. The distinction becomes clearer when we consider relationships that can exist between actors.

2) Relationships: Cloud service providers are characterized in the model by the kind of relationship they have with other providers, in particular, what kind of service they offer to others. A cloud service provider provides one of three kinds of service: **IaaS** (infrastructure-as-a-service), **PaaS** (platform-as-a-service), **SaaS** (software-as-a-service). These are the only kinds of relationships considered here between cloud service providers. Third parties are entities that enter into complex contractual relationships with cloud service providers, relationships that are not of the same kind. Further investigation is needed here, but for the purposes of modelling and simulation we do not need to restrict the kinds of relationship that third parties may have (with each other and with cloud service providers).

B. Modelling System Dynamics: Data Transfers and Accountability Mechanisms

While static modelling would enable us to simulate what effect particular assumptions might have on the obligations of a cloud service provider, *modelling system dynamics* enables us to simulate data flows between cloud service providers, data protection problems and their consequences when accountability controls are in place (and similarly when such controls have not been introduced). For a dynamic simulation, the main entities that need to be modelled are *personal data*; at each step of the simulation, personal data flow through a chain/sequence of service providers, which are predefined, and together constitute a model of a real-world cloud service provision chain. The purpose of the simulation becomes to show what happens to the data as they flow through the chain, and what effect these flows have on properties of the overall system.

So, what is our model? The entities modelled have been discussed in the previous subsection, along with their relationships; next, we discuss their expected behaviours, and the types of issues or problems that can be simulated using our model, and the responses that different entities can have and should have to such problems if accountability is to be achieved.

1) Behaviours corresponding to different types of role: First, consider the behaviours of individual data subjects. In our model, a data subject is an entity that can engage in one of the following actions at any time during a simulation:

- Create data (a datum is modelled simply as a pair of strings an identifier and a value)
- Modify/edit data
- Delete data
- Change preferences regarding usage of data (initially, a data subjects policy is simply a statement of for what uses data can be processed, and whether the data can be shared with third parties)
- Request summary of data and preferences held

For service providers, which are typically data controllers or data processors, the following actions are possible:

- Store data
- Encrypt and store data
- Decrypt data
- Check preferences and share data
- Create new policy over data
- Generate log of activity over data
- Enable/disable logging mechanisms
- Enable/disable monitoring mechanisms
- Enable/disable policy enforcement mechanisms

Regulators and auditors are modelled as having the following possible actions:

- Check compliance of data controller/processor with a specified rule or set of rules
- Check compliance of organizational policies with minimum requirements
- Create new rules specifying allowed uses of customer data, and penalties/remedies in case of non-compliance or other problem
- Create new rules specifying mechanisms that must be used to protect data subjects, and penalties/remedies in case of non-compliance or other problem
- Enforce penalty or other remedy in case of non-compliance or other problem
- Audit a data controller/processors system logs (esp. check origin, route, destination of data; intended use; protection mechanisms used, whether customers preferences were enforced)

Accountability agents are initially to be modelled as a variant of auditor, with the only difference that they cannot perform enforcement, only (implicitly) inform the organization they are associated with of any events of interest (e.g. failures, non-compliance). Further work may reveal other actions/events of interest.

2) Simulated Issues: In the simulation, we should be able to represent and visualize some of the issues discussed previously in Section II.A. Compliance failures, data breaches and direct attacks on a service providers infrastructure are specific events that we have so far considered in this work.

3) Simulated Responses: In Table II we can see how the different accountability mechanisms considered in our model can help to address the simulated issues. We note that this list is not exhaustive, as it only includes the mechanisms we have considered so far; other mechanisms could include, for example, automated tools for punishment or remediation; also, we have so far avoided detailing what types of rules are allowed in policies. In the Cloud Accountability Project, which is the context in which the simulation has been built, there is ongoing work on developing an accountability policy language; for the purposes of our simulation, we have so far assumed that rules restrict to whom and under what conditions data can flow; the distinction between data controller and data processor may well imply additional restrictions, and similarly there are restrictions on when data can be transferred to third parties (this is modelled as a preference that data subjects can set).

As we can see from the table, when none of the accountability mechanisms are enabled in a simulation, none of the problems considered trigger any response (thus allowing hazards and failures to occur). Of interest is the fact that hazards can then propagate (cascade) from one provider to another, and/or to any third parties. All other cases cause a response, thus demonstrating how accountability mechanisms work in practice.

V. Implementation

We have implemented three software components as part of the accountability simulator: a simulation engine, a web-based animation of data flows between cloud service providers, and a web service that draws data from the simulation engine. Currently we are continuing implementation work until all

TABLE II. PROBLEM AND THE SPECIFIC RESPONSES TRIGGERED BY ACCOUNTABILITY MECHANISMS IN THE SIMULATION MODEL

Problem	Mechanism			
	None	Policy enforcement	Monitoring	Logging
Compliance failure	X	All problems correspond to specific policy violations; Policy violation will be detected; Parties notified	Patterns of non-compliance can be detected	All failures will be logged and shown to auditors
Data breach	X		Monitoring interaction of service providers with untrusted third parties can provide advance warnings	All breaches will be logged and shown to auditors
Attack	X		Intrusion detection systems can be used to thwart attacks	All attacks will be logged and shown to auditors

three components have been fully integrated. In this section we present the functional structure of the simulator and then detail each of the implemented components.

The input file is a description of a scenario to be simulated, and is written in the accountability model description language, described in the next section. A scenario consists of a specified set of actors (so far we have not made the distinction between actors and roles in the language, but this is forthcoming in future versions), a specified set of relationships, configuration of options/parameters and the triggering of actions of particular actors.

When an input file is supplied to the simulator (via a web-based interface or through the command line), its contents are parsed using the language interpreter, which invokes appropriate methods in the accountability simulation engine. The accountability simulation engine contains the current state variables and the log of events executed so far; it constitutes the *backend* of the application and is written in plain Java.

In order to feed the state of the simulation, timings and outcomes to the web-based user interface, we have implemented a RESTful web service using the Java-based Restlet EE framework [14].

The initial version (designated *v1*) of the web-based user interface was implemented using HTML forms, and the data it receives from the web service consist of plain text strings that are displayed and updated as the simulation progresses, without any graphical animation.

The latest version (designated *v2*) of the web-based user interface has been developed separately, as a graphical animation, and work is ongoing to link it up to the web service. This will be discussed further in Section V-C below.

A. Accountability Model Description Language

Scenarios to be simulated are described by the user of the simulator in a custom modelling language we have built for this purpose. At this stage of development we have only included a core set of commands and mechanisms that can be included in scenarios, but we expect to add constructs in the language so as to allow inclusion of detailed privacy policies, access and usage controls, and other features. In particular, in the Cloud Accountability Project there is ongoing work on developing a dedicated accountability policy language, and it is likely that the constructs of that language will be incorporated into the language of the simulator.

Listing 1 shows the productions for the languages grammar, using the syntax of the SableCC [15] parser generator that we have been using to build the interpreter. The nonterminals in the grammar are *command* (for top-level commands), *declare* (used to declare actors of different types), *type* (representing the different actor types, namely user, cloud service provider, auditor and regulator), *servicetype* (for the different types of cloud service), *objectaction* (this is for expressions representing a property or action of a given actor), action (properties or actions), mode (for data protection problems that can be simulated using the trigger command), *mech* (for accountability mechanisms that can be enabled or disabled as needed). Commands *setgraph*, *setpolicy* and *setconstraint* are experimental; the command *graph* allows us to access the internal data structure of the accountability simulation engine and visualize it using AT&T GraphViz [16].

B. Accountability Simulator (Backend component)

The accountability simulation engine is responsible for maintaining and updating the current state of the simulation, and currently its main visible function is to display that state on the console or supply it (through a web service) to another application.

We can denote the internal state of the simulation engine by a tuple (see Equation 1)

$$(A, T, \rho, \sigma, M, \xi) \tag{1}$$

where A is the set of actors that have been declared, T is the set denoting the types of the actors, is the set of relationships between cloud service providers (the only relationships modelled are between these types of actor), is the store of data values held by the different actors (indexed by A), M is the set of accountability mechanisms enabled and is the output stream (this represents e.g. the standard output or a pipe to another application, or a web service).

```
command =
{ declaration } declare |
{ custdeclaration } customer lparen
        [ fst ]: identifier [q ]: comma
        [snd ]: identifier [z ]: comma
        servicetype rparen |
{action } objectaction |
{ trig } trigger mode identifier |
{setgraph } setgraph arglist |
{ setpol } setpolicy lparen identifier comma
        str rparen |
{setcons } setconstraint lparen [ fst ]: identifier
        [q ]: comma [snd]: identifier [z ]: comma
        str rparen |
{enablemech } enable mech |
{disablemech } disable mech |
{graphing } graph;
```

```
declare = {declplain} type identifier |
          {declqualified} type identifier str ;

type = {userdec} user |
       {cspdec} csp |
       {auddec} auditor |
       {regdec} regulator ;

servicetype = {iaastype} iaas |
              {paastype} paas |
              {saastype} saas ;

objectaction = {actionref} identifier dot action ;

action = {actionsenddata} senddata
         lparen identifier comma str rparen |
         {evalstate} state ;

mode = {databreach} databreach |
       {attack} attack ;

mech = {polenfmech} polenf |
       {logmech} logging |
       {monmech} monitoring;
```

Listing 1. SableCC grammar of the Accountability Simulator input language (only the main productions are shown)

The output of the simulation depends on which accountability mechanisms have been enabled; if no mechanisms are enabled (in which case the value of M above would be the empty set, $\emptyset$), then there is no change in the output ξ when a problem is triggered. However, when mechanisms are enabled and a problem is triggered, the effect (as described in Table II) is made visible on the output. In other words, the Function of the simulator (see Function 2) can be summarized operationally as a state transition of the form:

$$(A, T, \rho, \sigma, M, \xi) \rightarrow (A, T, \rho, \sigma', M, \xi') \qquad (2)$$

such that ξ' differs from ξ as it contains a notification of a compliance failure, data breach or attack when the trigger command is issued and one of the following is true:

$$(\{policyenforcement : enabled\} \in M) \ or$$
$$(\{logging : enabled\} \in M) \ or$$
$$(\{monitoring : enabled\} \in M)$$

It would not be difficult to use the above notation to derive a full operational semantics for the simulator, but for our purposes here it is sufficient to note that the main function of the tool, which is to behave differently depending on which type of problem is being simulated, and which mechanisms are enabled. So far we have assumed ξ represents textual output, namely strings describing the overall system state, such as lists of actors, their data values and more. Of course, the exact output consists of messages corresponding to the responses shown in Table II. In the next section we turn to work we have done on developing a graphical visualization.

C. Web-based Front-End

The vision for the web-based user interface has always been to have a graphical animation of data flows between individuals, cloud service providers, auditors and regulators and third parties. Demonstrating flows of data and the changes that occur to data and providers in the process emphasizes the dynamic aspect of the simulation. A screenshot of our current prototype of version v2 is shown in Figure 2.

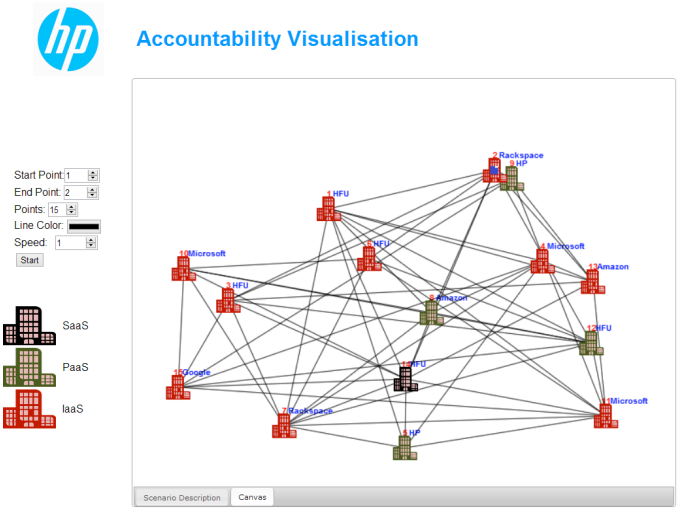

Figure 2. Visualisation front-end for the Accountability Simulator

It is very important to note that this version has been developed as a separate, standalone animation. Thus, it has not yet been linked to the web service component and, hence, the main simulation engine. However, it does show an instantiation of a random set of cloud service providers of different kinds (IaaS, PaaS, SaaS, and how a data item can be routed between providers. In the animation, the scenario is assumed to be random, rather than specified in the accountability modelling language; this is changing presently.

We expect to have dedicated controls (form buttons) to trigger particular data protection problems, and panels showing the responses produced by the simulator. In the screenshot in Figure 2 two tabs are shown at the bottom. While the (currently random) animation is shown on the *Canvas* tab, the other tab (titled *Scenario Description*) will allow the user of the simulation to supply an input file in future, written in the accountability modelling language of Section V-A, and this will be used to generate the animation in the next version.

VI. An Example Input File

Listing 2 shows an example input file that we have tested with the current version of the simulator. It describes a scenario in which there are two individual users, four cloud service providers, an auditor and a regulator. The relationships between the individuals and service providers are then declared. John and Mary then create some data, which is sent and stored on the specified service providers infrastructure. In line 15 a data breach problem is triggered; this has no effect when simulated as no accountability mechanisms have been enabled; the remaining lines enable different mechanisms, and trigger

a data breach and attack. Naturally the simulator produces a sequence of long warnings when interpreting lines 17, 20 and 22, as the policy enforcement, monitoring and logging mechanisms kick in.

```
User john "John Wayne";
User mary "Mary Wollstonecraft";
CSP salesforce " Salesforce .com";
CSP amazon "Amazon Web Services";
CSP rackspace "Rackspace";
CSP hpcs "HP Cloud Services";
Auditor kpmg "KPMG";
Regulator cnil "CNIL";
Customer(john,amazon,SaaS);
Customer(mary, salesforce , SaaS);
Customer(rackspace,hpcs,IaaS);
Customer( salesforce , rackspace , PaaS);
john.Senddata(amazon,"somedata");
mary.Senddata(hpcs," marydata");
Trigger Databreach salesforce ;
Enable Polenf;
Trigger Databreach salesforce ;
Enable Monitoring;
Enable Logging;
Trigger Databreach salesforce ;
amazon.State;
Trigger Attack amazon;
```

Listing 2. Example script written in the accountability simulation language.)

VII. Conclusion

In this paper we have presented the design and implementation of a simulator for accountability mechanisms in the cloud. We have discussed data protection problems, and how mechanisms for accountability such as policy enforcement, monitoring and logging can help to address such problems; the simulator we have built is a tool to assist understanding and modelling of real-world scenarios and will hopefully be a useful aid to cloud service providers, regulators and end users as it is extended with more features.

Future work will focus on integrating the v2 web-based UI with the accountability simulation engine and web service, and enriching that UI with more controls. Subsequently we will work on animating the accountability mechanisms and modelling additional ones.

It is also worth noting that a new EU Data Protection Regulation will eventually replace the current directive, which is under discussion in the European Parliament. This is likely to include new accountability rules and obligations, and must be taken into consideration in future work. For the purposes of this paper, however, we have focused on modelling the dynamics of accountability controls and how they impact data and data flows in cloud infrastructure.

Acknowledgment

Nick Papanikolaou wishes to cordially thank Siani Pearson and Nick Wainwright for their support and feedback during the development of this work. Special thanks are due to Fabian Reich, who spent a month at HP Labs working with Nick Papanikolaou and actually coded version v2 of the web-based GUI, navigating the vagaries of the HTML5 Canvas control. This work is supported by the European FP7 Programme A4CLOUD: Accountability for the Cloud and Other Future Internet Services.

References

[1] S. Pearson, "Toward accountability in the cloud," Internet Computing, IEEE, vol. 15, no. 4, pp. 64–69, July 2011.

[2] S. Pearson and N. Wainwright, "An interdisciplinary approach to accountability for future internet service provision," International Journal of Trust Management in Computing and Communications, vol. 1, no. 1, pp. 52–72, 01 2013.

[3] T. Haeberlen, L. Dupre, D. Catteddu, and G. Hogben, "Cloud computing: Benefits, risks and recommendations for information security," enisa - European Network and Information Security Agency, Tech. Rep., 2012.

[4] "Apec privacy framework," http://publications.apec.org/publication-detail.php?pub_id=390, [retrieved: 2014.04.08] 2005.

[5] "Directive 95/46/ec of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data," http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML, [retrieved: 2014.04.08] 1995.

[6] O. Q. Zhang, M. Kirchberg, R. K. L. Ko, and B. S. Lee, "How to track your data: The case for cloud computing provenance," HP Labs, Tech. Rep., 2012.

[7] "Amazon aws cloudtrail," https://aws.amazon.com/de/cloudtrail/, [retrieved: 2014.04.08].

[8] A. Law, Simulation Modeling and Analysis (McGraw-Hill Series in Industrial Engineering and Management). McGraw-Hill Science/Engineering/Math, 2006.

[9] B. J. Huffman, "An object-oriented version of simlib (a simple simulation package)," Informs Transactions on Education, 2001.

[10] M. Koelling, Introduction to Programming with Greenfoot: Object-Oriented Programming in Java with Games and Simulations. Pearson Education, 2009.

[11] "Mason multiagent simulation toolkit," http://cs.gmu.edu/~eclab/projects/mason/, [retrieved: 2014.04.08].

[12] "Matlab and simulink," http://www.mathworks.co.uk/products/simulink/, [retrieved: 2014.04.08].

[13] "Simio," http://www.simio.com/, [retrieved: 2014.04.08].

[14] "Restlet framework," http://restlet.org, [retrieved: 2014.04.08].

[15] E. Gagnon, "Sablecc, an object-oriented compiler framework," http://www.sablecc.org/, [retrieved: 2014.04.08].

[16] "Graphviz – graph visualization software," http://www.graphviz.org, [retrieved: 2014.04.08].

Privacy-Friendly Cloud Audits with Somewhat Homomorphic and Searchable Encryption

José M. López, Thomas Ruebsamen, Dirk Westhoff
Hochschule Furtwangen University
Furtwangen, Germany
{jose.lopez, thomas.ruebsamen, dirk.westhoff}@hs-furtwangen.de

***Abstract*—In this paper, we provide privacy enhancements for a software agent-based audit system for clouds. We also propose a general privacy enhancing cloud audit concept which, we do present based on a recently proposed framework. This framework introduces the use of audit agents for collecting digital evidence from different sources in cloud environments. Obviously, the elicitation and storage of such evidence leads to new privacy concerns of cloud customers, since it may reveal sensitive information about the utilization of cloud services. We remedy this by applying Somewhat Homomorphic Encryption (SHE) and Public-Key Searchable Encryption (PEKS) to the collection of digital evidence. By considering prominent audit event use cases we show that the amount of cleartext information provided to an evidence storing entity and subsequently to a third-party auditor can be shaped in a good balance taking into account both, i) the customers' privacy and ii) the fact that stored information may need to have probative value. We believe that the administrative domain responsible for an evidence storing database falls under the adversary model "honest-but-curious" and thus should perform query responses from the auditor with respect to a given cloud audit use case by purely performing operations on encrypted digital evidence data.**

***Index Terms*—Cloud Computing, Audit, Evidence, Computing on Encrypted Data, Somewhat Homomorphic Encryption, Searchable Encryption.**

I. Introduction

Recently, we face a growing need for accountability in cloud services, such that cloud-adopted digital evidence collection frameworks are getting momentum. Exemplary formats and frameworks have been proposed by the Common Digital Format Working group (CDESF) [18] or the Distributed Management Task Force (DMTF) [7]. According to [17], a generic evidence collection framework can be derived from the process of digital investigation defined by the National Institute of Standards and Technology (NIST) in [13]. This proposed process is split into a securing phase, an analyzing phase, and a presentation phase. In the securing phase the sources of evidence can be manifold, ranging from information from the network, to the host operating system (OS), the hypervisor and respectively the Infrastructure, Platform and Software as a Service (IaaS, PaaS, SaaS) layers. Also, the cloud management system (CMS) itself is a valuable resource of evidence. Additionally, more complex cloud service provision scenarios have to be considered, e.g., inter-cloud scenarios where complex provider chains are dynamically established to provision cloud services. Clearly, the entity which stores and bundles such a diversity of fine granular customer information has to be designed extremely carefully. Data leakage, regardless of whether it was caused by insider adversaries or by outsider adversaries of the cloud storage architecture, immediately causes loss of reputation and revenue for the involved cloud provider and, even more importantly, threatens the privacy of the cloud customer.

We believe that, besides other privacy preserving concepts like pseudonymization or pseudonym unlinkability, two predominant privacy preserving techniques have to be in place:

1) *Canniness of evidence data*: only such evidence data shall be collected and stored at a central entity which mandatorily are required to support a clearly defined cloud auditing use case;
2) *Encrypted evidence data*: digital evidence mandatorily needs to be encrypted when stored at a central entity;

With respect to the envisioned privacy preserving cloud audit agent system for third party audits, this means that an *Audit Agent Controller* (AAC) only deploys *audit agents* (AA) for evidence collection at several cloud subsystems which i) sparely collect digital evidence, and ii) encrypt digital evidence before transmitting such potentially sensitive information to a central entity for storing and subsequently pulling together data from different sources. Thus, the *Evidence Store* (ES) purely stores ciphered digital evidence from different subsystems and sources. Moreover, it shall never be able to gain the decryption keys. So, we feel what is essentially needed is a mixture of organizational aspects e.g. a privacy preserving storage policy and privacy preserving techniques like encryption. Flavors of the latter will be investigated in more depth in the remainder of this work. Our main contribution is to demonstrate the applicability of specific cryptographic schemes, i.e. SHE and PEKS, as privacy enhancement mechanisms for a software-agent audit system for clouds.

This paper is structured as follows: In Section II we summarize the existing related work. In Section III we describe the conceptual architecture and administrative domains of our approach. Later, in Section IV we give flavors of possible cloud audit use cases. In Section V we provide a brief introduction and definition of the required cryptographic primitives. Then, in Section VI we develop a cloud audit use case using our proposed framework. In Section VII we provide an estimation of the performance and the security and, finally, in Section VIII we conclude our work.

978-1-4799-5350-9/14/$31.00 ©2014 IEEE

II. Related Work

Our work requires the capability to collect digital evidence in the cloud. However, deploying additional software for "monitoring and evidence collection" inside the targeted virtual machine (VM) can not guarantee the reliability of the evidence collected. For example, a customer having full administrative control over the targeted VM might tamper the deployed monitoring tool to report fake data. A solution to this problem is to follow an out-of-guest approach like the one described by Carbone et al. in [8], where the authors propose to move the monitoring tool outside the focused VM in combination with function-call injection techniques. By following this approach, the monitoring tools are tamper-proof resistant against the customer resulting in more reliable evidence.

Wang et al. [20] propose a provable secure privacy-preserving third-party auditing protocol for verifying the integrity of data stored in the cloud. On behalf of the customer, this protocol allows an external third party auditor (TPA) to check storage correctness of the customer's data in the cloud. They consider the following requirements for asecure third-party cloud audit[1]: 1) the auditing process should not introduce new vulnerabilities towards user data privacy, 2) the auditor should be able to efficiently audit the cloud data storage without demanding the local copy of data.

Furthermore, we refer the interested reader to [17], [9], [8], [19]. There the authors provide frameworks for either: i) collecting and storing digital evidence, or ii) allowing a third-party auditor, who on behalf of the customer, is able to audit the cloud and verify the compliance of the contract.

III. Architectural Overview and Impact of Trust Boundaries

In Figure 1 we do illustrate the architectural overview of a cloud audit system using audit agents for the collection of digital evidence by following [17]. However, in addition to [17] we also highlight the involved administrative domains to derive trust boundaries and implicit trust delegations of the proposed architecture. Per *cloud provider* (CP) an *audit agent controller* (AAC) is deployed. This component can tap into several cloud subsystems and deploy *audit agents* (AA) for the evidence collection. The various AAs of the involved cloud providers continuously or periodically send the monitored digital evidences to the AAC, which, subsequently after aggregating input from the various AAs, sends such data to the *evidence store* (ES). Different ES per customer do exist. By using an evidence processor (EP), the (third-party) *auditor* receives reports with evaluation results from an ES eventually comparing it with pre-defined audit policies. However, as will be detailed in sections IV and VI-A, third-party auditors must be restricted in the amount and type of information they can request to the EP. It should be just enough to perform the specific audit goal. Obviously, the very fact of the availability of such an amount of customers' meta data from the different CPs stored at the ES is a valid reason to properly protect such a component.

It is important to stress that the above architecture consists in its most general case of at least $d = 2n + 2$ administrative domains where n is the number of involved CPs and, moreover, assuming a single provider for all per-tenant ESs, the third party auditor, and, finally, an administrative domain for each AAC per CP including its deployed AAs.

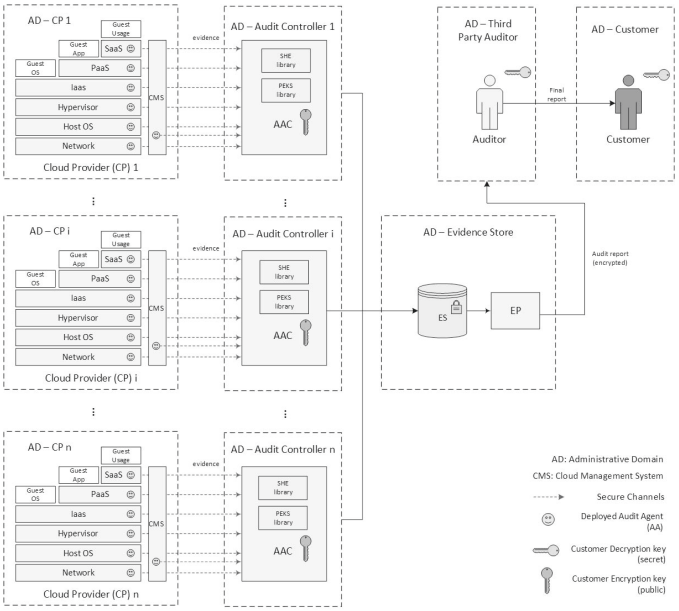

Figure 1. Architectural overview and administrative domains on a per-tenant view.

With the above trust boundaries in mind our extension to the architecture are as follows: Since an AAC is typically placed on a different server than the CP's services with its various AAs, we propose to securely tunnel all transmitted digital evidence. As said, since the AAC and the AAs belong to the same administrative domain, static security associations are available and, thus, secure tunneling can be established relatively easy by applying a trusted and secured VPN using PPTP, IPSec or SSL.

The privacy extension we propose in this work mainly addresses the *honest-but-curious* adversary model[2], since we can not exclude the EP to behave suchlike. In case of digital evidence data being stored in plaintext, this party could probably make use of such information (e.g., by selling or otherwise providing it to another party which is forceful enough to influence the ES). Please note that the express warranty of the integrity of evidence data is out of scope of this paper. However, the use of HomMACs [1] and homomorphic signatures [4] respectively, may be appropriate candidates in combination with SHE and PEKS to prevent malleability. This,

[1]Cloud Audit [17]: "The independent examination of records and activities to establish controls, policies, operational procedures and mechanisms, and expected means of remediation and to to recommend any indicated changes in controls, policy or procedures".

[2]A party is "honest-but-curious" if he executes the protocol correctly but tries to learn as much as possible.

we plan to investigate in our future work.

IV. Audit Event Use Cases

As previously mentioned, we assume *canniness of evidence data* to be a prerequisite and thus do only consider the collection of digital evidence at an audit agent which has a clear relevance for concrete cloud auditing use cases. Such use cases have been identified by the *Cloud Auditing Data Federation* (CADF) in [7].

The following cloud audit use cases will serve as a benchmark to evaluate whether a SHE or a PEKS scheme are appropriate building blocks for privacy-enhanced cloud audits.
Use case 1: Auditing access to a controlled resource. In this use case the customer would like to monitor and log all user "login" actions against all servers within their infrastructure, e.g. when a user attempts to login to a data base or to an internal service.

We consider the following example: A company *X* has outsourced some of its internal IT services to a cloud service provider *Y*. That service provider is specialized in providing Software as a Service (SaaS). A security policy at the company *X* states, that while any employee may use the service provided by *Y*, this may only be done when connecting from the company's network, to ensure additional security systems, e.g. transparent security proxys deployed by *X* are not circumvented. An audit of the CSP's access control logs is supposed to verify, that this policy is being adhered to, and, no access from an unauthorized location has occurred. Therefore, *Y* records service access attempts on a per-tenant basis including information about source of the access in the form of IP addresses, associated user, time and date, as well as its access control decision. Since this is information that can be considered sensitive, with respect to the possible leakage of the service user's behavior, encryption mechanisms shall be employed to protect the user's privacy. W.l.o.g. we assume in this example use case a specific size of *X*'s company network and the use of public IP addresses. In CIDR notation, e.g. a.b.c.d/y, the IP range of the company's network segment can be derived from the routing prefix size. In IPv4 this corresponds to:

$$N_{IP} = 2^{32-y} \tag{1}$$

Based on the previous example we consider the following *security policy* agreement between *X* and *Y*:

- IP addresses within the range 142.168.1.0/24 are classified as *authorized* or *unauthorized* otherwise.
- *X*'s employees can access the service provided by *Y* only if they are connecting from an authorized IP address.
- *X* authorizes *Y* to log access control information as evidence of compliance of this policy with the following restrictions:
  1) The location of the users (e.g. authorized IP addresses) is considered as sensitive.
  2) The duration for every connection to the service is considered as sensitive.
  3) Sensitive information should be protected against data leakage, insiders, outsiders as well as third-party auditors.

We argue that an IP address can be linked to a location, such that its revealment to third parties breaks the privacy of the user (X's employee). Additionally, the "connection time duration" represents the time an employee remains connected to the service. Such information may reveal some working habits of the employee. A suspicion, let it be reasonable or non-reasonable, could be that those which spend less time connected to the given service work less than those who do not.
Use case 2: Periodic monitoring resource status. Software per server provides periodic informational status of each server's CPU utilization along with metric data.
Use case 3: Aggregation of resource status into an audit event. The "monitoring server" summarizes these periodic measurements from the agents, by calculating an 'average utilization value' and then generates a single informational status.

Through the rest of this document we will focus on the cloud audit use case 1 to demonstrate the viability of our framework.

V. Encryption Candidates

In this section we provide a background of the chosen encryption mechanisms, namely "Homomorphic Encryption" and "Searchable Encryption". Additionally, we provide the definition for the chosen cryptographic primitives which will be referred to in Section VI.

A. Homomorphic Encryption

If someone wants to protect the confidentiality of a message, he could just encrypt it using standard encryption schemes. However, this produces a ciphertext in a way it is not possible to operate on it. In 1978 Rivest, Adleman and Dertouzos [16], proposed the idea of computing with encrypted data. They refer to it as *privacy homomorphisms*. This concept was later known as *Homomorphic Encryption* (HE).

By using HE, one could publicly perform computations on ciphertexts and obtain an encrypted result which, when decrypted, is the same as if the computations were performed on the plaintext. Let $\mathcal{E}$ be an encryption scheme and $*$ denote an operation in the ciphertext space, then:

- $\mathcal{E}$ is additive homomorphic if:

$$m_1 + m_2 = \mathcal{D}\left(\mathcal{E}(m_1) * \mathcal{E}(m_2)\right)$$

- $\mathcal{E}$ is multiplicative homomorphic if:

$$m_1 \cdot m_2 = \mathcal{D}\left(\mathcal{E}(m_1) * \mathcal{E}(m_2)\right)$$

Encryption schemes like RSA, ElGamal or Paillier cryptosystem support an *unlimited* number of additions or multiplications on the ciphertext, but *not both* at the same time. Thus, they are known as *partial homomorphic* encryption schemes (PHE). In [5] Boneh et al. presented an encryption scheme which supports an unlimited number of additions and one multiplication carried out in the ciphertext, introducing the

notion of *somewhat homomorphic* encryption (SHE). In [10] Craig Gentry presented the first construction of a *fully homomorphic* encryption (FHE) scheme, which is capable of evaluating unlimited number of additions and multiplications on the ciphertext. However, this scheme is far from being practical due to the intensive computation required.

Somewhat Homomorphic Encryption (SHE): There is an active research in the area of SHE with the aim of creating encryption schemes for *efficiently* evaluating *concrete* functions, which require the evaluation of unlimited additions and a *bounded* number of multiplications. As will be described in chapter VI-A, for this work we need to compute the mean and standard deviation on the ciphertext space, for which Brakerski et al.'s SHE scheme [6] is the best candidate due to its efficiency and functionality.

At next we present Brakerski and Vaikuntanathan's scheme [6], an efficient *somewhat homomorphic* scheme secure under the "ring learning with errors" (R-LWE) assumption [12].

Definition 1. (Somewhat Homomorphic Encryption Scheme). The scheme $\mathcal{SHE}$= (SH.KeyGen, SH.Enc, SH.Sum, SH.Mult, SH.Dec) consists of following algorithms:

1) SH.KeyGen(k): Given a security parameter k, this algorithm outputs the public and secret key pair (pk, sk).
2) SH.Enc(pk, m): Given a public-key pk and a message m, generates the ciphertext c. That is: $c \leftarrow$SH.Enc(pk, m).
3) SH.Add(pk, c_a, c_a): Given two ciphertexts c_a and c_b, output c_{add} as an encryption of the summation of the underlying messages. That is: $c_{add} \leftarrow$SH.Add(pk, c_a, c_b).
4) SH.Mult(pk, c_a, c_b): Given two ciphertexts c_a and c_b, output c_{mul} as an encryption of the multiplication of the underlying messages. That is: $c_{mul} \leftarrow$SH.Mult(pk, c_a, c_b).
5) SH.Dec(sk, c): Takes as input the secret key sk and a ciphertext c and returns the message m. That is: $m \leftarrow$Dec(sk, c).

The number of multiplications carried out in the ciphertext space is bounded to a parameter $D \in \mathbb{N}$.

B. Public-Key Searchable Encryption

The Public Key Encryption with Keyword Search (PEKS) scheme, proposed by Boneh et al. [3], enables the search for a *keyword* on encrypted data without revealing any information (other than the result of the search $\in \{0, 1\}$).

There are three parties involved in the process: the sender, the receiver and the operator. The *sender* encrypts the intended message m using a standard public key system denoted by $\mathcal{E}$. He then appends to the resulting ciphertext a PEKS ciphertext of each keyword. To send a message m with keywords $w_1, \ldots, w_n$ he sends:

$$\mathcal{E}(pk_R, m), PEKS(pk_R, w_1), \ldots, PEKS(pk_R, w_n)$$

Where pk_R denotes the public key of the *receiver*. The receiver, using his secret key sk_R, can give a *trapdoor* $T_{w'}$ to the *operator*, which will enable him to test whether the keyword w' is contained on the ciphertext or not, but without revealing the keyword itself.

Definition 2. (Non-interactive Public-key Encryption with Keyword Search (PEKS) scheme). Boneh's PEKS scheme consists of the following polynomial time algorithms:

1) SE.KeyGen(s): Executed by the receiver. Takes a security parameter, s, and generates a public/private key pair pk_R, sk_R. That is: $(pk, sk) \leftarrow$ SH.KeyGen(s).
2) SE.PEKS(pk_R, w): Executed by the sender. For a public key pk_R and a keyword w, generates a PEKS ciphertext of w. That is: $S \leftarrow$ SH.PEKS(pk_R, w).
3) SE.Trapdoor(sk_R, w): Executed by the receiver. Given the secret key of the receiver, sk_R, and a keyword w produces a trapdoor T_w. That is: $T_w \leftarrow$ SE.Trapdoor(sk_R, w).
4) SE.Test(pk_R, S, T_w): Executed by the operator. Given the public-key of the receiver, a searchable encryption ciphertext $S =$ SH.PEKS(pk_R, w), and a trapdoor $T_{w'} =$ SH.Trapdoor(sk_R, w'), outputs "true" if $w = w'$ or "false" otherwise. That is: $t \leftarrow$ SH.Test(pk_R, S, T_w), with $t \in \{0, 1\}$.

Our framework requires the customer to share his SHE and PEKS secret-keys with the third-party auditor. We argue that in practice, it does not represent a vulnerability since: i) the amount of information revealed to the auditor is restricted by the "auditor queries" (see Section VI-A), ii) the auditor and the cloud provider belong to different administrative domains, and iii) the auditor is someone the customer trusts.

VI. Protocol Flow and Storage Structure

This section explains the process for storing and retrieving the evidence collected. It is organized as follows: First we provide flavors of the type of information the auditor can request to the ES, then we detail the process of storing the evidence in the ES and finally give examples of responses to queries.

A. Auditor Queries to ES Database

Auditor queries should be designed by considering both the audit goal and the respective security policy. By doing so, the auditor is *restricted* on the sensitive data he can retrieve from the ES. Furthermore, the auditor queries should provide a good balance between the amount of sensitive information revealed to the auditor and the amount required to come to reliable audit results. We provide examples for cloud audit use case 1:

Use Case 1: Auditing access to a controlled resource

- $Q_{1.1}$: Separately, the total of successful and non-successful connections from unauthorized IP addresses.
- $Q_{1.2}$: For every successful connection from an unauthorized IP address, retrieve the user who accessed the service, its IP address, date and duration of the connection.
- $Q_{1.3}$: Probability of success when trying to connect from valid and invalid IP address;

- $Q_{1.4}$: Total time a user spent connected (by month, day, etc.).
- $Q_{1.5}$: Average time per connection and its standard deviation.

The queries previously introduced are in compliance with the respective security policy by restricting the amount of sensitive information revealed to the auditor. However, until now we have not yet solved the problem of a curious EP learning such information or data leakage to third parties. Encrypting the sensitive information seems to be the solution for these problems, however the use of standard encryption mechanism (e.g. RSA, AES) would prevent the EP from responding to the auditor queries.

We propose the use of Boneh's PEKS and Brakerski et al.'s SHE schemes as crypto primitives to guarantee the secrecy of sensitive information while allowing the EP to operate on such ciphered information. The overall process for generating an audit report is as follows:

- The auditor queries the EP for a per-tenant report.
- The EP uses the evidence stored in the ES to generate the query response and sends it to the auditor. Recall that the EP does not learn relevant information, neither from the query response nor from the ES database.
- The auditor decrypts the report using his secret key.

In the following sections we detail the process of storing the evidence (encryption) and then the process of responding to a predefined query (computing on encrypted data).

B. Storing the Evidence

In Figure 2 we illustrate the general protocol flow for storing digital evidence in the ES. As said, we assume the existence of secured VPN channels between the AAs and the AAC. During an audit, the EP will make use of stored evidence to respond to the auditor queries (see chapter VI-A). The protocol flow is as follows:

1) An AA continuously sends digital evidence m to the AAC protected with VNP secured channels.
2) The AAC receives m, encrypts the attributes considered as sensitive and sends m' to the ES for storage.

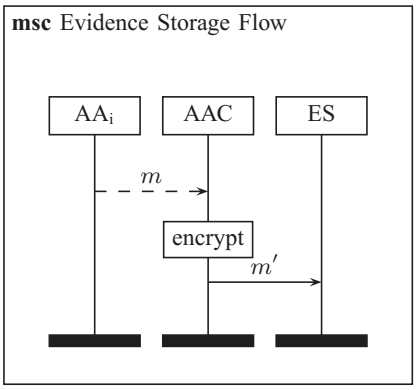

Figure 2. Protocol flow for storing evidence.

We highlight that the evidence is collected according to an audit goal and it should be stored in compliance with the defined security policy. More concretely, the evidence that shall be collected as well as the choice of the cryptographic primitives to protect the sensitive information, depend on the concrete cloud audit use case subject to audit which we describe at next:

Use Case1: Auditing access to a controlled resource
AAs are instructed to collect the data defined in Table I as evidence. Subsequently, an AA constructs m as a concatenation of such attributes together with a string to identify the use case and sends it to the AAC:

$$m :=< \text{"UC1"}, \text{Date}, \text{Observ}, \text{Init}, \text{IP}, \text{M}, \text{Out} > [3]$$

Attribute	Description
Date	The date when the evidence was collected
Observer	Id of the agent who collected the evidence
Initiator	Id of the user who tried to connect to the controlled resource
IP address	IP address from which the connection was attempted
Measurement (M)	The time-duration of the connection (e.g. in seconds).
Connection outcome	$out \in \{0, 1\}$, depending on whether the connection was successful or not

Table I
DATA TO BE COLLECTED FOR USE CASE 1.

The AAC receives m and validates its format. Then, by following the security policy, the attributes considered as *sensitive* are: "IP address" and "measurement" which are encrypted as follows:

- IP address: Encrypted using Boneh's PEKS scheme defined in Section V-B. First, the AAC encrypts the IP address using a standard public-key scheme $\mathcal{E}$ (e.g. RSA), then the AAC appends to the resulting ciphertext a PEKS ciphertext. Output IP' $\leftarrow \mathcal{E}(pk_c, IP)||\text{SE.PEKS}(pk_c, IP)$, where pk_c denotes the public-key of the customer.
- Measurement (M): Encrypted using Brakerski and Vaikuntanathan's scheme defined in Section V-A. This attribute represents the duration of every connection the user makes to the controlled resource (on-line time in seconds). Let val denote this measurement, where $val > 0$ for a successful login attempt and $val = 0$ otherwise. Output: M' $\leftarrow$ SH.Enc(pk_c, val), where pk_c denotes the public-key of the customer.

The AAC constructs m' and sends it to the ES for persistence as shown in Table II:

$$\text{m'} :=< \text{Date}, \text{Observ}, \text{Init}, \text{IP'}, \text{M'}, \text{Out}, \text{""} >$$

The last attribute "Type IP" is sent in blank by the AAC. It will be computed by the EP once the auditor provides the authorized IP addresses as *trapdoors*. The EP uses the

[3]In practice one could make use of headers to delimit each attribute.

Attribute	Description
Date	The date when the evidence was collected
Observer	Id of the agent who collected the evidence
Initiator	Id of the user who tried to connect to the controlled resource
IP_address (encrypted)	IP address from which the connection was attempted. Encrypted using Boneh's PEKS
Measurement M' (encrypted)	The time-duration of the connection (e.g. in seconds). encrypted using Brakerski's SHE.
Connection outcome	Stores $out \in \{0,1\}$, depending on whether the connection was successful or not
Type IP	Stores $t \in \{0,1\}$, depending on whether the IP is authorized or not.

Table II
TABLE USED TO STORE IN ES THE EVIDENCE FOR USE CASE 1.

trapdoors to test whether an IP address (which is in searchable encryption format) is *authorized* or *unauthorized* according to the respective security policy (whether a user is allowed to connect from such IP or not). This process is explained in Section VI-C.

As previously mentioned, we consider the EP follows the honest-but-curious model. By using the structure defined in Table II, we balance the amount of a customer's information available to the EP and the third party auditor, while still enabling the latter to perform the audit. The EP does not hold any decryption keys, such that *sensitive* attributes remain secret to it. Furthermore, since all sensitive attributes are encrypted using either searchable or somewhat homomorphic encryption schemes, the EP can still make meaningful computations on those ciphered data in order to respond to the auditor queries. The *predefined* auditor queries guarantee that the auditor will not have direct, random and unrestricted access to *sensitive* customer data but only to specific tasks, which reduce the risk of data misuse. For instance, the auditor can ask for those records in which a connection to a resource from an unauthorized IP address was successful and therefore violated a security policy. The result would only be revealing the unauthorized IP address, while still protecting recorded policy compliant access attempts.

C. Queries to ES

In this section we show how the EP responds to the predefined auditor queries. This requires the EP to generate SQL queries for the ES database and perform computations on encrypted data.

Use case 1: Auditing access to a controlled resource

In Figure 3 we illustrate the protocol flow for cloud audit use case 1, where:

1) as the first step, the EP is required to fill out the attribute "Type IP" in Table II, which classifies the respective IP address as either *authorized* or *unauthorized*.
2) once the underlying IP address has been classified, the auditor is enabled to query the EP.

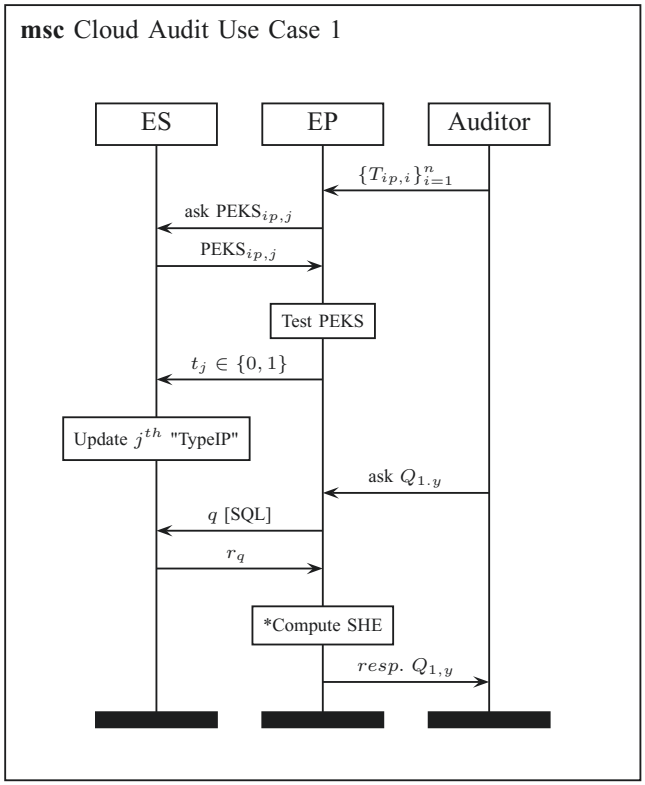

Figure 3. Protocol flow for querying the EP.

1. IP address Classification Process

This makes use of *trapdoors* generated by the auditor with searchable encryption (SE) algorithms described in Section V-B as we detail in the following:

We recall that the IP addresses are stored encrypted as:

$$\text{IP'} \leftarrow \mathcal{E}(pk_c, IP)||\text{SE.PEKS}(pk_c, IP)$$

Let Table II contain k records. Then for $1 \leq j \leq k$, the j^{th} IP address stored is denoted by:

$$\text{IP'}_j \leftarrow \mathcal{E}(pk_c, IP_j)||\text{SE.PEKS}(pk_c, IP_j) \quad (2)$$

1) Trapdoor Generation: Let $\{IP_i\}_{i=1}^{n}$ denote the set of authorized IP addresses according to the predefined security policy. The auditor generates a trapdoor $T_{ip,i}$ for every authorized IP address. That is, $T_{ip,i} \leftarrow$SE.Trapdoor(sk_c, IP_i), where $1 \leq i \leq n$ and sk_c denotes the secret key of the customer. The auditor sends $\{T_{ip,i}\}_{i=1}^{n}$ to the EP.
2) The EP asks the ES for the j^{th} searchable encrypted IP address.
3) The ES sends PEKS$_{ip,j}$:=SE.PEKS(pk_c, IP_j) to the ES, namely the right side of Equation 2.
4) Testing PEKS: Given a searchable-encrypted IP address $S :=$ PEKS$_{ip,j}$ and a set of trapdoors, the EP runs the SE.Test algorithm for every trapdoor $T_{ip,i}$ to determine if any of them correspond to the searchable-encrypted IP address or not. That is:

if $\exists\ T_{ip,i} \in \{T_{ip,i}\}_{i=1}^{n} \mid \text{SE.Test}(pk_c, S, T_{ip,i}) = 1$
then $t_j \leftarrow 1$
else $t_j \leftarrow 0$

5) The EP sends $t_j \in \{0,1\}$ to the ES, the latter updates the j^{th} record with "Type IP" $\leftarrow t_j$, i.e. whether the underlying IP address is unauthorized or authorized respectively.
6) Steps 2, 3, 4, 5 are repeated for every record in Table II, i.e. $1 \leq j \leq k$.

2. Processing Cloud Audit Queries

After the classification of the IP addresses is finished, the EP is able to respond to the auditor queries. Next, we give a precise construction for the cloud audit query $Q_{1.5}$.

1) The auditor asks the EP for $Q_{1.5}$, i.e. The average and standard deviation of the time a user spent connected.
2) The EP constructs q as in Listing 1.
3) The ES processes q and returns r_q to the EP. The response is $\{c_j\}_{j=1}^{k}$, the set of encrypted "Measurements".
4) The EP computes the *average* and *standard deviation* of the encrypted measurements as we explain below.
5) The EP sends the encrypted average μ' and standard deviation σ' to the auditor.
6) The auditor decrypts the result using the secret key of the customer.

```
select Measurement
from Table_II
where Type_IP = 1
        and Connection_Outcome = 1
        and Initiator = 'User1'
        [op] and 'Date'
        between 'date1' and 'date2'
```

Listing 1. SQL query for auditor query $Q_{1.5}$. On demand of the auditor, this query could be restricted to a specific date.

Audit use case query $Q_{1.5}$ requires the average μ and standard deviation σ to be computed. However, so far the research community does not have an algorithm to efficiently compute divisions or square roots of real numbers in the ciphertext space. Thus, when computing μ and σ, the numerator and denominator are returned as separate ciphertexts.

Computing the Average: Compute the average of the underlying measurements in $\{c_j\}_{j=1}^{k}$. Output $\mu' = (c_{sum} = \sum_{j=1}^{k} c_j, k)$, where c_{sum} is computed with Algorithm 1.

Algorithm 1 Sum of k ciphertexts using SHE

Input: $\{c_j\}_{j=1}^{k}$, with $k \geq 2$
Output: $c_{sum} = \sum_{j=1}^{k} c_j$

```
c_a ← c_1
for i = 2 to k do
    c_b ← c_i
    c_sum ← SH.Add(c_a, c_b)
    c_a ← c_sum
end for
```

The auditor receives μ', from which the mean can easily be computed with one division after decryption.

Computing the Standard Deviation: Compute the standard deviation of the underlying measurements in $\{c_j\}_{j=1}^{k}$. For the plaintext space, one could use the following equation:

$$\sigma = \sqrt{\frac{1}{k}\sum_{j=1}^{k}(c_j - \mu)^2} \tag{3}$$

However, in our setting the EP makes the computations on the ciphertext space which requires special treatment as we explain next.

1) μ is only known as the pair (c_{sum}, k), then we express Equation 3 as:

$$\sigma = \sqrt{\frac{1}{k^3}\sum_{j=1}^{k}(k \cdot c_j - c_{sum})^2} \tag{4}$$

Equation 4 requires to compute $k \cdot c_j$, namely the product of a ciphertext with a constant. This is not defined in Brakerski's SHE scheme, but the EP can encrypt k as $k' \leftarrow$ SH.Enc(pk,k) and then compute the multiplication of the two ciphertexts using SH.Mult.
2) The EP outputs $\sigma' = (c_{acc}, k^3)$ as the encrypted standard deviation, where c_{acc} is computed with Algorithm 2.

Algorithm 2 Standard Deviation of k ciphertexts using SHE

Input: $c_{sum}, k', \{c_j\}_{j=1}^{k}$, with $k \geq 2$
Output: $c_{acc} = \sum_{j=1}^{k}(k' \cdot c_j - c_{sum})^2$

```
for j = 1 to k do
    c_mul ← SH.Mult(k', c_j)
    c_res ← SH.Add(c_mul, −c_sum)
    c_sqr ← SH.Mult(c_res, c_res)
    if j = 1 then
        c_acc = c_sqr
    else
        c_acc = SH.Add(c_acc, c_sqr)
    end if
end for
```

3) Once the auditor has received σ', he computes the standard deviation σ as follow:
   a) $m_{acc} \leftarrow \text{SH.Dec}(sk_c, c_{acc})$
   b) $\sigma = \sqrt{\frac{1}{k^3} \cdot m_{acc}}$

VII. Performance and Security Analysis

In this section we provide an estimation of the performance of our framework as well as its security analysis.

A. Performance

At its current stage our work is purely conceptual, such that we are only able to provide an analytical performance evaluation of the cost of the computations required for a cloud auditing use case.

Use case 1: Auditing access to a controlled resource

To construct our scenario we consider the following parameters for a company X and a cloud provider Y:

- The audit is done once a year.
- X's IP address range: 142.168.1.0/24. Then, there are $g = 2^{32-24} = 256$ IP addresses.
- X has 100 employees. Each one connects to the service deployed in Y on average 3 times per day and there are

Algorithm	Run by	# Pairings	# Parings UC-1
SE.KeyGen	Customer	0	0
SE.PEKS	AAC	1	k
SE.Trapdoor	Auditor	1	r
SE.Test	EP	1	$k \cdot r$

Table III
NUMBER OR PAIRING OPERATIONS REQUIRED.

260 working days per year. Thus, Table II contains aprox. $k = 3 \cdot 100 \cdot 260 = 78 \cdot 10^3$ records at the end of the year.

At next we present the performance of the cryptographic schemes in the context of cloud audit use case 1.

Performance for PEKS: The construction of this scheme is based on a bilinear map $e : G_1 \times G_1 \rightarrow G_2$ (also known as *pairing*), which is the operation that requires most of the computational effort in the scheme. Thus, our analysis will be based on the number of pairing operations required.

We consider the Lynn's [2] benchmarking with the following parameters:

- PBC library [2].
- Super Singular Elliptic Curve $y^2 = x^3 + x$ of 512 bits.
- CPU: Intel Pentium III at 1 GHz.
- Evaluation of e: 11 ms.

In Table III we show the number of pairing operations required for one evaluation of the algorithm. The column "Parings UC-1" represents the number of pairings for the cloud audit use case 1.

The bottleneck is on the SH.Test algorithm which is executed by the EP when responding to an auditor query. In our setting, the complexity is $\mathcal{O}(k \cdot r)$. If we consider $k = 78 \cdot 10^3$, g= 256 and only one CPU, then the processing time is 61 hrs.

Performance for SHE: The performance of Brakerski's SHE scheme depends on the allowed number of multiplications to be carried out in the ciphertext. According to the cloud audit cases defined, we require to compute at most two multiplications (for calculating the standard deviation). We consider the results presented in [11] for our analysis, where:

- Implementation in the algebra system MAGMA.
- CPU: Intel Core 2 Duo at 2.1 GHz.
- Degree of the ring of polynomials: $n = 1024$.

Lauter et al. provide in [11] the running time of Brakerski SHE scheme for different configurations. We consider the setting which allows to compute up to two multiplications on the ciphertext space to construct Table IV. There we show the number of executions of each algorithm and the time estimation (in seconds) needed to compute the average and standard deviation for cloud audit query $Q_{1.5}$.

Algorithm	Run by	$Q_{1.5}$ -μ		$Q_{1.5}$ -σ	
		# Exe.	Time s	# Exe.	Time s
SH.KeyGen	Customer	1	0.26	1	$2.6 \cdot 10^{-1}$
SH.Enc	AAC	k	$2.7 \cdot 10^4$	k	$2.7 \cdot 10^4$
SH.Add	EP	k	78	$2k$	156
SH.Mult	EP	0	0	$2k$	$8.7 \cdot 10^3$
SH.Dec	Auditor	1	$1.8 \cdot 10^{-2}$	1	$1.8 \cdot 10^{-2}$

Table IV
PERFORMANCE FOR CLOUD AUDIT QUERY $Q_{1.5}$

From Table IV we can conclude that computing the average μ is quite efficient using Brakersky SHE scheme, since it only takes 78 s to compute the summation of $78 \cdot 10^3$ ciphertexts. It takes 148.2 min to compute its standard deviation.

According to the aforementioned numbers in Table III, IV, the major computational effort is done at the EP. Since it is part of the cloud, we argue there is a large number of CPUs which can process the cloud audit query in parallel. For example, testing if an encrypted IP is authorized or not (i.e., running the SE.Test algorithm) is a process which easily can be run in parallel. If we consider 100 CPUs, then this process is executed in 0.61 hrs. Additionally, we argue that responding an auditor query is not subject to real time constraints and that it is executed relatively rarely, e.g. only once a year.

B. Security Analysis

The proposed privacy framework makes use of SHE and PEKS as cryptographic primitives. These primitives coexist mutually exclusive in our framework due to the fact that there is no interaction between them neither explicitly nor implicitly since we do not apply the same key or encrypt the same plaintext with different crypto schemes. Consequently, we argue that the security of each scheme can be analyzed strictly separated. Both schemes are *provable secure*, i.e. the security can be proven using a reduction to a *hard problem*. At next we describe the complexity assumption for each scheme:

The security of Brakerski and Vaikuntanathan's SHE scheme relies on the RLWE problem which we define at next:

Ring Learning with Errors (R-LWE) Problem. It is a cryptographic complexity assumption similar to LWE problem [14] [15], however, it reduces the size of the keys to roughly linear, making the encryption schemes more efficient than those based on the LWE problem. This efficiency is possible by working over the ring $R_q := \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ (i.e. the ring of polynomials mod $\langle x^n + 1\rangle$ with coefficients in $\mathbb{Z}_q$), with the following restrictions:

- $q \equiv 1 \mod 2n$.
- $x^n + 1$ is irreducible over the rationals (achieved with n a power of two).

Definition 3. (Search R-LWE). Let R_q, q and n be as above. For fixed secret $\mathbf{s} \in R_q$ and some error distribution ψ over R_q, the search ring-LWE problem consists on recovering $\boldsymbol{s}$ given many Ring-LWE samples, i.e. samples of the form:

$$(\boldsymbol{a}, \boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e}) \in R_q \times R_q$$

where:

- $\boldsymbol{a} \xleftarrow{U} R_q$, $\boldsymbol{e} \leftarrow \psi$.
- $\boldsymbol{a}, \boldsymbol{e}$ are freshly generated for every sample.

Definition 4. (Decisional R-LWE Problem). Let $\boldsymbol{a}, \boldsymbol{s}, \boldsymbol{e}$ be as above. Given polynomially many samples of the form $(\mathbf{a}, \mathbf{b}) \in R_q \times R_q$, decide whether $\boldsymbol{b} = \boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e}$ or $\boldsymbol{b} \xleftarrow{U} R_q$. In resume, distinguish if the samples are *Ring-LWE vector* or *random*.

Lyubashevsky et al. proved in [12] the equivalence between the search and decisional variant of the R-LWE problem, by

giving a reduction from the search to the decisional case. Solving the R-LWE problem is known to give us a quantum algorithm for solving "short vector problems" on ideal lattices, which is believed to be exponentially hard.

Boneh's PEKS scheme is provable secure against chosen keyword attacks. Its security relies on the difficulty of solving the Bilinear Diffie-Hellman Problem (BDH).

Definition 5. (Bilinear Map). Let G_1 and G_2 be two groups of prime order p. A bilinear map from $G_1 \times G_1$ to G_2 is a function $e : G_1 \times G_1 \rightarrow G_2$ such that for all $u, v \in G_1$, $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$.

Definition 6. (Bilinear Diffie-Hellamn Problem). Let G be a cyclic group of order q and let g be a generator of G. Given $\{g, A = g^a, B = g^b, C = g^c\}$ where $a, b, c \in \mathbb{Z}_q$, compute $e(g, g)^{abc}$.

The success probability of any probabilistic, polynomial-time algorithm $\mathcal{A}$ in solving BDH in G is defined as:

$$Succ^{BDH}_{\mathcal{A},G} = Prob[\mathcal{A}(g, g^a, g^b, g^c) = e(g, g)^{abc}]$$

BDH assumption: For any probabilistic, polynomial-time algorithm $\mathcal{A}$, $Succ^{BDH}_{\mathcal{A},G}$ is negligible.

VIII. Conclusion and Future Work

In this paper, we proposed to apply crypto building blocks for a privacy enhanced cloud audit and showed that the use of Somehow Homomorphic Encryption (SHE) as well as Public-Key Searchable Encryption (PEKS) for evidence concealment support are relevant for cloud auditing use cases. We assumed the evidence store to act honest-but-curious and showed that with multiple addition operations and two multiplicative operations as provided by our chosen SHE, the evidence processor's computation on encrypted digital evidence data to a large extend can generate audit reports for relevant audit use cases. Moreover, with PEKS the auditor is enabled to perform queries on valid IP address ranges according to a given security policy. In our future work we will validate the integrity of the stored evidence. We believe HomMACs and homomorphic signatures can be applied to our framework to prevent malleability.

IX. Acknowledgement

The work presented in this paper was supported by the Federal Ministry of Education and Research (BMBF) within the project "Promotionsvorhaben zur Erarbeitung von Sicherheitserweiterungen für das Cloud Computing" (ProSeCCo). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ProSeCCo project or the BMBF.

References

[1] Shweta Agrawal and Dan Boneh. Homomorphic macs: Mac-based integrity for network coding. In *Proceedings of the 7th International Conference on Applied Cryptography and Network Security*, ACNS '09, pages 292–305, Berlin, Heidelberg, 2009. Springer-Verlag.

[2] Lynn Ben. *On the implementation of pairing-based cryptosystems*. PhD thesis, Stanford University, 2007. https://crypto.stanford.edu/pbc/thesis.pdf.

[3] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search, 2004.

[4] Dan Boneh and DavidMandell Freeman. Homomorphic signatures for polynomial functions. In KennethG. Paterson, editor, *Advances in Cryptology, EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 149–168. Springer Berlin Heidelberg, 2011.

[5] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Proceedings of the Second International Conference on Theory of Cryptography*, TCC'05, pages 325–341, Berlin, Heidelberg, 2005. Springer-Verlag.

[6] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. Cryptology ePrint Archive, Report 2011/344, 2011. http://eprint.iacr.org/.

[7] Cloud Audit Data Federation (CADF). Data format and interface definitions specification. Distributed Management Task Force, 2012. http://www.dmtf.org/standards/cadf.

[8] Martim Carbone, Matthew Conover, Bruce Montague, and Wenke Lee. Secure and robust monitoring of virtual machines through guest-assisted introspection. In Davide Balzarotti, SalvatoreJ. Stolfo, and Marco Cova, editors, *Research in Attacks, Intrusions, and Defenses*, volume 7462 of *Lecture Notes in Computer Science*, pages 22–41. Springer Berlin Heidelberg, 2012.

[9] Frank Doelitzscher, Thomas Ruebsamen, Tina Karbe, Martin Knahl, Christoph Reich, and Nathan Clarke. Sun behind Clouds - On Automatic Cloud Security Audits and a Cloud Audit Policy Language. In Tibor Gyires, editor, *International Journal on Advances in Networks and Services*, volume 6, 2013.

[10] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

[11] Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? Cryptology ePrint Archive, Report 2011/405, 2011. http://eprint.iacr.org/.

[12] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology, EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer Berlin Heidelberg, 2010.

[13] National Institute of Justice (U.S.). Electronic crime scene investigation: An on-the-scene reference for first responders. U.S. Dept. of Justice, Office of Justice Programs, National Institute of Justice, 2009.

[14] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, September 2009.

[15] Oded Regev. The learning with errors problem (invited survey). In *IEEE Conference on Computational Complexity*, pages 191–204, 2010.

[16] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In Leonid Hurwicz, David Schmeidler, and Hugo Sonnenschein, editors, *In Foundations of Secure Computations*, pages 169–177. Academic Press, 1978.

[17] T. Ruebsamen and C. Reich. Supporting cloud accountability by collecting evidence using audit agents. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 1, pages 185–190, Dec 2013.

[18] Bradley Schatz and Andrew J Clark. An open architecture for digital evidence integration. 2006.

[19] Bradley Schatz and Andrew J. Clark. An open architecture for digital evidence integration. In Andrew J. Clark, Mark McPherson, and George M. Mohay, editors, *AusCERT Asia Pacific Information Technology Security Conference : Refereed R&D Stream*, pages 15–29, Gold Coast, Queensland, May 2006. University of Queensland.

[20] Cong Wang, Qian Wang, Kui Ren, and Wenjing Lou. Privacy-preserving public auditing for data storage security in cloud computing. In *Proceedings of the 29th Conference on Information Communications*, INFOCOM'10, pages 525–533, Piscataway, NJ, USA, 2010. IEEE Press.

Agent-Based Evidence Collection in Cloud Computing

Philipp Ruf(✉), Thomas Rübsamen, and Christoph Reich

Cloud Research Lab, Furtwangen University of Applied Science, Furtwangen, Germany
{philipp.ruf,thomas.ruebsamen,christoph.reich}@hs-furtwangen.de

Abstract. Nowadays there are many offerings of cloud services all over the world which have various security requirements depending on their business use. The compliance of these cloud services with the predefined security policies should be proven. In a cloud infrastructure this is not an easy job, because of its immense complexity. This paper proposes an architecture which uses software agents as its core components to collect evidence across the different layers of cloud infrastructures (Cloud Managment System, Hypervisor, VM, etc.) and builds a chain of evidence to prove compliance with predefined security policies.

Keywords: Cloud computing · Evidence · Persistence · Accountability · Audit

1 Introduction

This work addresses the problem of collecting, processing and persisting evidence from different sources inside a highly dynamic environment and its automation dependent on a policy describing the contract between a `costumer entity` and its `Cloud Service Provider (CSP)`. The automation aspect is addressed by an `Software Agent` (in following `agent`) and is based on the previous work "Supporting Cloud Accountability by Collecting Evidence Using Audit Agents" [1] by Prof. Dr. Christoph Reich and M. Sc. Thomas Rübsamen. For example, a `CSP` is collecting information about policy violations and storing them using the service of a second CSP. The potentially confidential captured evidence need to be persisted in a integrity-verifiable way and protected from unauthorized access. This project's expected contributions are additional control mechanisms highlighting the transparency desired by the cloud service costumers and accentuating the trust in `CSP`s and their contractual awareness.

The proposed agent-based architecture, which we describe in the following, collects evidence to allow the detection of policy violations and generates policy violation reports while protecting sensitive information and respecting costumer privacy at the same time. Thereby using an agent framework supporting strong and weak agent migration [2] was necessary for distributing and delegating tasks on demand adjusted to their different destination environments. The data to

M. Felici and C. Fernández-Gago (Eds.): A4Cloud 2014, LNCS 8937, pp. 185–198, 2015.
DOI: 10.1007/978-3-319-17199-9_8

collect is depending on the assured policy contract between a cloud costumer and a cloud provider which, can be statutorily regulated, defined by the service provider or created out of user specific criteria. Through periodically audits, the implemented agents are able to provide the requested claims of evidence by persisting recognized policy violations.

While a Cloud Service itself potentially contains service interdependencies with external Service Providers, the sources of evidence to be covered by a trusted service are increasing, too. A `Chain of Accountability` can be formed implementing these centrally coordinated trusted (accounting) services along the supply chain. Pointing out the exact location of a occurred policy violation depicts how trust in a `CSP` is strengthened using services implementing evidence reveal- and notify mechanisms and therefore supporting accountability (e.g. members of the architecture to propose). In a multi-`CSP` scenario with service coherences inter-CSP collaboration is still a fundamental requirement. Currently, there is no standardized way for a cloud service costumer to check on his own whether or not he is affected by a policy violation occurred along the supply chain. The meanwhile established usage of `Web Objects` (like for example the `Amazon Simple Storage Service AS3`) extends the `Chain of Accountability` with dynamic interaction, which (usually) is transparent to the costumer. Potential evidence sources like these connections `on demand` and their potentially scalable content must be observed as of the time a active interaction with the service occurs. Therefore, the possibility of interacting with a `CSP`'s trusted service provides the transparency needed in a complex environment like the cloud.

This paper is structured as follows: in the first chapter of this paper we discuss related work (see Sect. 2). In Sect. 3, the evidence collection and persistence architecture including used technologies and agent coherences is described. Following that, the actual collection of evidence and the different collection agent types are explained in Sect. 4 followed by the persistence mechanism in Sect. 5. Following that, Sect. 6 describes the migration of agents in a scenario, where multiple cloud providers are involved. After that, an example of how service coherences are affecting policy violation evaluation, will be discussed in Sect. 7. We conclude this paper in Sect. 8 where our perception of further work is noted, too.

2 Related Work

Reich and Rübsamen empathized the need of policy violation audits and proposed how evidence collection has to be mapped to accountability in their previous work [1]. They are proposing an *Audit Agent System* which was the groundwork for the construction of the architecture presented in this paper. This work will not focus on the mentioned audit aspects but on the storing, presenting and processing of evidence.

The idea of using `Digital Evidence Bags (DEB)` [3] plays a key role as a solid evidence persistence structure in this paper. Based on the work by Turner, Schatz and Clark propose an extension for connecting evidence composing and

referencing DEBs using a Sealed Digital Evidence Bag [4]. This mechanism is a possible extension to this architectures persistence mechanism.

The usage of software agents also were proposed in the context of Security Audit as a Service (SAaaS) [5]. A related presentation layer and distributed sources of evidence are discussed in this paper but specializing on security policies and the guardedness of a source located at the input layer. Also, a *'security business flow'* modeler generating the policies to observe is differencing SAaaS from approach proposed in this paper.

Of course there are many tools offering a agent-based solution for monitoring network devices by collecting and analyzing a wide range of current system properties. Industry standards like Nagios [6] also providing an agentless monitoring solution which is a less capable but goes easy on resources. Also there are Software as a Service (SaaS) monitoring solutions like New Relic [7] which providing agents collecting data dependent on different scopes and devices. Besides the traditional monitoring of (network - e.g. cloud) resources it supports real-time analytics and a performance monitoring, which can be integrated into the application development process. Therefore, it is not surprising to be confronted with this system using a Platform as a Service (PaaS) like Cloud Control [8] or AppFog [9] simplifying the monitoring of scalable applications. Transmitting the current values to the cloud brings different advantages like rapid data analyzing using resources on demand. The architecture proposed in this paper also supports monitoring functions but differs by extending this aspect in focusing on active intervention like interacting with third party tools and their provided APIs.

3 Architecture Overview

The focus of this section is the brief introduction of this works architecture design including its components and used technologies. To understand the flow and properties of the proposed architecture, we have to take a closer look at the Java Agent DEvelopement framework (JADE) [10] which is the technological foundation of our work (Fig. 1).

JADE complies with the Foundation for Intelligent Physical Agents (FIPA) specifications [11,12], which define the internal behavior on action selection and execution as well as external agent interaction. The external agent interaction refers to the interaction context and the message creation, which draws on the FIPA ACL (Agent Communication Language). Other specified parts of JADE are system and platform services, which can be used for agent service registration or agent migration [13].

Using this powerful framework makes creating different agent infrastructures quite simple. Every JADE instance is denoted as a Container while multiple Containers are denoted as a Platform. Inside a Platform exactly one Main Container exists beside the different agent implementations, which itself contain agents for infrastructural provisioning. New agents, for example, can be added transparently as needed and contacted after their registration with a platform's Main Container. Using the recommended design guide [14],

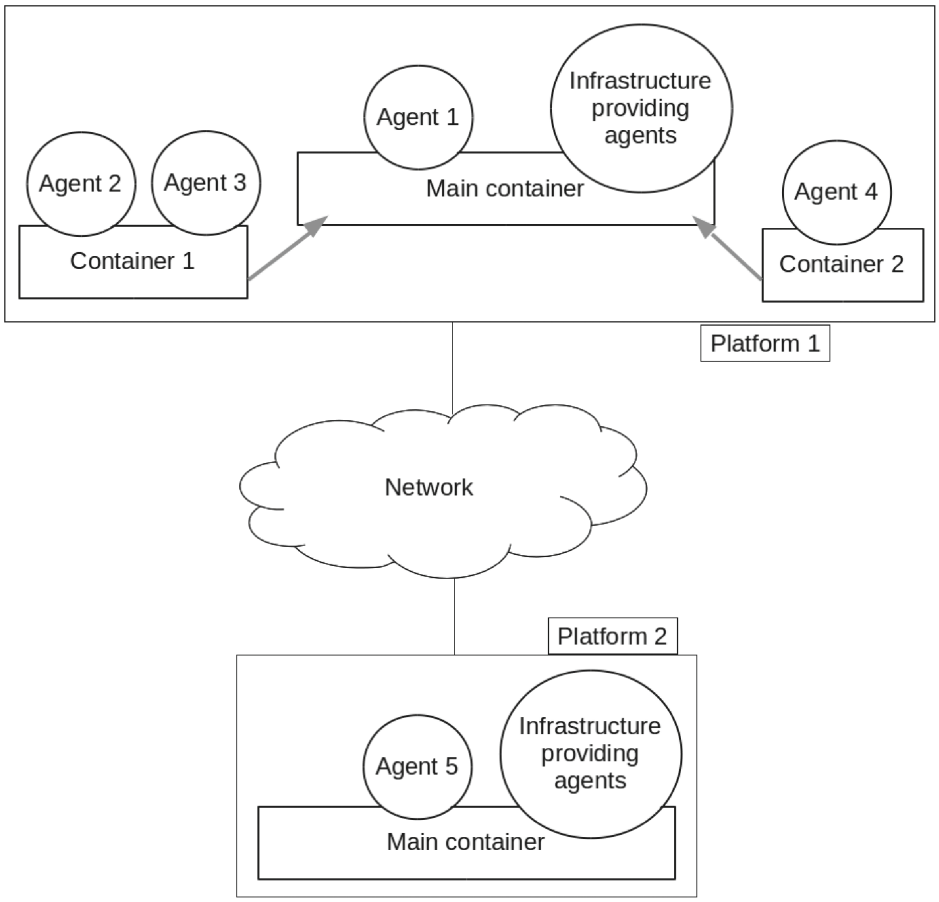

Fig. 1. JADE platforms and containers

the distributed agents are geared to each other and can be modularly extended. Also, the possibility of using `JADE` in `Public Key Infrastructure` mode is given by creating a configuration file containing the path to a `Key-Store`, key access password(s) and starting an agent with the configuration file as a parameter. Not only exchanged messages but also code transmission (used for agent migration - see Sect. 6) is encrypted if both participants are using the `JADE Public Key Infrastructure (PKI)` module/Add-On.

Being acquainted with the used agent framework lightens the contact with and comprehension of this architecture. A high level overview of the architecture is depicted in Fig. 2, revealing how distributed agents are communicating within it and in which way the different architectural components are interacting. All parts of this architecture are positioned inside one `JADE Platform` containing an evidence interpretation as well as a persistence agent which both interact with the distributed collection agents. Note, that additional services, which are provided out-of-the box by `JADE` (such as the centralized service registry and multi-platform interaction), are not pictured in Fig. 2.

To provide a chain of evidence, every trusted service of the supply chain contains a controlled agent which is responsible for evidence collection. To determine a policy violation, a variety of sources like the cloud management system, network packet data flows and storage units are browsed for conspicuous patterns. Also, external programs can be triggered to analyze their output.

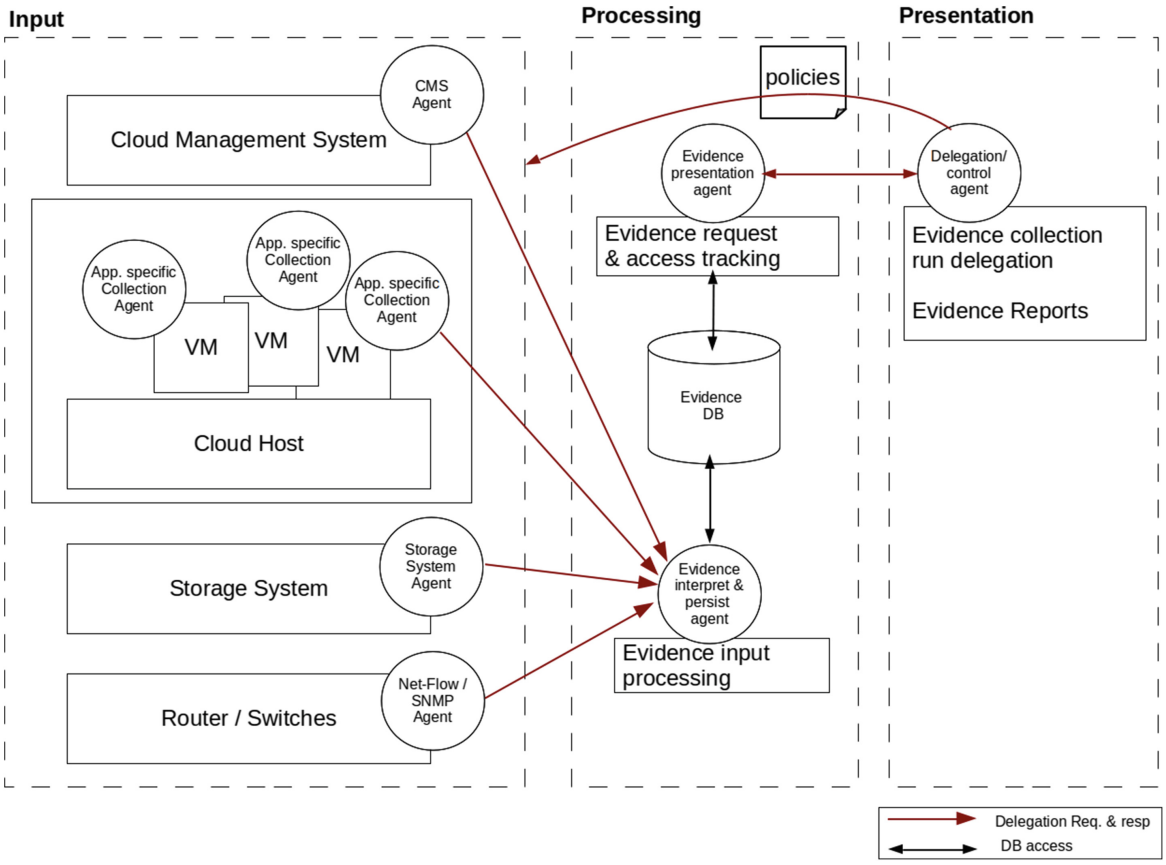

Fig. 2. High level architecture

To ensure data and therefore evidence integrity, every evidence collection run is persisted using a `Digital Evidence Bag` [3] which has been adopted for this approach (see Sect. 5). The DEB can contain raw data and diverse meta-data depending on the costumer contract or the kind of occurred policy violation, respectively. This architecture provides an interface delegating and coordinating tasks by interacting with the distributed `Evidence Collection Agents`.

The `Input Layer` contains different agents responsible for collection of evidence in their different scopes and locations. Currently covered evidence sources and their techniques are specified in Sect. 4. Each detected violation is noticed and the whole evidence including its meta- and additional control data is sent to the processing side. To guarantee information integrity, evidence is provided with a signature, which must be consistent during the evidence report process. This means, evidence integrity is ensured from the violation detection until the conversion from raw data to the output message has happened. At this point, a participating actor or an external system could be notified automatically. This automatic, transparency-strengthening process could therefore strengthen the trust in a `CSP` providing composite services [15].

The `Processing Layer` is responsible for the integrity observance and the evidence request access tracking which is expressed inside the `DEBs Tag Continuity Blocks (TCB)`, implemented as a H2 Database [16]. The `Evidence`

`Interpret And Persist Agent` is listening for successfully finalized or failed evidence collection records created by the Input Layer which triggers the persisting of new insights. Also, the `Evidence DB` is communicating with the `Evidence Presentation Agent`, which interacts directly with the `Delegation/Control Agent`. Its only purpose is the expressive presentation of requested evidence in due consideration of the requesting actor. It is also conceivable to provide a costumers exclusive evidence collection system. In this scenario, the compliance of policies (from CSP side) is possible using a separate `Cloud Service Provider Agent` with the required access rights, respectively the interaction with the costumers exclusive `Delegation Control Agent`.

The `Presentation Layer` is the only point of direct contact with a human actor (e.g., a customer or a trusted third-party auditor) inside this architecture. The user interaction handling and request transformation is handed over to the `Delegation/Control Agent`. Besides requesting meta data about currently active `Evidence Collection Agents`, a costumer entity is able to check the current status of the contractual compliance with its CSP. Also the explicit delegation of a evidence collection run is possible due communicating with this agent. Conveniently, the opportunity of adapting the JADE library to JSP based systems is given. Therefore, the orchestration of agent actions (which of course must follow agreed-upon policies) could be added for instance to a costumers private web interface.

Protective goals inside this forensic mechanism are integrity and confidentiality of collected data which have to be guaranteed until the evidence collection has finished and was persisted. On the other hand, collected data should only be requested by authorized auditors or other authorized entities such as cloud regulators.

4 Evidence Collection

This section is an introduction to the different evidence collecting agents, their evidence sources and the detection of policy violations. As shown in Fig. 2 the `Input Layer` potentially contains several distributed evidence sources located at Infrastructure as a Service (IaaS) -level. Besides the examination of log files [1] inside different systems, APIs and external applications are sources of evidence, which can be used to determine policy violations by different patterns. The evidence collection is performed either by using a so called `OneShotBehaviour` for a particular evidence collection initiation or by collecting evidence periodically/continuously. Currently the evidence sources are covered by the following `Evidence Collection Agents`:

- `CMS Agent:`
  This agent is able to interact directly with the central component of any cloud infrastructure. To detect policy violations, `APIs` provided by the `Cloud Management System` are used to gather needed information. In case of OpenNebula [17] the process of evidence collection could be the request for current storage, network or virtualization orchestration and of course the analysis of

log files, where events originating from cloud operations are recorded. For example, there is this internal project working on business secrets, which is placed on a separate hypervisor. Because of its critical data, this system would claim besides other transparency increasing measures the delegation of a `CMS Agent`. This `agent` would be responsible for gathering lifecycle information, tracking of occurred snapshots (which are relevant considering the aspect of needed confidentiality) and workload information of every (reachable) node.

- `Application Specific Agent:`
  These agent types collect a specific kind of evidence defined by the policy. For example, a policy could look like "It is not allowed to store email addresses inside a VM". To detect evidence of non-compliance, patterns matching email addresses need to be searched and recorded. Therefore, the agent is triggering an external program searching the VMs hard disk for the given pattern. This can be done using the `Cornell Spider` tool [18], which generates a log file containing all file paths that possibly compromise the given pattern including additional meta data. Of course the occurrence of false positives cannot be excluded automatically from the output.
- `Storage System Agent:`
  This agent is communicating directly with various `Storage Management APIs`. Besides performance monitoring this agent is also able to determine the exact location (e.g. datacenter) of a service. This feature can be used to verify the awareness of policies requiring a geographically aspect.
- `Net-Flow Agent:`
  This agent's task is the investigation of different network-enabled devices inside a CSPs network. Some policies will prohibit the network communication with certain addresses and/or address-ranges for a specific network device. By analyzing NetFlow logs, the policy violating communication can be tracked and used as evidence (e.g., communication endpoints, time and duration).
  In Sect. 6 there is a description of how to use this agent type along different CSPs in case of using their `XaaS` as supply chain.

After the evidence collection run has finished, the executing `Evidence Collection Agent` generating a evidence record as basis for the corresponding `DEB`. In some cases the evidence is a complete file which must be sent to the `Processing Layer` for purposes of conservation of evidence. Working large log files containing evidence can become a performance problem because each file containing a violation must be transmitted to the processing layer. In that case the evidence file must be transmitted inside a signed `Blob` and persisted at the processing layer (best encrypted, too) guaranteeing tamper evident properties.

5 Evidence Persistence

This section illustrates the different data structures (`Tables`) of the `Evidence Database` and how the different attributes are mapped to and reflect a `Digital`

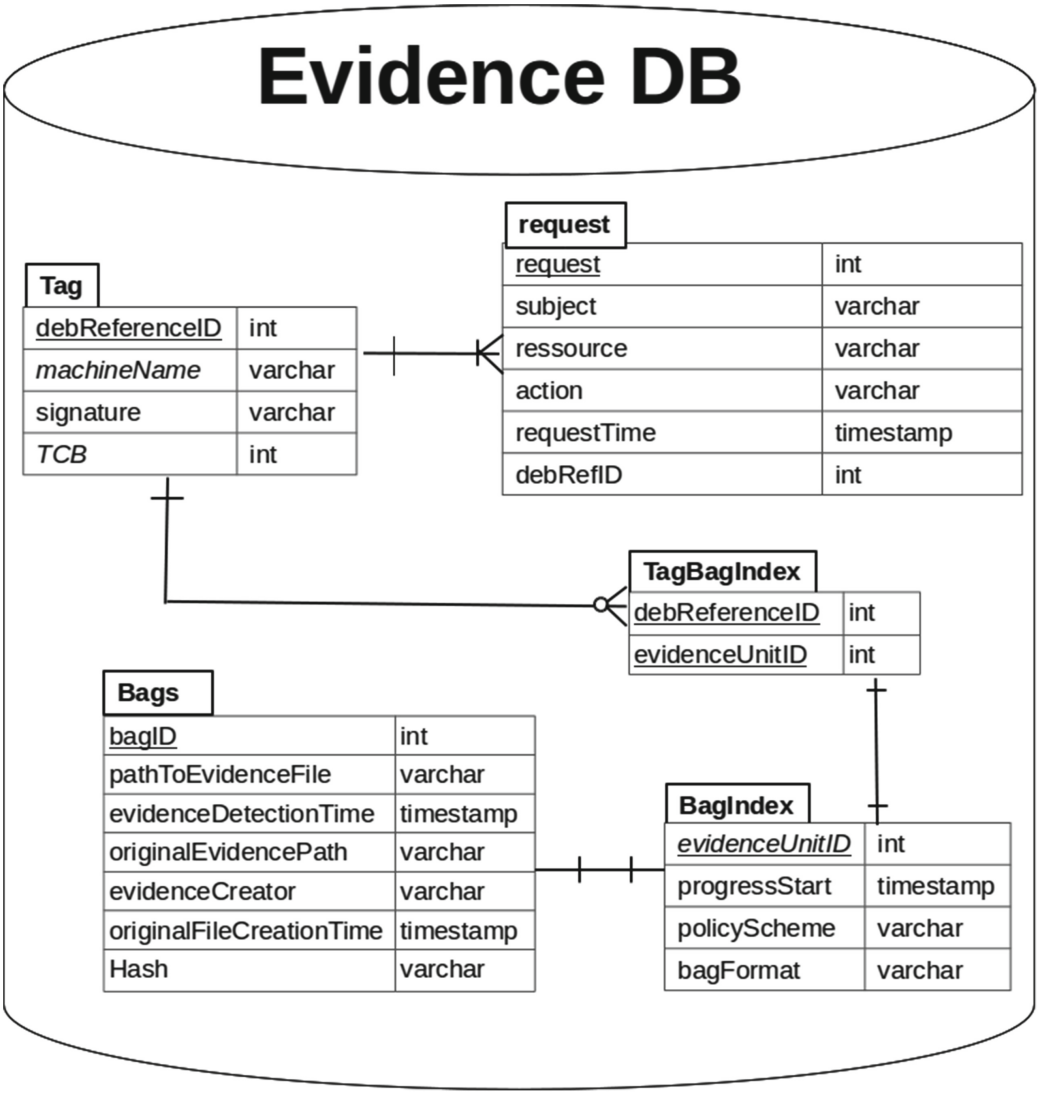

Fig. 3. Evidence database

`Evidence Bag (DEB)`. Like mentioned before, the persistence of evidence is triggered at the processing side as soon as the `Evidence Interpret and Persist Agent` is receiving a record from an `Evidence Collection Agent` registered with its platform. After verifying the dataset's integrity and adding the current `Request` record, every evidence request is processed by automatically forming the responding `Tag` row and its underlying layers. Therefore, the `Evidence Interpret and Persist Agent` is connected to a H2 `Evidence Database` (Fig. 3) using JDBC.

A `DEB` contains a `Tag` which has a `1:N` relation to an `Index`, which in turn is related `1:1` to a `Bag`.

– The `Tag Table` containing information about each evidence collection run in all because of its relation to the particular `Bag` and their overlying `Index Tables`. At this layer, meta data about the evidence record, information about the delegating agent and a reference to the last accessing subject of this evidence collection run are stored.

- To be aware of actions performed on a specific `Tag` or `Tag`-underlying layer the `TCB` entry references a row inside the `Request Table` to detect every single corresponding evidence access.
- The `1:N` relation between the `Tag` and the `Index` is established using an intermediate table.
- Also, containing a signature of every underlying relation altogether, the `Tag` table is a robust core element of this evidence persistence mechanism. Inside the evidence database the signatures are stored as `Base64 varchar`(String) and being restored as byte array for data integrity verification using the `Java.security API`.
- Besides the `Progress start` (at the `Evidence Collection Agents` side) and the `Policy scheme`, the `Index` table contains a `bagFormat` attribute to categorize the occurred evidence. As mentioned before every evidence collection run that scored no policy violation must be persisted, too. This can be done at this point. The `bagFormat` is a first indicator of the significance of persisted evidence. It can be *'structured text', 'raw binary data', 'archive', 'no policy violation'* or any other suitable categorization.
- By referencing the `Bag` table by its corresponding `evidenceUnitID`, a requesting subject is able to reach a stored policy violation. If the evidence is associated with a file, the file is persisted including its signature at the processing layers storage but is not stored inside the database because of performance loss inside the signing and verifying mechanism. However the path to this evidence file is stored inside this `Bag` table besides the `evidenceDetectionTime`, original evidence meta data and an additional Hash. Therefore, using the `JAVA Security API` every Tag is signed with the `Evidence Interpret and Persist Agent`'s DSA key, while the actual bag holds the `Evidence Collection Agent`'s signature, which was created during the collection process.

6 Agent Migration

This section focuses on a distributed evidence collection scenario by illustrating the possibilities of agent migration. Wasting system resources on not currently needed services (e.g. agents) can be avoided using the JADE API and/or the JADE GUI for manual agent orchestration [2]. Figure 4 depicts the scenario of a multi/inter CSP agent distribution on demand: According to a given policy, the `A-PPL Engine (Accountable Privacy Policy Language)` enforces accountability policies, like planned in the `A4Cloud Project` [19].

Also, the reaction on an occurred event is described (e.g., the notification of a subject about the analyzed evidence scoring a specific result). The `A-PPL` is currently work in progress. It extends `PPL` which extends the `eXtensible Access Control Markup Language (XACML)` [20] which is why this part is currently emulated inside this architecture using a `XACML Parser` deciding whether a function will be executed or not (e.g. the migration function).

Because of the `A-PPL Engine`'s SaaS aspects [21] the service probably will run inside a different ISP's virtual machine, which also possibly will be stored

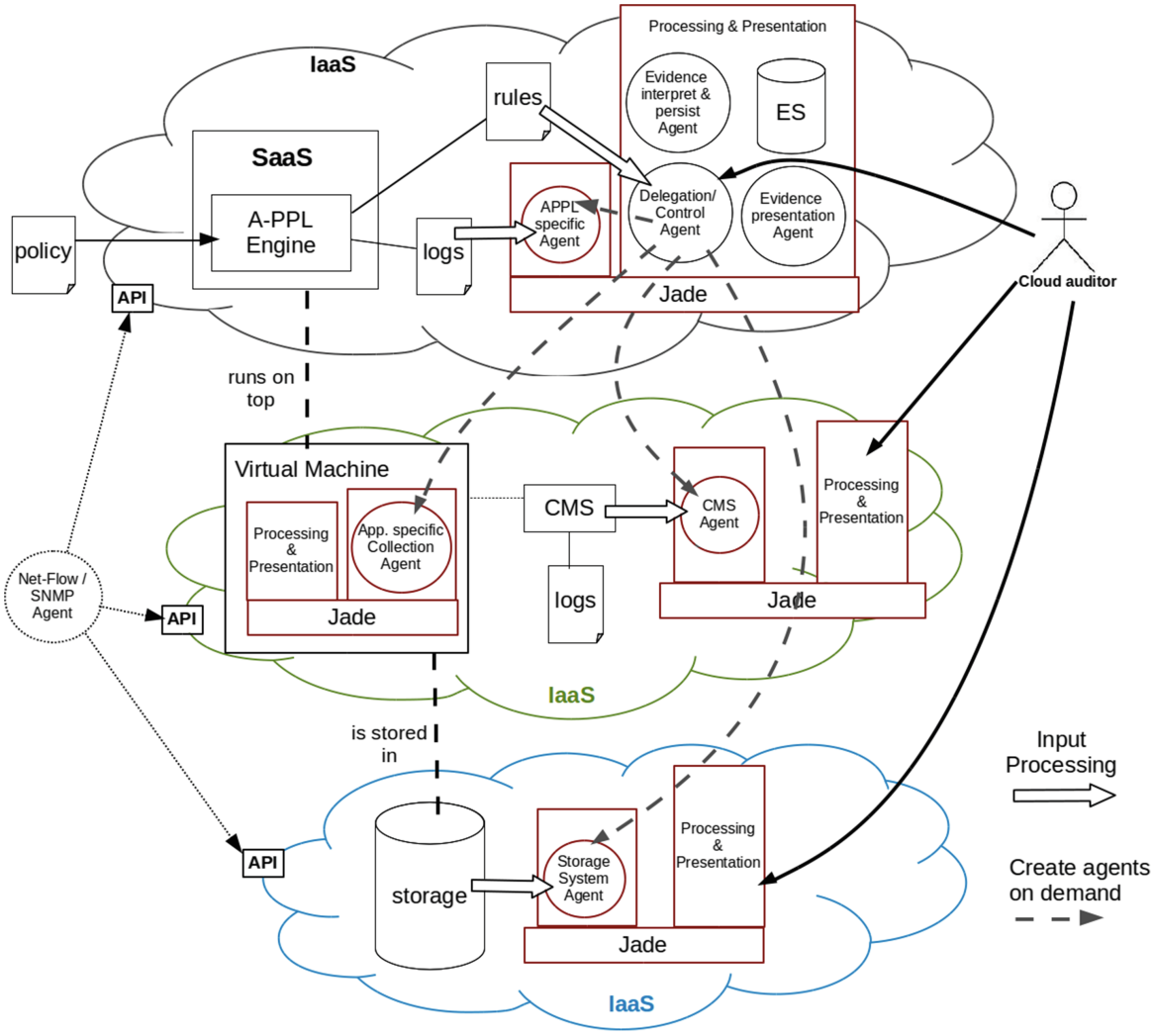

Fig. 4. Agent distribution on demand

inside a different ISP's Infrastructure. To provide a high availability the data is probably replicated to another data center, which possibly is located in another country and because of that following another juristic system [22]. Therefore, the actual avail of transparent data location in the cloud is to interpret as additional risk for a costumer.

Depending on the imported rules the `Delegation/Control Agent` distributes/ migrates the requested `Evidence Collection Agents` to their JADE destination platform. The different scarcely spawned `Evidence Collection Agents` initially performing a service registration after noticing their corresponding `Processing Layer` to communicate with. To avoid unnecessary overloading data traffic the evidence is always persisted inside their corresponding `Evidence Store` (weak migration).

The possibilities of network analytics are given anyway if the `Net-Flow Agent` is positioned ISP-local. To implement the `Net-Flow Agent` in a CSP-comprehensive way, every CSP must offer either a standardized API to check out the necessary connection stats or executing a (continuous) `Net-Flow Agent`

by their own to communicate with. Potentially this API will implement the Cloud Trust Protocol (CTP) [23]. To collect every available net-flow evidence the different CSPs must report their supporting services to the requesting system (e.g. this architecture) or respectively provide a way to check them out. While there is no standardized API the opportunity of using a Net-Flow Agent on every service providing CSP is given (see Sect. 7).

Also the possibility of temporary migration is given, where an agent will be transferred to another platform and migrated back to the ordering agents platform (strong migration). In this case, not only the agents executable is transmitted between the platforms but its currently objects containing new insights, too. Of course, distributed agents remove themselves from their corresponding platforms in case of a non-continuous evidence collection run. To collect evidence, the corresponding agent platform must be started with the required rights to access the resource (in most cases this is root).

7 Complex Service Provision Scenarios

A service provision chain scenario demands a correlation of evidence collected by all involved services and their platform's Evidence Collection Agents. The complexity of this service provision refers to the inclusion of multiple Service Providers and different companies, respectively.

Figure 5 depicts a scenario where CSP A is offering a (potentially public) Service A but using Service B provided by CSP B (transparent to costumers). In this example, a network communication with a country outside the EU takes place at the service provided by CSP B.

The agreed upon policy predicates that every processed data must been held inside the European Union but because of the supply chain CSP interoperability is needed, more precisely a trusted service is needed.

Both CSP A and CSP B are hosting services inside their own datacenters located inside the EU. All necessary evidence connections are provided using this evidence collection architecture trusting a central Locality Compliance Agent which is aware of all service relationships used by a potential costumer.

If CSP A respects the policy by hosting a service inside the EU but is in turn using a service provided by CSP B, CSP A alone is not able to guarantee the compliance to this policy.

Requesting evidence reports from the affected (distributed) platforms Delegation/Control Agent is how compounded evidence is come about. Depending on the data transmitted from Service A to Service B, potentially valuable information could be transmitted to a *'forbidden'* location, which is why the Evidence Collection Agent placed at CSP B diagnoses for network communication policy violations and creates an evidence record inside its platform's Evidence DB. Once the Locality Compliance Agent receives all necessary evidence reports from the participating CSP's Presentation Layers, the occurrence of interdependent policy violations are checked depending on the given EU data policy. Every new insight about interdependent policy violations will be

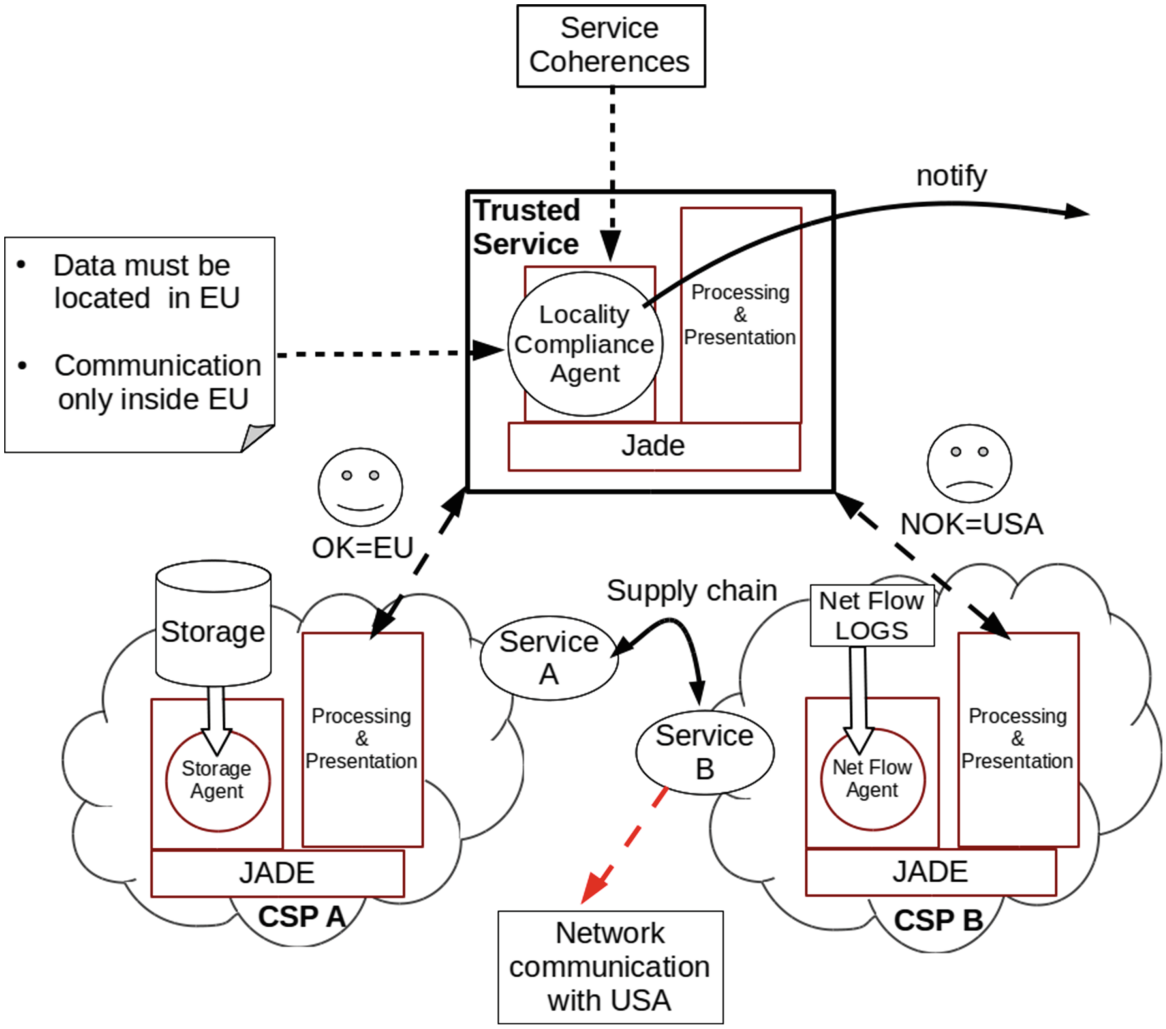

Fig. 5. Service coherences

forwarded to the `A-PPL Engine` which possibly will consider further steps (i.e., notifying stakeholders). Passing the new insights along the chain of accountability is relevant for a full conclusion but must go easy on network resources at the same time if this evidence architecture is applied in large datacenters. Therefore, for example a simple *'OK=EU'* or *'NOK=USA'* message on demand is sufficient for a pooled decision filling the dashboard of a customer's private web interface using the boolean product of all replies, group of replies respectively.

8 Conclusion and Future Work

This paper underlines the importance of structured evidence collection on demand running on nearly every device supporting transparency and therefore trust. The presented architecture enables distributed collecting and persisting of evidence following imported rules. Because of the possible distributed `JADE` platforms this construct will work in large `CSP` data centers without overloading the average performance.

To provide a quick evidence processing there must be a mechanism that excludes already known policy violations at `Evidence Collection Agent` side. The mentioned `Sealed Evidence Bag` [4] is a potentially evidence persistence extension. Also, the collected evidence must be presented in a convincing and distinct way at the user interface. Still existing challenges among other things are the performance evaluating of the defined architecture and scaling tests enabling the deployment of this architecture for highly dynamic services.

Acknowledgment. This research is closely related to the `A4Cloud Project`.

References

1. Reich, P.D.C., Rübsamen, M.S.T.: Supporting cloud accountability by collecting evidence using audit agents. In: 2013 IEEE International Conference on Cloud Computing Technology and Science (2013)
2. Bellifemine, F.L., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. Wiley Series in Agent Technology. Wiley, Chichester (2007)
3. Turner, P.: Unification of Digital Evidence from Disparate Sources (Digital Evidence Bags). QinetiQ
4. Schatz, B., Clark, A.: An open architecture for digital evidence integration. In: AusCERT Asia Pacific Information Technology Security Conference
5. Validating Cloud Infrastructure Changes by Cloud Audits
6. Nagios. www.nagios.com
7. New Relic. http://www.newrelic.com/
8. Cloud Control. www.cloudcontrol.com
9. App Fog. www.appfog.com
10. Italia, T.: Java Agent DEvelopement framework. http://jade.tilab.com
11. Poslad, S.: Specifying protocols for multi-agent systems interaction. ACM Trans. Auton. Adap. Syst. (TAAS) **2**(4), 1–24 (2007)
12. Foundation for Intelligent Physical Agents. http://www.fipa.org/
13. Reddy, P.I.P., Damodaram, D.A.: Implementation of Agent Based Dynamic Distributed Service
14. Nikraz, M., Caireb, G., Bahri, P.A.: A Methodology for the Analysis and Design of Multi-agent Systems using JADE. Telecom Italia Lab
15. Jansen, W., Grance, T.: Guidelines on security and privacy in public cloud computing. National Institute of Standards and Technology, U.S. Department of Commerce (2011)
16. H2 Database Engine. http://www.h2database.com
17. Open Nebula. http://opennebula.org/
18. Tchamdjou, M.Y.D.E.: Agenten zur Erkennung von sensiblen Daten und deren Schutz. HFU, Technical report
19. Accountability for the Cloud. http://www.a4cloud.eu/
20. XACML - Extensible Access Control Markup Language. www.oasis-open.org/
21. Benghabrit, W., Grall, H., Royer, J.-C., Sellami, M., Azraoui, M., Elkhiyaoui, K., Önen, M., Santana De Oliveira, A., Bernsmed, K.: A cloud accountability policy representation framework. In: CLOSER - 4th International Conference on Cloud Computing and Services Science, Barcelone, Espagne (2014). http://hal.inria.fr/hal-00941872

22. Bradshaw, S., Cunningham, A., Luciano, L.D.C., Hon, W.K., Hörnle, J., Reed, C., Walden, I. In: Millard, C. (ed.) Cloud Computing Law. Oxford University Press, Oxford (2013)
23. Cloud Trust Protocol. https://cloudsecurityalliance.org/research/ctp/

Supporting Cloud Accountability by Collecting Evidence Using Audit Agents

Thomas Ruebsamen, Christoph Reich
Cloud Research Lab
Furtwangen University of Applied Science, Furtwangen, Germany
{Thomas.Ruebsamen, Christoph.Reich}@hs-furtwangen.de

***Abstract*—Today's cloud services process data and let it often unclear to customers, how and by whom data is collected, stored and processed. This hinders the adoption of cloud computing by businesses. One way to address this problem is to make clouds more accountable, which has to be provable by third parties through audits. In this paper we present a cloud-adopted evidence collection process, possible evidence sources and discuss privacy issues in the context of audits. We introduce an agent based architecture, which is able to perform audit processing and reporting continuously. Agents can be specialized to perform specific audit tasks (e.g., log data analysis) whenever necessary, to reduce complexity and the amount of collected evidence information. Finally, a multi-provider scenario is discussed, which shows the usefulness of this approach.**

***Keywords*—Cloud Computing, Accountability, Audit, Evidence**

I. Introduction

Issues of transparency and control arise, when data moves from being stored locally to being stored remotely in the cloud. It becomes important to provide evidence for handling of confidential data in the cloud by remote parties throughout the whole lifecycle, also including deletion. However, this evidence is often not provided.

Currently, there is a lack of transparency and accountability from the provider side as for service provisioning/deprovisioning, tenant isolation, data processing and movement, privacy protection as well as many other aspects, which used to be fully under the control and monitoring of the consumer. Even if key terms are being added into cloud contracts (Service Level Agreement), processes and techniques must be developed to continuously and automatically monitor and audit these terms and ensure adequate transparency. Also, cloud providers must be prepared to provide adequate evidence about security and privacy in cloud scenarios.

A system for evidence collection that gathers, integrates and processes information, including logs, policies and context, in a way that preserves privacy and confidentiality and supports audit is needed. However, such an evidence framework for cloud computing does not exist yet.

The main contribution of this paper is establishing the conceptual basis of a system, which uses evidence generated from various sources in cloud service provision to conduct audits. These audits shall strengthen accountability by providing transparency regarding how, by whom and where data is collected, stored and processed in the cloud.

This paper is organized as follows: In Section II we summarize existing related work. After that, we describe the relations between cloud audits, accountability and evidence in Section III. Following that, in Section IV we propose an adopted evidence collection process based on digital forensics and discuss several evidence collection sources specific to cloud environments, as well as the most important data privacy issues in this context. In Section V we propose an audit agent system for cloud audits and evaluate this concept in Section VI. We conclude this paper in Section VII.

II. Related Work

Schatz and Clark [1] from the Common Digital Evidence Storage Format Working Group (CDESF) propose an evidence framework. Their architecture focuses on digital evidence bags (DEB), a generalized method for collecting information about evidence and evidence metadata, while keeping evidence integrity.

Flaglien et al. [2] evaluated currently used storage and exchange formats for handling digital evidence against criteria identified in recent research literature. Formats intended for storing evidence from highly dynamic and complex systems are characterized by incorporating additional information, which can be processed by data mining tools.

Our work focuses on collecting evidence in the cloud. Using digital evidence bags may addresss interoperability and evidence integrity, while additional metadata can be used to facilitate evidence processing.

An advanced approach for using the hypervisor for providing evidence in digital forensics is *Virtual Machine Introspection (VMI)* [3]. VMI leverages the capabilities of the hypervisor to look "inside" the virtual machine during runtime and using information collected this way for intrusion detection (e.g., detecting malware on the introspected VM). Garfinkel et al. conclude that this method is suitable for investigating cloud infrastructures, as long as there is access to the hypervisor.

Deploying additional software for evidence collection and monitoring inside virtual machines can be a problem in public cloud scenarios, where the customer has full administrative control inside the virtual machine. To overcome this issue, an out-of-guest-approach can be followed, like Carbone et. al present in their work [4]. By moving the monitoring tool outside the focused VM and leveraging function-call injection techniques as well as virtual machine introspection (VMI), the monitoring tool can be protected from the customer, resulting in more reliable information for evidence collection.

Dunlap et al. propose ReVirt [5], a logging and replay system for analyzing intrusions, that runs integrated with a

VM and performs the logging in the host OS. After an attack, it can replay the whole VM process for analysis.

The approaches proposed by Garfinkel, Carbone and Dunlap provide deep insight into what happens in a virtual machine. However, depending on the service model (i.e., how much administrative privileges a customer has) additional data collection points need to be considered to provide a detailed view on what happens in the cloud.

Poisel et al. [6] discuss digital forensics investigations at the hypervisor level of virtualized environments and introduce the topic of evidence correlation within cloud computing infrastructures.

Lu et al. [7] propose to adopt the concept of provenance in cloud computing by enabling a data object to report who created it and modified its contents. Therefore, data provenance could provide digital evidence for post investigations.

CloudAudit [8] provides a common interface and namespace with the goal to automate audits. However, how this can be achieved, what tools can be used, and how cloud infrastructures can implement these interfaces, is out of the scope of CloudAudit. These open questions are addressed by our proposed system.

Wang et. al [9] state requirements for secure third party auditors: 1) the auditor should be able to efficiently audit the cloud data storage without demanding the local copy of data 2) the auditing process should bring in no new vulnerabilities towards user data privacy. Regarding audits, this addresses the problem of an untrustworthy auditor, which needs to be considered by our proposed system as well.

III. Accountability Audits and Evidence

In this section, we describe accountability as it is understood in the A4Cloud project, how this concept relates to cloud audits and the relevance of digital evidence. In general, the A4Cloud project focuses on the handling of personal and confidential data in the cloud. All tools, operational procedures and mechanisms involving such data have to be audited to ensure accountability and give evidence in case of remediation.

Accountability for an organization, as described in the A4Cloud FP7 research project [10], consists of accepting responsibility for the stewardship of personal and confidential data with which it is entrusted in a cloud environment, for processing, sharing, storing and otherwise using data.

Derived from this, *Accountability Cloud Audit* can be defined as: *"The independent examination of records and activities to establish controls, policies, operational procedures and mechanisms, and expected means of remediation and to recommend any indicated changes in controls, policy, or procedures."*

For evidence, we use the definition by the Scientific Working Groups on Digital Evidence and Imaging Technology (SWGDE/SWGIT), which states digital evidence to be *"Information of probative value that is stored or transmitted in binary form"* [11].

Figure 1 depicts the relationship between cloud audit, evidence collection, contractual requirements and accountability attributes. Evidence is collected from multiple sources across a cloud system (a discussion of potential sources of evidence can be found in Section IV-A). These data are processed in accordance to the audit goal and presented as audit reports. Therefore, collected evidence supports cloud audits. Evidence should also be mapped directly to goals stated in service level agreements (SLA) and accountability contracts as well as other policy requirements. The compliance with or violation of such requirements can then be verified by performing audits and evaluating the available evidence.

For example, to make data processing (especially the compliance with or violation of data policies by the cloud provider) in the cloud more transparent, captured data-lifecycle events can be matched against policies in audits and thereby show the customer, that his data are handled appropriately. By providing evidence for the appropriate handling of data, the cloud provider demonstrates good stewardship of personal and confidential data.

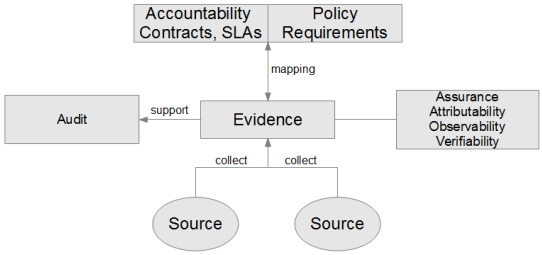

Fig. 1: Collecting Evidence and Mapping to Accountability

Accountability can be described by different attributes identified in the A4Cloud project. Several of these attributes are important in the context of evidence. *Assurance* for example, can be achieved by providing evidence for the compliance or violation of policies. *Attributability* is important to be able to attribute responsibility to actors. *Observability* describes how transparent a system is in terms of its operation and *Verifiability* captures the ability of an external actor to verify system operation against contractual agreements.

IV. Evidence Collection for Proving Accountability by Audits

Cloud forensics refers to digital forensics investigations performed in cloud computing environments. The process of a digital investigation can be separated into different phases as defined by the National Institute of Standards and Technology (NIST) in [12]. Each of these 3 phases has its own specific purpose in the chain of proving, that a cloud environment is accountable:

1) *Securing Phase*: The major intention is the preservation of evidence for analysis. The data have to be collected in a manner that maximizes its integrity. This is normally done by a bitwise copy of the original media. As can be imagined, this represents a huge problem in cloud computing where you never know exactly where your data are located and additionally do not have access to any physical hardware, which is solely under the control of the cloud service provider.

2) *Analyzing Phase*: In this phase, data from multiple systems or sources is pulled together to create an as complete picture and event reconstruction as possible. Especially in distributed system infrastructures, like clouds, this means that bits and pieces of data are pulled together for deciphering the real story of what happened and for providing a deeper look into the data. Because of the multi-tenancy of cloud infrastructures, the collected information at the cloud management system has to be filtered and then correlated with information collected at the customer's cloud resources.
3) *Presentation Phase*: The report, created in this phase, is a compilation of all the documentation and evidence from the analysis stage. The main intention of such a report is that it contains all results, it is complete and clear to understand. The timeline presented in the reports is also of high significance, because of the dynamic character of cloud environments.

With respect to accountability, requirements and other issues have to be stated in policies. According to these policies it can be decided which evidence data are important. One way to prove the compliance with these policies are cloud audits. Third-party cloud audits support the assurance of liability in case of data misuse, data breaches or data loss.

A. Sources of Evidence

The sources for evidence can be manifold, reaching from business process logging to operational logging. Operational logging could cover errors that concern a single cloud customer, critical conditions that impact all users, system related problems (e.g. failed resource access) and all activity that is executed by privileged accounts.

Data collected by logging systems is perhaps the most important type of data, from which evidence may be derived. In cloud environments, logging is performed on several architectural layers (see Fig. 2). For example, regarding the processing of data in the cloud, provenance information is of high importance. Logging where data originates, how, where and by whom it is processed, as well as where (in terms of geographical location) it is transferred is invaluable information when auditing against data processing policies.

As seen in Fig. 2 data can be collected at the network, hardware, host operating system, hypervisor, virtual machines and cloud management system (CMS). In the following we describe possible sources for evidence data collected across the different architectural layers:

Network: In a complex computing model, such as cloud, several stakeholders are involved. It should be possible to monitor networking resources which are utilized by a particular tenant. Networking resources can be either physical or virtual and can be shared among tenants. For instance in IaaS, a single network card in the host machine is utilized by several VMs and they may belong to different customers, which is typical in a multi-tenant cloud environment. For cloud audits, traffic flow information is important to reveal violations of security policies (e.g., compromised hosts, traffic flows to disallowed locations). Logging of traffic and communication endpoints is

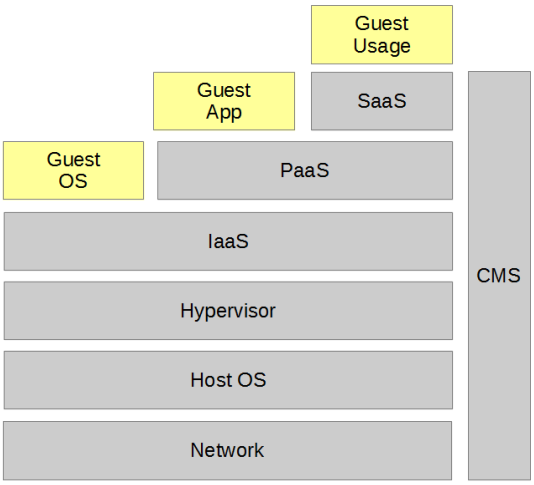

Fig. 2: Sources of Evidence

therefore an important source of evidence, especially in IaaS cloud deployments.

Host OS: Evidence collected on the host operating system level (i.e., servers hosting virtual machines) comprises load information, performance counters (e.g., network traffic counters) and various information collected by monitoring tools.

Hypervisor: The hypervisor is used to operate virtual machines. It has full control over virtual machines and assigns resources to them. It can provide runtime statistics, but also information derived using advanced techniques like Virtual Machine Introspection (VMI). Evidence collected from the hypervisor can be invaluable, since a VM can be observed from the outside.

IaaS: At this level, the most interesting evidence acquisition technique is VM snapshots. At any time, a copy of a VM can be created to preserve the actual state of a VM. Beside the VM, all cloud resources (e.g., storage, network, etc.) are important sources of evidence. It is noteworthy that there are still administrative domain problems. A cloud customer admin might alter information collected on this level and therefore make this evidence unreliable and useless.

PaaS: In a web service PaaS scenario evidence collection can be performed by the cloud provider as well as the cloud customer (i.e., platform developer). On the provider side, runtime environment logging (e.g., webserver, java runtime, database service, access control service) is of utmost importance. The major concern regarding information collection on this layer evolves around the segregation of multi-tenant log information at the provider-side. Additionally, the customer is free to implement its own evidence collection mechanisms (e.g., application logging). However, to what extent this information can be trusted is beyond the scope of this paper.

SaaS: On the SaaS level evidence may come from audit and logging APIs provided by the SaaS service provider. Such APIs may also include authentication and access records. However, the actual content of such data is highly dependent on the cloud provider and application type. Another source of evidence is

transient data stored on the client side. Evidence collection in such a scenario might need information from client side logging (e.g., javascript logging of operations) and other data stored in the browser's cache.

CMS: The Cloud Management System (CMS) is a huge source for evidence information. It is the central controlling component of a cloud infrastructure and provides information about user logins, cloud service usage, access rights, configuration, resource provisioning, policies, location etc.

Inter-Cloud: A service provision scenario with multiple cloud service providers demands a correlation of collected evidence from all involved providers. In such a scenario, inter-cloud collection of evidence is important.

B. Privacy Concerns

When collecting data on such a broad spectrum of different layers and components, privacy quickly becomes an important issue to consider. In the following, we want to raise awareness to the most important issues:

- *Protection of Evidence Source:* On every architectural layer of the cloud, data is generated, which may potentially serve as evidence during audits. The credibility and usefulness of audit results is tightly coupled with the quality of evidence sources. Therefore, the integrity of evidence data must be guaranteed. Data has to be protected from manipulation and data collected from evidence sources must be tamper-proof or at least tamper-evident. Without protection from tampering, the evidence collection system is not reliable and audits cannot be performed based on that data. Operational security mechanisms (such as tamper-proof logging) and organizational measures (restricting access to potential data sources and collected evidence to a minimal set of employees with authority over evidence collection) need to be put in place.
- *Data Retention:* Evidential data generated at the different layers may contain sensitive information (e.g., communication logs depicting communication partners, length and date). Such data must be handled carefully and deleted as soon as it is not needed anymore to ensure protection from misuse. However, continuous audits continuously evaluate a system. Therefore, a continuous stream of evidence may be needed, depending on the audit task.
- *Public Audit Interfaces for Third-Party Audits:* Public audit interfaces, which serve the purpose to strengthen transparency of cloud systems and ensuring proper usage of data in the cloud, come with several privacy issues. Cloud independent third-party auditors must be restricted in the amount of information, they can request from such interfaces. The amount of information should be just enough to perform the specific audits.

V. Accountability Audit Agent System

IT audit in general is defined as the process of collecting and assessing evidence to show that safeguards to protect against abuse and safeguards to maintain data integrity are in place, which will allow organizations to continue conducting business successfully. In this section, we will show how audits can support accountability. In particular, we will propose an automatic audit system based on software agents, which can be deployed across different architectural layers in cloud environments to collect and process evidence.

In previous work performed at the HFU Cloud Research Lab [13] we already presented our work on a cloud audit system, which enables customers of cloud services to perform audits of their cloud infrastructures (i.e., virtual machines in IaaS) using custom defined policies to check against. By describing policies in CAPL [14], providers and customers are enabled to define specific security policies and audit goals. These policies are checked using suitable tools (e.g., AV software, configuration parsers and analyzers). By rolling out software agents to the virtual machines, these tools are executed and relevant output and results is collected. By checking against thresholds and constraints defined in audit policies, audit results are generated and presented to the issuer of the audit process.

In the following, we conceptually extend this idea, to specifically support accountability. To support accountability, a more data-centric approach needs to be considered. While the assurance of compliance with security policies put in place by customers and providers is a very important issue, it does not cover problems relating to the correctness and policy compliance of data collection, transfer and processing in the cloud. To address these issues and providing transparency in these cases of data handling, using cloud audits, which focus on the virtual machine level, is not sufficient.

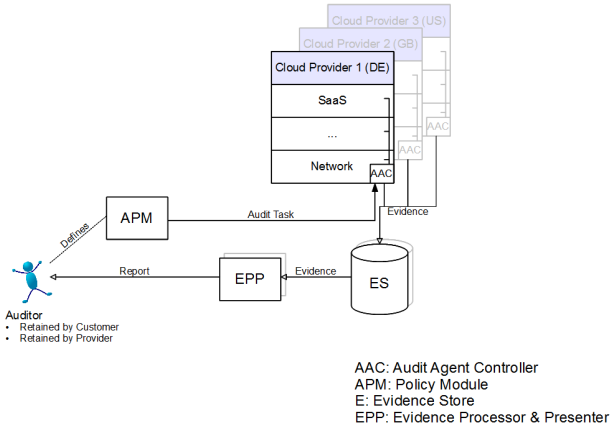

Fig. 3: Conceptual Overview of the Audit Agent System

Figure 3 depicts the conceptual overview of our proposed extension to the audit agent system to support accountability. Before the *Audit Agent System* is sketched the involved stakeholders are described:

- *Cloud Provider*: The cloud provider provides its customers with cloud services (e.g., IaaS, PaaS, SaaS...) with the need to consider multiple architectural layers (see IV-A for further discussion). Additionally, multiple cloud providers (depicted in the Figure as Cloud Provider 1, 2 and 3) from different regions (e.g., Germany, Great Britain and United States) can

be involved in the provision of a single service for the customer.

- *Auditor*: The auditor has technical know-how and sufficient resources to conduct audits on behalf of either the customer (external view) or provider (internal view). The goals and nature of policies which are audited, may differ, depending on the view. The view may also differ in case a trusted third-party auditor (TPA), who is independent from the customer and provider, but acting on behalf of any of those.

The *Audit Agent System* consists of the following components:

- *Audit Agent Controller (**AAC**)*: The AAC is a component, which can tap into several cloud subsystems (see IV-A) and deploy agents for evidence collection. A possible interface for auditors to this component could be CloudAudit [8] by the Cloud Security Alliance.
- *Evidence Store (**ES**)*: The ES is an isolated data store for each tenant, which stores evidence collected by the agents. Mechanisms to secure and protect the evidence are employed.
- *Audit Policy Module (**APM**)*): The audit policy module is used by the auditor to define audit policies. On the basis of these policies (describing the goal of what needs to be checked), a suitable audit tool is selected by the Auditor and configured properly. These preparation steps result in an *Audit Task*, which is processed by the AAC.
- *Evidence Processor & Presenter (**EPP**)*: The EPP is also a per-tenant component. By fetching evidence, verifying its integrity (e.g., verifying signatures and checksums) an evaluating against audit policies provided by the APM, a report with the evaluation result is generated.

Audits in general can be performed periodically or on-demand/continuously, when it is needed (e.g., change in the infrastructure). Our approach focuses on continuous auditing. The main problem of periodical audits in cloud computing is the dynamic change of the infrastructure and therefore, the risk of missing critical accountability issues if the interval is too big or if the interval is too short. Hence, a vast number of data from which evidence has to be derived may need to be analyzed. Simply storing data continuously without a specific audit goal in mind, just to have a large enough data-basis to perform an audit at a later point in time, does not necessarily improve chances of success nor is it a good idea in terms of data protection and privacy.

In our approach, audit policies are defined and evidence collection is optimized to only collect and store data relevant to the policy. By following this approach, we expect to reduce the amount of data stored in the ES.

Additionally, by providing the EPP and ES on a per-tenant basis, we try to reduce privacy risks compared to storing information related to all tenants in one single datastore. This form of isolation also enables a simplified deletion of the ES in case of service termination as well as backing up the ES for archival purposes.

VI. Evaluation

In this section we evaluate our proposed concept for an accountability audit agent system using a multi cloud provider scenario.

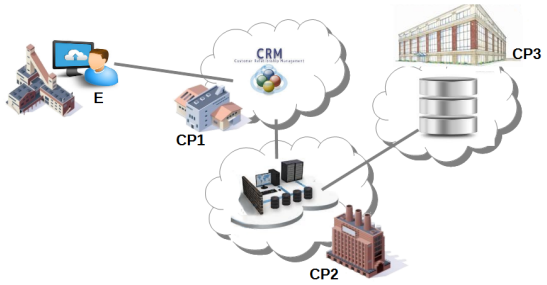

Fig. 4: Multi Cloud Provider Scenario

A. Scenario

An end-user (employee of company *E*) is using a Customer Relationship Management (CRM) software (see Figure 4). The CRM software is provided by Cloud Provider 1 (*CP1*) as a SaaS service. *E* interacts with the CRM application using a web-interface. *CP1* does not possess computing resources itself, but uses third-party providers. Cloud Provider 2 (*CP2*) is specialized in IaaS services and hosts the CRM software of *CP1*. Cloud Provider 3 (*CP3*) provides database as a service (DBaaS), which is connected to the CRM software. Both, *CP2* and *CP3* are constantly being evaluated by *CP1* regarding performance and cost.

B. Accountability

The end-user handles important business-related information in the CRM software on a regular basis (e.g., customer informations like names, addresses and history of business transactions). The company of *E* is therefore concerned about the security of this data, because they have no direct control over the infrastructure the CRM is hosted on. Therefore, they have negotiated the following accountability related contractual clauses with *CP1*:

1) All service providers involved in the service provision chain need to be ISO27001 certified to guarantee a baseline of security controls.
2) The data must only be stored inside the European Union (EU) regardless of *CP1*'s performance and cost evaluations of sub-providers.

C. Audit

CP1, *CP2* and *CP3* implement our proposed accountability audit agent system. *E* therefore wants to put an audit policy, for checking data location and communication end-points involved in the service provision, in place, which shall be continuously checked against. *E* can delegate this task to an external (e.g., information security professional, TPA) or internal (information technology department) auditor. The auditor describes the data locality task (i.e., which data collection tools to use, tool configuration and which sources of evidence are relevant) in

a machine-readable policy document. This policy maps the accountability related contractual clauses above and could look like this:
Policies based on contractual clause 1):

- If availability incident occurs, then check incident process and collect evidence.
- If Denial of Service (DoS) occurs, then check reporting and collect evidence.

Policies based on contractual clause 2):

- If log data of CRM (SaaS) indicates communication with entity outside the European Union, then collect evidence.
- If VM is migrated to another location, then collect evidence.
- If VM provisioning is located outside EU infrastructure, then collect evidence.
- If database write operations are outside EU IP domain, then collect evidence.

The policy document is sent to the AACs at *CP1-3*. The audit agent starts a tool for log parsing on the CRM instance. It specifically looks for communication events that happened over a specified period of time (in the audit task). By matching communication end-points and records these events (using a tamper-proof mechanism) and puts the so created evidence in the ES. At *CP2* the audit agent installs a network probe, which taps into network traffic of the virtual machines and records communication end-points. This information is also collected in the ES. At *CP3* the audit agent (similarly to *CP1*) parses database access and transaction logs and stores events in the ES. The EPP continuously analyzes the evidence put into the ES in defined intervals and creates reports stating whether or not a violation of communication and storage restrictions has been detected.

The report is presented to the auditor, who then notifies *E* about the accordance of *CP1* to policies put in place. Depending on whether or not the auditor is internal or external to *E*, the level of detail of the report is adjusted to be more or less. *E*1 may as well request reports directly from the EPP, which results in a detailed report about the performed actions, registered events and detected incidents.

VII. Conclusion

In this paper we emphasized the need of accountability audits and showed the importance of giving evidence to improve the trust into cloud computing environments. After adapting the general evidence process to cloud computing, possible evidence sources were presented and privacy concerns have been discussed. An agent based architecture has been introduced, which is able to do the audit processing and reporting continuously and selectively. Specialized agents can perform specialized audit tasks whenever they are needed. This reduces the amount of collected evidence information and make the audit architecture adaptable to dynamically changeable cloud environments. Finally, a multi-provider scenario shows the usefulness of this concept.

In our future work, we will refine the proposed system while considering the integrity of our approach. Important issues, which need to be addressed are the privacy issues outlined in this paper as well as protecting our system against agent manipulation and injection of malicious agents, while minimizing performance impact imposed by continuous audits.

Acknowledgment

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no: 317550 (A4Cloud).

References

[1] B. Schatz and A. J. Clark, "An open architecture for digital evidence integration," in *AusCERT Asia Pacific Information Technology Security Conference : Refereed R&D Stream*, A. J. Clark, M. McPherson, and G. M. Mohay, Eds. Gold Coast, Queensland: University of Queensland, May 2006, pp. 15–29.

[2] A. Flaglien, A. Mallasvik, M. Mustorp, and A. Årnes, "Storage and exchange formats for digital evidence," *Digital Investigation*, vol. 8, no. 2, pp. 122–128, 2011.

[3] T. Garfinkel and M. Rosenblum, "A virtual machine introspection based architecture for intrusion detection," in *Proc. Network and Distributed Systems Security Symposium*, February 2003.

[4] M. Carbone, M. Conover, B. Montague, and W. Lee, "Secure and robust monitoring of virtual machines through guest-assisted introspection," in *RAID*, 2012, pp. 22–41.

[5] G. W. Dunlap, S. T. King, S. Cinar, M. A. Basrai, and P. M. Chen, "Revirt: Enabling intrusion analysis through virtual-machine logging and replay," in *In Proceedings of the 2002 Symposium on Operating Systems Design and Implementation (OSDI*, 2002, pp. 211–224.

[6] R. Poisel, E. Malzer, and S. Tjoa, "Evidence and cloud computing: The virtual machine introspection approach," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 4, no. 1, pp. 135–152, 3 2013.

[7] R. Lu, X. Lin, X. Liang, and X. S. Shen, "Secure provenance: the essential of bread and butter of data forensics in cloud computing," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS '10)*. New York, NY, USA: ACM, 2010, pp. 282–292.

[8] Cloud Security Alliance (CSA), "CloudAudit A6 Cloud Security Alliance," http://cloudaudit.org/.

[9] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proceedings of the 29th conference on Information communications*, ser. INFOCOM'10. Piscataway, NJ, USA: IEEE Press, 2010, pp. 525–533.

[10] "A4Cloud FP7 Project," http://www.a4cloud.eu/.

[11] Scientific Working Groups on Digital Evidence and ImagingTechnology, "SWGDE and SWGIT Digital & Multimedia Evidence Glossary," https://www.swgde.org/documents/Current%20Documents/2013-04-08%20SWGDE-SWGIT%20Glossary%20v2.7.

[12] National Institute of Justice (U.S.), *Electronic crime scene investigation: an on-the-scene reference for first responders*. U.S. Dept. of Justice, Office of Justice Programs, National Institute of Justice, 2009.

[13] F. Doelitzscher, C. Reich, M. Knahl, A. Passfall, and N. Clarke, "An Agent Based Business Aware Incident Detection System for Cloud Environments," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 1, no. 1, p. 9, 2012.

[14] F. Doelitzscher, T. Ruebsamen, T. Karbe, C. Reich, and N. Clarke, "Sun behind clouds - on automatic cloud security audits and a cloud audit policy language," *International Journal On Advances in Networks and Services*, vol. 6, no. 1 & 2, 2013.

Cloud Audits and Privacy Risks

Thomas Rübsamen and Christoph Reich

Cloud Research Lab,
Furtwangen University of Applied Science,
D-78120 Furtwangen, Germany
{thomas.ruebsamen,christoph.reich}@hs-furtwangen.de

Abstract. In cloud computing users are giving up control over resources such as storage. Lacking transparency of cloud services (e.g. data access and data lifecycle reports) is an important trust issue, that hinders a more wide-spread adoption of cloud computing. Giving the customer of cloud services more information about data usage, compliance test reports and accordance to best-practices make the cloud more transparent. Reporting about such verifications is the main objective of cloud audits and is performed by third party auditors (TPAs). However, public auditing by TPAs can introduce new privacy problems. In this paper, a survey of current cloud audit privacy problems is given and techniques are shown how they can be addressed. Also, requirements for a privacy-aware public audit system are discussed.

Keywords: Audit, Privacy, Cloud Computing, Transparency

1 Introduction

Cloud computing as a paradigm is becoming more important for today's information technology. The shift from using traditional in-house datacenters towards on-demand provision of resources for delivering services has provided significantly more flexibility. However, a major aspect of adopting the cloud is giving up control of resources such as servers and network infrastructure. This also implies giving up control of data, which is stored in the cloud as well as services running in the cloud. The benefits provided are: increased flexibility, on-demand scalability and sometimes cost advantages.

Large companies like Amazon, Google and Microsoft have recognized the potential business opportunities and are providing well-established cloud services. Depending on the cloud service model, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), or Software as a Service (SaaS) [1] cloud customers have different levels of control over their resource stack for providing and using their services. For instance, a customer does have more control of their resources in a IaaS compared to a SaaS resource. Nevertheless some parts of the cloud infrastructure always remain hidden from the customer. This leads to a lack of transparency of cloud services, which is in turn the most important reason for a lack of trust in cloud computing. Highly visible security and privacy incidents of

major cloud providers also make this problem worse, e.g., as reported by CNN: "Google fires engineer for privacy breach" [2].

The Cloud Security Alliance (CSA) launched a new initiative to encourage transparency of security practices within cloud providers, called Security, Trust & Assurance Registry (STAR) [3]. Providers are encouraged to provide self assessment reports using STAR.

Some cloud providers address the transparency problem by providing tools such as Amazon's dashboard [4], but to increase trust cloud providers need to be able to prove compliance with external and internal regulations. Cloud audits are one instrument to provide cloud customers with more insight on cloud service delivery. They increase trust in cloud providers by proving the provider's compliance to regulatory requirements (SAS70, HIPAA, PCI) and industry best-practices (ISO27001, BSI Grundschutz, ITIL) regarding the implementation of privacy controls. Cloud audits may be characterized from different perspectives. They can be performed internally by the cloud provider or externally by a trusted third-party auditor. Additionally, audits may also be performed by the customer (e.g., public audit), by providing him with the necessary interface to conduct them.

However, cloud audit capabilities themselves can be problematic in multi-tenant environments such as the cloud. By providing an auditor access to cloud infrastructures, the customer's data might be at risk. In case of an untrustworthy auditor (third-party or internal) sensitive information is potentially at risk. The Privacy Rights Clearinghouse published such an incident, where an untrustworthy auditor used his access privileges to sell customer's credit card information obtained from the Amsterdam Hospitality Group [5]. Such an incident may also happen in cloud environments. Information about other tenants might also be leaked, when a customer is given the capability to audit the provider's services himself. Additional protection mechanisms beyond contractual terms are needed in cloud audit scenarios. Another problem arises from making the cloud more transparent in terms of data locality and audits of policies, which require certain types of data to be stored at distinct locations. However, tracking of data in the cloud is not the focus of this paper.

In this paper, we focus on the privacy implications of providing publicly available audit capabilities to cloud customers. In Section 2 we define transparency, privacy and cloud audit concepts, followed by Section 3 where we provide a survey on privacy issues, which stem from public cloud audits as well as some ways to address these issues. After that, requirements for a privacy-aware public audit system are discussed in Section 4. We end this paper with a conclusion in Section 6.

2 Transparency, Privacy and Cloud Audit

The dilemma is to provide the highest levels of transparency while maintaining strong privacy and being auditable by a third party. In this section, we describe the concepts of transparency, privacy and cloud audit.

2.1 Transparency

Transparency in the sense of visibility is an important issue in cloud computing. According to the NIST definition of cloud computing one essential characteristic of cloud computing is measured service [1]. Resources are monitored, controlled and reported on continuously. This provides some degree of transparency for the user and provider. However, this information might not be enough for showing *what* happens in the cloud. Additionally, technical (e.g. cloud architecture, configurations etc.) and operational information (information about security processes, incidence management, etc.) needs to be provided to make cloud services transparent. Also, the different cloud service delivery scenarios (IaaS, PaaS, SaaS) need to be considered. In IaaS the cloud user usually needs less information from the cloud provider, due to having full control about virtual machines at the operating system level. In PaaS and SaaS however, additional information needs to be provided by the provider because of the administration shift away from the customer.

2.2 Privacy

Privacy can be looked at from different angles. From a consumer's perspective it comprises the protection and appropriate use (according to the customer's expectations) of the customer's personal data. From an organization's point of view, privacy includes the application of laws, policies, standards and processes by which personally identifiable information (PII) is managed [6]. These aspects have to be considered carefully in cloud computing scenarios, where PII is processed and transferred.

Confidentiality is a concept, which focuses on the protection of data from unauthorized access. Usually, cryptographic mechanisms are used to achieve confidentiality. Confidentiality can be linked to privacy in the sense of protecting PII by using encryption techniques. However, in current cloud computing services, data is usually not encrypted to enable processing of data by the cloud provider. Therefore, other ways have to be found to protect customer's privacy.

2.3 Cloud Audit

Common cloud computing scenarios usually include at least one cloud service provider and a customer in the service provision chain. The introduction of a third party auditor (TPA) offloads time-consuming and cost-intensive tasks, as well as required knowledge to conduct audits from the customer. The entity, which performs audits is called cloud auditor. Generally, audits are used to "verify conformance to standards through a review of objective evidence" [7]. In cloud computing this means cloud services are examined regarding their performance, privacy, and security controls, which shall assure the cloud customer, that appropriate measures are in place. To enable the TPA to fulfill his role, public auditability becomes a requirement. The cloud provider has to provide auditors

with interfaces, techniques and necessary documentation as well as any additionally needed information to conduct audits. The auditor can than leverage his expertise and conduct audits on behalf of the provider (for external audits) or the customer.

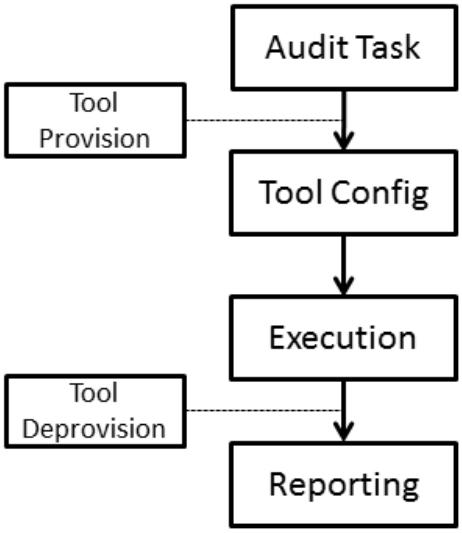

Fig. 1. Cloud Audit Process

Figure 1 depicts an automatic, tool-driven cloud audit process. Starting from a specific audit task, which describes a distinct objective, a tool is provisioned to perform the task (e.g., checking specific configuration files or parsing logs for certain events). The configuration of the tool specifies what has to be checked during execution. On the basis of the evidence collected by the tool, a report is generated and the tool is deprovisioned.

Auditors can be internal or external. External auditors are members of independent organizations specialized on performing audits. Internal auditors are usually contracted to the organization, whose entities are to be audited. They generally adhere to the same standards as external auditors with respect to performing an independent analysis.

3 Audit and Privacy

Cloud auditing needs to consider privacy of cloud customers. There can be several views on public cloud auditing scenarios, which will be shown in the following:

– Disclosure of customer's data
  Cloud customers store information in the cloud. Examples for this are cloud

storage services, which are used to store files, database as a service, which stores more structured data and data stored in SaaS services (e.g., customer relationship information).
- Disclosure of metadata about customer's data
  This kind of metadata includes data access and data usage logs, location data and data tracking information.
- Disclosure of metadata about customer
  This kind of metadata includes information about customer's usage profiles of cloud services.
- Disclosure of security relevant information
  Cloud customers may build virtual infrastructures. Auditing capabilities for such infrastructures may be provided by the cloud service provider as well as a third party. It is important to design them in a way, that no potentially harmful information is disclosed.

In this section, we provide an overview about recent research conducted in the area of privacy with respect to audits. We also give a short overview of how audits increase trust in cloud computing infrastructures.

3.1 Operational and Reliability Audit

By moving to the cloud, customer's (also owners of data) are placing control of their data into the hands of the cloud service provider. Two big arising issues are reliability and availability. Threats to these goals are for instance (silent) data corruption and (unintentional) deletion. Data corruption can have multiple causes like hardware faults in the networking or storage subsystems, or software bugs during processing of data. Deletion of data can be caused by careless provider personnel as well as software bugs. Risks evolving around these problems are for instance high costs caused by downtimes. On the provider side there is a risk of potential reputation loss. To mitigate these risks and provide the customer with an adequate level of assurance, that his data are safe with the cloud provider and stored correctly, audits can be used.

However, actually checking the correctness and retrievability of data is not a trivial task. Simply downloading data from the cloud and checking whether or not its integrity was harmed or parts are missing is not a feasible solution in cloud environments. Having to download terabytes of data and calculating checksums on that data would place a heavy burden on the customer or auditor, financially as well as in terms of resources needed.

Therefore, new ways of reliability auditing have to be found for cloud environments. There already exist several approaches, which aim at assuring data integrity and retrievability in the cloud while removing the need to download and check huge chunks of data. Most of the approaches focus on a third-party auditor (TPA) performing audits on cloud infrastructures.

Proof of data possession (PDP) [8] is a concept that uses homomorphic verifiability tags (HVTs) to prove that a remote party is in possession of a file.

This is done by generating metadata about the file prior to upload and verifying possession using a challenge-response protocol later on.

Proofs of retrievability (POR) [9] is a concept to enable an entity to verify the intactness of remotely stored data without requiring to retrieve large files. Thereby the server proves with a high probability to a data owner, that a file can be retrieved even in the case of some parts being corrupted. Most of the presented approaches leverage some kind of POR to enable reliability audits of cloud storage services [10].

In [11] Wang et al. propose a network architecture for secure data storage in the cloud. They acknowledge the previously described availability and privacy problems in current cloud storage systems. In their solution, a trusted TPA conducts audits on behalf of the data owner. They assume, that there is no incentive for the TPA to violate the privacy of the data owner. However, by proposing a solution, where the TPA is denied access to content, data leakage to the TPA shall be prohibited. Therefore, they propose the requirement, that a TPA must not know the contents of data, which is audited. Further goals of their approach are the support of dynamic data updating, batch auditing, and minimization of auditing overhead (e.g., network bandwidth). They propose cryptographic techniques like using homomorphic authenticators, and Merkle hash trees [12] to fulfil those requirements.

Boyang et al. take the concept of public auditability one step further and propose a system, where shared data stored in the cloud can be verified while preserving the identity privacy of each signer of a data block [13]. By using ring signatures to construct homomorphic authenticators the TPA is able to verify data, while not leaking identity information.

Furthermore there exist projects, that include TPAs and public auditability principles to enable secure and trustworthy cloud storage systems [14–17].

Performance auditing and the privacy issues introduced by it have been investigated to a much lesser degree. Large cloud provider such as Amazon, Google or Microsoft usually provide their customers with specialized interfaces (e.g., Amazon CloudWatch) for extracting performance monitoring information. Such information includes, but is not limited to, CPU utilization, network I/O statistics. The degree to which such information is published (e.g., on a per VM basis or details about the actual cloud infrastructure performance) is chosen by the cloud provider. Privacy problems may be linked strongly to the number of details published and the multi-tenant nature of cloud environments. Performance counters published to one customer might be used to deduce information about tenants using the same shared resources.

TPAs auditing cloud performance might need elevated privileges on the examined services (e.g., for measuring loading and saving times in a SaaS scenario). A TPA also has to assess the accounting system of cloud providers. This is usually tied very closely to the collection of usage logs and performance counters. To make a statement about the correctness of a provider's accounting processes, the TPA needs access to such information. However, without proper measures to protect the customer's privacy (e.g., proper level anonymization) a TPA can

easily extract cloud usage profiles (e.g., service interaction and communication) from this information. One thing to consider is the reversal of anonymization by combining multiple sources of anonymized data.

Operational and reliability audits are used to assess a providers procedures, systems, records and activities in order to test the adequacy of controls in place.

3.2 Regulatory Compliance Audit

Another form of audits is for regulatory compliance. Some businesses require cloud service providers to be compliant to or certified against certain regulations. Prime examples of such industries are healthcare, where sensitive medical information is processed, or finance, where sensitive financial data about individual subjects are processed. Typical examples are SAS70 [18] reports or HIPAA [19], which define how such data may be processed. Furthermore, there exist ISO27001 [20] which addresses information security management and CSA CloudAudit [21], which combines several of the previously mentioned audit frameworks to address cloud specific issues. However, these kinds of audits usually contain a lot of non-automatable audit tasks such as questionnaires or interviewing experts at the provider.

Reports generated by these frameworks are often not made available to the public. Amazon for example only releases their SAS70 report, when a customer contacts the Amazon support and requests it specifically. Regarding the privacy of other customers, these reports might reveal security flaws in the provider's processes, which could be exploited.

Regulatory compliance audits are used to assess a providers compliance in order to:

- Test the adequacy of controls in place
- Verify that a provider complies with established policy
- Verify that a provider complies with operational procedures (e.g., COBIT)

3.3 Security and Information Privacy Audit

Cloud security audits are supposed to uncover flaws and vulnerabilities in cloud infrastructures and service delivery chains, for instance: reveal unauthorized access to services and data, destruction of data and denial of service (DoS). The goal of this kind of auditing is to assure an appropriate level of protection especially by following industry best-practices.

However, information that is used during a security audit usually contains highly sensitive data, such as access logs and performed actions by customers. Special care has to be taken when providing such information in a public audit system.

Security audits are used to asses a provider's security issues in order to:

- Detect breaches or potential breaches of compliance
- Detect badly configured services

Information privacy audits are used to assess a providers compliance especially with respect to customer data managing in order to:

- Verify that a provider complies with established data policy
- Verify that a provider complies with privacy acts

4 Proposed System Requirements for a Public Cloud Audit System

In this section, we propose a privacy-respecting system for public cloud audits and address the most important requirements.

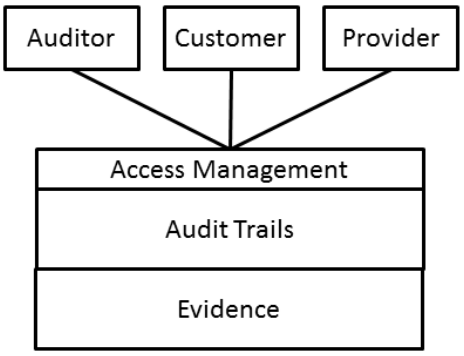

Fig. 2. Privacy-respecting Views on Cloud Audit Reports

Figure 2 depicts a high-level scenario for such a system. The foundation comprises of an evidence base. Evidence is any information needed to provide proof for specific audit tasks. This may include logs, configurations, data provenance information, timestamps, checksums and any other kind of data and metadata, which might be useful in constructing audit trails. Audit trails provide reliable proof about a certain audit task. However, as previously described audit trails might leak PII. Therefore, different views on audit reports must be provided to different stakeholders. This is done using the access management layer, which, depending on the actor requesting the audit report, provides reports with different levels of detail.

Sample Scenario: For example, consider the following scenario:

> A task for auditing the data life-cycle events of a customer's data in the cloud is requested. The sources of the relevant information are the cloud management system (CMS), which tracks high-level events such as provision/deprovision of cloud storage and the storage backend, which provides detailed information about data access, location and usage. A report is generated as a result depicting the audit result.

Mapped to the approach described, three different views on this report are provided by the audit system:

- Customer view:
  The customer view is the most detailed. It provides information about the CMS events, actions performed on the storage, such as who accessed what including timestamps, retrievability checks results.
- Auditor view:
  The auditor's view is less detailed. According to the checks defined in the audit request, the auditor is provided with more high-level results. Evidence accompanying the report is anonymized as needed.
- Provider view:
  The provider's view on the audit report contains mostly information stemming from the CMS.

Cloud Audit System Requirments: From this we derive requirements for the cloud audit system:

- Interfaces: public audit interfaces may differ among providers which complicates interoperability. In complex service provision chains, this hinders efficient auditing (see Section 5).
- Formats: differing data formats for the same information among providers also hinders interoperability. Also, tools used to extract information for specific parts of the audit trail usually use proprietary formats.
- Data collection: auditing requires the collection of data across all architectural levels. However, no more information than actually needed to provide proof for the audit task at hand shall be collected.
- Specific audit tasks: audit tasks shall focus on specific tasks. This shall enable reducing the amount of data needed to provide proof for the task.
- Dynamics: cloud environments are very dynamic. Therefore, a dynamic mechanism for collecting relevant evidence is needed. The audit system must support a mechanism to react address dynamic infrastructures.

5 Audit Challenges in Cloud Service Delivery Chains

Complex cloud service provision scenarios, where multiple service providers are chained for service composition, introduce new audit challenges. Typically, these chains are hidden from the customer. However, the customer's data are transferred along the service provision chain. For example a SaaS service provider may use an IaaS service provider's infrastructure to deliver its service. The IaaS service may be chosen depending on pricing and performance parameters and can also be exchanged without the customer's notice. This is a rather simple example for a cloud service delivery chain. More complex scenarios are thinkable, when multiple SaaS, PaaS and IaaS providers are involved.

Auditing a service and the subsequent services it depends on, introduces the following challenges:

– Audit of each involved service: each service involved in the service provision chain needs to audited.
– Audit of data transfer between services: the communication between services along the provision chain needs to be audited.
– Regulatory compliance: some parts of the provision chain may be located abroad, placing them under different jurisdictions.

6 Conclusion

In this paper we presented public cloud audit as a possible solution to increase trust in cloud computing by providing a proof and higher cloud transparency. We thereby focused on privacy concerns which arise in public cloud audit scenarios and addressed them in our proposed solution. By providing cloud stakeholders with different access views on cloud audit reports and therefore privacy of cloud customers can be protected.

In our future work, we will refine the high-level requirements of this system. Additionally, more complex service delivery chains, which involve multiple cloud service providers will be analyzed in detail to make public cloud audits practicable and service delivery chains auditable.

Acknowledgment

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no: 317550 (A4Cloud).

References

1. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. Technical report, National Institute of Standards and Technology, Information Technology Laboratory (2011)
2. CNN: Google fires engineer for privacy breach. `http://edition.cnn.com/2010/TECH/web/09/15/google.privacy.firing/` (Sep. 2010)
3. Cloud Security Alliance (CSA): Security, Trust & Assurance Registry (STAR). `https://cloudsecurityalliance.org/star/`
4. Amazon: Amazon's service health dashboard. http://status.aws.amazon.com/
5. Privacy Rights Clearinghouse: Amsterdam Hospitality Group. `https://www.privacyrights.org/data-breach-asc?title=amsterdam`
6. Pearson, S., Benameur, A.: Privacy, security and trust issues arising from cloud computing. In: Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on. (2010) 693–702
7. Badger, L., Bohn, R., Chu, S., Hogan, M., Liu, F., Kaufmann, V., Mao, J., Messina, J., Mills, K., Sokol, A., Tong, J., Whiteside, F., Leaf, D.: US Government Cloud Computing Technology Roadmap Volume II Release 1.0 (Draft) - Useful Information for Cloud Adopters. Technical report, National Institute of Standards and Technology, Information Technology Laboratory (2011)

8. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. Cryptology ePrint Archive, Report 2007/202 (2007) http://eprint.iacr.org/.
9. Bowers, K.D., Juels, A., Oprea, A.: Proofs of retrievability: theory and implementation. In: Proceedings of the 2009 ACM workshop on Cloud computing security. CCSW '09, New York, NY, USA, ACM (2009) 43–54
10. Juels, A., Kaliski, Jr., B.S.: Pors: proofs of retrievability for large files. In: Proceedings of the 14th ACM conference on Computer and communications security. CCS '07, New York, NY, USA, ACM (2007) 584–597
11. Wang, C., Ren, K., Lou, W., Li, J.: Toward publicly auditable secure cloud data storage services. Network, IEEE **24**(4) (2010) 19–24
12. Merkle, R.C.: Protocols for public key cryptosystems. In: IEEE Symposium on Security and Privacy. (1980) 122–134
13. Wang, B., Li, B., Li, H.: Oruta: Privacy-preserving public auditing for shared data in the cloud. In: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. (2012) 295–302
14. Li, L., Xu, L., Li, J., Zhang, C.: Study on the third-party audit in cloud storage service. In: Cloud and Service Computing (CSC), 2011 International Conference on. (2011) 220–227
15. Patel, H., Patel, D.: A review of approaches to achieve data storage correctness in cloud computing using trusted third party auditor. In: Cloud and Services Computing (ISCOS), 2012 International Symposium on. (2012) 84–87
16. Zhu, Y., Hu, H., Ahn, G.J., Yau, S.S.: Efficient audit service outsourcing for data integrity in clouds. J. Syst. Softw. **85**(5) (May 2012) 1083–1095
17. Shah, M.A., Swaminathan, R., Baker, M.: Privacy-preserving audit and extraction of digital contents, cryptology eprint archive, report 2008/186 (2008)
18. SAS70: SAS70. http://sas70.com/sas70_overview.html
19. U.S. Government Printing Office: HIPAA. http://www.gpo.gov/fdsys/pkg/PLAW-104publ191/html/PLAW-104publ191.htm
20. ISO: ISO27001:2005. http://www.iso.org/iso/catalogue_detail?csnumber=42103
21. Cloud Security Alliance (CSA): CloudAudit A6 Cloud Security Alliance. http://cloudaudit.org/

Evidence for Accountable Cloud Computing Services

Thomas Rübsamen[1], Christoph Reich[1], Aryan Taherimonfared[2], Tomasz Wlodarczyk[2], and Chunming Rong[2]

[1] Cloud Research Lab,
Furtwangen University of Applied Science,
D-78120 Furtwangen, Germany
{thomas.ruebsamen,christoph.reich}@hs-furtwangen.de
[2] Department of Electrical Engineering and Computer Science,
University of Stavanger,
N-4036 Stavanger, Norway
{aryan.taherimonfared,tomasz.w.wlodarczyk,chunming.rong}uis.no

Abstract. Evidence that allows assurance of accountability services, verification of compliance with the principles of accountability by service-providers and attribution of responsibility for breaches within the chain of accountability is essential. This paper defines how evidence may be required and proposes suitable ways of treating key accountability concepts. It shows the importance of verification and assurance, monitoring and auditing, and challanges of evidence in cloud computing. A discussion of logging and evicence gathering points complete the paper.

1 Introduction

Issues of transparency and control arise, when data moves from being stored locally to being stored remotely on the cloud. It becomes important to provision evidence for handling of confidential data in the Cloud by remote parties through whole lifecycle, also including deletion. However, this evidence is often not provided; transparency and verifiability are missing in the cloud context (especially at PaaS and IaaS levels). Moreover, there are additional related issues including cloud computing and globalization, increasing foreign government surveillance, the potential for light-touch self-regulation by the back door, weak certification for accountability, and weak links in terms of data protection along the service provision chain.

Currently, there is a lack of transparency and accountability from the provider side as for service provisioning/de-provisioning, tenant isolation, data processing and movement, privacy protection as well as many other aspects which used to be fully under the control and monitoring of the consumer. Even if key terms are being added into cloud contracts (Service Level Agreement), processes and techniques must be developed to continuously and automatically monitor and audit these terms and ensure adequate transparency. Cloud providers must be also prepared to provide adequate evidence about security and privacy provision.

A system for Evidence Collection that captures, integrates and processes the information including logs, policies and context in a way that preserves privacy and confidentiality and, supports audit and attribution is needed. An evidence framework for Cloud Computing does not exist yet. The main contribution of this paper is establishing necessary requirements for provisioning of evidence in a Cloud environment and how these requirements influence the tasks of monitoring and audit.

This paper is organized in the following way. In Section II we summarize existing related work. In this context, in Section III, we discuss general requirements necessary to provision evidence handling in a Cloud environment. In Section IV we discuss how these requirements influence the tasks of monitoring and audit. In Section V we summarize challenges of evidence provisioning in Cloud Computing. We conclude the paper in Section VI.

2 Related Work

One initiative towards evidence framework for Cloud Computing is an open architecture for digital evidence integration [1] by Schatz, B., and Clark, A. J. from the Common Digital Evidence Storage Format Working Group (CDESF). The architecture focused on digital evidence bags (DEB), a generalized method for collecting information about evidence and evidence metadata while keeping evidence integrity.

In Dykstra's paper [2] investigates how to obtain forensic evidence from cloud computing using the legal process by surveying the existing statues and recent cases applicable to cloud forensics. A sample search warrant is presented that could provide a sample language for agents and prosecutors who wish to obtain a warrant authorizing the search and seizure of data from cloud computing environments.

The paper from Haeberlen et al. [3], an accountable virtual machines (AVMs) has been introduced, which can execute binary software images in a virtualized copy of a computer system and can record non-repudiable information that allows auditors to subsequently check whether the software behaved as intended. Since this approach is basically VM logging and replaying, it is effectively the same as our full integrity checking, potentially with a lot of overhead.

In the paper of Poisel et al. [4] discuss digital forensics investigations at the hypervisor level of virtualized environments and introduce the topic of evidence correlation within cloud computing infrastructures.

The acquisition and analysis of digital evidence in cloud deployments is more complex, because data could be encrypted before being transferred to the cloud or it could be stored in different jurisdictions resulting in data being deleted before investigators have access to it [5].

Flaglien et al. [6] evaluated currently used storage and exchange formats for handling digital evidence against criteria identified in recent research literature. Formats intended for storing evidence from highly dynamic and complex sys-

tems are characterized by incorporating additional information, which can be processed by data mining tools.

Lu et al. [7] proposed to adopt the concept of provenance to the field of cloud computing by enabling a data object to report who created it and modified its contents, provenance could provide digital evidences for post investigations. Provenance information would have to be secured in cloud environments as leaking this information could breach information confidentiality and user privacy.

Marty's [8] approach utilize logging facilities to generate and collect relevant data to support the digital forensics investigation process.

The chain of custody documents how evidence was handled in the context of the digital investigations process [9]. The documentation describes how evidence was collected, analyzed, and preserved to be approved in court.

3 Accountability and Evidence

The A4Cloud FP7 research project [10] approach encompasses legal and regulatory mechanisms and a range of technological enhancements that can provide the necessary basis for trust. Customers, providers and regulators should be supported by preventive, detective, and corrective task (see [11]) and, for example, give cloud customers more control over their cloud services, ensure providers to meet their obligations, and enable cloud audits.

Technology can provide assistance in ensuring proper implementation of accountability. In particular, technology can be used to strengthen the enforcement and monitoring of policies and to help provide evidence, assurance and transparency. Hence, in accordance with Recommendation 5 from (Castelluccia et al, 2011 [12]), our approach is that privacy assessment, assurance, verification or enforcement should be evidence-based, and that these evidences might be derived from a number of sources, events and traces at different architectural layers.

The A4Cloud project identified a number of accountability attributes, like obligation, responsability, remediation, attributability, liability, sanctions, assurance, transparency, remediation, observability and responsiveness. These attributes have different importance from the perspective of a framework of evidence and identification of evidence types. We can divide these attributes into two general groups, those that reflect on accountability as a concept and those that reflect on how such concept should or could be implemented. Evidence of the following accountability attributes are of primary interest:

1. Attributability: Attributability describes a property of an observation that discloses or can be assigned to actions of a particular actor (or system element).
2. Observability: Observability is a property of an object, process or system that describes how well the internal actions of the system can be described by observing the external outputs of the system.
3. Assurance can take the form of evidence. An accountability system can produce evidence that can be used to convince a third party that a fault has

or has not occurred. In the context of accountability, assurance could refer to provision of ex ante evidence for compliance to governing rules, and possibly also to evidence that the governing rules and other factors provide appropriate grounds for trustworthiness .

4. Verifiability can be defined as the ability of an external party to observe a given aspect of a contractual relationship through the collected evidence. The quality or level of verifiability depends directly on the available evidence.

Remaining difficulties addressed by A4cloud is the development of mappings between the accountability contracts/SLAs and evidence available through logging. The framework should build an evidence base from which mappings of low level distributed remote IT logs can be mapped to high level policy requirements and service level agreements (SLAs). Evidence of accountability can therefore be provided and input to certification schemes or trustmarks. Figure 1 shows an overview of these relationships with log data being collected as evidence and evidence supporting auditing as well as assuring the previously mentioned accountability attributes addressed by the A4Cloud project.

Environments in which there are diverse and heterogeneous service providers, make provision of protocols and models for trust verification and assurance difficult. The CloudTrust Protocol [13] defines some evidence categories, but has not covered other categories such as legal liability of the involved parties.

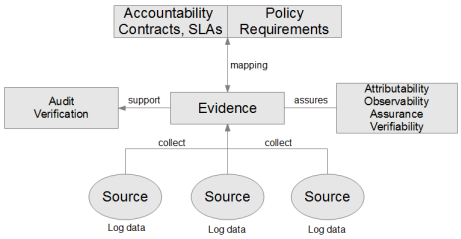

Fig. 1. Collecting Evidence and Mapping to Accountability

There are no efficient mechanisms available to gather convincing evidence from verified log data in distributed multi-tenancy environments, even if cloud providers would be willing to provide this. Although there are a number of existing logging approaches, they do not fit cloud computing very well. For example, EGEE LB log solution in grid computing is mostly used for debugging purposes only, as it keeps track of jobs. Even if verified log data is available, there are still challenges to make them compatible and interoperable. As different cloud providers implement and operate their systems differently, there is no guarantee that they all provide the same kinds of log information, which may expose weaknesses in their systems. There is currently no standard on log information to be delivered and there is no financial or regulatory incentive for the providers to provide such information. Furthermore, there is no accountability model for cloud, and therefore it is impossible to assign responsibilities even if the evidence exists. Neither are there any mechanisms for assigning responsibilities when the incident involves more than one provider based on gathered evidence in distributed systems.

4 Monitoring and Audit

Accountability mechanisms must be justified and Bennett [14] points out that a important process is independent testing of practices, provision of evidence that is taken into account, including auditing against the ISO 27001 series and associated cloud security standards. Evidence is provided by tools into trusted third party auditing processes against such standards.

ISO standards cover audit requirements at a high levelwhich is to maximize the effectiveness of and minimize interference to/from the information systems audit process. These solutions are not currently linked to formally defined accountability models, as accountability models only currently exist in terms of regulatory frameworks or point technical solutions. Accountability (for complying with measures that give effect to practices articulated in given guidelines) has been present in many core frameworks for privacy protection, like the Organization for Economic Cooperation and Development (OECD)'s privacy guidelines.

A4Cloud provide an approach based around a model of accountability that is interdisciplinary in approach, in which we build an evidence repository that provides evidence for preventive, detective and corrective accountability mechanisms by means of associated mechanisms for obtaining and negotiating obtaining these events from remote monitoring parties, and mechanisms for mapping the low level IT logs to what is in our repositories to policies and service level agreements (SLAs). In this way we bridge from distributed remote logs to high level policy requirements, and can detect policy violations. Audit capabilities in conjunction with external audit frameworks should be enhanced in order to strengthen the obligation for compliance and improve detection of violations.

5 Challenges of Evidence in Cloud Computing

Cloud forensics refers to digital forensics investigations performed in cloud computing environments. The process of a digital investigation can be separated into different phases as defined in the National Institute of Standards and Technology, "Electronic Crime Scene Investigation: An On-the-Scene Reference for First Responders" [15] each having its own specific purpose:

1. Securing Phase: The major intention is the preservation of evidence for analysis. The data has to be collected in a manner that maximizes its integrity. As can be imagined, this represents a huge problem in the field of cloud computing where you never know exactly where your data is and additionally do not have access to any physical hardware.
2. Analyzing Phase: Data from multiple systems or sources is pulled together to create as complete a picture and event reconstruction as possible.
3. Presentation Phase: Reporting all results in a clear and understandable way.

Current techniques in computer forensics can only analyze the evidence left behind by a careless intruder. We will use a combination of legal, technical and regulatory approaches to provide traceability, logging mechanisms and tools for

determining information provenance in distributed systems. This will underpin liability assignment and validation of insurance claims made in case of data breach or data loss. Evidence provided by our tools will enhance existing and developing certification schemes within the cloud.

With respect to the notion of evidence, it is important to differentiate between accountability and forensics. Digital forensics looks for unintended evidence, i.e. evidence that some party was not planning to leave and which collection was not planned ahead.

5.1 Sources of Evidence by Logging

The sources for logging can be manifold reaching from business relevant logging and operational logging. Operational logging could cover errors that concern a single cloud customer, critical conditions that impact all users, system related problems (e.g., failed resource access) and all activity that is executed by privileged accounts.

Sources of evidence to log, based on requirements and attributes, should be strengthened through the use of formal methods (e.g., formal logic). This is necessary to ensure the evidence quality in a situation where the amount of evidence-related data exceeds human reasoning capabilities.

Logging will need to be carried out at various stages of abstraction, i.e. at the system level, at the data level, at the service level, at the business level to determine when data is accessed, shared, moved, etc. The type of things that need to be logged at the data level are:

- *data creation:* the creation of a new data item, and the policies associated with this new item. The new item may be created by a user, or may arise from the automated copying or processing of data already in the system.
- *data access:* who accessed which data, for what purpose, the role of a person accessing the data, whether consent was obtained for usage from the data subject
- *data flow:* where the data is sent (including the jurisdiction), who shared data with whom
- *data type:* the type of data (e.g., is it personal, sensitive, etc.)
- *data deletion:* when was the data deleted, which erase method was used (unlink, delete data, delete backup, etc.)
- *data handling:* how data is handled to check conformance with some policies (e.g., data is stored password-protected or encrypted), data policy changes by the service provider, timing information (for example, for conformance to data retention policies)
- *data notification:* triggering and satisfaction of obligations

Subsequently, this information can be used in order to analyse whether organizational, regulatory and legal policies have been followed (this is a detective control, as opposed to checks made within the system associated with access control, etc. which are preventive). More specifically, we may want to focus on the following:

– segregation of duties; trans-border data flow; assurance that access control policies have been met
– assurance that obligations have been met
– records about how information was shared, with the context and associated obligations/sticky policies

5.2 Evidence Gathering Points

There are various locations to gather evidential data. As seen in Fig. 2 data (log data, memory, databases, etc.) can be collected at the network, hardware, host OS, hypervisor, the VMs, the CMS, the network and evidence data across other cloud platforms.

Network: In a complex computing model, such as Cloud, several stakeholders are involved. It should be possible to monitor networking resources which are utilized by a particular stakeholder. Networking resources can be either physical or virtual. Moreover, these resources can be shared among stakeholders. For instance in IaaS, a single network card in the host machine is utilized by several VMs and they may belong to different customers. Distinguishing between customer's traffic, which are hosted in a common set of substrates, is a key issue for accountability. This can also be applied to other service models of cloud, when traces of stakeholders' network activities must be available as an evidence type. However, existing networking devices and monitoring solutions are not compatible and efficient for such a multi-tenant environment.

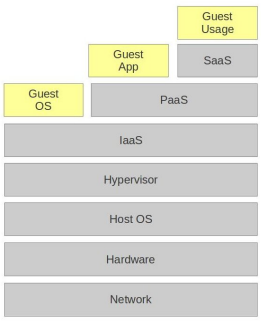

Fig. 2. Evidence Gathering Points

Hypervisor: The usage of data from hypervisors to prove various actual situations has been referred to as "virtual machine introspection" (VMI) and data gathered from this level of access supported the operation of Intrusion Detection Systems (IDS). It is suitable for investigating cloud infrastructures as long as there is access to the hypervisors.

VM: In order to obtain information from within VMs it could be helpful to install additional software inside the VMs. Carbone et al. [16] follow this approach by developing a secure and robust infrastructure called SYRINGE. The monitoring application is protected because it is put into a separate virtual machine as known from the out-of-guest approach. Nevertheless, it is possible to invoke guest functions by utilizing the function-call injection technique. The VM introspection make use of the guest OS knowledge of the deployed software architecture and can only be accessed with the customer's permission. A disadvantage arises from this component being susceptible to compromise from malicous entities.

CMS: The Cloud Management System (CMS) is a huge source for information gathering. It is the central controlling component of a cloud infrastructure and provides information about user logins, cloud service usage, access rights, configuration, resource provisioning, policies, etc.

IaaS: Except for traditional forensic acquisition at the virtual resources most interesting are VM snapshots which can accommodate preservation letters or serve as the acquisition image. Public clouds do not allow live forensics and access to volatile data. The storage is logical and focused on allocated space. Images can include data remnants or unallocated disk space. The logging may be co-located or spread across multiple and changing resources.

PaaS: In a web service PaaS the log data analysis can be carried out with the aforementioned methods, but relies on the cloud service provider. Multi-tanent log data must be separated or merged together from multiple resources.

SaaS: Access to application / authentication logs are possible to get and the SaaS application features may assist with network forensics. The logging information is located on the provider side and highly dependent of the application. The information may be inconsistent across API.

InterCloud: Cloud sources may be distributed over many providers and therefore collecting evidence over multiple sides is even more complex and difficult. There is a need of standardization of an evidence protocol, similar to the TrustCloud protocol.

6 Conclusion

The accountability approach taken in the EU FP7 A4Cloud project should help organisations meet their obligations and give cloud customers more control in cloud services. An evidence framework will be developed to assure accountability by building an evidence base gathering information. This information is collected at different level of the cloud stack and distributed in the infrastructure.

References

1. Schatz, B., Clark, A.J.: An open architecture for digital evidence integration. `http://eprints.qut.edu.au/21119/1/c21119.pdf`
2. Dykstra, J.: Seizing electronic evidence from cloud computing environments. In Ruan, K., ed.: Cybercrime and Cloud Forensics: Applications for Investigation Processes, Hershey, PA: Information Science (2013) 156–185
3. Haeberlen, A., Aditya, P., Rodrigues, R., Druschel, P.: Accountable virtual machines. In: Proceedings of the 9th USENIX conference on Operating systems design and implementation. OSDI'10, Berkeley, CA, USA, USENIX Association (2010) 1–16

4. Poisel, R., Malzer, E., Tjoa, S.: Evidence and cloud computing: The virtual machine introspection approach. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA) **4**(1) (3 2013) 135–152
5. George, E., Mason, S.: Digital evidence and cloud computing. Computer Law & Security Review **27** (September 2011) 524–528
6. Flaglien, A., Mallasvik, A., Mustorp, M., Årnes, A.: Storage and exchange formats for digital evidence. Digital Investigation **8**(2) (2011) 122–128
7. Lu, R., Lin, X., Liang, X., Shen, X.S.: Secure provenance: the essential of bread and butter of data forensics in cloud computing. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS '10), New York, NY, USA, ACM (2010) 282–292
8. Marty, R.: Cloud application logging for forensics. In: Proceedings of the 2011 ACM Symposium on Applied Computing. SAC '11, New York, NY, USA, ACM (2011) 178–184
9. Casey, E.: Digital Evidence and Computer Crime - Forensic Science, Computers and the Internet, 3rd Edition. Academic Press (2011)
10. Pearson, S., Tountopoulos, V., Catteddu, D., Sudholt, M., Molva, R., Reich, C., Fischer-Hubner, S., Millard, C., Lotz, V., Jaatun, M., Leenes, R., Rong, C., Lopez, J.: Accountability for cloud and other future internet services. In: Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on. (2012) 629–632
11. Pearson, S., Wainwright, N.: An interdisciplinary approach to accountability for future internet service provision. In Thampi, S.M., ed.: International Journal of Trust Management in Computing and Communications. Volume 1., INDERScience Publishers (2013) 52–72
12. Fischer-Hübner, S.: Transparency enhancing tools & hci for policy display and informed consent. In: Privacy, Accountability, Trust Challenges and Opportunities : ENISA Report. European Network and Information Security Agency, Technical Competence Department (2011)
13. Cloud Security Alliance (CSA): Cloud Trust Protocol. `https://cloudsecurityalliance.org/research/ctp`
14. Bennett, C.: 2. In: The Accountability Approach to Privacy and Data Protection: Assumptions and Caveats. Palgrave MacMillan (August 2012) 33–48
15. National Institute of Justice (U.S.): Electronic crime scene investigation: an on-the-scene reference for first responders. U.S. Dept. of Justice, Office of Justice Programs, National Institute of Justice (2009)
16. Carbone, M., Conover, M., Montague, B., Lee, W.: Secure and robust monitoring of virtual machines through guest-assisted introspection. In: RAID. (2012) 22–41