2021

On Trust Optimisation for Decentralised M2M Services

Shala, Besfort

UNIVERSITY OF
PLYMOUTH

ON TRUST OPTIMISATION FOR DECENTRALISED M2M SERVICES

by

BESFORT SHALA

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

DOCTOR OF PHILOSOPHY

School of Engineering, Computing and Mathematics

June 2021

Acknowledgements

First and foremost, I would like to thank my supervisors Prof. Bogdan Ghita, Prof. Armin Lehmann and Prof. Stavros Shiaeles, for their comprehensive support and guidance during this research work.

Second, I would like to express the deepest gratitude to my supervisor Prof. Ulrich Trick for his continuous support throughout the research work. He is an exceptional contributor to the success of this thesis. I truly thank him for his encouragement, constructive critique and perpetual availability throughout the past years.

Third, many thanks go to fellow researchers of the Research Group for Telecommunication Networks at the Frankfurt University of Applied Sciences. Thank you for your friendship, support and great inspiration during the past years. Warm thanks also go to the graduate school members and the CSCAN Network at Plymouth University.

Last, I am most grateful to my loving wife Iliriana for her positivity in times of hardship and all of the sacrifices she has made on my behalf, especially during the COVID-19 pandemic. Our children, Siera and Ben, have grown into strong personalities by the time the thesis evolved, and their love was the biggest motivation for the thesis. I also owe my deepest gratitude to my father Naim, my mother Lumnije and my brother Lum for their unwavering belief in me and their continuous encouragement and support.

Thank you, my family, for your spiritual support throughout this thesis and my life in general. You have given me the necessary energy to perform even on difficult days. This PhD thesis is dedicated to you.

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 Doctoral College Quality Sub-Committee.

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

Relevant scientific seminars and conferences were regularly attended at which work was often presented and several papers prepared for publication.

The following publications are the own work of Shala. The other authors provided guidance.

- Shala, B., Wacht, P., Trick, U., Lehmann, A. (2017), “Optimised Test and Security Solution for P2P-based M2M Applications”, ITG-Fachtagung Mobilkommunikation, pp. 105-110, Osnabrück, Germany.
- Shala, B., Wacht, P., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2017), "Framework for Automated Functional Testing of P2P-based M2M Applications", IEEE International Conference on Ubiquitous and Future Networks (ICUFN), pp. 916-921, Milan, Italy.
- Shala, B., Wacht, P., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2017), "Trust Integration for Security Optimisation in P2P-based M2M Applications", IEEE Trustcom/BigDataSE/ICESS, pp. 949-954, Sydney, NSW, Australia.
- Shala, B., Wacht, P., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2017), "Automated Functional Testing of P2P-based M2M Applications", IEEE Internet Technologies and Applications (ITA), pp. 29-34, Wrexham, UK.
- Shala, B., Wacht, P., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2017), "Ensuring Trustworthiness for P2P-based M2M Applications", IEEE Internet Technologies and Applications (ITA), pp. 58-63, Wrexham, UK.
- Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2018), "Trust-based Composition of M2M Application Services", IEEE International Conference on Ubiquitous and Future Networks (ICUFN), pp. 250-255, Prague, Czech Republic.
- Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2019), "Blockchain-based Trust Communities for Decentralized M2M Application Services", International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), Taichung, Taiwan. Lecture Notes on Data Engineering and Communications Technologies, Springer, vol. 24, pp. 62-73
- Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2019), "Distributed Ledger Technology for Trust Management Optimisation in M2M", Mobile Communication - Technologies and Applications, ITG-Symposium, pp. 1-6, Osnabrueck, Germany.
- Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2019), "Novel Trust Consensus Protocol and Blockchain-based Trust Evaluation System for M2M

Application Services", Internet of Things: Engineering Cyber Physical Human Systems, Elsevier Journal, vol. 7, 100058.

- Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2020), "Trusted, Decentralised and Blockchain-based M2M Application Service Provision". International Conference on Broad-Band Wireless Computing, Communication and Applications (BWCCA), Antwerp, Belgium. Lecture Notes in Networks and Systems, Springer, vol. 97, pp. 210-22.1
- Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2020), "Synergy of Trust, Blockchain and Smart Contracts for Optimisation of Decentralized IoT Service Platforms". International Conference on Advanced Information Networking and Applications (AINA), Caserta, Italy. Advances in Intelligent Systems and Computing, Springer, vol. 1151, pp. 547-558.
- Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2020), "Blockchain and Trust for Secure, End-User-Based and Decentralised IoT Service Provision", in IEEE Access, vol. 8, pp. 119961-119979.

Presentations and Conferences attended:

- Collaborative Research Symposium on Security, E-learning, Internet and Networking (SEIN 2016), Frankfurt, Germany, July 2016
- 22nd VDE ITG Mobilkomtagung, Osnabrück, Germany, May 2017
- 9th International Conference on Ubiquitous and Future Networks (ICUFN 2017), Milan, Italy, July 2017
- International Conference on Trust, Security and Privacy in Computing and Communication (IEEE TrustCom-17), Sydney, Australia, August 2017
- 7th International Conference on Internet Technologies and Applications (ITA 17), Wrexham, England, September 2017
- 10th International Conference on Ubiquitous and Future Networks (ICUFN 2018), Prague, Czech Republic, July 2018
- 13th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2018), Taichung, Taiwan, October 2018
- Blockchain Summit, Frankfurt, Germany, March 2019
- Collaborative Research Symposium on Security, E-learning, Internet and Networking (SEIN 2019), Darmstadt, Germany, March 2019
- 24th VDE ITG Mobilkomtagung, Osnabrück, Germany, May 2019
- 3rd Blockchain Public Day Deutsche Bahn, Frankfurt, Germany, October 2019
- 14th International Conference on Broad-Band Wireless Computing, Communication and Applications (BWCCA-2019), Antwerp, Belgium, November 2019
- Blockchain Conference for Financial Markets: Infrastructure for Digital Assets, Frankfurt, Germany, November 2019
- 34th International Conference on Advanced Information Networking and Applications (AINA-2020), Caserta, Italy, April 2020

Word count of main body of thesis: 49.890 Signed …….……

Date ……26.11.2021……

On Trust Optimisation for Decentralised M2M Services

Besfort Shala

Abstract

The service provision in traditional Machine-to-Machine Communication (M2M) ecosystems relies on commercial service providers. Their services are implemented using centralised infrastructures and focus on specific business processes. Recently, the role of end-users and their personal environments became interesting in terms of service provision. Intelligent devices located in the end-user's environment can be used to create M2M services. These services enhance the competition in the M2M marketplace and provide more service variety. The end-user integration creates a decentralised network of service providers, coming with benefits such as energy efficiency or platform independence. However, inexperienced end-users and missing authorities monitoring the service creation workflow can lead to unreliable M2M services. Decentralised environments also increase the risk of uncontrollable behaviour and untrustworthy activities. In such scenarios, trust relationships are essential to protect these networks and render them a perspective. Trust positively affects the overall security of decentralised systems and should be maintained using suitable trust management systems.

The aim of this thesis is to optimise the overall trustworthiness in decentralised M2M communities. This research initially identifies limitations related to three topics: decentralised M2M service provision, service testing, and trust approaches. Decentralised M2M service platforms suffer from a lack of trust. Existing M2M service testing concepts are not end-user-friendly, use centralised architectures, and do not follow end-user-based service provision aspects. Current trust management systems are using incomplete trust models not covering several attributes of an entity. They are also not considering the trust scores of new M2M services or service providers. This aspect allows malicious nodes to join the community and harm other members. Another drawback is that existing trust approaches do not provide a secure storage system for all the computed trust data. Overall, the review highlights the need for a new framework addressing these issues.

This thesis proposes a novel framework for trust evaluation and corresponding functional testing of decentralised M2M services. It presents a comprehensive trust approach that considers several aspects such as the trust status of new M2M entities, secure trust data storage, and reliable decision-making among the nodes. In order to overcome the lack of trust information about new M2M entities, service testing is incorporated within the trust evaluation process. Additionally, this research utilises blockchain technology to increase data integrity and optimise different parts of the framework. In terms of blockchain, this thesis also identifies drawbacks to existing consensus protocols and proposes a novel trust-based one. Furthermore, all M2M community members maintain the various trust activities using a community-based approach that relies on blockchain principles. Finally, this thesis also presents a prototypical proof of concept evaluation of the proposed framework components. The evaluation results confirm the benefits of this approach and its resilience against attacks in comparison to others. Overall, the combination of blockchain, service testing, and trust optimises the security in decentralised M2M communities.

Contents

List of Figures

List of Tables

1 Introduction

Traditionally, central commercial marketplaces like Google or Amazon provide services to the community. The Big Tech focuses on various specific business processes, continuously expanding their product range, and ever increasing their influence on their customers. At the same time, the number of smart devices within the end-user's environment is growing, providing new opportunities to create different Machine-to-Machine Communication (M2M) services that cover business cases that were previously not considered or only superficially. These alternative services also increase the diversity of services on the market and promote competition between service providers.

The integration of the end-user into the service provision process is necessary to implement the integration of local devices and make them accessible to external entities. The end-user integration creates a decentralised network of service providers. This alternative network or marketplace becomes very relevant for consumers in times of rising issues with global commercial service providers acting as “Big Brother”, such as data hoarding, privacy, or activity tracking. There is also a call of independency from supervisory authorities or specific platform rules. End-users respectively consumers want to control their identity, activities, and data. Decentralised marketplaces have the power to solve these issues. They provide an open market where service providers and consumers can join and operate with low fees (e.g., due to the missing of intermediaries). They also enable restrict-free self-organisation at all levels of service provision and use.

A major advantage of a decentralised approach is that local communities can easily be formed to meet specific needs. An example of such a community is a neighbourhood

where different households share their smart devices to cover the specific use case for building and street monitoring. The monitoring application is a composed (multi-owner) service which integrates different single services, such as temperature, humidity, motion, camera, evaluation, alarm, or notification services. These are provided by individual end-users within the local community using the several available devices (actuators, sensors) within their personal environment. In the local monitoring application, a camera service could detect a certain motion in the neighbourhood and trigger the alarm service, which turns on the siren and blue light. Then the alarm service activates the notification service, which sends short messages to the people concerned. There is no controlling authority and nodes are connected peer-to-peer in the network. A central commercial approach for this use case would mean higher acquisition costs and handing over personal data to less trusted entities.

The literature review shows that there are various M2M service platforms (as described in (Kim *et al.*, 2014)) used for different purposes. However, analyses have shown that most of the existing approaches do not rely on decentralised architectures and do not integrate end-users in the service provision process. Some projects (Lopez *et al.*, 2011; Kim *et al.*, 2012; Lin *et al.*, 2013; Steinheimer *et al.*, 2017a) introduce alternative approaches where end-users and their personal environments are part of M2M service provision. For instance, the approach in (Steinheimer *et al.*, 2017b) presents a user-friendly service platform. The service platform manages all available devices and services present in the personal environment of the end-user and integrates remote services provided by other end-users. It allows end-users to graphically combine devices or services and create complex M2M services that can be made available to other end-users. The project also proposes a Peer-to-Peer (P2P) network for communication and

information storage to avoid centralised entities. All participating peers (service providers and service consumers) are part of an M2M community for social networking purposes.

However, fully decentralised environments, where only end-users take responsibility for service provision, increase the risk of failures and malicious behaviour of the participants. There is no controlling entity that ensures that the created services meet the conditions for being deployed. Less technical knowledge or inability to police the behaviour of the end-users could lead to serious problems such as wrong, malicious, or not working services. These failures could happen intentionally or accidentally from the end-user side. Other limitations include the lack of trust in the network, missing access control schemes for joining/leaving service providers, and untrustworthy mechanisms for deploying M2M services. Existing approaches for decentralised M2M service provision do not consider these problems, nor do they provide an approach for testing such services. Moreover, they do not include concepts for creating trust relationships among end-users and services in the M2M community. Overall, trust in decentralised M2M communities is overlooked by existing approaches.

The inherent challenges of an end-user-based and decentralised M2M service provision approach lead to two main issues addressed in this research work. The first issue focuses on missing trust relationships among all M2M entities, including M2M service providers and M2M services. The aim is to optimise the overall trustworthiness in the M2M community. The second one includes the correct functionality of end-user-based M2M services as an essential element of decentralised M2M service provision.

The aims and objectives of this research thesis are presented in section 1.1, followed by an outline of the thesis structure in section 1.2.

1.1 Aims and Objectives

The aim of this research is to propose a framework for trust evaluation and corresponding functional testing of decentralised and end-user-based M2M services. This work produces an approach for evaluating the trustworthiness of M2M entities part of a decentralised network. It leads to the overall optimisation of trust in the community. Moreover, concerning the decentralised nature of the M2M community and the network of end-users (with possible less technical knowledge) acting as service providers, this research work also provides a methodology to test the functionality of M2M services. The overall research work results in a comprehensive framework covering both issues – trust and testing.

The main objectives of this research are subsequently outlined:

1. To analyse end-user-based and decentralised M2M service provision approaches, highlight their limitations, and define initial requirements for optimisation in terms of testing and trust.
2. To review existing approaches for service testing in M2M, highlight their benefits and limitations, and derive specific requirements for optimising the testing process.
3. To review existing approaches for ensuring trust in M2M communities, highlight their benefits and limitations, and derive specific requirements for optimising the trust evaluation process.
4. To define the architecture and associated methodology of the proposed framework for trust evaluation and corresponding functional testing of decentralised and end-user-based M2M services.

5. To specify a concept for end-user-friendly service testing by enabling community-based test activities.
6. To design a concept for evaluating the trustworthiness of new services or service providers joining the M2M community without previous history.
7. To design an overall trust evaluation framework considering different trust aspects such as service quality, end-user behaviour, or community task participation.
8. To specify a data management system enabling secure and trustworthy storage of computed and collected service and trust data in the M2M community.
9. To propose the integration of trust for service or service provider relationships and optimise different aspects of end-user-based and decentralised M2M service provision. These aspects include the admission control for the M2M community, the registration of M2M services, and cooperative M2M services.
10. To produce a proof of concept implementation to verify and evaluate the proposed framework for trust evaluation and corresponding functional testing of decentralised M2M services based on defined scenarios.

The key idea of trust optimisation in this research is to present a decentralised and end-user-based environment where all nodes provide reliable services to each other, service and trust information is stored in a tamper-proof way, and the whole community is maintained by all members actively participating in different tasks.

The order of objectives declared above corresponds to the general structure of this thesis, as presented within the following sections.

1.2 Thesis Structure

Chapter 2 describes the theoretical background of this thesis by introducing the principles of Machine-to-Machine Communication (M2M) systems and the Internet of Things (IoT). It also provides an overview of the concept of end-user-based and decentralised M2M service provision. The overview is given by describing the approach in detail and highlighting its limitations.

Chapter 3 presents existing approaches, which are related to the research field. First, relevant approaches for testing M2M/IoT services are reviewed. Then, several trust evaluation/management approaches in M2M/IoT are evaluated. This chapter concludes with the definition of requirements for a novel framework for ensuring correct services and trustworthy M2M entities in a decentralised M2M community.

The requirements identified in chapter 3 are used to propose a novel framework for trust evaluation and corresponding functional testing of end-user-based and decentralised M2M services in chapter 4. Moreover, this chapter introduces an architectural overview of the proposed framework and briefly explains its components.

Chapter 5 starts by introducing blockchain technology and presents a novel trust-based consensus mechanism used throughout the proposed framework. Additionally, this chapter highlights the relevance of integrating service testing in the trust evaluation process and introduces a novel approach for evaluating the trustworthiness of new services based on functional and performance test results.

Chapter 6 presents a comprehensive trust evaluation system by including several trust-related aspects in the process. The trust evaluation is supported through a new trust

computation and aggregation scheme. This chapter also proposes a novel blockchain-based approach for optimising the overall trustworthiness of the trust evaluation and management system.

Chapter 7 discusses the proof of concept implementation of the proposed framework components and the evaluation of the proposed framework.

Chapter 8 summarises the research work by highlighting achievements and limitations. Moreover, this chapter includes potential areas for future research and development.

2 Decentralised Service Provision in Machine-to-Machine Communication Systems - Limitations

As indicated in the introductory chapter, the main objectives of this research are ensuring trustworthiness and functional reliability of decentralised and end-user-based M2M services. Therefore, section 2.1 gives an overview of M2M communications/Internet of Things systems and provides necessary information regarding M2M system architecture and terminology. Moreover, section 2.2 describes the variety of M2M service platforms and highlights the focus of this research on end-user-based service provision approaches. Furthermore, section 2.3 introduces the foundations for service provision with end-user integration in a decentralised environment. Section 2.4 highlights identified limitations which serve as a basis for optimisation purposes conducted during this research.

Some parts in this chapter have been published in (Shala *et al.*, 2018; 2019a; 2019c; 2020b; 2020a).

2.1 Fundamentals of Machine-to-Machine (M2M) Communication and Internet of Things (IoT)

Nowadays, the number of intelligent devices is growing and emerging technologies are enhancing the quality of life in the world. A connected world of devices or so-called things often is associated with the terms of Machine-to-Machine (M2M) communications and the Internet of Things (IoT). In this context, various valuable services are created and part of daily activities. These services are made available to consumers via different available service platforms using different service provisioning approaches. This section provides the basics about M2M and IoT and clarifies other related concepts relevant to this thesis.

The literature is often using the terms M2M and IoT interchangeably. According to (ETSI, 2010; Holler *et al.*, 2014; Hassan, 2018), M2M communications can be described as solutions allowing communication via wired or wireless communication networks between devices (same devices are also considered in IoT) without direct human intervention. M2M solutions cover a wide range of application fields such as telematics, metering, remote monitoring, fleet management, security, ATM (Automated Teller Machine), and POS (Point of Sales) (Holler *et al.*, 2014). M2M communications support the optimisation of productivity gain or cost reductions. One example is industrial production facilities where relevant sensor data are captured (e.g. temperature sensor), forwarded to other entities, and used to trigger other related business processes (Holler *et al.*, 2014; Hassan, 2018). The essential elements of M2M reinforce the statement that

M2M can be seen as the forerunner of IoT. The IoT can be viewed as a more generalised term used to cover a broader range of application fields. Furthermore, the IoT is defined as "a set of technologies, systems, and design principles associated with the emerging wave of the Internet-connected things" (Holler *et al.*, 2014). According to (Cirani *et al.*, 2018), IoT "encapsulated a vision of a world in which billions of objects with embedded intelligence, communication means, and sensing and actuation capabilities will connect over IP networks". In general, the difference between M2M and IoT is that the former relies on point-to-point communications (using hardware modules and dedicated protocols), whereas the latter depends on IP-based networks (using open protocols) (Hassan, 2018). However, this difference has no significance in this research as the focus is on the service layer. Thus, M2M and IoT are used synonymously in this research thesis when providing a literature review of existing approaches. Moreover, this thesis will use the terminology of M2M for providing more information on service provisioning, existing limitations, and presenting an optimisation approach.

Figure 2.1 shows a general high-level layer model highlighting the different components and their relationships within M2M/IoT systems. The reference models defined for M2M (ETSI, 2013b) and IoT (ITU-T, 2012) show similarities. The layer model consists of the device layer, network layer, service capability layer, and application layer. The device layer consists of the devices used to run M2M/IoT applications, the device capabilities including, for instance, direct or indirect interactions of devices with the communication network, and the gateway capabilities providing, among others, multiple interface support (Boswarthick *et al.*, 2012; ITU-T, 2012; ETSI, 2013b; Holler *et al.*, 2014). The network layer consists of networking and transport capabilities. The first one focuses on network connectivity issues, and the second one on transportation, control, and management

information (Boswarthick *et al.*, 2012; ITU-T, 2012). The service and application support layer consists of capabilities providing different functions derived from core network functionalities and used to support applications (ITU-T, 2012; ETSI, 2013b). Finally, the application layer of the layer model contains the M2M/IoT applications (ITU-T, 2012; ETSI, 2013a). Permission to reproduce Figure 2.1 has been granted by ITU.

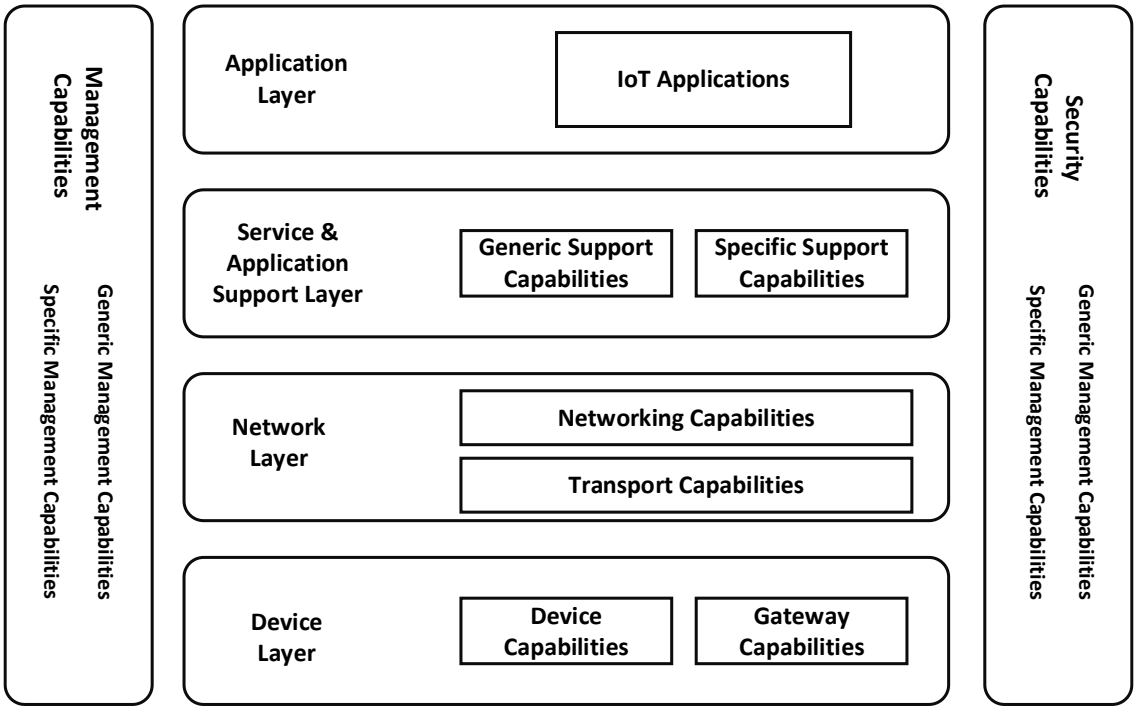

Figure 2.1: M2M/IoT high-level architecture (ITU-T, 2012)

The (ETSI, 2013a) and (oneM2M, 2019) present a collection of technical terms related to M2M. Thus, according to (ETSI, 2013a), an M2M service provider provides M2M services to other M2M application service providers or end-users. M2M application service providers use these services to create/operate M2M applications. The service logic of M2M applications is used to implement M2M application services provided to others for consumption. This research will not distinguish between the terms “application

service", "application" and "service" as specified in (ETSI, 2013a; oneM2M, 2019). For simplicity, the generalised term "M2M service" is used in the following chapters.

Service platforms play an essential role in an M2M/IoT system. They provide services and ensure connectivity between devices in a system. Other tasks include the management of participating nodes and the information exchange between entities. The detailed functionalities of service platforms in the context of M2M are specified in (oneM2M, 2016d) and (oneM2M, 2016b). According to (oneM2M, 2016c), the M2M ecosystem consists of end-users, service providers, and network providers. Similarly, (ITU-T, 2012) identifies different business roles in the IoT ecosystem: application customer, application provider, platform provider, network provider, and device provider. The (ITU-T, 2014) highlights that the roles of end-users and service providers are different.

The following sections provide general information on existing service platforms focusing on decentralised ones with end-user integration in the service provisioning process.

2.2 M2M/IoT Service Platforms

An increasing number of M2M/IoT service platforms address different fields of interest and business models. Several publications survey existing platforms focusing on specific characteristics based on their research scope (Castro *et al.*, 2012; Kim *et al.*, 2014; Guth *et al.*, 2018; Hejazi *et al.*, 2018; Fridelin Panduman *et al.*, 2019). The analysis of various approaches concludes with centralised and decentralised architectures used in existing service platforms (Mocnej *et al.*, 2018; Asemani *et al.*, 2019). However, limitations of

centralised approaches are large resource requirements for service development and maintenance, high costs for platform operation, and single points of failure of centralised entities. The integration of the end-user and its M2M environment in the service provisioning process enables flexibility, decentralisation, service variety, and energy efficiency to the community. Thus, smart environments can be supported by the integration of intelligent devices as part of end-user environments. This integration provides great potential to complement existing services and create new complex ones accessible to other end-users. Therefore, in the following, some relevant decentralised platforms with end-user integration are briefly described.

The literature review identifies a collection of some relevant M2M/IoT service platforms aiming to be decentralised (Kim *et al.*, 2012; Kitagami *et al.*, 2014; Shih *et al.*, 2014; oneM2M, 2016a; Steinheimer *et al.*, 2017a), to include end-users in the service creation/providing process (Lopez *et al.*, 2011; Kim *et al.*, 2012; Lin *et al.*, 2013; Steinheimer *et al.*, 2017a), and to use the devices part of end-user environments for service creations (Xiaocong and Jidong, 2010; Lopez *et al.*, 2011; Kim *et al.*, 2012; 2014; Doukas and Antonelli, 2013; Lin *et al.*, 2013; Padilla *et al.*, 2013; Steinheimer *et al.*, 2017a). However, only the approaches in (Kim *et al.*, 2012; Lin *et al.*, 2013; Steinheimer *et al.*, 2017a) provide a fully or partially decentralised and end-user-based service platform. Table 2.1 summarises the evaluation of the different M2M service platforms.

Table 2.1: Evaluation of M2M service platforms

Requirements	Service Platforms										
	(Kim *et al.*, 2012)	(Kitagami *et al.*, 2014)	(Shih *et al.*, 2014)	(oneM2M, 2016a)	(Lopez *et al.*, 2011)	(Lin *et al.*, 2013)	(Doukas and Antonelli, 2013)	(Xiaocong and Jidong, 2010)	(Kim *et al.*, 2014)	(Padilla *et al.*, 2013)	(Steinheimer *et al.*, 2017a)
Decentralised	o	o	-	o	-	o	-	-	-	-	+
End-user-based service provision	o	-	-	-	o	o	-	-	-	-	+
Personal environment integration	+	-	-	-	+	+	+	+	+	+	+
Support trust building	-	-	-	-	-	-	-	-	-	-	-
Support service testing	-	-	-	-	-	-	-	-	-	-	-

Assessment notation: + satisfied, o partial satisfied, - not satisfied

The authors in (Kim *et al.*, 2012) introduce an enhanced Dynamic Service Overlay Network (e-DSON) platform focusing on distributed service provision. The presented platform supports the provision of individual user services and operates on distributed servers on the internet. Moreover, it allows the combination of decentralised device resources and services for complex service implementation. However, the platform is maintained by a centralised operator and does not involve the end-user directly in the service composition process. Next to them, the authors in (Lin *et al.*, 2013) present a decentralised and end-user friendly M2M service platform. The end-user has access to the local devices and can be integrated into the application design via the platform. However, the approach introduced in (Lin *et al.*, 2013) is limited only to end-users inside their specific smart home and services are not made accessible to external entities. The approach in (Steinheimer *et al.*, 2017a) relies on a Peer-to-Peer (P2P) network for communication and information storage to ensure a completely decentralised

environment. The end-user-based service provisioning is enabled through a user-friendly web interface where the service logic can be graphically designed.

The review concludes that most of the reviewed M2M service platforms are not (entirely) decentralised as they include centralised system elements or single service platform providers in their concept. Besides them, many approaches exclude (partly) the end-user from the service creation/provision process. Only the work in (Steinheimer *et al.*, 2017a) provides a fully decentralised architecture and involves the end-user in the service provision process. Therefore, it serves as a basis for further investigations in this research thesis.

The review also highlights that none of the reviewed approaches supports trust building between the entities. Moreover, they do not consider service testing in their approaches. Thus, trust and service reliability in end-user-based and decentralised service provision communities are neglected. Both issues are addressed in this research thesis and lead to a novel framework for trust evaluation and corresponding functional testing of M2M services.

The next section describes the foundations of decentralised and end-user-based M2M service provision. Additionally, identified limitations and problems supplemented with initial requirements are presented in section 2.4.

2.3 Decentralised and End-User-Based M2M Service Provision

As derived in the previous section, the authors in (Steinheimer *et al.*, 2017a) present an entirely decentralised M2M system architecture where the M2M service platform itself is not provided by a platform operator but by end-users of the platform itself. End-users can design individual services and make them available for other end-users or central service providers. It is also possible for end-users to cooperate in the provision of complex M2M services. The approach distinguishes between technical peers (e.g. device) and non-technical peers (e.g. human). An end-user can provide services to others using the M2M service platform running on a technical device (e.g. DSL router, personal computer, smartphone).

M2M Service Platform

The M2M service platform consists of a local Service Creation Environment (SCE) and a Service Delivery Platform (SDP) (Steinheimer *et al.*, 2017b). Moreover, the service platform (illustrated in Figure 2.2) includes all available devices (actuators, sensors, combined) and services present in the personal environment of the end-user and integrates remote services provided by other end-users. The SCE provides a Graphical User Interface (GUI) for designing the behaviour of an M2M service (service logic) graphically and for combining building blocks representing the M2M service components, M2M devices, and multimedia service components. The SCE automatically transforms the service logic into a formal service description that can be parsed, interpreted and executed by every underlying execution environment. Thus, the designed service can be provided

as a service to other end-users via the SDP. Permission to reproduce Figure 2.2 has been granted by the author of the referenced publication.

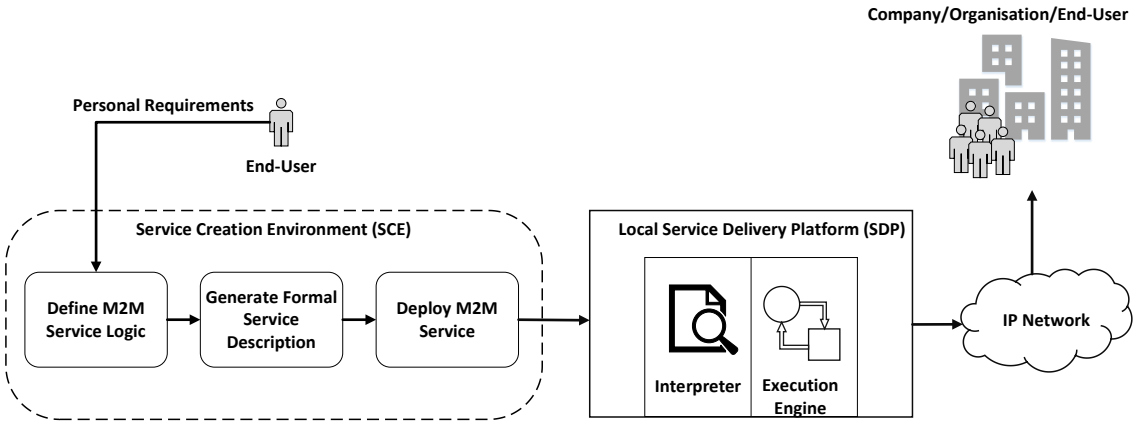

Figure 2.2: Service creation and provision process (Steinheimer, 2018)

Decentralised Architecture

The information exchange and communication between the peers for service utilisation and generation signalling is done directly (end-to-end) using M2M communication protocols. The information storage is implemented through a distributed data storage using a P2P overlay network (e.g. Chord - Distributed Hash Table (DHT)) formed by all existing end-user nodes (peers).

An M2M community (Steinheimer *et al.*, 2013) serves for social networking between all the nodes and shows the linking of end-users at the interest level. Furthermore, the M2M community can address different application fields or geographical locations. End-users and the services they provide can be part of several sub-communities at the same time. The M2M community provides the possibility to release services for a specific user group or restrict it to a specific geographic area. Moreover, the M2M community forms the legal basis for operating a decentralised platform. Any node is voluntary joining the M2M

community and accepting the usage conditions for data processing. Figure 2.3 shows the structure of P2P connected peers within an M2M community, where some of them are acting as service providers by providing services to other peers (service consumers). Permission to reproduce Figure 2.3 has been granted by the author of the referenced publication.

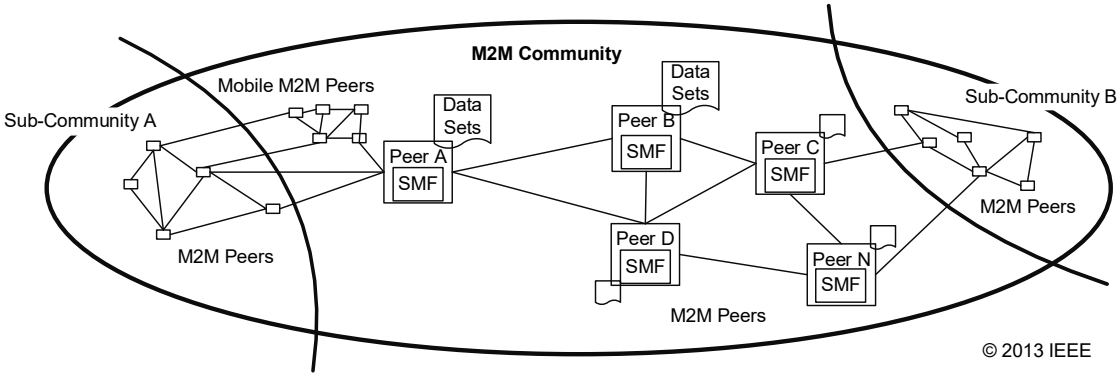

Figure 2.3: M2M community (Steinheimer, 2018)

Service Types and Descriptions

In principle, the service provision process produces single and composed M2M services (Kim *et al.*, 2012; Reetz, 2016; Steinheimer *et al.*, 2017a). Single M2M services can be used locally in the personal environment of an end-user. They are implemented by single M2M devices consisting of input/config and output parameters. It is also possible to make them available for external entities (as remote M2M services). The combination of different single M2M services forms a composed M2M service which can also be used locally or by other M2M community members. All available M2M services are made visible for every end-user via the GUI.

The service behaviour is defined using state machines and described through a platform-independent model (PIM). An end-user creates the services by designing an abstract

behaviour model of M2M services. The abstract modelling allows the end-user to describe the system without specific knowledge about the execution platform.

For describing the M2M service logic, a formal service description language is used (Steinheimer *et al.*, 2019). The service description is deployed in the local environment of the end-user, which is responsible for the service execution. Specifically, M2M services are described by machine-readable State Chart extensible Markup Language (SCXML) (W3C, 2015). They are consumable for others via their Interface Descriptions (IFD).

M2M Service Registration

The service registration process starts with the service provider who stores the interface description (also containing service ID and contact information) of the new M2M service in the DHT database (called service/application registry (SAR)). Every M2M community member has access to the DHT and can request one of the available M2M services by first acquiring the corresponding interface description. In some cases, several service providers offer the same services in the form of identical IFDs but with different contact information. The same services acting as individual instances are stored with the same service ID in the SAR. Service consumers can use a keyword to find specific services and freely select a service instance for usage.

Cooperative M2M Services

Cooperative M2M services are created by the composition of different single M2M services provided by various service providers. Usually, an end-user acting as a service designer creates the logic of a cooperative M2M service by combining the different

building blocks (service components) in the GUI. The service logic is created based on the personal interests of an end-user, and in cases of a cooperative service, it has the form of a service chain (see Figure 2.4). The created service logic is represented in the form of an SCXML service description which is sent to all community members for configuration. The configuration process includes selecting the different service instances responsible for a specific position in the service chain. However, the M2M community consists of several same or different M2M services. Multiple end-users can offer different instances of the same M2M service. The fact that there are multiple instances of the same service makes the configuration and implementation of the cooperative M2M service challenging. The authors in (Steinheimer *et al.*, 2017a) suggest a random selection of the available service instances for composition. However, this is not a secure method and could lead to selecting insecure or low trusted M2M entities. This could lead to unstable and inefficient M2M services as the service chain could contain unwanted components.

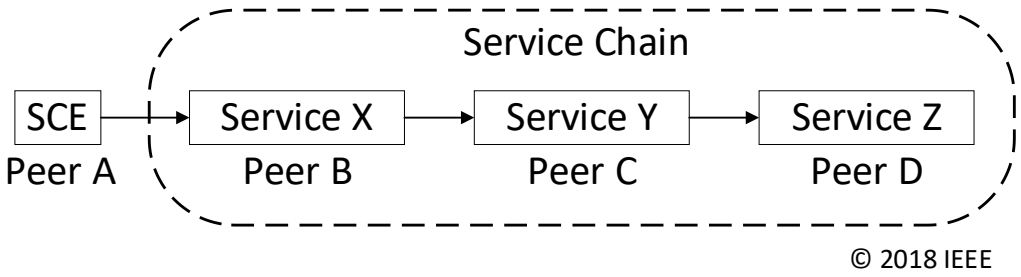

Figure 2.4: Service chain for cooperative M2M service (Shala *et al.*, 2018)

Example

An example of a modelled M2M service is shown in Figure 2.5 and can be explained as follows: Service 1 receives sensor values from the M2M gateway. Based on predefined configurations, it detects a specific sensor value and forwards this information to the next service, in this case, service 2. Service 2 is providing a service for determining the

consumer that should get be alarmed. Service 2 receives the information from service 1 and does the determination process based on predefined criteria. These results with contact information about the consumers are sent to service 3. Then, service 3 informs the relevant consumers about the temperature values by sending a text message.

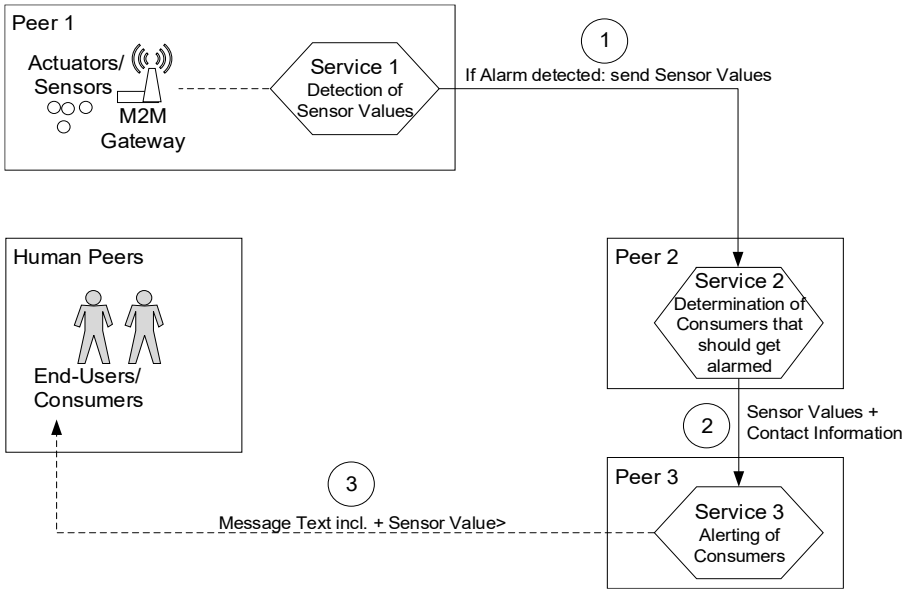

Figure 2.5: Use case temperature surveillance

2.4 Trust- and Test-Related Issues

The end-user integration into a fully decentralised M2M environment for service provision is a promising alternative to traditional approaches. However, there is a risk in terms of trust among the participating entities and the service reliability. The reviewed approaches in section 2.2 do not consider how decentralised M2M services will be tested and do not provide strategies to handle trust relationships within the M2M community. This section identifies existing limitations with a focus on service/peer trustworthiness

and service functionality. The limitations are classified in trust- and test-related issues. These issues are used to derive initial requirements serving as a basis for optimisation purposes in this research.

Trust-Related Issues

- Joining and leaving the M2M community: One issue in decentralised M2M communities is the way nodes join or leave the network. Misbehaving nodes could harm the system when entering the community. They could try to withdraw from the system unresponsively or unexpectedly whenever a request is received. The same also appears for new services as they can be registered without any functionality or security considerations. Others have no information regarding the trustworthiness of these services or service providers.
- Providing M2M services: Untrustworthy peers may also pretend to offer services in order to confuse other peers. Moreover, a peer can provide several services, some of which are trustworthy and some of which untrustworthy. In this context, it is important to know the trust scores of all M2M entities, either if they are new or existing ones.
- Selecting M2M service instances: Multiple service providers can also offer different instances of the same M2M service. Decentralised networks avoid centralised coordination entities for selecting specific service instances to create a cooperative M2M service, and relying on random or manual selection of service instances is not fair and secure. It can lead to a selection of insecure or untrusted services building up an unreliable cooperative service provided to the M2M

community. Therefore, a trust selecting principle is essential to assign peers to an M2M service based on their trust scores.

- Storing information: Another issue in decentralised networks is the way nodes store information among themselves. P2P storage approaches suffer from the falsification of entries by malicious nodes. The probability of receiving wrong information from different P2P-based entities is very high (as there is no controlling authority and malicious peers could falsify information).

Trust evaluation of the entities (services and peers) in the M2M community is crucial to mitigate the above-mentioned trust-related issues.

Test-Related Issues

- Unreliable M2M services: Missing centralised entities make the process of service provision uncontrollable. No one can ensure that newly created services meet the necessary conditions to be provided to service consumers. End-users acting as service providers can link wrong services with each other or do wrong service configurations. As a result, wrong or not working M2M services are made available to the M2M community.
- Inexperienced service providers: End-users may have little technical knowledge for creating M2M services. Additionally, they are also facing difficulties in performing testing activities in the M2M community.
- Not complete service information: The service creation process follows not the traditional lifecycle. Services are created based on the personal requirements of end-users (service providers) and not in consultation with other service consumers. Thus, there is no natural language description and requirement

specification of M2M services presented. The main issue is the generation of tests from existing information such as service descriptions and interface descriptions of distributed M2M services.

These issues reveal the necessity to test M2M services and provide an end-user friendly service testing platform.

Initial Requirements

The different trust- and test-related issues lead to the definition of initial requirements serving as a starting point for this research. Figure 2.6 summarises the conditions and problems in a decentralised community with end-user integration in the service provision process. Moreover, it shows the starting requirements derived from these conditions and problems.

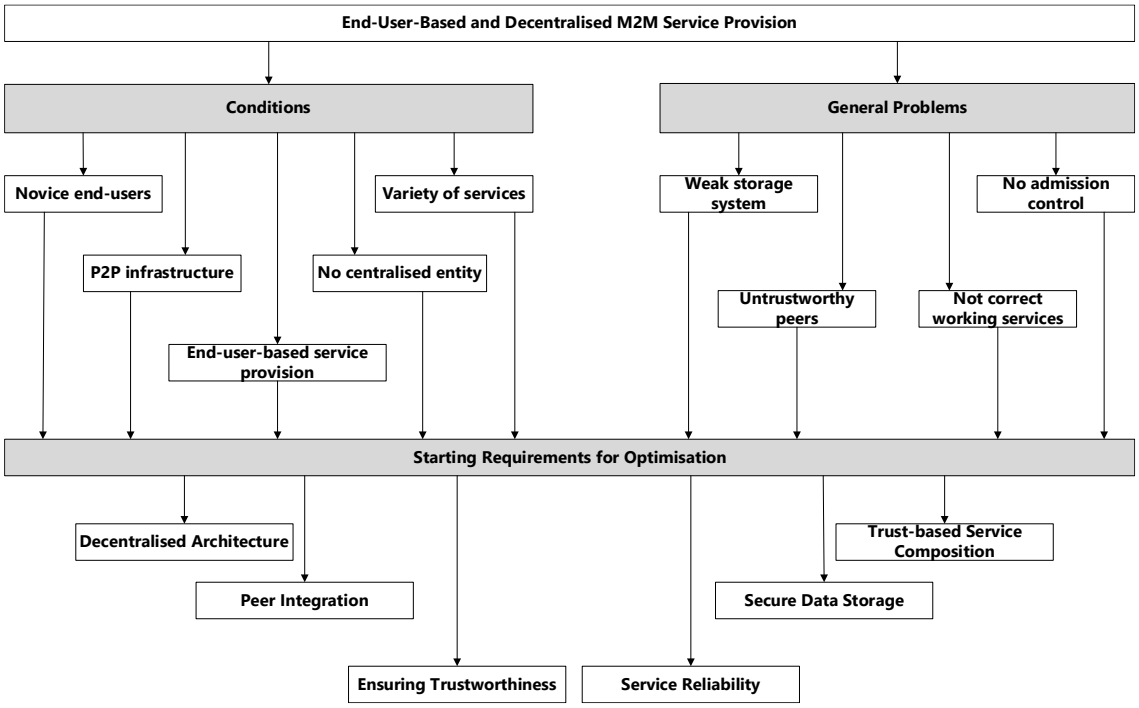

Figure 2.6: Overview of general conditions and requirements

The starting requirements for a new framework are briefly explained in the following:

- Ensuring trustworthiness: The framework should evaluate the trustworthiness of all M2M entities. Trust should be used to build relationships among the nodes. The trust status should be considered for new and existing M2M entities. For instance, the trust score of new M2M services should be computed to support that service consumers do not use untrusted services and possible malicious services are avoided in advance.
- Service reliability: Every M2M service should be tested to ensure that only the correct working ones are operated and provided to the M2M community.
- Secure data storage: All information in the M2M community, including the trust data, should be stored trustworthy among the nodes.
- Decentralised architecture: The framework should use a completely decentralised architecture to avoid the limitations of centralised ones.
- Peer integration: All peers of the M2M community should be involved in trust and testing activities. The involvement is necessary to maintain the decentralised structure and enable reliable service provision, trust evaluation, and test execution outcomes. The integration and interaction of peers should be incentivised through challenging activities.
- Trust-based service selection and composition: Only trusted M2M services should be selected for service consumption or for creating more complex (composed) M2M services.

Two factors influenced the research performed for this thesis. The basic factor is the limitations and requirements of decentralised service provision approaches described

above. The core factor is the drawbacks of existing trust and testing approaches identified in chapter 3. This thesis will lead to an overall framework for trust evaluation and functional testing that overcomes current limitations and optimises the overall trustworthiness in the M2M service provision process.

2.5 Conclusion

As stated in chapter 1, this thesis aims to ensure reliable and trustworthy M2M services and service providers. It considers end-user-based M2M service provision and decentralised M2M communities as the main objects of this research. Therefore, this chapter investigated the general environment and the main principles of M2M/IoT, service testing, trust, and blockchain.

Section 2.1 introduced M2M/IoT fundamentals and provided useful information on service platforms (section 2.2) focusing on decentralised ones. In this context, decentralised and end-user-based service provision approaches in M2M were discussed in section 2.3. Furthermore, potential limitations leading to initial requirements for optimisation in terms of trust and testing were identified in section 2.4.

The following chapter will identify relevant existing approaches in the fields of testing and trust management in M2M/IoT. Furthermore, the highlights and disadvantages of these approaches will be determined. The outcomes of the review will be used to derive the main requirements for defining a novel framework enabling trust evaluation and functional testing in end-user-based and decentralised M2M service platforms.

3 Challenges for Reliable, Trusted, Decentralised, and End-User-Based M2M Service Provision

This chapter starts (section 3.1) by presenting an overview of existing test approaches within the M2M/IoT field and other potential related works. The review includes the identification of existing limitations. Section 3.2 briefly summarises existing trust management and evaluation approaches in M2M/IoT. Their deficits are also highlighted in the section. Finally, section 3.3 summarises the findings of the conducted review and presents the derived requirements used in this research work for a novel framework for trust evaluation and corresponding functional testing of end-user-based and decentralised M2M services.

Some parts in this chapter have been published in (Shala *et al.*, 2019b).

3.1 Related Work on Current Test Approaches in M2M/IoT

One of the aims of this research work is to optimise the service reliability in M2M communities. Therefore, new and existing M2M services have to be tested continuously with a focus on functionality. According to (Wacht, 2017), service testing plays an essential role in avoiding errors during service development and preventing failures during service consumption. Functional testing can be described as a process that verifies and validates the functional behaviour of a system based on predefined functional

requirements. For more information on functional testing see (IEEE, 1990; Baker *et al.*, 2008; Pezzè and Young, 2009; German Testing Board, 2014).

The literature review shows only a few existing test approaches within the domain of M2M/IoT. Moreover, it also points out that functional testing in IoT systems is overlooked as only a few existing publications can be found. Therefore, the following review also covers some non-functional testing approaches that include aspects relevant to this research. Different existing approaches are identified, and the highlights are presented in the next paragraphs. Figure 3.1 presents a classification of these approaches based on their main characteristics. The outcomes of the review are presented at the end of this section.

The authors in (Reetz *et al.*, 2012) introduce a business-oriented service composition approach of IoT services and a formalised integration of semi-automated testing. They propose to use a comprehensive model-based testing approach to automate the test design/execution process and introduce a test-driven life cycle management to merge the service creation process with the testing phase. This cycle starts with the modelling phase, where the service is created based on functional specifications. Afterwards, the composition of different simple services builds composed services and generates service and interface descriptions. Then, metadata are identified for service provisioning, including semantic descriptions of the service contract and the run-time service environment where the service is to be deployed. The service and test execution happen in a sandbox environment. The execution phase is used to monitor and evaluate the running service based on its semantic description. For efficient test case generation, a

semantic test model is introduced. This model contains the service description, which includes the service interface description and the resource interface description.

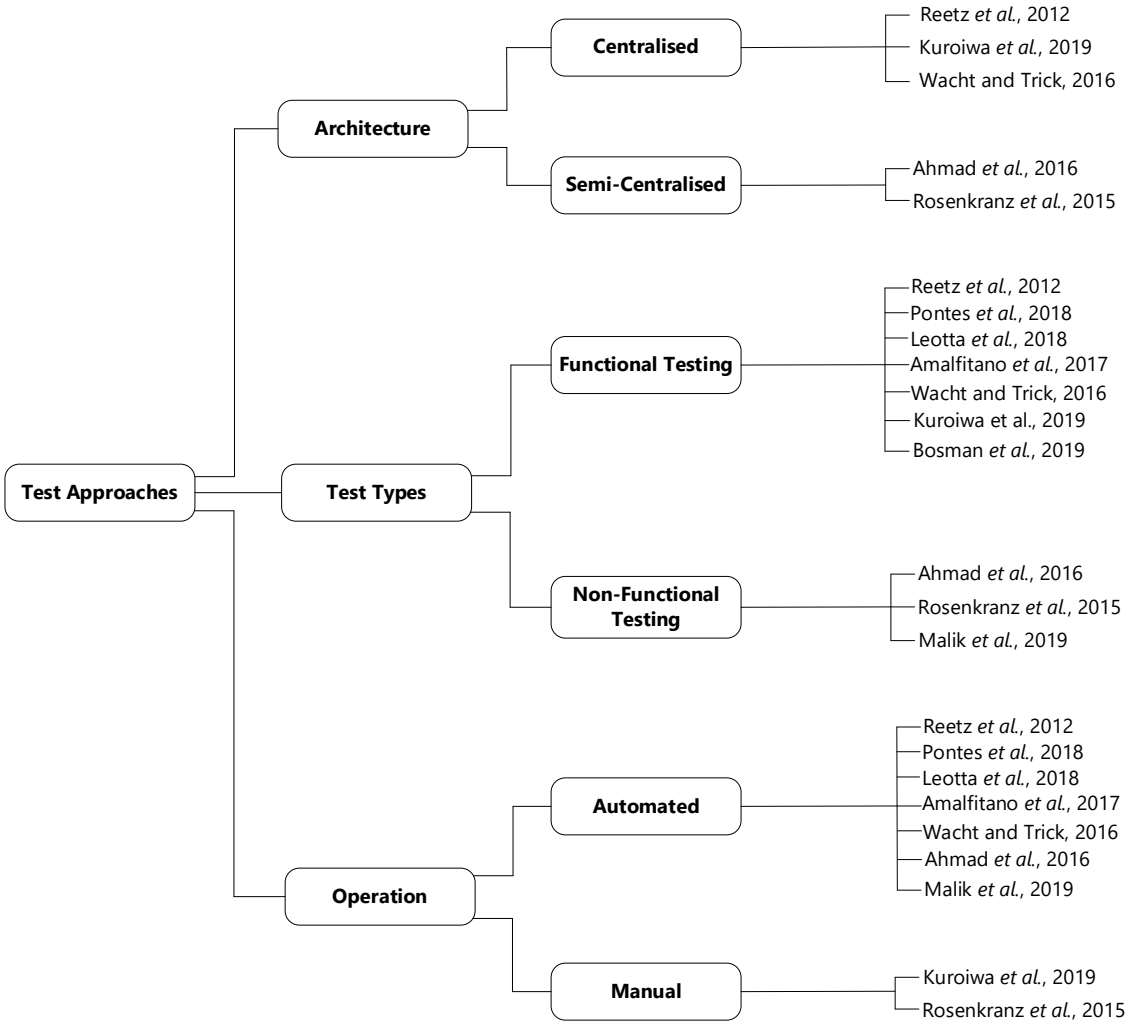

Figure 3.1: Main characteristics of existing test approaches in M2M/IoT

The authors in (Pontes *et al.*, 2018) suggest a test automation framework based on patterns used for integration testing in IoT. First, the authors propose a feature model to present the plurality of components and features within an IoT ecosystem. The feature model can support testing processes by selecting appropriate test patterns based on the functionality that will be tested. The authors also state that recurring behaviours of IoT systems could be described as test patterns. These patterns are used to check the capability of whether

readings and measurements are performed, commands are executed, alerts are sent, or actions are executed by the corresponding IoT entity. Another feature of the patterns is that users with less technical knowledge can easily instantiate them. Thus, specifications, such as SUT and data source settings, trigger information, and expected outcomes are required to operate the proposed framework.

An acceptance testing approach for IoT systems is proposed in (Leotta *et al.*, 2018). The authors apply their approach to a diabetes mobile health use case scenario consisting of sensors/actuators, smartphones, and a cloud system. The proposed approach focuses on system level testing and relies on three key elements. First, they introduce the formalisation of the system behaviour to be able to apply model-based techniques for test case derivation. Therefore, they use Unified Modelling Language (UML) state charts to model the behaviour of the system. Second, they propose to virtualise the whole IoT systems (physical devices) to overcome testing limitations under real conditions (more expensive and complex). The third element includes the definition of the test scenarios and test cases using the information from the UML charts (defined transitions and assertions). To define the number of test cases, the authors propose to apply the ad-hoc path coverage criterion, which allows defining more interesting scenarios using fewer paths. Finally, the authors define the implementation of the test cases where the main point is the localisation and interaction with the user interface (UI) components.

The authors in (Kuroiwa *et al.*, 2019) analyses regression tests on IoT systems to verify that system configuration changes do not produce functionality failures of the system under test. Therefore, they propose a so-called hybrid testing environment consisting of emulators, a test case generator, and a model checker. The emulators serve as a system

under test and are used to implement different system configurations. Moreover, using emulators, the problems with reconstructing IoT systems in the laboratory, such as cost and space restrictions, are avoided. Key elements are the system configurations written in the eXtensible Markup Language (XML) and the test scenarios written in Promela (model description language used for the model checker SPIN). The test case generator uses both to generate executable test cases for the regression tests. These tests are performed by the test execution unit called SPIN in all possible execution orders of communication.

The authors in (Bosmans *et al.*, 2019) present a different testing approach combining simulation and real-life testing for IoT systems. They focus on testing the functional requirements of IoT systems at the system level and integrate the impact of the behaviour in the process. The authors identify that modelling human behaviour is not as simple as for other local entities and review two modelling techniques: explicit (agent-based) modelling and data replay (datasets to replay entity behaviour). The proposed hybrid test system uses simulation or real-life testing depending on the development phase. Thus, at the beginning of the development, the testing mainly relies on simulation and is substituted by real-life testing during other steps. The approach presented in (Bosmans *et al.*, 2019) provides an interesting point by incorporating different IoT entities such as human actors and their behaviour in the testing process.

The authors in (Amalfitano *et al.*, 2017) highlight the benefits of integrating the *x In the Loop* (xIL) approach to the IoT domain. The approach enables virtualised testing of systems at model, software, and hardware level. The authors propose to use a model-driven approach to verify and validate IoT systems. Moreover, they state that an IoT

application is a composition of different so-called Things. First, their proposed approach starts with the so-called Model in the Loop (MiL) testing, where the model of the Thing is tested in the early stages of its development. This step also includes the simulation of the Thing and the context in order to test the functionality. Afterwards, in the second step, the generated source code of the Thing model is tested against the simulated context (Software in the Loop (SiL). It concludes with the outcomes of whether the software of the Thing interacts well or not with the context. The last step includes the Hardware in the Loop (HiL), where the Thing is deployed on real hardware to test its behaviour when interacting with the simulated context. These steps are part of the proposed Thing in the Loop (TiL) approach. The abstract test cases are automatically generated based on the defined test model. These test cases are used in a later step to generate specific test cases under the different proposed loop techniques.

The approach described in (Wacht and Trick, 2016) defines a methodology for testing the functional behaviour of value-added services. Therefore, the authors introduce a specific description language called Service Test Description for value-added services. This description is used as a basis for test case generation and is a combination of service and test specifications. The Service Test Description is divided into two parts. The first part is the architectural perspective which contains general information about the value-added service. The second part is the behavioural perspective which includes requirements specifying the functionality of the value-added service. In order to create and execute tests automatically, a so-called Test Creation Framework is introduced. It consists of different components. These components are used to graphically specify test description instances, create reusable test modules, generate behaviour models based on state charts, and generate abstract and executable test cases.

For testing IoT platforms, the authors (Ahmad *et al.*, 2016) present an approach called Model-based Testing as a Service (MBTAAS). MBTAAS is a combination of model-based testing (MBT) and service-oriented solution. The whole process starts with the test architect, which defines test objectives derived from specific requirements. These objectives are used to model the SUT, which contains static and dynamic views of the system. This test model is the basis for test generation using a coverage-based test selection criterion. The test generation, test publication, and test execution are automated processes. The MBTAAS process includes the implementation of web services. Through the communication between web services, all testing steps are implemented. There is also a publication service that produces a customised test description file from the MBT results file. An execution service is used to get this test description for test execution. A reporting service collects and sends the test results to the database, where also input data for test execution are stored. In connection with the database, the web front-end service provides the results from the reporting service to the end-user and is also used to configure/ launch test campaigns. For test case derivation, the specific test description is used, which contains the MTB model, the configuration file, and test data.

The authors in (Rosenkranz *et al.*, 2015) propose a test system architecture that can be used for open-source software development. They highlight the importance of ensuring interoperability in distributed IoT systems and propose a test approach for interoperability testing using crowd computing principles. Therefore, they introduce a test architecture consisting of a centralised entity called continuous integration (CI) broker and a test cluster (distributed element) consisting of different clients. The CI acts as a coordinator, triggers test executions, and reports the resulting test outcomes. The test cluster builds test sources and performs test executions on test platforms. Multiple scripts are used for

the test execution of the derived test cases. These scripts also contain information about the SUT and the test configuration (information about the build environment and test platform requirements). The interaction between test scripts and the test platform is implemented using a newly introduced abstraction layer and its unified Application Programming Interface (API). Moreover, the authors highlight the benefits of using the Network Experiment Programming Interface (NEPI) as a description language in the platform abstraction layer.

The authors in (Malik *et al.*, 2019a) propose an automated testing model for distributed IoT systems. The introduced approach uses model-based principles and can be used for different types of testing. These testing methodologies are incorporated within a so-called IoT testing as a service model. The authors use a distributed cloud service to support automated testing. The first test service of their model is the distributed interoperability testing. It is based on distributed remote devices and enables automated interoperability and conformance testing. Instead of using test case suites, the authors propose to use distributed test plugs containing quick test case responses. The next test service is the conformance testing based on oneM2M where test plugs are used to test different system specifications. The third test service is securing IoT using distributed systems analysis. It is used to identify system vulnerabilities and increases system reliability. Finally, semantic and syntactic validation is used to validate the semantic/syntactic correctness of corresponding data using different ontologies. The authors provide some specific details on each of the different test methodologies in their work.

The review of the approaches mentioned above has conducted the following outcomes.

- **Architecture:** Most of the reviewed approaches are missing to provide details regarding the level of centralisation/decentralisation (Amalfitano *et al.*, 2017; Leotta *et al.*, 2018; Pontes *et al.*, 2018; Bosmans *et al.*, 2019). Others propose different optimisation elements within traditional testing methodologies where test developers act as centralised entities (Reetz *et al.*, 2012; Wacht and Trick, 2016; Kuroiwa *et al.*, 2019). In (Ahmad *et al.*, 2016) and (Rosenkranz *et al.*, 2015), a decentralised architecture is intended. However, using centralised entities/brokers leads to a more centralised architecture.
- **Test Generation:** The authors in (Reetz *et al.*, 2012) present an approach for test generation using semantic descriptions which contain information for functional and non-functional testing. The limitation of the work is that there is no possibility of deriving test cases for cooperative services composed of distributed services. Moreover, the authors provide less information on the functional and non-functional description of services. In (Wacht and Trick, 2016), a comprehensive test description used for test generation in value-added service environments is presented. However, the approach leads to an enormous amount of generated test cases. It does also not provide the possibility of filtering relevant information on distributed system models to build a relevant test model used for the test generation process. Another drawback is missing information in the so-called Service Test Description. For instance, non-functional properties are not included, which could be used to generate security tests or figure out possible security concerns. The authors only mentioned non-functional properties within their framework without specifying the details. In (Kuroiwa *et al.*, 2019), only a few details are provided. Test cases have to be manually generated, which add more

complexity to the whole test process. The authors in (Pontes *et al.*, 2018) and (Ahmad *et al.*, 2016) neglected to provide more details on test description and test case generation.

- **Test Automation:** The authors in (Reetz *et al.*, 2012) proposed a test architecture used for semi-automated service testing. The authors in (Pontes *et al.*, 2018) claim to provide automated testing but lacks full automation by requesting manual generation of test configurations. The work in (Amalfitano *et al.*, 2017) contains some interesting points, such as the so-called in the loop test phases and the test automation. In (Rosenkranz *et al.*, 2015), the test cases have to be defined individually by each tester by themselves. Similarly, in (Kuroiwa *et al.*, 2019), test cases have to be manually generated, adding more complexity to the whole test process.
- **Other identified limitations:** The limitation of the work in (Reetz *et al.*, 2012) is that they focus on testing IoT services running on centralised servers. Classical centralised service providers create the IoT services in (Reetz *et al.*, 2012) without the participation of end-users in the service creation process. The authors in (Ahmad *et al.*, 2016) do not support end-user-based M2M services. Some other approaches are not end-user friendly, and the testing activities, including the specification of the whole system, have to be done by specialised test developers (Leotta *et al.*, 2018; Kuroiwa *et al.*, 2019). These facts make the presented testing approaches useless in end-user based M2M environments.

3.2 Related Work on Current Trust Approaches in M2M/IoT

Decentralised M2M networks contain several distributed and heterogenous nodes connected P2P. Trust plays a very important role in enabling fairly collaboration between nodes. Based on trust relationships, many attacks can be mitigated in the system. Missing trustworthiness in the M2M ecosystem has a high impact on the overall security of the system, and the design of appropriate trust management systems considering the special nature of the M2M community is seen as a necessary countermeasure against trust limitations. The principles and importance of trust provisioning are highlighted in (ITU-T, 2016; 2017b).

The main focus of this research (as stated in chapter 2) is to enhance the overall trustworthiness in the M2M community and among the community members. Therefore, this section provides a selection of the most relevant and recent trust management approaches in M2M/IoT. The literature review shows that there are trust approaches proposed for general purposes (general systems) or for specific use cases in IoT (specific systems). In the following, a brief overview of each of the existing works is given, and identified limitations relevant for decentralised and end-user based M2M services are presented.

3.2.1 Trust Approaches for General M2M/IoT Systems

This section briefly describes different identified trust approaches used for general purposes in M2M/IoT systems and highlights the outcomes of the review. Figure 3.2 shows an overview of general characteristics derived from existing trust approaches and

to which of these approaches they belong. The outcomes of the review are presented at the end of this section.

The authors in (Mendoza and Kleinschmidt, 2015; 2016) propose a distributed trust management system where participating nodes locally assess the trust value of their neighbours. Moreover, the authors highlight the problem of nodes trying to harm the system by performing selective attacks or on-off attacks. Selective attacks are performed when a node tries to save its resources by providing services with minimal computational requirements. On-off attacks are attacks where nodes switch back and forth from good behaviour to bad behaviour to confuse the system and remain with good trust scores. Their approach aims to overcome these problems. The authors state that the trust evaluation should be made based on the context in which a service is provided and the resource capabilities of the service provider. The trust evaluation is separated into different phases and done based on direct experience information between nodes.

The authors in (Chen *et al.*, 2016) introduce a trust management approach focusing on Service Oriented Architecture (SOA)-based IoT systems. Moreover, they aim to provide a solution to support trustworthy service compositions. Therefore, they propose a distributed trust approach where every user of the network can do the evaluation and store trust information in high-end devices. Users are socially connected using a social network. Direct interaction experiences and recommendations are used for the trust evaluation process. For instance, a direct interaction experience is when a user performs ratings based on non-functional characteristics, such as response time, failure probability, or prices. In order to get trustful recommendations for indirect interactions, a user considers “social similarities” with other users. The social similarity metrics include

friendship, social contact, and the community of interest perspective. The authors in (Chen *et al.*, 2016) solve the problem of limited storage space of IoT devices by proposing a storage management system that only considers trust values of nodes "with the highest trust value and recent interacting nodes as these nodes are most likely to share common interests".

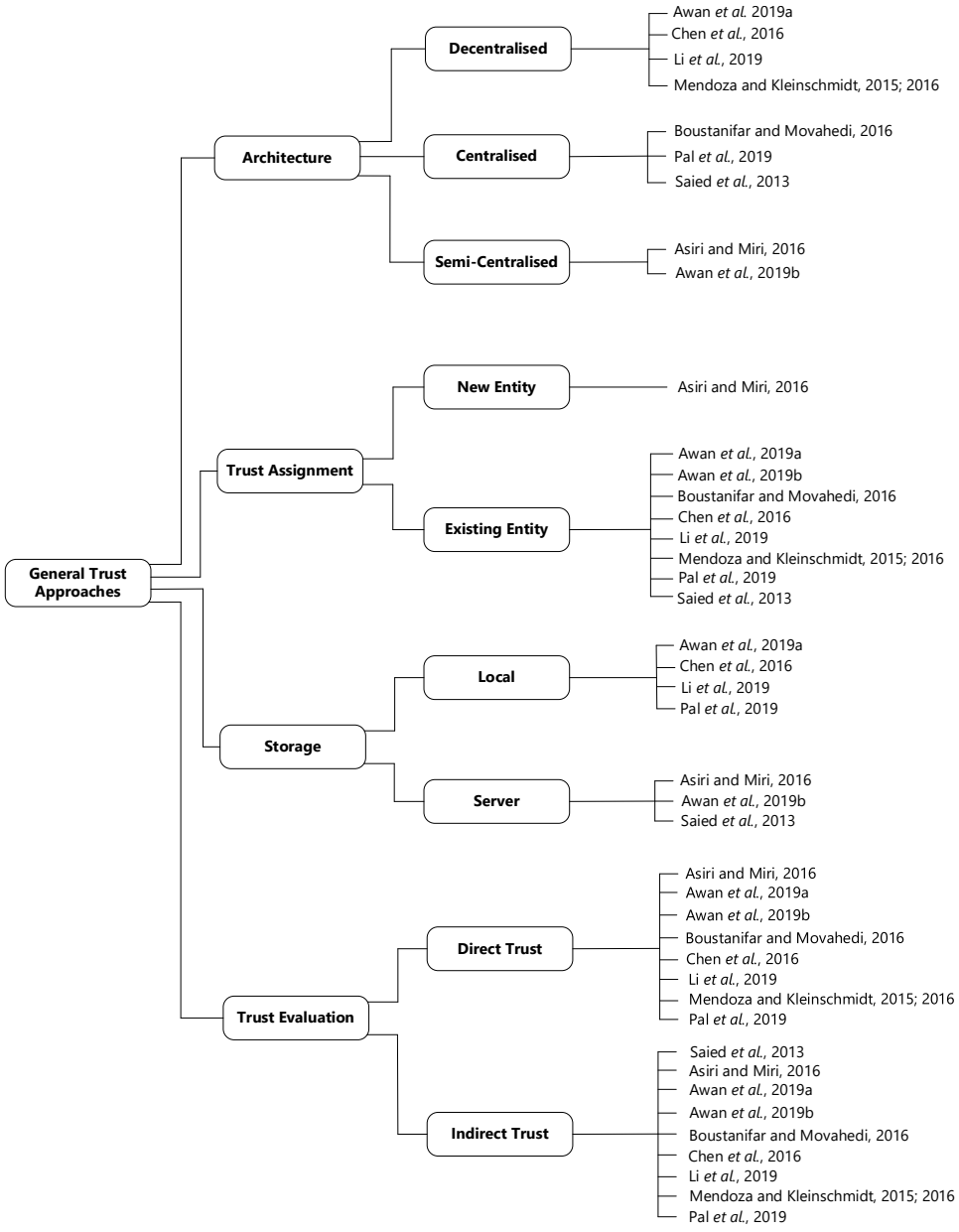

Figure 3.2: General characteristics of existing trust approaches

In order to handle several attacks in IoT and to detect malicious behaviour in the system, the authors in (Asiri and Miri, 2016) introduce a distributed and recommender-based trust management system. To distinguish trustworthy nodes from malicious ones, the authors integrate a probabilistic neural network. The peers themselves do the trust evaluation process. The presented trust model separates the nodes in the network into so-called alpha nodes and normal nodes. Alpha nodes are nodes with a high level of "computation, communication and other available resources" and configured "once at the model's setup". These nodes act as supernodes maintaining the whole trust evaluation process by "assigning jobs and distributing processing among" other nodes. Nodes are rating their completed transactions with each other and store this information in their rating tables. The trust score of a node is then calculated from these entries (taking into account the corresponding weighting). New nodes without previous interactions will get an average rating score.

Considering the different IoT characteristics, the authors in (Saied *et al.*, 2013) propose a centralised trust management system where different trust management servers are responsible for specific geographical locations. They evaluate the trust scores of IoT nodes based on the history of past behaviours. The system assumes that all peers are trustworthy from the beginning. The trust level is assessed using recommendations and the quality of recommendations. The trust evaluation process is separated into several operation steps. Reports will be used to rate the trust level of other nodes. In order to fill the gap of missing ideal reports, the authors propose to calculate the context similarity with a focus on the type of service and node capabilities. Further, the significance of the reports is computed to determine the most relevant reports for the trust evaluation process.

The only work explicitly considering the notion of M2M in connection to trust management systems is found in (Boustanifar and Movahedi, 2016). They propose a trust management system for mobile M2M communications, aiming to decrease the overhead of communications between nodes and reduce energy consumption. The authors identify the problem of malicious nodes trying to break the functionality of the system by increasing the job completion time and energy consumption. The presented trust approach considers the interaction history between the nodes and the quality of service they provide. A reinforcement learning algorithm is introduced to reward/punish nodes based on their interactions. All the nodes are considered to have the same initial trust score at the beginning. Tasks are finished by the initiator node or, in the case of lower capabilities, by a neighbouring node. The difference between the "local execution time" and "remote execution time" of a task is used to rate the interaction between two nodes.

The authors in (Awan *et al.*, 2019b) propose a trust management model for different IoT domains based on multilevel central authorities. Therefore, the IoT domains are divided into different communities and roles based on the similarities and interests of the participating IoT nodes. Thus, a dedicated community server is responsible for calculating the trust score of IoT nodes inside the community. The IoT nodes do not have the capability to store trust information. The domain server deals with coordination issues among the communities, other domain servers, and the trust server. As the main point of the proposed architecture, the trust server is responsible for domain trust evaluation and the generation of trust certificates. The trust score calculation of IoT nodes includes direct and indirect trust. Direct trust is evaluated based on the estimation of compatibility, honesty, and competence of IoT nodes. Indirect trust is evaluated based on recommendations from other interacting nodes.

Another trust approach is presented in (Awan *et al.*, 2019a), where devices are made capable of distributing trust among other nodes. The trust score is aggregated from direct observations and recommendations. A decentralised architecture is used for the trust evaluation process. Every node computes the trust score about another node (trustee) locally. Trust is classified into three levels. Level one is the knowledge of a node towards another node and is conducted by computing compatibility, integrity, and feedback. Level two is the reputation of the interacting nodes based on honesty, reliability, and cooperativeness. For each interaction (level three), the experience of the nodes is assessed based on recommendations, competence, and credibility parameters. Moreover, past information is used to assess all parameters and scale the evaluation based on trustee performance. The overall trust score is conducted by using the summation function on the results of the direct observations and recommendations. To overcome storage limitations in IoT devices, the authors propose a storage strategy where only scores about past experiences are stored for the future.

The authors in (Pal *et al.*, 2019) propose a trust management framework that tries to improve the access control scheme, including decision-making processes in IoT systems. The system model is separated into different domains consisting of device managers and user devices or service providing devices. Each domain consists of a so-called trust authority (trust manager and evidence database) used for trust value calculation and data storage. Direct, recommended, and derived trust are used to evaluate the trustworthiness of entities. These trust types are used in the trust evaluation process for a given context, category, and relationship. Furthermore, the total derived trust considers all contexts, categories, and relationships between two entities. Trust is represented in (Pal *et al.*, 2019) as an opinion metric, where belief, disbelief, and uncertainty are used as parameters for

given trust relationships. Opinions between entities are expressed through positive, negative, and uncertain experiences one entity has with another one. For the trust score calculation, the authors propose to use subjective logic and highlight its benefits for modelling and analysing uncertainty in IoT systems.

The authors in (Li *et al.*, 2019) present a trust management system for distributed IoT applications considering different contexts in which a service is requested. Direct and indirect observations are used to evaluate the trust score of an entity. Direct observations have a higher importance in the trust evaluation and are only supported by recommendations when direct information is limited. First, direct and indirect observations for a specific service are collected. Afterwards, an entropy-based trust evaluation with a focus on a target context is started. Based on the trust evaluation outcomes, the decision for proceeding or not the transaction is made. Each participating device stores trust information about its contacts locally. To conduct context-aware computations, the authors introduce the concepts of weighted-average context and context distance. Also, they propose to enable context-based trust estimation to predict the trust score of a service/service provider based on past trust information for a given context.

The review of the approaches mentioned above has conducted the following outcomes:

- **Architecture:** Centralised, semi-centralised and decentralised approaches are presented for trust management and evaluation. For instance, the authors in (Saied *et al.*, 2013) design a centralised trust management system with servers covering trust activities in specific geographical places. The centralisation level is improved in (Awan *et al.*, 2019b) and (Asiri and Miri, 2016). Here, supernodes

are performing different trust activities. Other reviewed approaches (Mendoza and Kleinschmidt, 2015; 2016; Chen *et al.*, 2016; Awan *et al.*, 2019a; Li *et al.*, 2019) use the benefits of decentralised architectures. They propose different approaches to distribute the different trust tasks among the network. These tasks are conducted locally by every participating node and, in some cases, only for direct neighbours (Chen *et al.*, 2016).

- **Trust evaluation:** In general, direct and indirect trust information (as also described in (ITU-T, 2017b)) is used for the trust evaluation process. The differences of the approaches consist of the level of priority for either direct or indirect trust and for the specific attributes used. It has to be pointed out that some of the reviewed approaches did not specify the attributes used for direct or indirect trust information. For instance, the authors in (Li *et al.*, 2019) only define a higher importance for direct interactions, but do not provide more details. The approach in (Saied *et al.*, 2013) utilises only recommendations and their quality to assess indirect trust. In summary, the reviewed approaches are not providing a trust evaluation scheme covering different aspects of a service or node.
- **Trust assignment:** Except of the approach in (Asiri and Miri, 2016), all others focus on getting the current trust scores of already deployed services and nodes in the network. They are missing to provide trust details for new services and peers joining the community.
- **Storage:** Some rely on centralised storage systems (Saied *et al.*, 2013; Awan *et al.*, 2019b; Pal *et al.*, 2019), others on decentralised ones (Chen *et al.*, 2016; Li *et al.*, 2019). For instance, in (Saied *et al.*, 2013) and in (Awan *et al.*, 2019b), all trust information is stored by the responsible cluster trust managers. Other

approaches, such as in (Chen *et al.*, 2016; Li *et al.*, 2019), prefer to store relevant trust information locally at each participating node. However, only the authors in (Awan *et al.*, 2019a) consider a storage strategy to handle storage limitations in IoT devices. Further efforts to improve the storage systems in existing approaches are not present, and the integrity of trust data, whether stored centralised or decentralised, is almost neglected.

3.2.2 Trust Approaches for Specific M2M/IoT Systems

This section briefly describes different identified trust approaches used for specific purposes in M2M/IoT systems and highlights the outcomes of the review. Figure 3.3 shows an overview of general characteristics derived from existing trust approaches and to which of these approaches they belong. The outcomes of the review are presented at the end of this section.

The authors in (Benkerrou *et al.*, 2016) present an honest- and credit-based trust management system to enable trust relationships in cooperating hierarchical IoT systems. The IoT environment consists of community managers (supervision role) and other normal nodes (service requester and service provider). The normal nodes do not have an initial trust score. The community manager acting as trust manager, forwards the requests among the nodes and coordinates their collaborations. Credit is used to start a collaboration between two nodes. For instance, a service requester pays for a service and this amount is refunded by the provider based on a specific agreement. After using the service, the requester will rate the provider based on its own experience. Then, a recommendation is sent to the community manager, who evaluates the credibility of the

recommendation based on previous honesty values and the credibility of the collaborator's rating. Finally, these trust parameters are used to evaluate the trust level of a node in the network.

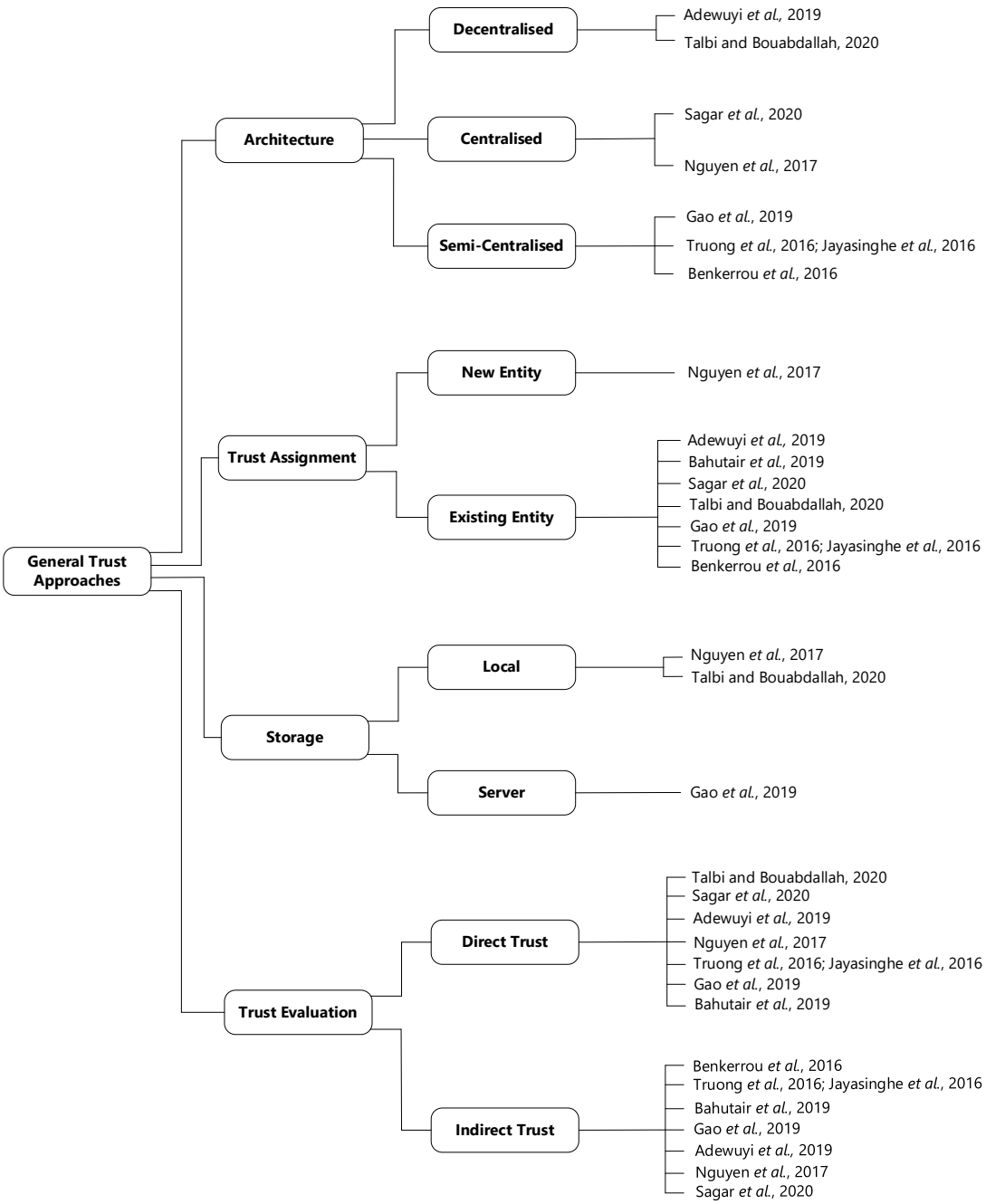

Figure 3.3: General characteristics for specific trust approaches in M2M/IoT

The authors in (Nguyen *et al.*, 2017) introduce a trust model for assessing the trust level of new IoT devices before interactions with others happen. Their trust model is applied in a personal space IoT system consisting of implanted and wearable devices and smartphones. To determine the initial trust level of a new device, the authors in (Nguyen *et al.*, 2017) use the challenge-response mechanism. This mechanism evaluates the uncertainty level of a device based on its behaviour and stores these results in the tested device. The personal space controller performs different challenges on the new device to evaluate its behaviour and permit admission into the community. After every challenge, the controller compares the response of the device with its expected behaviour and assesses the result of the current challenge. Afterwards, all challenge results are used to form the uncertainty level of the device via information entropy. Then, the initial trust level is computed using the uncertainty level.

The authors in (Adewuyi *et al.*, 2019) suggest a dynamic trust model for collaborative applications in IoT, where collaborations are carried out between unknown nodes (also in terms of trust) to increase the efficiency of a specific task. The authors state that trust parameters should also be based on the contextual functional properties of a service. These properties support determining whether a service provider is reliable and provides a good service. They introduce a trust model covering different objective and subjective trust properties used for the trust assessment. The objective properties include transaction speed (network speed), reliability, rate of work, proximity, cost of a service, and stake in the collaboration. The subjective properties include honesty, cooperativeness, or friendliness. The weighting of the different parameters is done dynamically based on the current subjective opinion of each node (acting as trustor). It can be adjusted at any time

during a session of interactions. In order to get the overall trust score of a node, the authors propose to use a weighted sum function for the trust aggregation process.

An adaptive approach for trust evaluation of crowdsourced IoT services is presented in (Bahutair *et al.*, 2019). The trust evaluation process consists of four stages used to assign a service trust score based on consumer usage. The service trust score is called adaptive trust. The assessment process uses service-related and usage-related data. The first stage of the trust assessment framework consists of detecting factors affecting the trust score of a specific IoT service. The authors suggest using rating data (from previous interactions) for different services of the same type for factor detection. In the second stage, a model is created that predicts the trustworthiness of a service for each factor. Stage three focuses on creating a model that can detect necessary factors for a given usage. The trust factors used for the trust score evaluation can vary for any different service type. The trust score of the service is obtained in stage four, where the results from the second and third stages are aggregated to get the final result.

The authors in (Sagar *et al.*, 2020) introduce an efficient trust model aiming to isolate misbehaving nodes in the Social Internet of Things (SIoT). Therefore, trust computation is done based on direct observations and indirect recommendations. The direct trust metric includes direct observations focusing on friendship similarity, the community of interest, cooperativeness, and reward/punishment. The friendship similarity is measured based on the interactions among participating objects and their importance regarding a task or context. The community of interest attribute is measured based on the similarity of nodes in terms of community interest. The cooperativeness attributes are measured based on the level of cooperativeness between two nodes. The measurement is done by

checking the balance of their interactions. The reward/punishment attribute is used to upgrade or downgrade nodes based on their behaviour. Regarding indirect trust and the associated reputation value, the authors proposed requesting this value only from nodes having at least one friend in common between trustor and trustee. The results of direct and indirect trust are used to compute the overall trust score of a node.

The authors in (Talbi and Bouabdallah, 2020) propose an interest-based trust management system for Social Internet of Things, where IoT nodes are evaluated based on the interest preferences of entities. The idea of the approach is to optimise the level of cooperation between various IoT entities. Similar to other reviewed approaches, trust is evaluated based on direct and indirect experiences. The direct experience values are conducted based on past interactions between two nodes. Each type of interest preference of a trustor is considered in the trust evaluation and is used to compute the global direct trust score of the trustee. The trustor subjectively weights each type of interest. In contrast to them, indirect trust is only evaluated if two nodes do not have past interactions with each other. This process involves other nodes providing recommendation regarding a trustee. To increase the relevance of indirect recommendations, the authors suggest considering the trust relationship between the trustor and the recommender as well as the similarity of the interest preference between them. Trust is modelled using beta distribution.

Another trust approach for SIoT is presented in (Truong *et al.*, 2016) and (Jayasinghe *et al.*, 2016). The authors introduce a trust model based on knowledge, reputation, and recommendations. Specifically, the knowledge trust metric is considered as direct trust and consists of first-party information from the trustor about the trustee based on personal requirements (and specific attributes). On the other side, reputation and recommendations

are used to assess the indirect trust. They are conducted by third-party information. Recommendations are trust opinions from surrounding or direct nodes. Reputations are trust opinions from others (global opinion). For calculating the overall trust score, the authors propose the weighted sum method using a personalised multi-criteria utility theory-based mechanism. They also clarify, that upon specific use cases, other relevant trust aggregation methods can be incorporated.

An appropriate trust approach for devices in service-oriented IoT edge environments is presented in (Gao *et al.*, 2019). In this approach, the IoT edge devices evaluate trust, which is categorised into the so-called capability trust, direct trust, and indirect trust from feedback. The capability trust is conducted based on the service request of the consumer and the resource status of the service providing device. Service trust can include the following parameters: serviceable duration, CPU computation power, energy maintenance capability, the time cost of each service unit, storage space size, and transmission bandwidth. Direct trust is conducted based on historical interactions (service tasks performed) between two entities. To face up with misbehaving devices, the authors propose a punishment system for unsuccessful interactions. Finally, the indirect trust is conducted based on the feedback of other entities for the evaluated IoT entity. The overall trust score is computed from the minimum of the capability trust score and the weighted sum of direct and indirect trust score.

The review of the approaches mentioned above has conducted the following outcomes.

- **Architecture:** The trust management and evaluation systems introduced in (Adewuyi *et al.*, 2019; Gao *et al.*, 2019; Talbi and Bouabdallah, 2020) are using a completely decentralised architecture, where individual nodes are responsible

for computing and storing the trust scores of others. The authors in (Benkerrou *et al.*, 2016; Jayasinghe *et al.*, 2016; Truong *et al.*, 2016; Gao *et al.*, 2019) rely on semi-centralised solutions. For instance, in (Benkerrou *et al.*, 2016) and (Gao *et al.*, 2019), different community managers or edge servers are used for the trust activities. A centralised approach is presented in (Nguyen *et al.*, 2017), where a centralised controller evaluates the trust of new devices in a personal environment.

- **Trust evaluation:** In (Benkerrou *et al.*, 2016), recommendations from normal nodes are used to evaluate the trust score of service providers. However, it is not enough to consider only recommendations ignoring the performance and the availability of the services. Instead, the direct interaction between the trust manager and the service provider should also play a role in the trust evaluation. Other works use both direct and indirect trust metrics to evaluate the trust score of entities. However, the authors in (Sagar *et al.*, 2020), for instance, propose to use indirect trust information only from nodes having at least one single friend in common between trustor and trustee. A different view on the usage of direct and indirect trust is presented in (Talbi and Bouabdallah, 2020). Here, the authors state that indirect trust evaluation is only done if two nodes do not have past interactions with each other. The authors in (Jayasinghe *et al.*, 2016; Truong *et al.*, 2016) distinguish indirect trust information into recommendations (trust opinions from surrounding or direct nodes) and reputations (trust opinions from others (global opinion)). A general problem of existing approaches is that there are no specific details on the trust metric parameters.
- **Trust assignment:** The problem of evaluating the trust score of new entities is an overlooked issue. Only the work in (Nguyen *et al.*, 2017) provides a challenge-

response process to assess the trust score of new IoT devices. However, there are no details about the executed challenges and how these challenges are derived from the controller. Additionally, the challenge-response and the uncertainty level trust metric parameters are not enough to measure the trust level of an entity.

- **Storage:** The authors in (Nguyen *et al.*, 2017) consider storing the evaluated trust score on the already tested devices, which do not provide a secure storage environment for others. In (Talbi and Bouabdallah, 2020), trust information is stored locally, and there is no global view of all nodes. Next to them, edge servers are used in (Gao *et al.*, 2019) to store the trust data of the evaluated entities.

3.3 Requirements for a New Framework for Trust Optimisation of Decentralised M2M Services

The research and development activities, described in section 3.1 and 3.2, provide an overview for testing M2M/IoT services and evaluating/managing trust between M2M/IoT entities and services. The previous sections also identified the main limitations of existing works. None of the reviewed approaches contributes to an overall framework that could test M2M services after their deployment and evaluate their behaviour to compute their trust score. This chapter aims to derive the main requirements for a framework providing test and trust functionalities. The requirements that will be defined in this section represent the basis for the proposed framework. They are classified in general, test-based, and trust-based requirements. General requirements are derived from the initial conditions/requirements and general problems regarding decentralised and end-user

based M2M services introduced in 2.4. Test- and trust-based requirements are defined based on the highlights and limitations of previously introduced approaches (refer to section 3.1 and 3.2). Test-based requirements consist of requirements necessary to do functional testing of decentralised and end-user-based M2M services. Trust-based requirements are important for designing a trust system that covers different trust activities, evaluates the behaviour of services/service providers, and securely shares the trust values in the M2M community.

The increasing number of smart devices and the deployment of many services in the community offer several advantages (as mentioned in section 2.2). They also come with limitations, such as services that are not working correctly or not at all. The high amount of services could lead to serious security attacks where trust plays an important role in mitigating them. Therefore, as mentioned at the beginning of this section, it is crucial to have a flexible framework that could test M2M services immediately after their deployment and while in operation. The framework should also evaluate the trust score of M2M services/service providers to classify them for further provisioning in different categories. The literature review shows that there are solutions either for testing services (refer to section 3.1) or for evaluating trust of services (refer to section 3.2). No overall framework is presented in the literature. Another important feature is the level of decentralisation. The M2M service provision concept presented in section 2.3 does not contain any central instances. The whole process for using or providing a service, the composition of services, and the storage of information is done using a decentralised architecture. Therefore, it is important to provide a test and trust framework that considers the same decentralisation level and works with the same principles to benefit from the

different advantages mentioned in chapter 2. Next to them, the framework should provide the ability to consider not only traditional M2M services but also decentralised and end-user-made ones. One of the general conditions of this work is that end-users are actively involved in the M2M community and provide and use services. Regardless of their less technical knowledge, they should address different issues (service provision, service testing, trust evaluation). In order to ensure a reliable working M2M community, it is necessary to integrate the end-users in testing and trust evaluation activities. Moreover, they should also be integrated for storage purposes, as a lot of data is in circulation. Another aspect is how the different end-users are motivated to participate in the different M2M community tasks. Passive or not-well behaving activities should be punished. During the service selection and composition process, end-users configure and select different services based on their preferences. This selection leads to the combination of several individual services to create a cooperative M2M service. It could be that different peers provide the same services and that randomly selecting one of these services is not secure. Furthermore, it could lead to the problem of selecting services provided by insecure or untrusted peers. This problem could result in an unstable and unreliable composed service. Therefore, the trust score of the service should be used to select services and perform service compositions. Based on these aspects, the following general requirements are derived:

- Test and trust support - Framework should provide the possibility to test and evaluate the functional behaviour of M2M services. Moreover, the framework should ensure trust among the M2M services/service providers.

- Decentralised architecture - Framework should avoid centralised management and control by implementing the whole testing and trust processes without central instances.
- Support decentralised and end-user-based M2M services - Support of M2M services with the following features: a) decentralised b) provided by end-users with less technical knowledge.
- End-user-integration – Framework should enable end-users to actively participate in testing, trust evaluation, and data storage activities.
- End-user-friendly – Framework should consider the low knowledge level of an end-user and provide end-user-friendly functionalities.
- Incentivisation mechanism – Framework should support incentivisation mechanisms to motivate participating nodes to improve how they act in the M2M community (from passive to an active node or from untrustworthy to trustworthy).
- Trust-based service selection and composition – Framework should consider the trust score of M2M services in the selection/composition phase.

As mentioned above, none of the presented approaches provides an overall solution, including testing and trust evaluation of decentralised M2M services. Regarding the decentralised capability, only the trust projects presented in (Mendoza and Kleinschmidt, 2015; 2016; Chen *et al.*, 2016; Awan *et al.*, 2019a; Li *et al.*, 2019; Pal *et al.*, 2019) support a fully decentralised architecture avoiding centralised authorities and problems of a single point of failure. Other approaches evaluated in this research are using semi-centralised or centralised architectures. Almost all of the evaluated works do not integrate or involve the end-user in any community activity. Only a few approaches consider slightly or partly

some side-support from end-users for testing or trust activities. Moreover, only the work in (Pontes *et al.*, 2018) claims to provide an end-user-friendly platform for testing purposes. Another important requirement is the ability of the platform to compose services with each other based on their trust scores. Only the approach introduced in (Chen *et al.*, 2016) considers the evaluated trust scores of the services for service composition. Reward/punishment systems are not present in the reviewed testing approaches. Some of the trust approaches utilise incentivisation systems to increase the number of trustworthy participation in the community (Saied *et al.*, 2013; Mendoza and Kleinschmidt, 2015; 2016; Boustanifar and Movahedi, 2016; Gao *et al.*, 2019; Sagar *et al.*, 2020; Talbi and Bouabdallah, 2020).

The service provider could be a user with no technical background who causes faulty or malicious services in the community. For ensuring correct working services, it is required to check the functional correctness of new services. The M2M service should be tested in the deployment phase and throughout its lifetime in the community. Thus, the framework should ensure that services are tested at the initial state and continuously. The integration of the end-user in the testing process is unavoidable. However, often end-users are not able to easily perform service testing. Another important element is the test description which is required for the test generation process. The M2M services provide information such as service descriptions and interface descriptions. These descriptions should be used to create a suitable test description. The test description should contain all the relevant information needed to generate test cases. Furthermore, the test case generation process should automatically produce efficient test cases to perform functional testing of M2M services. As there is an end-user-based M2M community, it would be necessary to

automate several steps in the workflow, including the test processes. Thus, the framework should support the automation of creating and executing tests as it simplifies the testing process for end-users in the community. Based on these aspects, the following test-based requirements are derived:

- Functional testing of new services - The functional behaviour of new provided M2M Services should be tested.
- Functional testing of existing services – The functional behaviour of existing M2M services should be tested.
- Test automation – The framework should support the automation of different testing steps, starting from definition to execution.
- Test description of M2M services - A suitable general description of M2M services should be provided for test case generation.
- Test case generation - Test cases for functional testing should be automatically generated using the available service information.

The reviewed testing projects support functional testing of newly provided services. However, none of them satisfies the requirement of testing services continuously after they are part of the service marketplace. The automation of different testing steps is supported in (Ahmad *et al.*, 2016; Wacht and Trick, 2016; Amalfitano *et al.*, 2017; Leotta *et al.*, 2018; Malik *et al.*, 2019a). Thus, the generation/execution of test cases and the preparation of the test result reports are mostly automated. Regarding the formal description of M2M services, the work in (Wacht and Trick, 2016) provides an interesting approach with a so-called Service Test Description. However, this description has missing information regarding security-related questions which could be used for trust

evaluation. The approach in (Reetz *et al.*, 2012) also includes semantic models in the service description. This description is used to generate test cases considering functional and non-functional properties. Nonetheless, they do not describe in detail the description definition procedure and the test case generation.

As stated in chapter 1, one aim of this research is to evaluate the behaviour of decentralised M2M services and to compute from these evaluation trust scores. These scores could be used to mitigate attacks in the community by banning services and peers with low trust scores from the community. Most of the approaches presented in section 3.2 provide solutions for evaluating the trust score of existing services. They exclude initial trust scores for newly provided services or new service providers and open the doors for malicious entities from outside. For ensuring trust from the beginning, the initial trust score of the service must be considered and evaluated. An initial trust score enables other peers to figure out faster trustworthiness among peers and services. The overall framework should support both trust scores for new M2M services and existing ones. The initial trust score should play an important role in determining the reliability position of the service in the community. The trust evaluation process will generate a significant amount of trust data among peers. Therefore, the framework has to provide a mechanism for securing the storage of trust data. Moreover, it should ensure the trustworthiness of services/service providers and the trustworthiness of trust data. Another important aspect is the trust model and the metrics used for trust evaluation. Most of the presented trust metrics in the literature are for a specific application field and do not consider the characteristics of decentralised M2M services. Moreover, the presented trust metrics do not support or are not applicable for evaluating trust for new services/service providers.

Finally, trust approaches are exposed to various trust attacks in the M2M community. These attacks aim to falsify the trust information about participating M2M entities. The framework should provide prevention techniques in order to be resistant to these attacks. Based on these aspects, the following trust-based requirements are derived:

- Initial trust score – It should be possible to determine the trust level for newly provided services/peers.
- Ongoing trust score - It should be possible to determine the trust level for existing services/peers.
- Secure trust data storage - Trust data determined by trust evaluations and sent between the peers should be stored securely among the nodes.
- Trust model completeness – The trust model should support different aspects for the trust evaluation of M2M entities.
- Trust attack resilience – The trust approach should be resilient against different trust attacks.

Most of the trust management approaches in M2M/ IoT presented in this chapter do not provide any possibility for evaluating the trust of new services/service providers. Only the authors in (Asiri and Miri, 2016) and in (Nguyen *et al.*, 2017) propose methods for initial trust assessment of new services/devices. However, the average rating method proposed in (Asiri and Miri, 2016) does not consider the characteristics or the behaviour of a new node. The missing information on how challenges are derived for the challenge-response process used for initial trust evaluation and the centralised controller for performing challenges on the device are drawbacks of the approach presented in (Nguyen *et al.*, 2017). Most trust approaches do not provide or consider any solution for a secure

data storage system of trust-related data. The authors in (Chen *et al.*, 2016) try to solve the storage management problem by considering only nodes with good trust values and high impact in the community. However, it is still possible to falsify trust information on the different storage nodes. Regarding suitable trust metrics used for trust evaluation, the works in (Ma and Wang, 2016) and (Nakahira *et al.*, 2015) provide interesting trust parameters that could also be reused for the framework presented in this research. However, they are not sufficient and should be supported by additional trust parameters. The requirement for a complete trust model is only satisfied by the work in (Adewuyi *et al.*, 2019). The authors also consider the contextual functional properties in the trust evaluation process. Moreover, the authors propose different objective and subjective trust properties for the trust process. Finally, none of the reviewed approaches satisfies the requirement for a resilient trust approach against different attacks.

The following tables summarise the requirements specified above, including the evaluation of the related work regarding these requirements. Table 3.1 contains general- and test-based requirements. It shows the evaluation of the testing approaches presented in section 3.1. Table 3.2 and Table 3.3 show the evaluation for existing trust approaches. They contain general- and trust-based requirements. Requirements the related work satisfy are marked with "+"; requirements not satisfied by the related work are marked with "-"; requirements the projects partially satisfy are marked with "o". Evaluations that cannot be determined based on the published information about the related works are marked with "/".

The different specified requirements will serve in chapter 4 to propose a novel framework for trust evaluation and functional testing of decentralised M2M services. Chapter 5 and 6 will then explain the underlying concept of the framework.

Table 3.1: Evaluation of testing approaches based on derived requirements

Requirements		Functional and Non-Functional Testing Approaches									
		(Reetz *et al.*, 2012)	(Pontes *et al.*, 2018)	(Leotta *et al.*, 2018)	(Kuroiwa *et al.*, 2019)	(Bosmans *et al.*, 2019)	(Amalfitano *et al.*, 2017)	(Wacht and Trick, 2016)	(Ahmad *et al.*, 2016)	(Rosenkranz *et al.*, 2015)	(Malik *et al.*, 2019a)
General	Test and trust support	-	-	-	-	-	-	-	-	-	-
	Decentralised architecture	-	/	/	-	/	/	-	o	o	/
	Support decentralised and end-user-based M2M services	-	-	-	-	-	-	-	-	-	-
	End-user-integration	-	o	-	-	-	-	-	o	-	-
	End-user-friendly	-	+	-	-	-	-	-	o	-	-
	Incentivisation mechanism	-	-	-	-	-	-	-	-	-	-
	Trust-based service composition	-	-	-	-	-	-	-	-	-	-
Test-based	Functional testing of new services	+	+	+	+	+	+	+	-	-	-
	Functional testing of existing services	-	-	-	-	-	-	-	-	-	-
	Test description	o	-	o	o	-	o	o	o	/	/
	Test case generation	o	-	o	o	-	o	o	o	/	/
	Test automation	o	o	+	-	/	+	+	+	-	+

Table 3.2: Evaluation of general trust approaches based on derived requirements

Requirements		General Trust Approaches								
		(Awan *et al.*, 2019b)	(Awan *et al.*, 2019a)	(Pal *et al.*, 2019)	(Li *et al.*, 2019)	(Chen *et al.*, 2016)	(Mendoza and Kleinschmidt, 2015; 2016)	(Asiri and Miri, 2016)	(Saied *et al.*, 2013)	(Boustanifar and Movahedi, 2016)
General	Test and trust support	-	-	-	-	-	-	-	-	-
	Decentralised architecture	o	+	-	+	+	+	o	-	-
	Support decentralised and end-user-based M2M services	-	-	-	-	o	-	-	-	-
	End-user-integration	-	-	-	-	o	o	o	-	-
	End-user-friendly	-	-	-	-	-	-	-	-	-
	Incentivisation mechanism	-	-	-	o	-	+	-	+	+
	Trust-based service composition	-	-	-	-	+	-	-	-	-
Trust-based	Initial trust score	-	-	-	-	-	-	o	-	-
	Ongoing trust score	+	+	+	+	+	+	+	+	+
	Trust model completeness	o	o	o	-	o	-	-	-	-
	Secure trust data storage	-	-	-	-	o	-	-	-	-
	Trust attack resilience	-	-	-	-	-	-	-	-	-

Table 3.3: Evaluation of specific trust approaches based on derived requirements

Requirements		Specific Trust Approaches							
		(Adewuyi *et al.*, 2019)	(Bahutair *et al.*, 2019)	(Sagar *et al.*, 2020)	(Talbi and Bouabdallah, 2020)	(Gao *et al.*, 2019)	(Jayasinghe *et al.*, 2016; Truong *et al.*, 2016)	(Benkerrou *et al.*, 2016)	(Nguyen *et al.*, 2017)
General	Test and trust support	-	-	-	-	-	-	-	-
	Decentralised architecture	+	/	-	+	o	o	o	-
	Support decentralised and end-user-based M2M services	o	o	o	o	o	o	-	-
	End-user-integration	-	o	-	o	o	-	-	-
	End-user-friendly	-	o	o	-	-	-	-	-
	Incentivisation mechanism	-	-	+	+	+	-	-	-
	Trust-based service composition	-	-	-	-	-	-	-	-
Trust-based	Initial trust score	-	-	-	-	-	-	-	o
	Ongoing trust score	+	+	+	+	+	+	+	-
	Trust model completeness	+	-	o	-	o	o	-	-
	Secure trust data storage	-	/	/	-	-	-	-	-
	Trust attack resilience	-	-	-	-	-	-	-	-

3.4 Conclusion

This chapter presented a literature review in the fields of testing and trust. Sections 3.1 and 3.2 reviewed several existing test and trust approaches in M2M/IoT and identified their highlights and limitations. The literature review in 3.1 has shown that functional testing in M2M/IoT has been mostly overlooked, and only a few approaches are available. To enrich the level of knowledge regarding M2M/IoT testing, section 3.1 also included

non-functional testing approaches in the related work. Section 3.2 selected the most relevant trust approaches used in M2M/IoT systems and classified them based on their use case scenario in general and specific approaches. Section 3.3 summarised the findings from 3.1 and 3.2. Moreover, initial conditions and requirements from section 2.4, together with highlights of existing test and trust approaches, are used to derive requirements relevant for this research thesis. The identified requirements are classified in general, test, and trust-based requirements and are used to evaluate existing approaches.

The overall review concluded that none of the existing approaches enables both testing and trust evaluation of M2M services. Moreover, most of the approaches are not suitable to be used for end-user-based and decentralised M2M communities neither provide an end-user-oriented platform. Besides them, they neglect a completely decentralised architecture and miss to use trust or incentivisation methods to increase the level of reliability in the community.

The outcomes of section 3.1 regarding the test-based requirements showed that most of the existing testing approaches cannot test ongoing M2M services. Their focus is more on testing services before their deployment, not continuously. Other drawbacks are identified regarding the service descriptions used for test case generation and the test case generation process. Existing approaches provide less information or inefficient solutions regarding these processes.

The review results regarding the trust-based requirements in section 3.2 concluded that none of the trust approaches considers the initial trust score of new M2M services or new M2M service providers in the community. Another identified drawback is the insecure storage possibilities provided by existing works. Evaluated trust scores are not securely

stored, and the integrity of trust data is not ensured. The presented trust models are not complete considering only one or few trust aspects of a service/service provider for the trust evaluation process. The resilience against different trust attacks is also very low in existing approaches.

The evaluation results and the identified requirements are used as major criteria for the proposed framework in the upcoming chapter.

4 Proposed Framework for Trust Evaluation and Functional Testing of Decentralised M2M Services

Chapter 2 highlighted the need to optimise the overall trustworthiness in decentralised M2M communities. Chapter 3 identified several limitations on existing trust and testing approaches used in the M2M/IoT field. As a result of the performed review, this research thesis proposes a novel framework for trust evaluation and corresponding trust evaluation. First, this chapter starts with the description of the decentralised M2M ecosystem and the definition of the general proposed concept (section 4.1). Then, section 4.2 introduces the overall framework architecture and its main components. Sections 4.3 and 4.4 highlight the methodologies and main elements used to perform trust evaluation and corresponding functional testing of decentralised M2M services.

Some parts in this chapter have been published in (Shala *et al.*, 2018; 2019c; 2020a).

4.1 General Concept

End-user-based and decentralised M2M service provision is very promising (as described in section 2.3). Its aim is to make local resources available to external users and create new competitive services. Moreover, it aims to achieve independence from central service providers and ascend the role of the end-user to a new dimension. Thus, it tends to enhance competition in the M2M/IoT marketplace and provide more service variety to

the community. However, as identified in sections 2.4 and 3.4, the concept for decentralised service provision does not consider several issues that lead to different limitations in terms of service functionality and trustworthiness in the whole M2M community. The proposed framework in this research thesis aims to overcome these limitations and to optimise the overall trust.

Figure 4.1 shows a fully decentralised M2M ecosystem (based on section 2.3), which serves as a basis in this research work. It consists of a decentralised P2P network with many end-users (peers) acting as service providers or service consumers. Every end-user can manage (1) M2M devices available in their personal environments and design/configure easily M2M services themselves. Moreover, they can provide (2) the functionality of local M2M devices to other end-users as a service. New services are announced and registered by storing (3) their service descriptions in the P2P overlay network. Other end-users can retrieve (4) existing descriptions and subscribe to a service to consume it (5). The whole workflow is implemented without centralised entities (central service providers) or centralised execution environments for M2M. Therefore, every end-user can use their own devices such as routers, smartphones, or notebooks as local execution environments, fulfilling the hardware requirements to act as execution systems for service provider activities. The end-users also can cooperate in order to create complex M2M services (service composition).

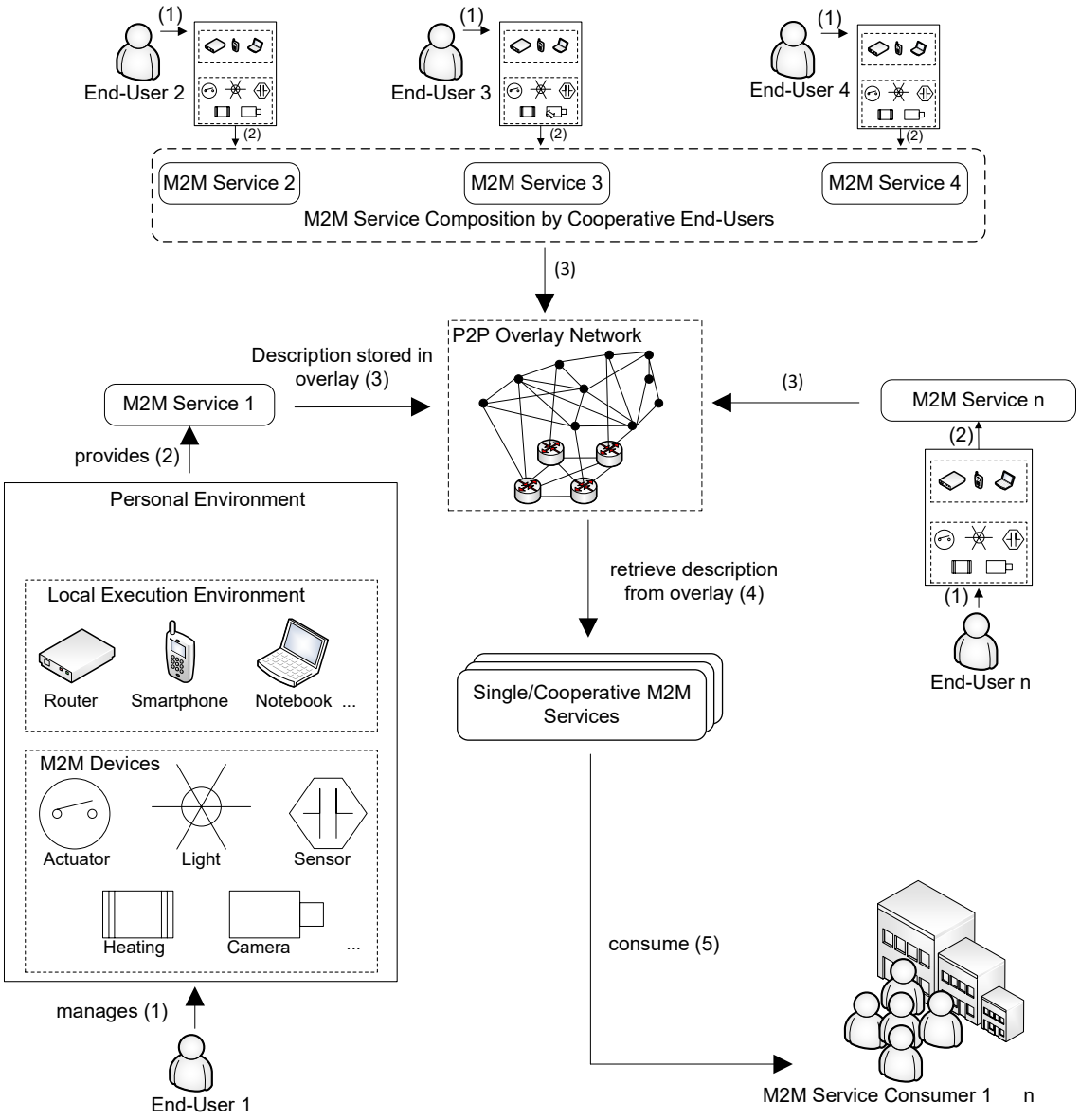

Figure 4.1: Decentralised M2M ecosystem (Shala *et al.*, 2020a)

This research identifies the following characteristics of end-user-based M2M service provision in the decentralised M2M ecosystem:

- End-user service provision without technical knowledge: Every end-user part of the M2M community can participate in service provision activities. They can use an end-user-friendly GUI to design, configure, and deploy M2M services easily. There are end-users in the M2M community with less technical knowledge. They

can also unintentionally or intentionally design, configure, or deploy malfunctioning services. Moreover, it could be that they do not have any skills regarding service and test development.

- Decentralised system architecture and P2P network: There is no centralised entity in the M2M service provision architecture and no central stakeholder involved in different processes. All participating end-user nodes are connected via a P2P network. This network is used for information exchange and data management.
- Single and cooperative M2M services: Every end-user can provide single M2M services to others. It is also possible to combine different M2M services to create a cooperative M2M service.
- Independent service logic description and service interface description: The M2M service logic is described using a formal description language. The description enables the service execution independently of the used execution environment. There is also an M2M service interface description to make services available to other end-users outside the local area.
- No trust at all: The whole workflow of the M2M service provision and the M2M community members do not include any trust elements. For instance, trust is not used to select service providers or to combine individual services.

Together with limitations of existing trust evaluation and service testing approaches (refer to chapter 3), these characteristics form the basis of this research work.

The general concept in this research work considers the end-user as part of trust and test activities. These activities can be done without expert knowledge and with minimum

effort. End-user nodes and the services they provide have their own trust scores. They also maintain all the trust information and use them for building trust relationships.

The concept also includes a decentralised architecture for trust evaluation and functional testing. In line with the decentralised service provision approach, every node of the network participates in different community tasks. Thus, all nodes build together a decentralised network and share service, test, and trust information.

The service provision process produces single services and cooperative services. Their functionality could change during operation. Thus, an M2M service may have a different functional behaviour after its creation and a different one while operating. The proposed concept considers this fact and supports the testing at different operation phases. The same idea is also applied to trust activities. The trust score of new services is essential to support consumer decisions on using or not using a service. Furthermore, the trust score of well-established services should be monitored to identify misbehaving trends. This research project proposes to maintain the trust evolution of M2M services. Moreover, the trust score determination of end-user nodes is also part of the trust concept.

In general, there are no trust relationships between the different entities (end-user nodes, services) in the M2M community. Therefore, this research thesis proposes to integrate trust activities among the nodes. Every participating node should have its trust score. Moreover, if a node is acting as a service provider, its services should also be evaluated. Therefore, a service provider does have a node trust score component and a service trust score component. Both together result in the total trust score of the service provider. The concept also considers evaluating trust for new and existing M2M entities.

The trust evaluation and functional testing activities are the two main parts of the framework proposed in this thesis. Moreover, both activities are interconnected with each other, building a feedback system. Functional testing of services is essential to ensure that new and existing M2M services continuously provide the right functionality. Thus, this research proposes integrating testing within the trust evaluation system and using the functional behaviour of services as a trust attribute (among others). The total trust score of a service provider node results from the aggregation of the different trust scores obtained by evaluating the service provider and its services. Related work has also shown that most of the existing approaches do not provide a secure way to store trust scores and other related trust data. This thesis describes the utilisation of blockchain technologies in order to enable a high level of data integrity.

This research thesis interprets the concept of trust as a value for measuring the reliability/correctness of different working M2M services and entities. The goal of the framework is to establish trust between the different M2M entities up to the level where one entity does what it claims to do and does it in a reliable way. This research does not cover issues with identity management, authentication, or access control. It assumes that nodes have an immutable identity tied to a public/key pair. Furthermore, it does not consider hacker activities and security-related attacks on the system. It assumes a network of end-users, where inadvertent or unaware mistakes can happen. Overall, the proposed framework includes the following features:

- Service reliability: A service performs what it is setting to do based on the service information and nothing else. Any kind of service malfunctioning or service misinformation is due to mistakes by the end-user.

- Tamper-proof data storage: Service and trust data is securely stored using blockchain technology. All the community members can rely on the integrity of the data as there is no possibility to modify or remove them. With the blockchain, one kind of manipulation in the system is no longer possible. Other manipulation attacks, such as modifying the data through communication and storing false entries in the blockchain, are out of the scope of this thesis.
- Community maintenance: All community members are actively engaged to maintain the decentralised M2M community. Any node is involved in various community activities.
- Trust-related attacks: Attacks, such as bad-mouthing or ballot-stuffing (ITU-T, 2017b), aiming to harm the trust system by providing false trust information are avoided. The proposed framework can protect against attacks where untrusted nodes provide good trust scores for bad nodes or bad trust scores for good nodes. It also protects against the quick trust score devaluation of particular nodes. The framework effectively guarantees no short-term or instant actions that could result in mistrust in the network. However, the proposed trust approach does not protect against attacks performed by more than 51% of trustworthy nodes. It also does not protect against long term data poisoning, where untrustworthy nodes change for a long time their behaviour from untrustworthy to trustworthy.
- Security-related attacks: As mentioned, this research focuses only on trust-related attacks explicitly mentioned in the thesis. It also protects against inadvertent node failures. However, any other type of attacks, such as denial of service, eavesdropping, long term data poisoning, or spoofing, is out of the scope of this research thesis.

4.2 Overall Framework Architecture and Components

Section 4.1 described the general concept for the framework for trust evaluation and corresponding functional testing of decentralised M2M services. Figure 4.2 shows the general architecture of the proposed framework and its different components. In general, the framework consists of the following parts: Local M2M Environment, M2M Community, P2P Storage Engine, Test Engine, and Trust Engine. The architecture shown in Figure 4.2 represents the local perspective of the workflow. There are local and external M2M entities. Both are part of an M2M community and are connected P2P. Every peer is locally operating Test and Trust engines, which perform the corresponding test and trust activities. All M2M entities use the same decentralised storage system to store service, test, and trust information. The remainder of this section will provide a brief description of each component.

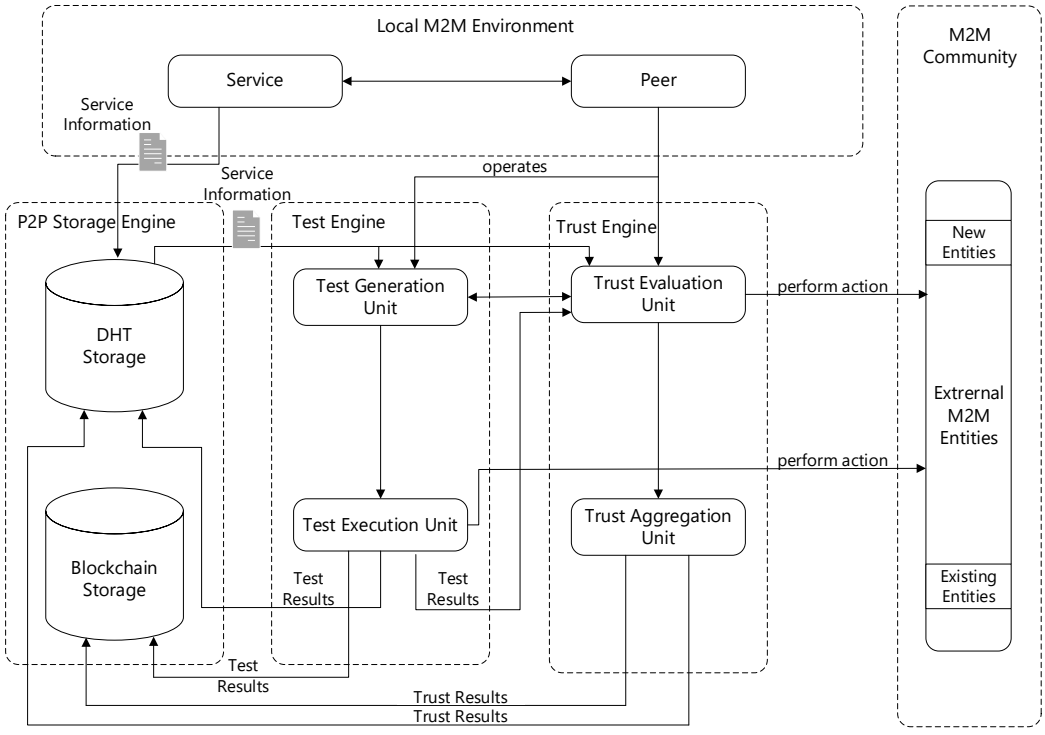

Figure 4.2: Test and trust framework architecture

Local M2M Environment

A peer is equivalent to an end-user in terms of service provision and utilisation. Every peer can provide one or more services to other end-users. Both peers and services are considered to be M2M entities. Each M2M service has its service information which can contain service description and service interface description. These descriptions are stored in the P2P storage. Every peer can act as a test and trust agent by operating the Test and Trust Engine. These engines are covering the activities of functional testing and trust evaluation. The local M2M environment is part of the global M2M community, where other M2M entities with similar roles are present.

M2M Community

All participating peers and the different services they are providing are part of the M2M community (refer to section 2.3). This community is fully decentralised and is operating as a P2P network. Moreover, it enables social networking functionality for all participants. Every interested entity can join the P2P network.

The M2M community consists of new and existing M2M entities. Their joining status is relevant for the test and trust activities. After a new entity joins the community, all participants will receive a joining notification. Then, other community members will test these new services before their deployment. Moreover, they evaluate the initial trust score of the new M2M services. Test and trust information about new entities supports other entities in their decisions to use corresponding M2M services. Besides them, the M2M community also continuously updates the test and trust information of existing M2M entities.

Test Engine

Every peer part of the M2M community can perform functional testing of new and existing M2M services. The Test Engine covers all relevant test steps required to evaluate the functional behaviour of services and consists of the Test Generation Unit and the Test Execution Unit. Both units provide automated operations and require less end-user action. The Test Generation Unit receives from the P2P storage relevant information about an M2M service. Based on this information, test cases are generated and afterwards executed on the SUT. The Test Execution Unit produces the test outcomes and sends them to the P2P storage and the Trust Engine for further processing.

The Test Engine is also interconnected with the Trust Engine. Testing activities can be used to check the correctness of the service functionality. On the other side, testing activities can be incorporated within the trust activities. Here, the outcomes of the tests serve as input for the trust evaluation.

Trust Engine

Like the Test Engine, every participating peer can evaluate the trust score of other peers or services. Moreover, the evaluation includes new and existing M2M entities. The Trust Engine consists of the Trust Evaluation Unit and the Trust Aggregation Unit. The Trust Evaluation Unit covers the various operations used to assess other entities based on several trust attributes. Moreover, it sends the outcomes of these operations to the Trust Aggregation Unit, which uses a dynamic weighting system to compute the trust score of M2M entities. Afterwards, these scores are sent to the P2P storage. The upcoming sections provide more details on the trust evaluation and aggregation.

P2P Storage Engine

The P2P Storage Engine consists of a DHT storage and a Blockchain storage. The DHT storage is used to store service descriptions, contact information of service providers, and test and trust evaluation results. The DHT architecture enables distributed storage functionalities and fast access to data. Additionally, the Blockchain storage is incorporated to store relevant data and enhance data integrity in the network. The cryptographical principles of blockchain make its use for data storage very powerful. Data can be stored tamper-proof inside the blockchain. The Blockchain storage consists of data regarding service, test, and trust activities. The combination of DHT and Blockchain storage enables integrity check-ups. Thus, a peer can quickly query a specific datum from the DHT and get the same datum from the blockchain for comparison in case of doubts.

4.3 Workflow of Functional Testing

The previous section has introduced the Test Engine, dealing with the different activities. This section provides more details about the testing workflow. This research aims to provide an end-user friendly functional testing approach where various steps are performed automatically or require minimal effort from the end-user. The proposed approach for functional testing can be used for two contexts: service testing and trust evaluation.

Usually, the service development cycle includes the interaction between the service provider and the service consumer. Therefore, the consumer provides a non-technical and natural language description of a service that should be delivered. The description serves

as a basis for the experienced service developer, which can use a service creation environment (part of a traditional M2M service platform) to create a formal service description. The service implementation code is generated from this description, and the service is deployed on a server. Similar steps are also performed by the experienced test developer when doing service testing. The test developer gets an informal service description from the service consumer, which is used in further steps to generate appropriate test cases. These test cases are afterwards run against the deployed service. The service and test development processes are interconnected and follow an agile approach. Both service developer and test developer work together on providing a final product to the consumer based on its requirements.

This research project focuses on end-user-based environments where the end-user takes the responsibilities of a service provider. Moreover, the whole M2M community is a decentralised network of several independent M2M services and service providers. The service development lifecycle does not rely on centralised entities nor specialised developers. The end-user is natively involved in the service provision process. The provided services are not based on consumer requirements, but on the personal requirements of the service provider. Moreover, there is no tester responsible for test development. Thus, a normal service and test development lifecycle is not followed and guaranteed for end-user-based M2M service provision. Overall, reliable and correct working M2M services are not ensured in end-user-based service provision approaches. The testing process is not defined, and the role of an experienced tester is missing.

As depicted in section 2.3, the end-user-based M2M service provision starts with the Service Creation Environment (SCE), which is part of the personal environment of the

end-user. The SCE provides a GUI that enables the simple creation of a state machine representing the behaviour of the system. Every end-user can design services based on personal requirements. The defined service logic serves as a basis to generate the formal service description of that service. Afterwards, the SCE sends the service description and the interface description to the P2P storage, where all other participating peers have access to it. Moreover, the SCE deploys the M2M service to the execution environment of the corresponding service provider. Figure 4.3 shows the workflow of the service creation process.

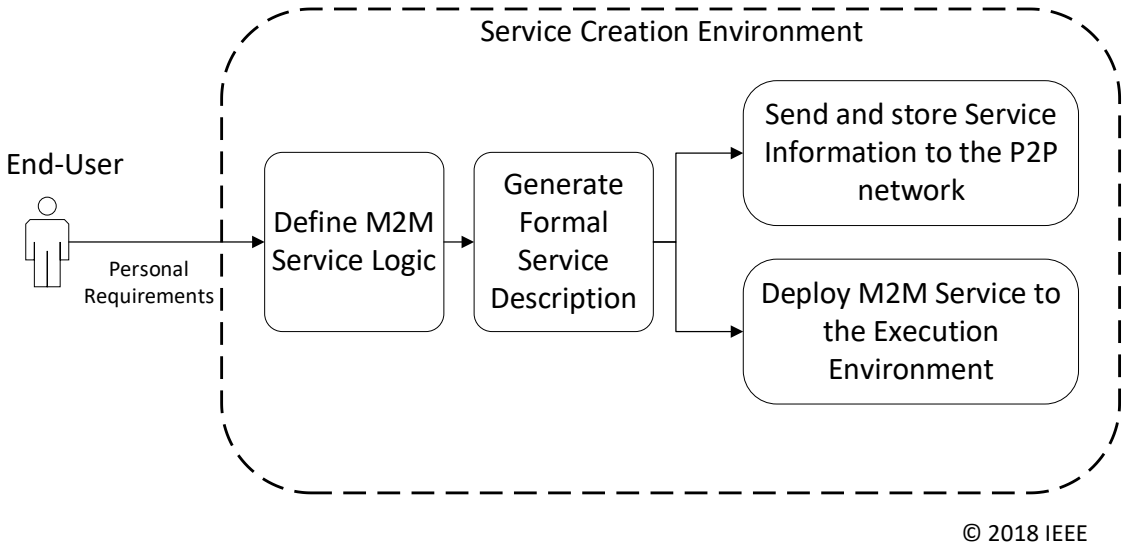

Figure 4.3: Service Creation Environment (Shala *et al.*, 2018)

The benefits of end-user integration, respectively, the enhancement of the end-user role in service provision processes make it necessary to define the role of the testing entity within the proposed framework. The challenge is to identify which entities are going to perform the necessary testing activities required to evaluate the functional behaviour of M2M services. Moreover, this research aims to use a fully decentralised environment (as stated in section 3.3). Tests in the M2M community should avoid the usage of centralised

coordinators. Therefore, this research proposes to integrate all participating end-users (peers) in the service testing process. Every end-user can test other M2M services and contribute to an excellent working M2M community. Figure 4.4 shows the integration of other end-users (acting as testers) in the testing process. A service provider makes a new M2M service available to the M2M community. Therefore, the service provider sends the service information (1) to the P2P storage, informing all the community members about the new service (2). Afterwards, every peer can individually decide to use the service information to perform tests (3) on the M2M service.

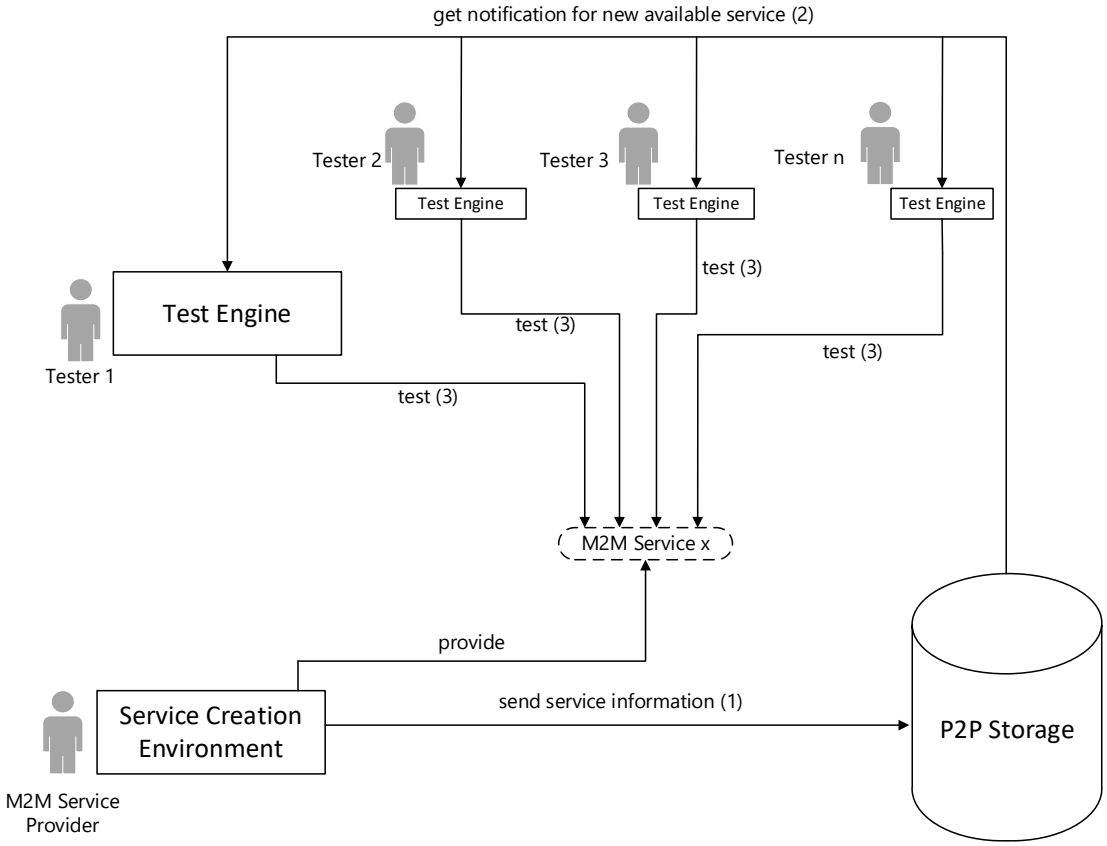

Figure 4.4. End-user integration for service testing

Adequate test suites are required to perform testing activities for each single service in the M2M community. Basically, every service provider should include alongside the

service information also executable tests and mocks, that can be used for test purposes. For testing composed M2M services, it is assumed that services are defined in such a way that a simple composition of individual test descriptions is possible.

In cases, when a service provider fails to provide the necessary test information to the testers, basic tests have to be done. These tests follow a simple testing workflow as shown in Figure 4.5. Therefore, the Test Generation Unit uses the service information got (1) from the P2P storage to identify (2) the SCXML-based system model (as defined in (Steinheimer *et al.*, 2019). This model already includes test data for performing simple tests based on the current configuration of the M2M service. It is used to create (3) an appropriate test model, which serves as an input (4) for the Test Generator to derive different test cases (instructions) for functional testing. Then, the test instructions are sent (5) to the Test Execution Unit. The Test Execution Unit receives the test instructions and automatically performs the test execution (6) on the SUT respectively on the M2M service. The execution outcomes are the test results containing a verdict whether the service has passed or failed the tests. Finally, the Test Execution Unit analyses the results and sends them to the P2P storage for sharing with other community member.

The evaluation of the proposed testing workflow is shown in chapter 7. It has to be pointed out that this research is covering basic service testing. The proposed testing concept enables end-users in the M2M community to invoke functional tests that would give them some increase confidence of the service works like it should. It is assumed that every service provider deploys the necessary test information alongside the service. A complete comprehensive test suite generation methodology for every function of a service is out of the scope.

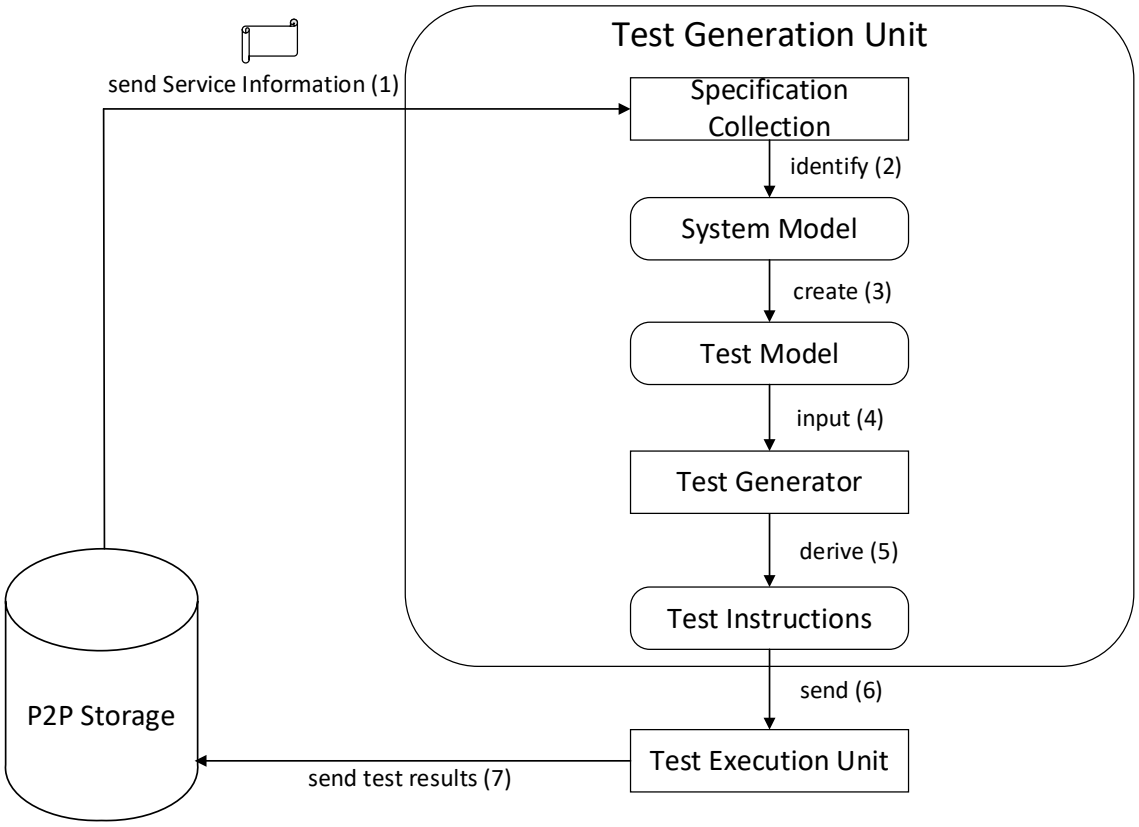

Figure 4.5. Service testing workflow

4.4 General Description of the Trust Model

Trust in decentralised M2M communities is the main element of this research work. Several existing trust approaches in M2M/IoT were evaluated in chapter 3. The result is that most of the publications do not consider the possibility to assign the right trust score for new M2M entities. Another limitation is that the several presented trust models cover only one or two aspects of a service or peer. They are not presenting an accurate global trust view on the M2M entity.

In order to overcome the limitations of existing trust approaches, this research project introduces a comprehensive trust evaluation model. The trust approach presented in this research thesis integrates aspects, such as service functionality, data integrity, and

community activities within the trust evaluation. Figure 4.6 presents the trust evaluation layer model with its different layers and components. The layer model should be seen from the perspective of the trust evaluator, beginning from the bottom layer.

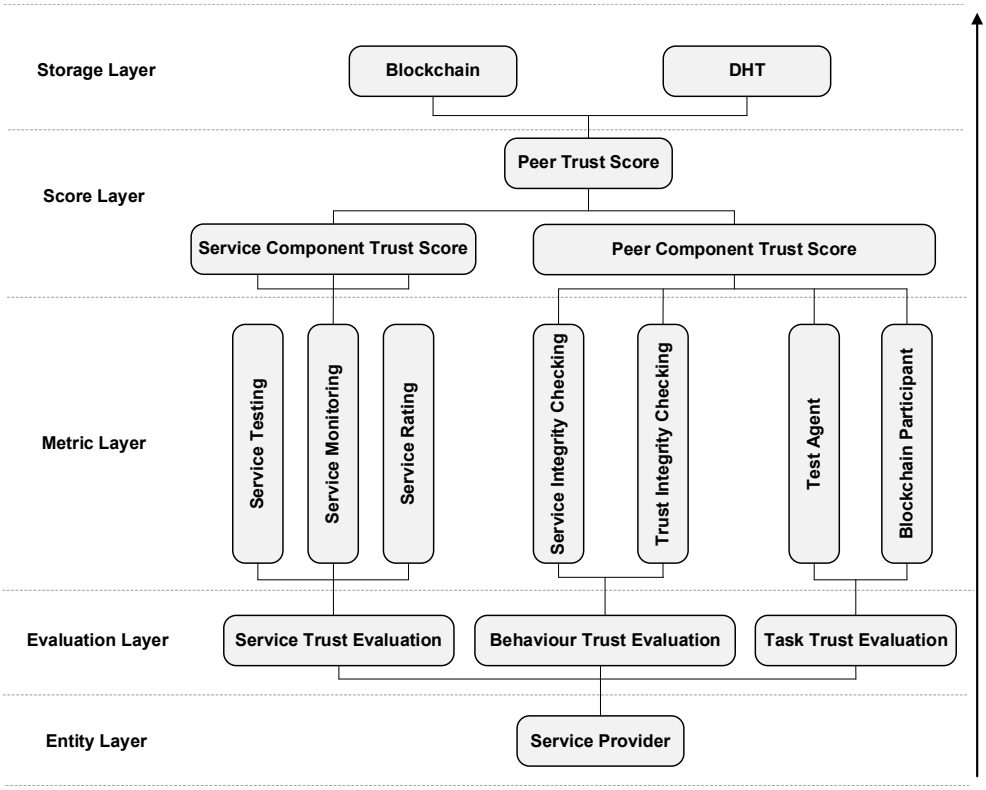

Figure 4.6: Trust evaluation model (Shala *et al.*, 2020a)

In the following, a short description of each layer starting from the Entity Layer is presented:

Entity Layer: The trust evaluation can be done by every peer part of the M2M community. Any peer can evaluate the trust score of another one.

Evaluation Layer: This research proposes three types of trust evaluations: service evaluation, behaviour evaluation, activity evaluation (participation in community tasks).

Metric Layer: The following metrics are used for service evaluation: service testing, service monitoring, and service rating. The following metrics are used for behaviour evaluation: checking the integrity of service and trust data. For activity evaluation, the following metrics are used: checking the participation of the end-user in different community activities such as service/peer testing or performing blockchain tasks.

Score Layer: The total Peer Trust Score of the end-user consists of the Service Component Trust Score and the Peer Component Trust Score. The Service Component is computed using the results from the Service Trust Evaluation. The Peer Component is computed from the behaviour and task trust evaluation. The trust evaluation concludes with a trust score ranging from 0 to 1 (worst to best), indicating the level of trustworthiness.

Storage Layer: For integrity reasons, the resulting trust scores are stored in the Blockchain and Distributed Hash Tables (DHT). The first provides a tamper-proof storage feature; the second one enables a fast lookup of data.

Figure 4.7 shows the workflow of the proposed trust evaluation approach in the M2M community. Every end-user (peer) can evaluate the trust score of other end-users or the services they provide. In order to perform the trust evaluation activities, an end-user can use different trust metrics. These metrics are used to cover several peer and service attributes. They are assigned to three evaluation categories defined as Service Trust Evaluation, Behaviour Trust Evaluation, and Task Trust Evaluation.

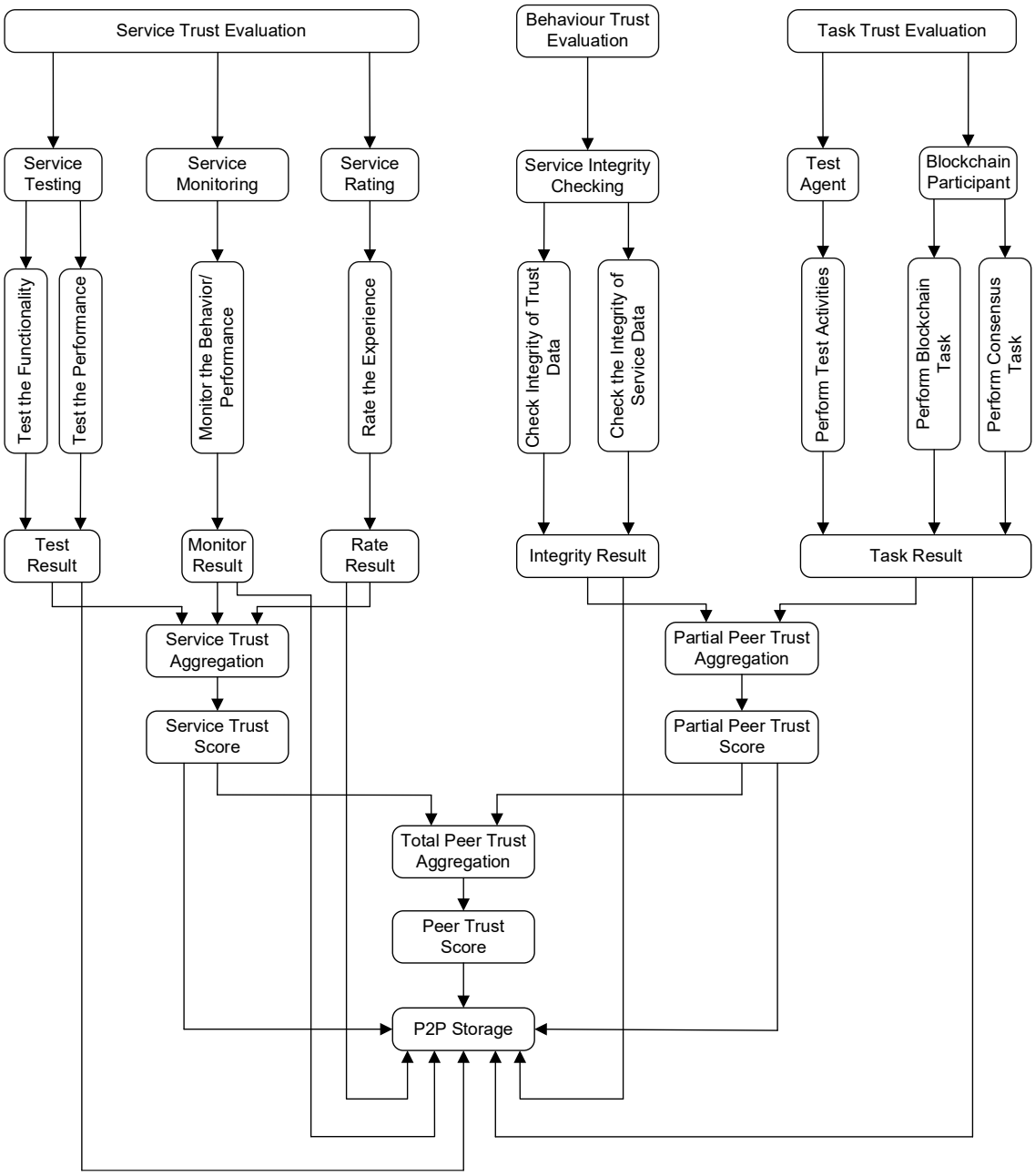

Figure 4.7: Trust evaluation workflow (Shala *et al.*, 2019c)

The Service Trust Evaluation category consists of Service Testing, Service Monitoring, and Service Rating. In Service Testing the functional behaviour and the performance of a service are evaluated. For instance, using functional testing, the service behaviour is tested and concludes whether the service is performing like it is mentioned in its service description or not. The functional testing activities are covered by the Test Engine

(introduced in the previous section). Next to functional testing, this research suggests using performance testing to confirm the service performance regarding different performance parameters (defined in the service description). Both are tasks within the Service Testing Metric. This metric is used for two purposes. First, to compute the initial trust score of new services provided by new or existing service providers. The initial trust score supports other end-users in their decisions to use or not the new service after publication to the M2M community. As conducted in chapter 3, existing trust approaches fail to provide realistic and objective trust information about newcomers. The second purpose of the Service Testing Metric is to be incorporated within the regular trust evaluation system, where the trust score of existing entities is evaluated. Two other trust metrics part of the Service Trust Evaluation category are service monitoring and rating. For the first one, the trust evaluating peer monitors the behaviour and the performance of a service by focusing, for instance, on the ratio of positive/negative responses or online/offline actions. The service rating considers the service experience of the end-user (acting as a service consumer) in relation to a service. The results of Service Testing, Service Monitoring, and Service Rating are used to calculate the Service Trust Score of the service provided by a peer.

Another trust metric of the proposed trust approach is the Service Integrity Metric (belongs to the category Behaviour Trust Evaluation), where data stored in the Blockchain and the DHT storage is compared with each other. Service and trust data is stored in both storage systems, and every end-user can check whether the data was tampered with or not. If data is changed, then the responsible peer gets at a lower trust score. As mentioned, this research aims to present a completely decentralised network where end-users are performing different activities. In order to incentivise end-users to do so, this research

proposes to integrate their activities (tasks) within the trust evaluation (category Task Trust Evaluation). Participation in different community tasks is considered as an additional trust metric of the proposed trust model. The history of participation in community activities is stored in the blockchain. Thus, any peer can be checked whether it was actively involved in different tasks or not. The community tasks include the trust evaluation activities or the maintenance of the blockchain (and its consensus mechanism). The results of the Behaviour Trust Evaluation and Task Trust Evaluation are used to compute the Partial Peer Trust Score of the peer. In order to compute the overall trust score of the peer, the Service Trust Scores of all services provided by the peer are combined with the Partial Peer Trust Score of the peer gained by behaviour evaluation and task participation.

The different trust evaluation activities are performed independently and in different time intervals by M2M community members. A corresponding result is conducted for each of the trust metrics (test, monitor, rate, integrity, and task result; see Figure 4.7). These metric evaluation results are stored in the blockchain and in DHT (both represent the P2P storage). Other community members use the information stored to evaluate the overall trust score of a given peer. It is also possible to evaluate only the overall trust score of an individual service provided by the peer. In order to get the overall trust score of a peer, all relevant evaluation results (performed through the different trust metrics) are collected from the blockchain or DHT. Then, these values pass through a trust aggregation function where dynamic weighting is used to calculate the most recent trust score of a peer. All trust-related data, whether it is an evaluation score from the metrics or a service/peer trust score, is stored in the P2P storage.

In order to store evaluation metric results and trust scores in the blockchain, they are included in blockchain transactions and sent throughout the network. One principle of blockchain is that a consensus mechanism is required to confirm blockchain transactions. However, existing consensus mechanisms suffer from limitations such as high energy consumption or unreliable elements in the internal workflow. Therefore, this research project proposes a novel trust consensus protocol optimising the overall trustworthiness of the blockchain. The blockchain is not only used for storage purposes. The proposed trust approach also uses blockchain principles to provide a decentralised network and support the consensus-achieving process in the M2M community. The outcomes of the ratings and trust evaluations have to be confirmed by all community members using the proposed consensus protocol. Thus, this research interconnects blockchain and trust. Both support each other for their corresponding purposes. The trust approach uses blockchain to enable high data integrity. Blockchain includes trust elements to increase the security level in a trustless environment. Chapter 6 provides more details on the blockchain-based elements of the proposed framework.

Features

The comprehensive trust model proposed in this research optimises the overall trustworthiness in a decentralised and end-user-based M2M community. The end-users aim to maintain a high trust score as it enables them to get easier access to other services and be more attractive as service providers to others. The benefits of the trust model can be illustrated with an example represented by a monitoring application in the neighbourhood. The application consists of different single services, such as temperature, camera, evaluation, alarm, or notification services, provided by individual end-users

within the local community. The proposed trust model ensures that the end-users continuously check their services in order to maintain a stable good trust score. Therefore, they need to verify whether their services provide the right functionality and performance as mentioned in the service description. They also receive feedback from others highlighting their satisfaction with the services. Furthermore, the end-users of the neighbourhood are motivated to participate in various activities, such as service provision, trust evaluation, and blockchain maintenance. A lack of support for the community leads to a lower trust score. New end-users joining the neighbourhood and willing to provide one of the requested services for the monitoring application are first checked for the functional correctness of these services. These checks serve as a basis to form an initial trust score which mitigates the risk of unreliable new services/service providers joining the community. The blockchain elements of the trust model ensure that end-users can rely on the integrity of the stored data. Overall, the proposed trust model mainly intends to protect against inadvertent node failures or end-user faults that affect the reliability of services in a decentralised M2M community.

4.5 Conclusion

This chapter introduced a novel framework for trust evaluation and corresponding functional testing of decentralised M2M services.

First, section 4.1 introduced the preconditions used to define an overall concept. These include the decentralised nature of the M2M community and the inclusion of end-users in the service provision processes. Preconditions are also the status of M2M services, whether they are new to the community or already operating in it, and the service

descriptions used. Finally, the whole environment without trust elements plays a crucial role in defining the proposed trust concept. The overall concept congregates these preconditions and forms the basis for the definition of the proposed framework.

Then, section 4.2 described the framework architecture and its different components where the test and trust engine are highlighted. The test engine covers all relevant test activities to assess the functional behaviour of services. The trust engine handles the trust evaluation of services and peers using different trust metrics. It also handles the trust aggregation, leading to an overall trust score for a specific M2M entity. Both engines are operated by a corresponding peer and are supported by the storage engine for storage purposes.

Section 4.3 introduced the functional testing part by outlining the methodology. It starts with the service provision process, where the end-user designs a service and deploys it to the M2M community. The methodology continues with the test generation unit, which uses service relevant information to form a test model. This model is used to generate test cases which afterwards are executed through the test execution unit.

The trust model and workflow of the trust evaluation are presented in section 4.4. Different trust metrics and corresponding attributes are used to assess the trust score of M2M entities. The outcomes of these activities are aggregated and stored in the blockchain for integrity reasons. Blockchain principles are used to maintain a decentralised network and achieve trust-based consensus-building among the M2M community members.

Following this overview of the proposed framework, chapter 5 provides more details on the blockchain incorporation for several activities within the M2M community. It also presents the integration of service testing for trust evaluation purposes. Chapter 6 describes the main part of this research thesis, the blockchain-based trust evaluation system with its trust model and different metrics. Furthermore, the overall trust evaluation workflow and the trust aggregation process are explained.

5 Blockchain and Service Testing for Trust Optimisation

The previous chapter proposed a novel approach for trust evaluation and functional testing of decentralised M2M services. Furthermore, the methodology for testing the functional behaviour of services and the comprehensive trust model covering several aspects has been introduced. Chapter 4 also highlighted that both testing and trust are fixed components within the same system and are interdependent. This chapter covers the parts of the framework dealing with service testing and evaluating their initial trust score.

Section 5.1 introduces blockchain technology in order to ensure data integrity in the M2M community. Additionally, several existing trust approaches using blockchain within M2M are reviewed, and their limitations highlighted. Section 5.1 also identifies that consensus protocols used within the blockchain technology play a crucial role when considering them for integration into the M2M environment. This section assesses existing blockchain-based consensus protocols and proposes a novel trust-based consensus protocol.

This research aims to present a trust management system that covers not only the trust evaluation for existing M2M services but also that for new ones. Chapter 3 has highlighted the limitation of existing trust approaches regarding the initial trust score problem for new M2M entities. Section 5.2 continues with a literature review of approaches in other application domains dealing with the initial trust score problem. The review concludes with identified highlights and limitations of existing works. Based on

these results, section 5.2 presents a novel concept for trust evaluation of new M2M services by merging the functional testing process with the trust process. Moreover, the testing process is extended through the integration of performance testing. Section 5.2 concludes with the workflow for building the initial trust score of new M2M services.

It has to be pointed out that the concept of trust in this research is also interpreted as a value for measuring the reliability and correctness of different working M2M services. These services are provided by different peers and used in several composed M2M services. As any peer can provide many services, it can be considered that the total trust score consists partly of the trust scores of the services it provides. Therefore, this research primarily focuses on trust evaluation based on service trust scores. The upcoming chapters will also include the so-called peer trust score.

Some parts in this chapter have been published in (Shala *et al.*, 2018; 2019a; 2019b; 2019c; 2020a).

5.1 Blockchain for Trust Management Optimisation

Building trust relationships within an M2M community is essential to ensure a secure and decentralised environment. Existing trust approaches in the literature provide interesting points regarding trust evaluation. However, none of the reviewed works in chapter 3 is considering the integrity of conducted trust information. A trust evaluation system aims to provide more trust in a network. However, if the trust information, respectively, the trust scores resulting from the evaluation are not securely stored, how can we trust the trust evaluation system?

The Distributed Ledger Technology (DLT)/blockchain technology gained popular attention by providing benefits of data immutability and process automation in open or closed communities without the need for centralised entities. The authors in (ITU-T, 2017a) state that DLTs provide favourable features, such as non-reversible/modifiable data entries, privacy and security capabilities, automated data synchronisation, and transparency. Detailed information on DLT/blockchain technology can be found in (ITU-T, 2017a; 2019d; 2019a; 2019b; 2019c).

The usage of cryptographical principles and consensus mechanism are the key factors for the strengthens of blockchains. This research thesis takes advantage of blockchain technology and provides a concept which enables trustworthy management of trust data within the M2M community. The proposed approach enables a high level of data integrity and overcomes the limitations of decentralised networks.

5.1.1 Blockchain-Based Trust Approaches in M2M

The benefits of using blockchain technology to enhance the privacy, security, and trust level within a system are acknowledged by researchers of different disciplines such as in (Gattermayer and Tvrdik, 2017; Alowayed *et al.*, 2018; Goka and Shigeno, 2018; Strobel and Dorigo, 2018; Kandah *et al.*, 2019; Yang *et al.*, 2019). Recently, some works are also conducted within the field of trust management in IoT. This research thesis has identified the most relevant trust approaches and presents their highlights and limitations in the following. The outcomes of the review are summarised at the end of this section.

The authors in (Di Pietro *et al.*, 2018) propose a blockchain-based trust system in an IoT community with different IoT domains. Initially, the authors introduce a platform called

the obligation chain, which operates a distributed credit-like system and a reputation mechanism. The system enables service providers to handle service consumer obligations in terms of accepting or declining them. The signed obligations are stored in the obligation chain, which is a distributed ledger. Obligations are signed outside the blockchain. To protect against malicious nodes, the authors propose to use so-called proof of commitments for service providers and proof of fulfilments for service consumers, which are using the information regarding obligations and fulfilments stored in the distributed ledger. The service provider and consumer cooperate by exchanging terms of use (created by service providers) and obligations (defined by consumers). The authors combine their obligation chain with the standard bitcoin blockchain to assess the credibility of the obligation issuer. On top of the obligation chain, a three-way handshaking protocol is suggested to bridge trust between different domains. This protocol consists of the setup, spend, and fulfilling phase, using information stored in the distributed ledger to handle service handlings between service providers and service consumers. Reputation scores of service consumers are stored in the distributed ledger and, if required, evaluated locally by every service provider based on the average obligations fulfilled on time by the consumer.

Another approach is presented in (Lahbib *et al.*, 2019), where the system relies on a network model with different manufacturing zones. These zones consist of physical resources such as IoT devices, an authenticator acting as an authorisation entity, a trust manager managing and evaluating the trustworthiness of the zone members, and miners collecting trust information in a block, broadcasting them and verifying other blocks. The authors use direct observations of the packet delivery behaviour and recommendations from other nodes for trust evaluation. Specifically, the entities cooperativeness,

competence, community of interest, and credibility toward recommendations are used as trust metrics. The authors utilise a private blockchain called multichain for blockchain activities using a round-robin algorithm for approving transactions minimising complex computation resources. The trust computation of the trust manager also considers the experience scores computed from devices based on their communication with direct neighbours.

The authors in (Asiri and Miri, 2018) present a blockchain-based approach combined with smart contracts to evaluate the trust of IoT devices. Smart contracts are used to set endorsement policies for new transactions. Thus, after a transaction is proposed by a client (IoT device), other nodes called endorsing peers will evaluate them, leading to the acceptance or rejection of the transaction. Transactions with chain code are only accepted if the trust score of participants is high enough. Blockchain activities are done by peer nodes that are considered trustworthy. Only authorised clients can join the blockchain networks. Participants are assigned trust points that are updated based on rating resulting from interactions between two participants. Interactions between two nodes are enabled using smart contracts that check the trust points related to the thresholds. The balance of trust points is stored in the blockchain. The authors refer to trust evaluations in two cases. First, trust evaluation using packet delivery rate as trust indicator and performed directly after each interaction between two participants. Second, the trust evaluation of the blockchain nodes after a peer initiates a transaction proposal (seems more to be a trust checking smart contract).

The authors in (Malik *et al.*, 2019b) present a dynamic trust evaluation system that uses a consortium blockchain to track interactions among supply chain participants. The trust

score is assigned based on these transactions and has a special focus on this trust data. The following elements are stored in the off-chain: raw data, supply chain data (produced by sensor devices), trade events between entities, and regulatory endorsements. Their hash values are sent to the blockchain via transactions that trigger smart contracts to automatically calculate trust and reputation scores based on the provided data. The blockchain is used, among others, to store the hashes of the supply chain data and the digital profiles of all entities (containing the trust information). Smart contracts are also used to include quality assessments between the participants. These assessments support the incentivisation of participants to behave trustworthily. Entities without past reputation that join the network for the first time are assigned with a minimum trust score by default. The authors in (Malik *et al.*, 2019b) state that the overall trust score of an entity is calculated based on the overall reputation score and some other feature scores (e.g. consumer feedback). The weighting factors for the trust evaluation are determined by the business network administrator, who also manages the blockchain network and defines the business network model.

The authors in (Kouicem *et al.*, 2018) introduce a trust management architecture where the trust values of the service providers are stored in the blockchain. The system architecture proposed in (Kouicem *et al.*, 2018) consists of one layer with distributed IoT devices providing services to each other and a second layer with distributed fog nodes responsible for managing and controlling IoT objects. The fog nodes also maintain a blockchain used by the IoT devices to store trust information. The blockchain transactions are validated by the fog nodes using the Proof of Stake (PoS) Algorithm. The trust model used in (Kouicem *et al.*, 2018) to evaluate the trust level of IoT objects considers only honest IoT devices for reporting recommendations (based on the interaction experience)

about other IoT service providers. Interaction experience means the recommendation of an IoT device toward other IoT service providers regarding a used service. The transactions containing trust information are sent by the home fog node to other fog nodes part of the blockchain for validation. Information about available services provided by potential service providers is stored in a Distributed Hash Table.

The authors (Dedeoglu et al., 2019) propose a blockchain-based approach to improve end-to-end trust in different IoT applications. Therefore, they introduce a layered trust architecture for IoT blockchain, where trust is considered for data observation and blockchain validation. Data observation includes data from different sources, such as from IoT devices. The hash values of the data are stored off-chain and in the blockchain (via transactions) for integrity reasons. For trust management, the authors present a data trust module and a gateway reputation module. The data trust module evaluates the trustworthiness of observation data based on the behavioural information of the data source and related information from other sources. The reputation module provides reputation information about participants to the blockchain and the application layer. The presented trust model in (Dedeoglu *et al.*, 2019) relies mainly on the confidence of the observations taken between the nodes, the confidence of the data sources, the reputation of data sources, and the evidence taken from other observations. The gateway nodes participate in block generation, validation, and distributed consensus in the private blockchain network, where only nodes with permission can participate. No competition for block generation among them is required. Based on the information included in blockchain transactions, the gateway nodes (are selected periodically) calculate the evidence and the sensor reputations to assign trust values for the sensor observations. The generated block includes transactions containing observation data, public key and

signature of data sources, assigned trust value for the observation, and the updated reputation of the data source. Furthermore, the authors propose a reputation-based block validation by considering validations of data stored in the blockchain with data provided by nodes and the reputation scores of block generating nodes.

The authors in (Ali *et al.*, 2019) propose using a behaviour monitor system in IoT-blockchain infrastructures that can store IoT device data and classify normal or malicious behaviour based on these data. Their system model considers different declared IoT zones used for different IoT use cases. Each zone has its local blockchain network used to store all kinds of communications between devices in the form of blockchain transactions. Every zone has its master node selected based on the resource capability and used for main blockchain activities, such as creating new blocks. Moreover, the master node centrally processes all incoming and outgoing transactions to and from a zone. The authors propose a behaviour monitor for each zone (configured on the master node), which classifies the behaviour of every device and compute a level of trust in each zone using learning neural networks.

The authors in (Boussard *et al.*, 2019) present a blockchain-based trust evaluation for IoT devices (focusing on the home network), where reported histories stored in a blockchain are used to compute the trust scores for each class of devices. Initially, the authors propose to isolate groups of devices in network slices using their own defined home controller. Through a simplified risk assessment scale, users are able to assign desired trust levels to those isolated slices. The controllers then use this information to check if the devices are meeting the expectation of the user. Therefore, the current trust score of the devices is evaluated and compared to the expected trust level. The trust score is evaluated using the

proposed trust assessment system, which consists of the Terms of Use (TERMS). The TERMS consists of the properties and capabilities of a device designed by its manufacturer. Deviations and reports are also part of the trust assessment system. The first one defines the behaviours that do not follow the TERMS. The second one is the behaviour feedback monitoring done by network controllers of devices. These three elements (with crowd-sources nature) are stored in the blockchain and used to compute the trust scores of devices based on observed behaviour history. The authors also introduce an analyser element as a local trust assessment instance in home networks. They believe that global trust assessments are not suitable in environments with different policies and security or privacy requirements. The analysers examine the data from the blockchain and evaluate trust for the devices.

This research considered different requirements/elements to assess the presented trust approaches (Asiri and Miri, 2018; Di Pietro *et al.*, 2018; Kouicem *et al.*, 2018; Ali *et al.*, 2019; Boussard *et al.*, 2019; Dedeoglu *et al.*, 2019; Lahbib *et al.*, 2019; Malik *et al.*, 2019b). Some of the requirements have already been derived in chapter 3. The others are related to blockchain technology in combination with smart contracts. In the following, the blockchain-related elements are briefly outlined:

- Trust data storage: Are trust information stored in the blockchain (on-chain) or outside the blockchain (off-chain)?
- Blockchain type: Is the approach using a private or public blockchain?
- Blockchain operation mode: Is the blockchain operated closed (permissioned) or open (permissionless)?

- Consensus protocol: Which mechanism is used to ensure that the participating nodes agree on the same copy of the ledger?
- Smart contract: Does the approach use smart contracts to automate processes and avoid the usage of third parties?
- Smart contract use case: For which use case is the smart contract integrated?

Next to them, Table 5.1 and Table 5.2 give an overview of the characteristics of different trust approaches in relation to the used assessment elements.

Table 5.1: Review of blockchain-based trust approaches – part I (Shala *et al.*, 2020a)

Elements	Trust Approaches			
	(Di Pietro *et al.*, 2018)	(Lahbib *et al.*, 2019)	(Asiri and Miri, 2018)	(Malik *et al.*, 2019b)
Decentralisation	partially	partially	not	partially
End-user integration	no	no	no	no
Trust incentivisation	no	no	yes	yes
Data storage type	locally, blockchain	locally, blockchain	blockchain	blockchain
Trust data storage	off	on	on	on
Blockchain type	public	private	private	consortium
Blockchain operation mode	open	closed	closed	closed
Consensus protocol	PoW	round robin	n/a	n/a
Smart contract integration	no	no	yes	yes
Smart contract use case	n/a	n/a	check trust score	calculate trust score
Trust score assignment	ongoing	ongoing	ongoing	ongoing
Trust evaluation entity	local, individual, untrusted	local, individual, untrusted	local, individual, untrusted	distributed, untrusted
Trust model completeness	low	moderate	low	moderate
Trust aggregation	n/a	weighted average	n/a	weighted sum
Trust attack resilience	low	low	low	moderate
Suitability for decentralised and end-user-based M2M services	low	low	low	low

Table 5.2: Review of blockchain-based trust approaches – part II (Shala *et al.*, 2020a)

Elements	Trust Approaches			
	(Kouicem *et al.*, 2018)	(Dedeoglu *et al.*, 2019)	(Ali *et al.*, 2019)	(Boussard *et al.*, 2019)
Decentralisation	fully	fully	not	partially
End-user integration	no	no	no	part
Trust incentivisation	no	yes	no	no
Data storage type	DHT, blockchain	blockchain	blockchain	locally, blockchain
Trust data storage	on	on	on	off
Blockchain type	public	private	private	n/a
Blockchain operation mode	open	closed	closed	n/a
Consensus protocol	PoS	Periodical selection	individual	n/a
Smart contract integration	no	no	no	no
Smart contract use case	n/a	n/a	n/a	n/a
Trust score assignment	ongoing	ongoing	ongoing	ongoing
Trust evaluation entity	local, individual, untrusted	distributed, trusted	local, individual, untrusted	local, individual, untrusted
Trust model completeness	low	moderate	low	low
Trust aggregation	n/a	n/a	machine learning	n/a
Trust attack resilience	low	moderate	low	low
Suitability for decentralised and end-user-based M2M services	moderate	moderate	low	low

The main limitations of the reviewed trust approaches (Asiri and Miri, 2018; Di Pietro *et al.*, 2018; Kouicem *et al.*, 2018; Ali *et al.*, 2019; Boussard *et al.*, 2019; Dedeoglu *et al.*, 2019; Lahbib *et al.*, 2019; Malik *et al.*, 2019b) can be summarised as follow:

- Blockchain activities are utilised to store trust data. However, they are not used to merge the benefits of blockchain and consensus within the trust evaluation system to optimise the trust evaluation process.
- As with the trust approaches reviewed in chapter 3, the trust models used do not cover new M2M entities (new M2M services, new M2M service provider).

- Used consensus protocols have several limitations in terms of security and performance.
- Most of the approaches are using private and closed blockchains. Their focus is more on having a controlling/managing node in the network.
- Existing trust approaches do not consider the nature of end-user based M2M services, nor do they provide an end-user friendly platform and integration of the end-user into the trust processes
- Most of the blockchain-based trust approaches are not fully decentralised. Instead, they contain centralised elements in the community. Usually, supernodes (centralised nodes monitoring other nodes in a specific area) are used for different community activities. Others do not contest task activities. Moreover, the trust scores of the supernodes are not considered. This results in low credibility regarding their provided trust evaluation results.
- Their resilience against trust attacks (such as bad-mouthing attacks) varies from moderate to low, opening the doors for malicious nodes to harm the system.
- Only a few of the evaluated approaches (Asiri and Miri, 2018; Dedeoglu *et al.*, 2019; Malik *et al.*, 2019b) include reward/punishment systems in the network to incentivise good behaviour among the nodes.
- The benefits of process automation and fully decentralisation are also mostly ignored in existing IoT trust approaches. Only the authors in (Asiri and Miri, 2018) and (Malik *et al.*, 2019b) use smart contracts for checking the trust score, respectively computing it.

- The computation and aggregation of the trust score are only addressed in (Dedeoglu *et al.*, 2019; Lahbib *et al.*, 2019; Malik *et al.*, 2019b) by using weighted average or machine learning techniques in order to get the final score.

Similar to the reviewed trust approaches in chapter 3, it can be concluded that the blockchain-based trust approaches are less suitable for using them in end-user-based and decentralised M2M service provision approaches.

5.1.2 Fundamental Concept of Blockchain Integration for Data Integrity

Decentralised M2M communities are facing several problems. One of them is the increasing number of nodes joining/leaving the network and their unpredictable behaviour. That means different nodes acting independently from each other could behave maliciously by changing or removing sensitive data relevant to all community members. In terms of trust and trust evaluation within the M2M community, malicious nodes could manipulate or remove trust data from the network. Therefore, this research recognises the strengths of blockchain technology and proposes using it to optimise the storage system of the introduced trust approach. Thus, trust data is stored tamper-proof in the blockchain, and all community members have the possibility to check its integrity.

The workflow of the proposed approach starts with a service provider making a service available to the M2M community. All community members may evaluate the service trustworthiness (using the concept presented in chapter 4). Each M2M community member can act as a test agent. After a test agent has performed the trust evaluation

process and computed the trust score of the M2M service, it sends a blockchain transaction to the blockchain network. The blockchain transaction consists of information such as service ID, service instance (contact information about the service provider), evaluated trust score, and test agent contact information). The transaction broadcasted to all nodes part of the network is going to be part of a block. The created block has to be validated from the blockchain nodes. This process concludes with a so-called consensus indicating that the nodes agree on the same version of the blockchain. Figure 5.1 shows the blockchain integration used for storing trust data in the blockchain. Permission to reproduce Figure 5.1 has been granted by Springer Nature.

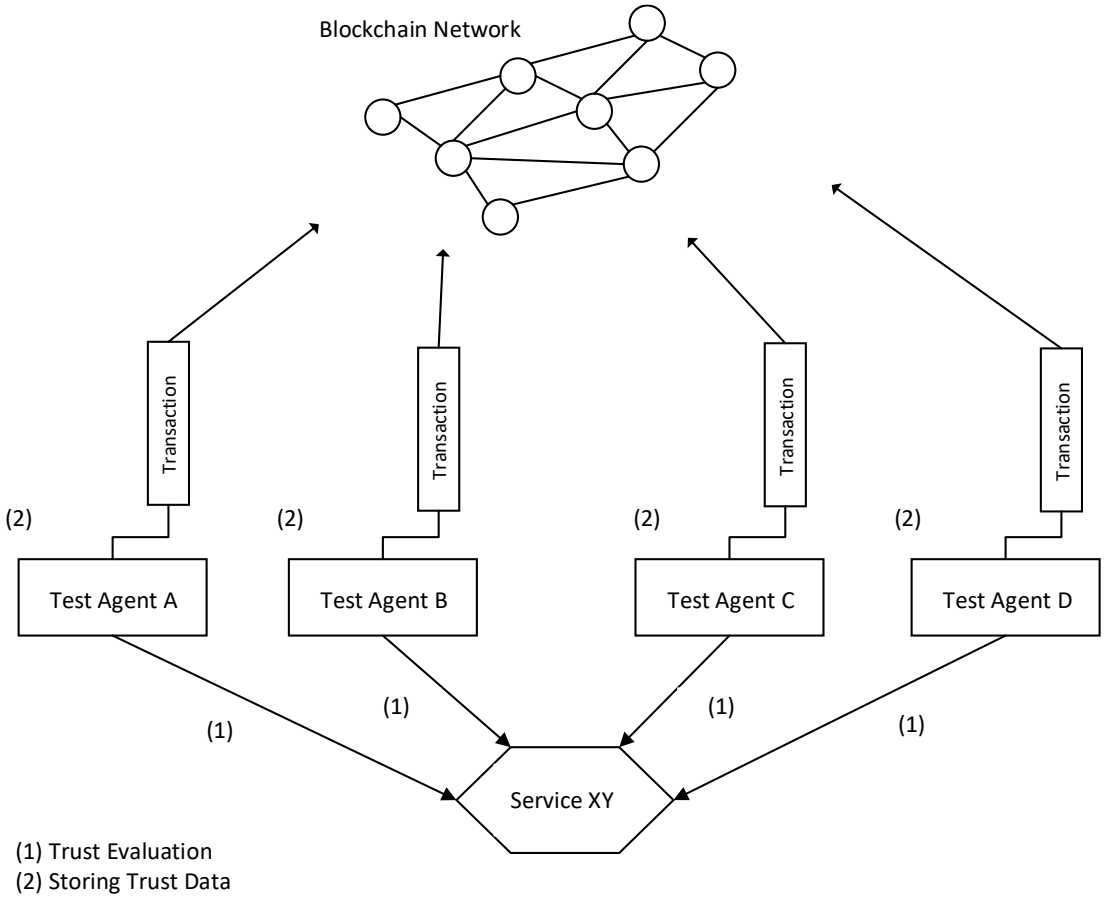

Figure 5.1: Integration of blockchain for storing trust data (Shala *et al.*, 2019a)

This research also suggests combining the DHT and blockchain storage types. Therefore, a DHT architecture is used to store the evaluated trust data. Specifically, the DHT storage contains only the current trust information about the M2M entities and enables fast and efficient lookups. The blockchain is used to store the whole data history and enable tamper-proof data. An end-user can utilise the DHT to find specific trust information quickly. On the other side, for integrity checks and further investigations, the blockchain can be used. If an end-user wants to check the integrity of a specific datum, it requires the information from the P2P overlay and compares/verifies it with that from the blockchain. Positive matches indicate that the end-user can rely on the data, and further actions can be taken.

The optimisation concept with P2P and blockchain can also be mapped to several other aspects of decentralised and end-user-based M2M service provision. The M2M community, with its members and services, provides a source for various data. There is information about the community members, such as contact details, admission details, and the list of services they provide. Furthermore, there is data about the registration of services and their descriptions. Figure 5.2 illustrates a general overview of storing different data categories in DHT and blockchain and using this combination for integrity check-ups. Permission to reproduce Figure 5.2 has been granted by Springer Nature.

A critical but also powerful and essential element of blockchain technology is the consensus protocol. The literature provides several proposed consensus approaches used within the blockchain technology (for instance, some are summarised in (Chalaemwongwan and Kurutach, 2018)). The most famous representation of consensus protocols is the Proof of Work (PoW) algorithm presented in Bitcoin. However, these

existing consensus approaches have several limitations. The next section will provide a review of existing consensus protocols, highlight their drawbacks, and propose a novel trust-based consensus protocol. The new consensus protocol is used within the framework proposed in this thesis and can also be used beyond it.

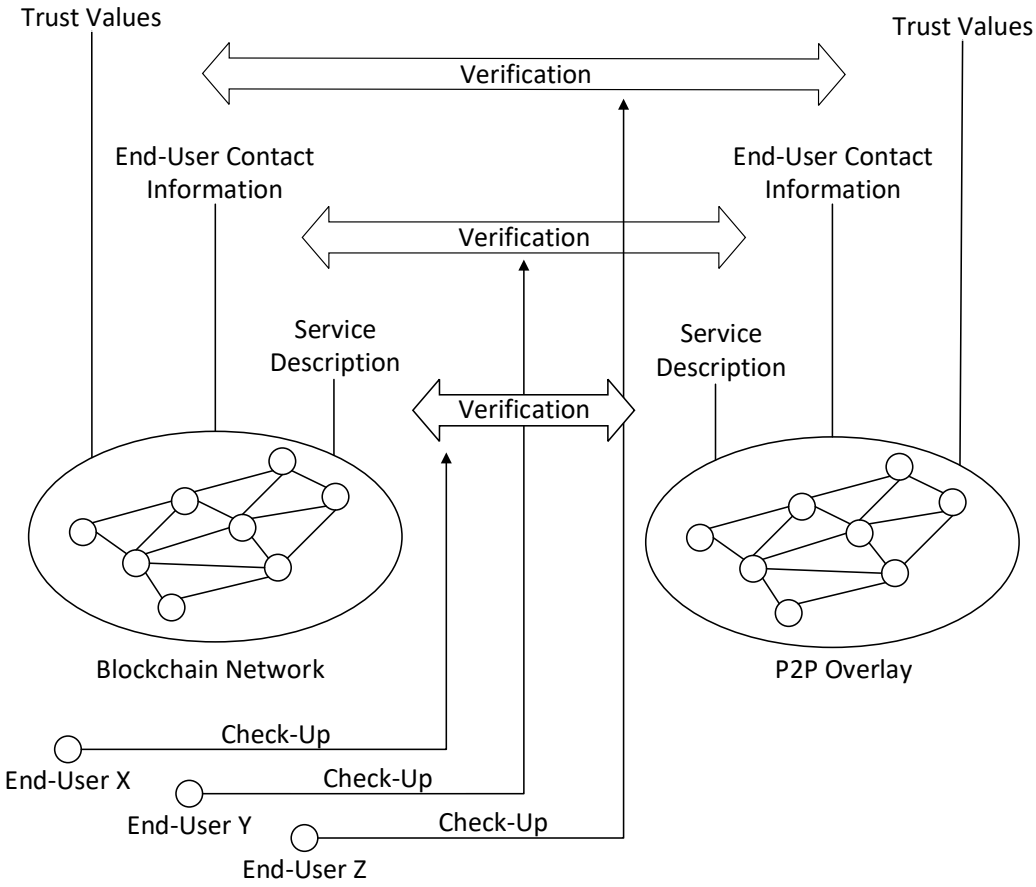

Figure 5.2: DHT and blockchain for the storage of different data categories (Shala *et al.*, 2019a)

5.1.3 Review of DLT-Based Consensus Protocols

As mentioned, consensus between nodes is required to agree on the same state or copy of a ledger. The literature review provides several definitions for the terms consensus, consensus protocol, and consensus mechanism (Christidis and Devetsikiotis, 2016; METI, 2016; Natoli and Gramoli, 2016; Bashir, 2017). One paper defines the consensus

as a process to reach a global view of all transactions between the nodes in the blockchain (Christidis and Devetsikiotis, 2016). Another definition (METI, 2016) states that consensus is "a series of procedures from approving a transaction as an official one and mutually confirming said results by using" consensus mechanisms or protocols. A consensus mechanism is defined in (Bashir, 2017) as "a set of steps that are taken by all, or most, nodes in order to agree on a proposed state or value". The main properties of a consensus are "agreement indicating that if two non-faulty participants decide they decide on the same block, validity indicating that the decided block should be one of the blocks that were proposed and termination indicating that eventually a correct participant decides" (Natoli and Gramoli, 2016).

There are different consensus protocols present in the literature. The literature review in this section separates them into two categories. The first one deals with the most relevant consensus protocols proposed within the DLT community. The second one consists of approaches that combine trust and consensus elements to optimise the whole workflow.

This research identifies several requirements that have to be fulfilled by a consensus protocol to comply with the characteristics of an end-user-based and decentralised M2M environment. A well-suited consensus protocol optimises the synergy of blockchain, trust, and M2M and should fulfil the following listed requirements:

- Less computational effort: M2M entities part of the M2M community are lightweight devices. They do not have the computational capability to perform high computing actions to maintain the distributed ledger and achieve consensus. Therefore, the protocol should support less computational effort to perform all these actions.

- Trusted selection of the block creator: One node will be selected to create a new block in the blockchain. This node is called block creator. The way a peer is selected as a block creator is crucial, as a peer with bad intention could be selected and add fake transactions to the ledger.
- Trusted transactions and block creation: Distributed ledger technologies provide a secure way to store data. However, the trustworthiness of the data and the data originator are not the aims of it. The consensus protocol should consider the differentiation between the transactions and the transaction originator trustworthiness. Besides selecting the block creator, the blocks created should also be trusted, which means they should not include fake transactions. This implicates that only trusted transactions are added within a block.
- Decentralised architecture: The DLT network should be completely decentralised. Centralised entities controlling the blockchain activities should be avoided. A decentralised architecture enables the defence against single points of failure and possible attacks from a supervising node. Besides them, the initial requirement of this research is to provide an overall decentralised approach within the proposed framework.
- Trust reward/punishment: Peers part of the M2M community should be motivated to participate in the consensus-building process. Harmful activities should be punished and the right ones honoured by the system.
- Dynamic and reliable trust model: The trust model should adjust itself dynamically based on the trust situation of the community. Moreover, it should generate reliable trust scores and dynamic allocation of them.

- Robustness/resilient against fake transaction attacks: The consensus should be resilient against fake transactions or blocks flooded by one or many peers in the network.
- Reliable validation decision process: The process and the decisions of the community to include a new block in the blockchain should be reliable.

The evaluation of the reviewed consensus protocols is done based on these identified requirements. In the following, a brief overview of traditional- and trust-based consensus approaches is presented.

Traditional Consensus Approaches

The Proof of Work (PoW) consensus protocol was introduced by Bitcoin and is one of the first mechanisms to achieve consensus between nodes in a blockchain on the same ledger. The PoW consists of nodes acting as "miners" by trying to solve a computational puzzle called hash. The node that solves first this puzzle will validate and add a new block to the blockchain. The performing activities are rewarded for the first successful miner for validation. (Nakamoto, 2008) However, performing PoW requires hardware with high computational power, and the mining process is an energy-intensive process.

Another consensus mechanism is Proof of Stake (PoS), which tries to offer a more efficient way to validate transactions in the blockchain. This mechanism does not require high computing power and selects nodes randomly for mining based on several criteria (depends on the PoS version). The first version of PoS defines proof of ownership of a currency and the coinage consumed by a transaction as criteria to select nodes to add new blocks. The blockchain with the highest score is selected as the main chain in different

concurrent chains in the network. This score is computed based on the consumed coinage of every transaction which is part of the block. (King and Nadal, 2012) PoS protocol minimises the computing effort and provides more energy efficiency. However, selecting leaders based on the stake ownership percentage could lead to centralisation and monopolisation.

One of the extended versions of PoS is the Delegated Proof of Stake (DPoS), where other nodes elect the block producers. The votes are weighted based on the network stake of each voter. In DPoS the number of block producers is fixed, and every block producer is allowed to produce one block per round. If a block producer does not perform the right actions, he can be voted out by the community. The DPoS provides a pretence democracy by enabling voting. (Snider *et al.*, 2018) However, one of the disadvantages of this protocol is the possibility to build a monopoly as votes are weighted based on the stake owned by each node. Thus, rich nodes could increase their impact on the network. Another problem is the fixed small number of block producers that can be elected. As a result, the level of centralisation increases.

Nano is a consensus protocol that uses a block-lattice structure where each block contains only one transaction, and every node (account holder) has its blockchain (not the whole but only a single view about it). Sending funds from one account to another requires a *send transaction* from the sender of the funds and a *receive transaction* from the receiver. If there is a conflict on a transaction, the system can start a voting mechanism where representatives are chosen on behalf of account holders and vote for transactions. Their voting power is calculated based on the sum of all account balances that have chosen them. In order to mitigate a double-spending attack, Nano uses a lightweight version of

the PoW. (LeMahieu, 2018) The Nano protocol is similar to the Proof-of-Stake protocol and leads to supernodes in the form of representatives. Moreover, monopoly is also an issue here.

A different consensus approach is provided in Ripple where each node has a Unique Node List (UNL) and a ledger. The UNL contains a list of nodes chosen by a node, assuming that they will not behave maliciously. The consensus works in that way that a node votes by comparing the transaction received from the UNL with other transactions from previous rounds or other nodes if they match. Transactions that receive a negative vote are considered for the next consensus round or discarded from the network. The voting mechanism runs until the transaction receives 80% of the votes, which qualifies them to be included in the ledger. (Schwartz *et al.*, 2014) Ripple is an energy-saving protocol that relies on voting based on transaction similarity and does not require PoW computing. However, the drawback of Ripple is that the consensus is achieved by a fixed number of peers, which leads to a less decentralised system.

Another consensus protocol is introduced in IOTA, which uses Tangle as an underlying distributed ledger technology. It is based on a directed acyclic graph (DAG). In a Tangle, all transactions are linked with each other, where unconfirmed transactions are called tips and confirmed transactions sites. In order to deploy a new transaction to the community, a node first needs to validate two previous transactions by performing a lightweight PoW (used to avoid double-spending) and other validating steps (for instance, if the transaction conflicts with the history of the tangle). After validating two other transactions, the current transaction is linked to them and waits until others validate it. Additionally, IOTA adds weight to the sites (transactions). The weight of each site is calculated based on the

time a node spent doing PoW for confirming that transaction. Each transaction also has a cumulative weight. (Popov, 2018) Tangle is a lightweight consensus protocol that uses a coordinator to protect the system from double-spending attacks. Therefore, the coordinator has to issue transactions periodically called milestones. Other transactions are confirmed if they are referenced by a milestone (Wall, 2017). The integration of a coordinator is a centralised element in Tangle and can be considered as a drawback.

Trust-Based Consensus Approaches

The authors in (Zou et al., 2019) propose using a blockchain to log service transactions part of a crowdsourcing site. To avoid the limitations of PoW, they introduce a consensus mechanism where a ledger management group maintains the blockchain and the leader of it is selected using a voting-based election algorithm. Moreover, the selected leader selects a *service transaction validation group* to validate transactions that have to be included in the blockchain. The list of the *service transaction validation group* is sent to other members part of the blockchain that vote based on their own trust database for or against that list. Based on the received votes, the leader forms the validation group and broadcasts the final list to the network. The validation nodes select transactions for including them in the next block and broadcasts them to other nodes. These other nodes, called consortium nodes, also vote on the proposed transactions by the validation nodes. The leader summarises the votes of both the validation nodes and consortium nodes to select the winning transactions to be included in the block. The consensus protocol introduced by (Zou *et al.*, 2019) avoids Proof of Work and adds trust to the whole process. However, deciding on a leader for blockchain management leads to centrality and a single point of failure. The authors also do not describe the motivation of the participating nodes

in the overall process. There are also missing information regarding the local trust databases on whether they differ from node to node. Another point is that the authors consider all transactions to be part of the blockchain without checking the trustworthiness of the transactions.

Another trust-based consensus protocol is proposed in (Bahri and Girdzijauskas, 2018), where the authors combine a so-called trust graph encoded in the blockchain and PoW. Based on the trust graph, the trust score of every node can be derived and used to "assign the amount of energy that has to be sent for traditional PoW". First, a trust graph is created where every node expresses its trust towards other nodes. Based on that graph, trust metrics for all consensus nodes are computed. In order to avoid centralisation, all nodes with the highest trust score are discarded from the consensus part. The consensus protocol presented in (Bahri and Girdzijauskas, 2018) randomly selects a node in each round relative to the overall trust derived from the trust graph. The selected node can decide on transactions to be deployed in the blockchain. The authors in (Bahri and Girdzijauskas, 2018) also claim that the difficulty of the cryptographic puzzle depends on the trust score of the selecting node proposing a new block. Nodes with a higher trust level have to mine with a lower difficulty level of the puzzle. The approach presented in (Bahri and Girdzijauskas, 2018) represents a lightweight version of the PoW that minimises the computing power required for mining and integrates trust elements in the overall process. However, they do not explain how transactions are validated after a node with a high trust score has proposed a block. Moreover, generating a trusted candidate set by the previous miner adds centralisation to the concept. The authors also miss adding more trust to the process of creating a block or selecting transactions. Moreover, the participation of the nodes in the consensus process is not rewarded or punished.

The authors in (*COTI*, 2018) propose a consensus protocol operating in a DAG and where transactions receive trust scores based on the trust score of the sending node. In order to deploy a transaction to the DAG, the node has to validate two prior unconfirmed transactions. Moreover, the node has to perform a lightweight form of PoW to disable double-spending attacks. These unconfirmed transactions are selected based on the current trust score of the transaction to be deployed (trust scores that are close to each other). "This results in the formation of trustchains in the cluster. The cumulative trust score of such a chain is the sum of the trust score of all the transactions making up the chain". (*COTI*, 2018) To check whether a transaction is confirmed, the cumulative trust score of its trustchain must be compared with the preset global confirmation threshold. The advantage of the approach presented in (*COTI*, 2018) is that, compared to other approaches, it also considers the trust score of the transaction originator and the transaction in the consensus process. Moreover, the confirmation of the transaction depends on its trust score and the trust score of the trustchain. The authors in (*COTI*, 2021) describe the trustchain as a network where the trust score of nodes is considered to evaluate the level of proof of work necessary to confirm a transaction. However, the authors in (*COTI*, 2021) state that a small number of highly trusted nodes implement the consensus. The disadvantage is the risk of centralisation and monopoly during the consensus process.

In order to enable a decentralised smart contract settlement and validation, the authors in (ICASH, 2018) introduce a Proof-of-Trust protocol. The idea of this protocol differs from the others reviewed above. It evaluates and pre-validates information entered in the smart contract, which is contested by other participant nodes. "Whenever data input is contested by one of the smart contract parties, a decentralised network of qualified participants

validates the data before the execution of the contract" (ICASH, 2018). The Proof-of-Trust protocol is performed by so-called delegates, who are initially considered trustworthy and legitimate through a known-your-costumer process. For each smart contract, a delegate is assigned, and the contract is "executed with a valid input". The delegates have at the beginning the same high trust score (100%), which remains or is changed based on the inputs provided by them. If there is a "contestation to an input, a full network call is incited and all delegates in the network vote in an auditing process of the original query to provide majority consensus" (ICASH, 2018). Based on this vote, the trust score of the delegate is changed or not. The authors in (ICASH, 2018) introduce a reward/punishment system for the inputs provided by the delegates. Another advantage of this approach is that the inputs added to a blockchain are pre-checked, and a certain level of trust is attached to a transaction. However, the selection of the delegates is disadvantageous, as they are initially considered trustworthy without going through a trust evaluation process.

Review Outcomes of Existing Consensus Protocols

The benefits and limitations of the different consensus protocols can be summarised as follow:

Most of the reviewed approaches (Nakamoto, 2008; King and Nadal, 2012; Schwartz *et al.*, 2014; *COTI*, 2018; Bahri and Girdzijauskas, 2018; LeMahieu, 2018; Snider *et al.*, 2018) require computational effort for achieving consensus and validating new transactions. However, the protocols described in (Schwartz *et al.*, 2014; Zou *et al.*, 2019) minimise this effort by removing the need to perform "Proof of Work" and solve a computational puzzle. Traditional-based consensus protocols (Nakamoto, 2008; King and

Nadal, 2012; Schwartz *et al.*, 2014; LeMahieu, 2018; Popov, 2018; Snider *et al.*, 2018) also do not fully satisfy the requirement to fairly select a node that produces a new block. They use computational effort or amount of stake to decide on a block creator. In contrast to them, trust-based protocols (*COTI*, 2018; Bahri and Girdzijauskas, 2018; ICASH, 2018; Zou *et al.*, 2019) add some trust elements to the selection process by considering the trust score of peers. Another point is how new blocks are created and if they include correct transactions. Except for the protocol in (*COTI*, 2018), which consider the trustworthiness of the inputs in the blockchain, all other reviewed protocols (Nakamoto, 2008; King and Nadal, 2012; Schwartz *et al.*, 2014; Bahri and Girdzijauskas, 2018; ICASH, 2018; LeMahieu, 2018; Popov, 2018; Snider *et al.*, 2018; Zou *et al.*, 2019) do not or partially satisfy the requirement for trusted block creation. In order to avoid monopoly and single points of failure, a decentralised architecture with decentralised responsibilities is required. Only the PoW fully satisfies this requirement by including all participating nodes in the consensus and block validation process. Other reviewed approaches (King and Nadal, 2012; Schwartz *et al.*, 2014; *COTI*, 2018; ICASH, 2018; LeMahieu, 2018; Popov, 2018; Snider *et al.*, 2018; Zou *et al.*, 2019) partially satisfy the requirement of a decentralised architecture. Mostly, semi-centralised representatives or only a limited number of nodes involved in consensus building prevent the system from being fully decentralised. Another important element of a consensus is how the network motivates the nodes to participate in the consensus process and deal with misbehaving nodes. Specifically, a trust reward/punishment system is required. The authors in (*Bitcoin - Developer Guides*, 2021) suggest a ban score assigned to misbehaving nodes that broadcast false information. However, this fact only partially fulfils the requirement as it does not include incentives for nodes to participate in all consensus steps. Another

approach that partially fulfils the defined requirement is (ICASH, 2018), which introduces a reward/punishment system, but only for nodes that send an input to the blockchain without considering the validating nodes or block creators. None of the reviewed approaches, except the protocol introduced in (Bahri and Girdzijauskas, 2018), use or describe any trust model that can evaluate the trustworthiness of the participating nodes of the blockchain. The trust level of the nodes ensures a reliable and secure network mitigating several misbehaving attacks. In this context, participating nodes could flood the network with fake transactions and try to bring false information into the blockchain. Protocols like (Nakamoto, 2008; Schwartz *et al.*, 2014; *COTI*, 2018; ICASH, 2018; Popov, 2018; Zou *et al.*, 2019) partially satisfy the requirement of being resilient against these attacks. The high computing power required to add new blocks to the blockchain in (Nakamoto, 2008) mitigates the integration of blocks with fake transactions in the blockchain as not every malicious node can solve the cryptographic puzzle. The approach in (Schwartz *et al.*, 2014) requires transaction similarity to add a transaction to the blockchain. Therefore, a malicious node must maintain a high percentage of nodes to proceed with its fake transactions in the blockchain. The Tangle protocol also partially satisfies this requirement by considering the cumulative weight of transactions to decide which transactions are valid. Most of the trust-based approaches (*COTI*, 2018; ICASH, 2018; Zou *et al.*, 2019) consider trust weight or scores of transactions in their solution. However, they do not cover all considered attacks of fake transactions. Therefore, they partially fulfil the requirement of being resilient against fake transaction attacks. Finally, only the protocols introduced in (Schwartz *et al.*, 2014; ICASH, 2018; Zou *et al.*, 2019) partially support a reliable validation decision process. In Ripple (Schwartz *et al.*, 2014),

the validation is done through transaction similarity check-ups, and in (ICASH, 2018; Zou *et al.*, 2019), through voting of all community members.

The outcomes of the review are also illustrated in the following Table 5.3.

Table 5.3: Evaluation of existing consensus protocols (Shala *et al.*, 2019c)

Protocol		Requirements								
		Computational effort (1)	Trusted block creator selection (2)	Trusted block creation (3)	Trusted transaction (4)	Decentralised architecture (5)	Trust reward/punishment (6)	Dynamic and reliable trust model (7)	Resilient against fake transaction attacks (8)	Reliable validation decision process (9)
Traditional-based	PoW	-	o	-	-	+	o	-	o	-
	PoS	o	o	-	-	o	-	-	-	-
	DPoS	o	o	-	-	o	-	-	-	-
	Nano	o	o	-	-	o	-	-	-	-
	Ripple	+	n/a	o	-	o	-	-	o	o
	Tangle	o	-	o	o	o	-	-	o	-
Trust-based	Zou	+	+	o	-	o	-	-	o	o
	Bahri	o	+	-	-	-	-	o	-	-
	COTI	o	+	+	+	o	-	-	o	-
	i-Cash	n/a	n/a	n/a	o	o	o	-	o	o

Assessment notation: + satisfied, o partially satisfied, - not satisfied, n/a not applicable

5.1.4 Novel Trust-Based Consensus Protocol for Blockchain

The previous section has identified several limitations on existing consensus protocols. One main issue is the fact that these approaches do not consider trust in their methodology. Trust inside the M2M community is a crucial aspect of this research. To overcome existing problems and to be applicable for decentralised M2M environments, this research project presents a novel trust-based consensus protocol. This protocol is used in the blockchain network and other activities within the proposed framework (will be presented in upcoming chapters). The consensus protocol aims to combine several benefits of existing ones and include trust as a fundamental element. It is called Trust Consensus Protocol (Trust-CP) as it integrates trust within the different steps of the consensus-building workflow.

Fundamental Principles

The Trust-CP aims to solve several trust issues in a decentralised community where the nodes do not need to spend too much energy and waste computational power for performing blockchain activities. The main component of the proposed consensus protocol is the dynamic trust model (Trust Evaluation System), which is introduced in chapter 4 and is used to assign trust scores to peers from whom blocks and transactions inherit the scores. This novel trust model considers all relevant trust aspects of a node, including the initial trust level, community participation, and experience rating between the nodes. Another important component of the Trust-CP is the trust selection of block creator nodes (nodes that can collect transactions and add a new block to the blockchain). In order to enable reliable verification of new blocks, a trust-based voting system is integrated into the consensus protocol. For encouraging the nodes to participate positively

in the whole blockchain process, a trust reward/punishment mechanism is included in the Trust-CP.

Lifecycle of Trust-CP

This research considers that all nodes of the M2M community also participate in the blockchain network. As the M2M community is a decentralised environment without a centralised authority, the participating nodes are continuously evaluated to derive their current trust score. Part of the trust evaluation is also the active participation in the blockchain network. As mentioned, all nodes in the blockchain network will have their trust score. During the lifetime, transactions are sent from one node to other nodes. All transactions are assigned with the trust score of the transaction initiator (trust score of the sender node). In this research, the blockchain is used to store information about the trust scores of nodes and services. For instance, node A evaluates the trust score of node B using the trust model described in chapter 4. After the trust score is computed, node A sends a transaction to node B containing the trust information. Other nodes also send a transaction like that. These transactions are unconfirmed and are waiting to be approved by the blockchain network. Before the approval process starts, the transactions have to be included in a block. This is done by so-called block creators, who are nodes selected from the blockchain network to perform these tasks.

The remainder of this section describes the lifecycle of the Trust Consensus Protocol through an example.

The lifecycle description of the Trust-CP starts with a given network of five nodes. It is assumed that these nodes have been in the network for some time and that various

activities have been carried out. Furthermore, it is also assumed that the trust scores of these nodes have been evaluated (using the trust model explained in section 4.4) during the time and that there is sufficient data to support this. The bootstrapping of these nodes and the trust evaluation activities are not relevant and not described at this point. Exemplary the following trust scores of the five nodes are given: Peer A = 0.58; Peer B = 0.89; Peer C = 0.77; Peer D = 0.48; Peer E = 0.81). The description of the Trust-CP also considers that the five nodes are and behave trustworthily.

The proposed Trust Consensus Protocol consists of five main phases (see Figure 5.3): Trust Peer Filtering (see Figure 5.3, Phase I), Random Selection (see Figure 5.3, Phase II); Block Creation (see Figure 5.3, Phase III); Trust Weighted Voting (see Figure 5.3, Phase IV); Trust Reward/Punishment (see Figure 5.3, Phase V). These phases are explained in the following paragraphs.

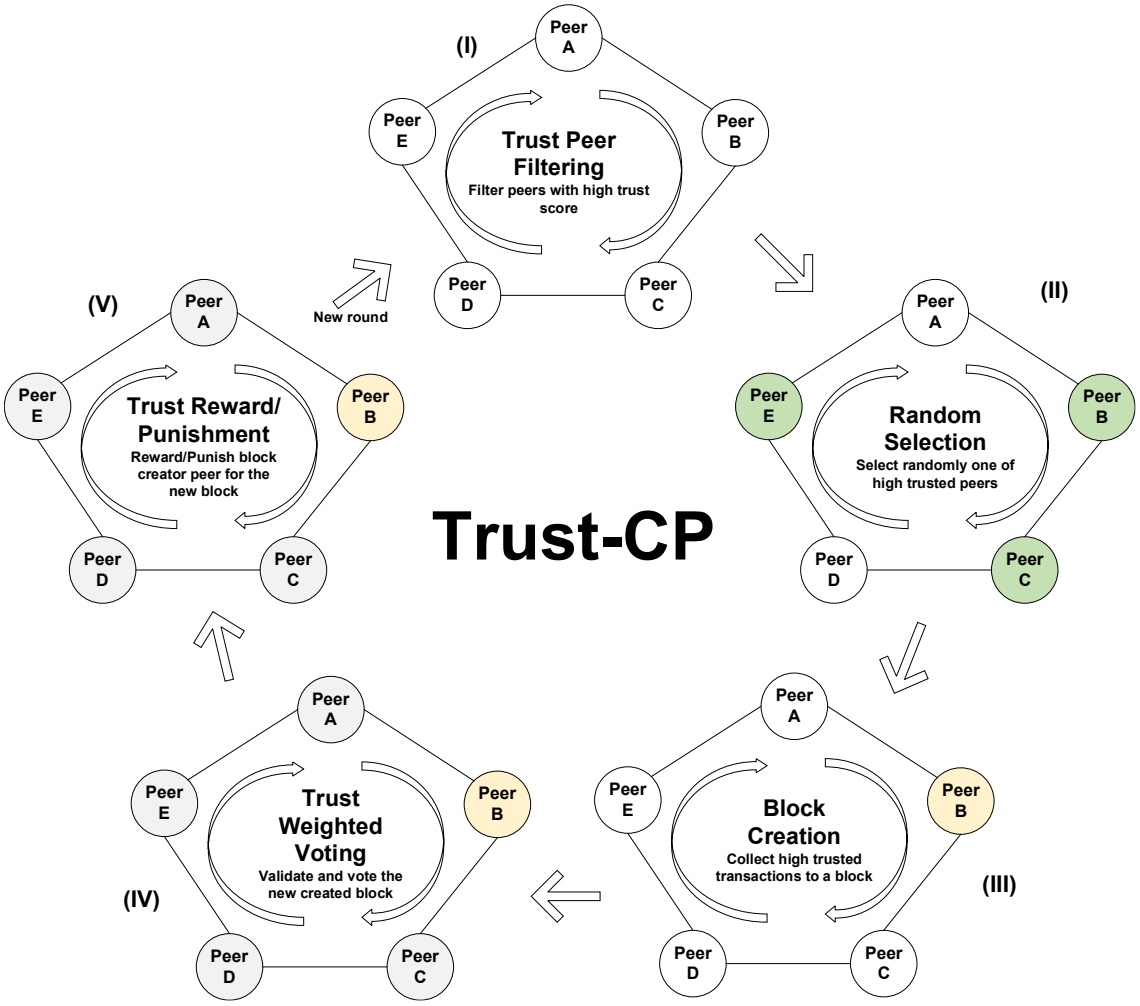

Figure 5.3: Trust-CP (Shala *et al.*, 2019c)

The Trust-CP process is triggered every round of a new block generation, where unconfirmed transactions are collected and included in a new block. In Phase 1 of the Trust-CP (see Figure 5.3), only the nodes with a high trust score (0.75 or higher) are filtered out and considered for Phase 2. The threshold of 0.75 is set initially and is changed through a dynamic threshold adjustment system. In this example, the trust peer filtering identifies three highly trusted peers (Peer B, Peer C, and Peer E) for the random selection phase (Phase 2). All other nodes are not considered to be selected as block creators. If there are no nodes with a trust score above the threshold, it is assumed that there is an untrustworthy or compromised network at all. This kind of scenario is out of the scope of

this thesis. Alternatively, the selection of the block creator could be based on the voting of all nodes part of the blockchain. Every node will vote for another node to be selected as a block creator. The nodes are voting for nodes part of their neighbour list. The voting is weighted with the trust score of the voting node and the node being voted on.

Phase 2 (see Figure 5.3) of the Trust-CP consists of randomly selecting one of the highly trusted nodes as the block creator for the new round of block generation. In this case, the algorithm selects one of the three peers filtered out in Phase 1. Exemplarily, Peer B is selected to act as a block creator in Phase III.

Overall, Phase 1 and 2 of the Trust-CP rely on a trust-based selection mechanism where the block creators are selected based on the combination of trust score and randomness. For every round of block generation, an algorithm runs through the nodes to randomly select a block creator based on its trust score.

In Phase 3 (see Figure 5.3), the selected block creator (in this example Peer B) collects pending transactions to a block. Another information added to the block is the trust score of the block creator and the voting information about its selection process. Additionally, the created block contains block ID, previous block header hash, timestamp (no difficulty target and no nonce value compared with the Proof of Work), Merkle root, and transactions. Ideally, only transactions with a high trust score are considered to be part of the block. Transactions below the threshold are not considered to be part of the block. If the block creator node does not have a pending transaction that meets the criteria, then the block creator node drops its task, and a new block creator must be selected using the algorithms in Phases 1 and 2.

After the block is generated, it will be broadcasted to other nodes for validation and confirmation (see Figure 5.3, Phase IV). Other nodes will receive the block and verify it by checking the trust score of the block creator node, the trust score of the transactions part of the block, and the hash values of the block. Suppose the block contains the right information and also fulfils the criteria of the system. In that case, it will be voted positively by the validating node and the block is forwarded to other nodes. The criteria are fulfilled if a trusted block creator creates the block, the block contains the right hashes, and the transactions within the block are also trusted. If the block does not meet the conditions, it receives a no vote. The votes are weighted based on the trust score of the validators. If the block receives a majority of positive votes from all highly trusted nodes, it is part of the blockchain.

This research also proposes to reward or punish nodes for their actions within the blockchain network. The reward/punishment system motivates the nodes to participate in blockchain activities and increases the overall functionality of the network (see Figure 5.3, Phase V). If, for instance, a block creator provides a fake block, it will get a lower trust level from the validating nodes and will be downgraded by losing the possibility to create blocks for further rounds. In general, the block creator node will receive positive or negative trust scores based on the approval or rejection of the proposed block.

Overall, the following benefits and limitations are identified:

- The trust score of the participating entities is considered in all phases of the consensus lifecycle. Only trustworthy nodes can participate in the consensus process, and only trustworthy transactions and blocks are allowed to be included in the blockchain.

- The trust scores used for the Trust-CP are based on the proposed trust model in section 4.4, which considers different trust aspects of an M2M entity and thus provides a good basis for an expressive trust score. Combining the Trust-CP with the trust model complicates the process for malicious nodes to increase the trust score.
- Spending too much energy and wasting computational power like the PoW consensus protocol is avoided.
- Fake transactions broadcasted from untrustworthy nodes are ignored and not included in the blockchain (refer to the experiments conducted in 7.2.4).
- Resilient against an increasing number of malicious nodes (with low trust scores) in the network and fake transactions spread by them (refer to the experiments conducted in 7.2.4).
- The Trust-CP actually works in an environment where all trustworthy nodes participate correctly and faithfully in all consensus activities. Attacks or malicious activities from nodes with good trust scores are not considered in the proposed consensus protocol.
- The resiliency of the proposed consensus protocol decreases as more trustworthy nodes with bad intentions cooperate in the network.
- Trust-CP fails when attacked by collaborating nodes that joined the network, worked hard to get high trust scores, and now control the entire system. These nodes can push out other nodes from the network, add wrong information to the blockchain, and manipulate the trust scores of others. Nevertheless, compared to other consensus protocols, the Trust-CP resists this type of attack for longer because of the high effort a malicious node has to put in to increase its trust score.

- Typical attacks on traditional consensus protocols, such as denial of service, double-spending, or identity attacks, are not considered in the evaluation of the Trust-CP and are out of the scope of this thesis.

5.2 Integration of Service Testing within Trust Evaluation Activities

The different M2M/IoT trust approaches reviewed in chapter 3 present solutions for evaluating the trust score of existing services, and no or few publications consider new joining entities. This is a disadvantage because malicious peers could enter the community with malfunctioning/malicious services and manipulate the ranking of others to gain a better reputation for themselves. As a result, they can cause trouble in the system. For that reason, it is essential to consider the initial trust score of services and peers when entering the M2M community. This ensures faster trustworthiness among the peers and the services. Thus, trust relationships between entities are established more quickly and peers with bad intentions are identified.

The upcoming subsections will present a literature review of trust approaches outside the M2M/IoT domain that recognise the problem of trust for new entities (section 5.2.1). Moreover, the integration of functional testing within the trust evaluation (subsection 5.2.2) and the extension with performance testing will be introduced (section 5.2.3). Finally, section 5.2.4 presents the workflow for building the initial trust score of new M2M services.

5.2.1 Existing Trust Approaches for Initial Trust Score Evaluation

Trust information about new M2M entities entering the M2M community supports the decisions of existing peers to start interactions with them or not. This research has identified a few publications dealing with initial trust scores for services in other application fields. In the following, a selection of the most significant works for this issue is presented. The outcomes of the review are summarised at the end of this section.

Some approaches are dealing with the trust bootstrapping problem for web services (Aljazzaf *et al.*, 2011; Nguyen *et al.*, 2012; Tibermacine *et al.*, 2015). The term trust bootstrapping is defined in (Aljazzaf *et al.*, 2011) as the "mechanisms to assign trust rate for a new service that its trustworthiness is unknown and before having any requestor interacting with it". In (Nguyen *et al.*, 2012), it is defined as a process for establishing the initial trust score "for newcomers who do not have or have very limited record of their past behaviours in the community". The approach in (Aljazzaf *et al.*, 2011) considers the subjectivity of trust where different nodes have a different opinion about the observed trust score of the service. Moreover, it includes also the different trust scores of a service in different situations. Initially, every service provider publishes provider and service information. The system uses different information to perform the evaluations. The process includes different methods as defined in (Aljazzaf *et al.*, 2011): monitoring (Objective Trust Metric), certification (Subjective Trust Metric), or feedback (Provider Trust Metric) approach. More in detail, the following metric parameters are used: execution time, response time, latency, throughput, remedies, security, privacy, payment satisfaction, output satisfaction, brand, competence, honesty, website, and physical location. The evaluated trust metrics are stored in the registry of the system and can be

used by the service requester for selecting a service based on its trust preferences on a set of trust metrics.

The authors in (Nguyen *et al.*, 2012) aimed to overcome the initial trust score problem for web services by providing three generic mechanisms. The first one is the inheritance mechanism. Here, a web service gets the trust score of the service provider. This is applied to a new service when the service provider is already part of the community and has past trust scores. The second mechanism is based on referrals, where a web service gets the trust score based on referrals from other communities. This includes the situation when the service provider of the new web service is also new to the community, but the web service has been used in other communities. The third one is the guarantee mechanism, where the web service gets a temporary trust score under guarantee conditions. This is applied when service and service provider are new, and the other two mechanisms are not relevant. In this case, the new service agrees to provide a guarantee policy. For example, by paying back part of its cost to the consumer if it does not perform as promised in the agreement. The community will manage this process using a monitoring engine in the system. (Nguyen *et al.*, 2012)

Another approach for new web services is introduced in (Tibermacine *et al.*, 2015). The initial reputation values are estimated based on the Quality of Service (QoS) attributes of a service and the similarities with other services. First, a new service joins the community, including a set of initial QoS attributes. Then, the system checks the history of the service provider offering that service. Based on the history, the system can classify the service provider as a new or existing one. New service providers will get a zero-reputation score which means untrustworthy. If it is an existing service provider, reputation is calculated

based on the quality of services provided in the past. In the next phase, the system will check the similarity rate of the new service with other existing services. First, the system builds a model from the QoS vectors and reputation values of similar services. Similar services are derived using the comparison of the Web Services Description Language (WSDL) files of the services. The new service will then get the higher reputation value of the provider reputation value and the reputation value created by the system. The third phase considers services and service providers who are new in the system without past interactions. Therefore, the system combines all QoS and reputation data of all long-standing web services to build a multiple regression model to estimate the reputation of these new entities. (Tibermacine *et al.*, 2015)

The authors in (Yu *et al.*, 2015) give a special focus on computing the initial trust score of defence agents, which are part of computer network collaborative defence models. Different types of agents are present in such an environment: management agent, evaluation agent, and new coming agent. The new coming agent will undergo the trust bootstrapping process to get an initial trust score, whereas the management agent and the evaluation agent are considered trustworthy. The trust evaluation process starts with the registration of the new agent to the management agent. The latter selects an appropriate evaluation agent for this process. Afterwards, the evaluation agent sends test tasks to the new node and waits for the task execution results. The results are used to classify the trust type of the defence agent, which is related to the behaviour pattern of the agent and is identified by analysing its feedback. Moreover, the trust evaluation process includes identifying constraints by analysing the benefits and costs for performing defence task between two entities. Additionally, this analysis includes also the benefit that the trust brings to the entities by calculating the gained trust utility. Afterwards, assigning the

initial trust score of the new defence agent is done using the methods of assigning corresponding values and computation of weighted averages for the defence agents. (Yu *et al.*, 2015)

The authors in (Djamaludin *et al.*, 2013) aim to provide an approach for assigning initial trust scores in Delay Tolerant Networks (DTN). Therefore, they propose a distributed trust system where initial trust scores are computed during the deployment and initialisation of new nodes. The authors in (Djamaludin *et al.*, 2013) support the key management in networks without a Public Key Infrastructure (PKI). Their initial trust score approach is called Leverage of Common Friends (LCF), which focuses on common contacts between two nodes meeting for the first time. In detail, the LCF approach considers the number of contacts, the number of common contacts, and the public key validity as elements to establish the initial trust score between two entities. (Djamaludin *et al.*, 2013)

Another trust approach for evaluating the initial trust score of entities is presented in (Wang, 2020). The author introduces a trust model including direct and indirect trust for the Internet of Vehicles (IoV) environment. Direct trust includes historical interaction experience between vehicles, whereas indirect trust includes recommendations from other nodes. The problem of trust initialisation between two vehicles is solved by using the social relationships of vehicle owners. These kinds of relationships are considered to be used for building a so-called offline trust. The offline social relations are classified based on the social proximity between entities and includes the following types: relative, friend, ordinary, and strange. The type *relative* means that the entities have a blood relationship, whereas the last type, *strange*, stands for no previous interactions. Finally, initial, direct,

and indirect trust are used to compute the overall trust score of an entity (using also weighted sum calculation).

The authors in (Wahab *et al.*, 2020) deal with the initial trust score problem of cloud services and introduces a machine-learning approach where the evaluating entity uses specifications of new and similar cloud services to compute the initial trust score. Specifically, a user assesses the trust score of each interaction based on the own satisfaction level. Suppose a user decides to participate in the bootstrapping process for another new cloud service. In that case, it will use its existing trust data on similar cloud services and the new one to train a decision tree machine learning classifier. This process results in the trust labels *trustworthy* or *untrustworthy new cloud service*. Many users can be part of the trust bootstrapping process. All their trust labelling outcomes are aggregated to compute the final initial trust score for the new joining cloud service. An incentivisation system ensures the participating of community members in the trust bootstrapping process. (Wahab *et al.*, 2020)

In contrast to existing trust approaches in M2M/IoT, the works described in this section provide some interesting points on how to solve the problem of new joining entities. However, similar to the M2M/IoT trust approaches presented in chapter 3, they suffer from the limitations of insecure storage and trust assessment systems. None of the approaches provides details on the storage system and its architecture (centralised or decentralised). Moreover, there is no information if security measures are used to protect sensitive trust data. The following presents some identified limitations regarding the trust assessment schemes provided in the approaches mentioned above.

- Self-declaration: In (Aljazzaf *et al.*, 2011), the service providers have to specify the trust metrics themselves. Thus, they are able to falsify the information. This results in not reliable trust data and has a bad impact on the trust evaluation process.
- Inheritance approach: Some design issues are also present in (Nguyen *et al.*, 2012). For instance, the inheritance mechanism where a new service inherits the trust score of the service provider's existing services. It is not very secure to rely on other services from the service provider, as a new service may behave differently. Thus, the trust score assignment for a new service should not only be based on the trust scores of existing services. Similar, the authors in (Tibermacine *et al.*, 2015) use the values of other services provided by the service provider for assigning the initial score for new joining services.
- Nodes similarities: Some of the approaches (such as in (Djamaludin *et al.*, 2013) or (Wahab *et al.*, 2020)) are relying on the similarities between two entities for assigning the initial trust score. Moreover, the author in (Wang, 2020) includes social proximity between entities in the offline world. In (Djamaludin *et al.*, 2013), the focus is on common friends between two entities. However, in a decentralised M2M community, it is assumed that there are not so many existing social relationships between new joining entities and existing ones.
- The referral and guarantee mechanism presented in (Nguyen *et al.*, 2012) are also not efficient and secure because considering only the behaviour of the service in past communities without considering its initial behaviour is not enough to provide a reliable trust level of services.

- The authors in (Yu *et al.*, 2015) lack information regarding the evaluation steps taken by the evaluation agents for assigning the initial trust score for new agents.

5.2.2 Integration of Functional Testing for Initial Trust

The previous sections have outlined the need for establishing trust in the initial states of M2M service provisioning. The initial trust score problem has been overlooked so far in existing M2M/IoT approaches (see chapter 3). Thus, new M2M services and service providers join the network with no or generalised trust information. Few approaches in other domains (see the previous subsection) dealing with this issue provide only a limited solution to be applied for the M2M domain. However, they do not provide a fully independent and decentralised trust management system and mostly rely on recommendations or trust scores from similar services, which are not reliable sources for assigning initial trust scores.

No initial trust information in the M2M community exhibits shortcomings. The service creation process of the end-user results in a new single or composed service in the community. In comparison with existing services, there is no prior knowledge of the new service in the community, nor are there some recommendations/observations or historical data, which could give a short overview of the behaviour of the new service. Moreover, there is no transaction list and no rating score about the new service compared to existing ones in the community. This knowledge gap about the service could lead to enormous problems such as security attacks performed by newly joining end-users. The M2M community and the participants are not able to check whether they can trust the new M2M entities. Therefore, it is recommended to provide an approach for evaluating the trust

score of newly provided services in the community to enable other entities to decide whether to trust the service and start an interaction or ignore and ban it out of the community.

This research project proposes to test services immediately after publication and use the testing outcomes to create an initial trust score. The testing process includes the integration of functional tests. The functional testing concept (already introduced in section 4.3) consists of end-users acting as test agents and their test environment (Test Engine (TTE) and Trust Engine (TUE)). All end-users of the M2M community can act as test agents and perform test activities on other M2M services independently. In general, certain activities of the M2M services are reviewed and evaluated through so-called test agents. The test environment handles the whole testing process by analysing the service information, producing test cases, executing them, and analysing the test outcomes. Together with defined criteria, these outcomes are used to compute the initial trust score of a service and verify whether or not the service that entered the community is trustworthy. The whole test process should confirm the functionality of the new M2M service and forms the starting point on whether to trust it or not. Thus, the initial behaviour of an M2M service represents its initial trust score. The trust scores are shared among the M2M participants and stored in the P2P storage. The M2M community also uses this trust information for other related trust issues after the service is already part of the M2M community. Moreover, the initial trust scores are part of the process when evaluating the ongoing trust scores of M2M entities (will be explained in chapter 6).

5.2.3 A Perspective for Performance Testing Integration within the Trust Model

Different trust metric parameters are defined in the trust management approaches presented in chapter 4, which are not completely suitable or enough for trust evaluation in decentralised M2M environments. As mentioned, they are also limited only to the computation of ongoing trust scores. This section analyses several testing methodologies that could be used to evaluate the trustworthiness of new M2M services. Moreover, it introduces a novel combination of test techniques and identifies additional trust metric parameters for new M2M services.

Model-based testing comes with several benefits (as mentioned in chapter 2), such as automatic test case generation or the system behaviour specification. After an end-user provides a new M2M service, it should be tested. Basically, functional testing is done against the new M2M service. This test should verify its functionality and consider the SUT as a black box by analysing only the input and output values. Testing the functionality of an M2M service provides the first impression of its behaviour. The previous section states that the initial behaviour is used to build a first initial trust score of the M2M service. However, considering only the basic functionality for trust evaluation of new services is not enough and should be improved by further trust metric parameters.

Another important aspect of testing the system requirements is security testing. This kind of testing can be classified as security functional testing, which validates the correct implementation of security features in the system and security vulnerability testing, trying to identify unintended system vulnerabilities. (Felderer *et al.*, 2016) This research focuses

on decentralised M2M services, which are created by end-users with basic or no technical background. Therefore, it can be assumed that decentralised M2M services do not contain security features that have to be verified using security functional testing. The second category of security testing is security vulnerability testing and could be interesting for this research because unintended vulnerabilities can happen in end-user provided M2M services. However, the authors in (Felderer *et al.*, 2016) state that security vulnerability testing "requires more specific testing techniques" by manually defining and evaluating several attacks. This is incompatible with an automated framework considering the decentralised and distributed architecture in the M2M community. The authors in (ETSI, 2015) describe several activities that are part of security testing, such as risk assessment and risk-based security testing, functional testing of security features, performance testing, robustness testing, penetration testing. Most of these activities focus on testing the security attacks or their impact on the system under test. In contrast, performance testing verifies that the system under test can handle a constant load of a service request without being affected to respond within a required response time (ETSI, 2015). Moreover, performance testing aims to find the performance drawbacks of the system and provides the possibility to identify the stress level "that will result in denial of service" (ETSI, 2015). This leads to the assumption that good performance results of new M2M services indicate higher trustworthiness and provide the possibility to identify the willingness level of a service to participate in interactions with others.

This research project proposes to combine functional and performance testing results to compute the trust score of new M2M services. As mentioned above, the functional testing step is accomplished using model-based functional testing where, based on the system model, adequate test cases are generated and used for test execution. A model-based

approach can also be used for performance testing by building so-called performance models from system components and interactions. After the end-user creates the new M2M service, the tester will verify its correct functionality. Moreover, the tester will do performance testing to confirm the participation willingness of the M2M service. Then these results are combined and calculated to obtain the final verdict. For example, the M2M service will successfully pass the functionality test and respond positively to a predefined number of requests using performance testing. This gives a first trust overview about the initial behaviour of the M2M service and can be used later for computing the ongoing trust score of the already established M2M service.

This research project proposes a general concept of how performance testing can be used in combination with functional testing to enforce a reliable trust evaluation of new M2M entities in the community. Moreover, this research initially describes how this kind of testing can be mapped to the already presented framework for functional testing and trust evaluation. Future work could include a more in-depth analysis of other non-functional testing possibilities and detailed research on performance test case generation in decentralised networks. Thus, the findings presented in this subsection serve first to complete the initial trust evaluation campaign within the scope of this research and for future research directions (as this topic seems to be overlooked in the M2M/IoT literature).

5.2.4 Workflow for Building Initial Trust Scores

The previous section indicated that the functional behaviour and performance characteristics of an M2M service are useful information to evaluate its initial trust score.

The tests are generated based on available service information. It should also be mentioned that, when considering new M2M entities, there should be a distinction between new M2M services provided by a new joining service provider and new M2M services from already established providers.

- New service versus new service provider: the functional behaviour and the service performance are the only indicators to evaluate the initial trust score of the new M2M service and assign this score to the provider.
- New service versus existing service provider: The overall ongoing trust score of the service provider is considered in the calculation together with the functional/performance trust evaluation results.

The following will consider the process for initial trust evaluations in the M2M community. Figure 5.4 shows the workflow of the initial trust score evaluation using functional and performance testing.

An end-user provides a new M2M service (1) and stores the service information in the P2P network (2). Afterwards, the community participants are notified that a new service is available (3). Then, every interested community member can act as a test agent and evaluate the new M2M service. First, using the service information retrieved from the P2P storage (4), each test agent will generate (5) appropriate test cases (methodology described in section 4.3) and execute them on the new M2M service (6). The testing outcomes will be analysed (7) and used to compute the initial trust score of the M2M service. The trust score is then sent to the P2P network for sharing with others. In this illustration, many test agents are performing the test and trust evaluation activities. Other end-users have the possibility to use these scores to decide whether or not to use the new

service. Moreover, these scores can also be used in combination with other trust scores of M2M service, which are concurrently evaluated by other end-users acting as test agents.

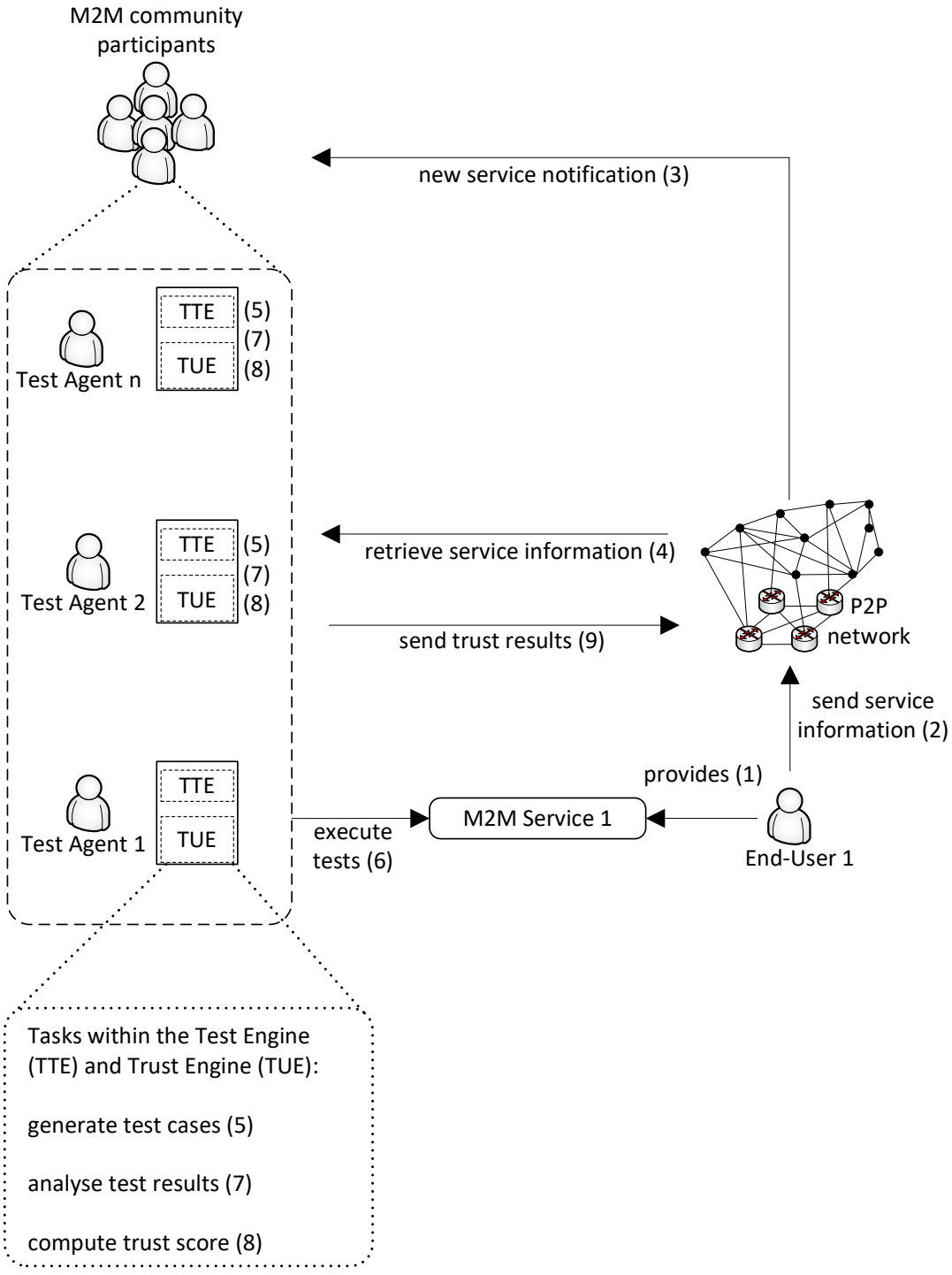

Figure 5.4: Functional tests for initial trust score evaluation

Section 6.2 provides the mathematical model and further information regarding trust computation and aggregation.

5.3 Conclusion

This chapter proposed the integration of service testing within the trust evaluation process. The overall reliability of the trust relationships between the M2M entities even in the initial stages is increased through this synergy.

Section 5.1 focused on blockchain technology and its usability in the M2M domain. First, this section presented a literature review of existing blockchain-based trust approaches and outlined their benefits and limitations. Then, section 5.1 introduced the general concept to use blockchain for trust management optimisation. Thus, trust for the different M2M services and service providers are evaluated and stored trustworthy among the nodes. This enables a trustworthy trust evaluation system to provide trustworthy values to the M2M community. Section 5.1 also identified some issues with consensus protocols used within blockchain technology. Therefore, a comprehensive review of existing consensus approaches within the DLT-domain is done, and relevant outcomes are presented. Based on the evaluation, this research proposed a novel trust-based consensus protocol called Trust-CP. It comprises beneficial elements of existing protocols and integrates trust in different steps of block creation and consensus-building. The introduced Trust-CP avoids wasting resources on computation and increases trust in the whole process. Throughout this research, the Trust-CP is not only used within blockchain technology, but also for trust evaluation and M2M community activities.

Section 5.2 initially outlined the benefits of combining testing and trust processes to assess trust for new M2M entities. It highlighted the lack of initial trust score evaluation in existing M2M trust approaches and presented a literature review in M2M adjacent domains. The outcomes of this review were pointed out, and the integration of functional testing for service testing at the initial stages of service provision was proposed. Performance testing was also incorporated within the trust evaluation process to optimise the significance of the initial trust score. Finally, section 5.2 presented the trust-building process for new M2M services and service providers.

This chapter forms one pillar of the trust evaluation system that will be introduced in the next chapter. Moreover, the service testing part will be integrated not only for initial trust score evaluation but also for computing ongoing trust scores of M2M entities.

6 Comprehensive Trust Evaluation System

Chapter 4 and 5 introduced the principles for performing tests within decentralised M2M communities and presented the integration of test activities within the concept for trust evaluation. This chapter comprises the main parts of the framework for trust evaluation of decentralised M2M services.

Section 6.1 introduces the enhanced trust evaluation system for decentralised M2M services and peers. The trust evaluation system covers different aspects of service providers and services. The already presented testing approach (chapter 5) is merged with the trust evaluation part. Furthermore, activities such as monitoring and rating M2M services are introduced. Section 6.1 also proposes to consider the behaviour of peers within the M2M community when evaluating trust. This comprises the integrity of the services they provide and their effort to participate in various community tasks. Additionally, this section presents the different parameters and their mathematical representation. The outcomes of sections 5.1, 5.2, and 6.1 are used to present a novel blockchain-based trust evaluation approach in section 6.2. Furthermore, section 6.2 describes the scheme for computations and aggregations of trust scores. The scheme includes steps to calculate the initial trust score of new M2M services and the overall trust score of existing M2M services/service providers.

Some parts in this chapter have been published in (Shala *et al.*, 2019c; 2020a).

6.1 Enhanced Trust Evaluation for New and Existing M2M Services

Chapter 4 provided a general overview of the proposed trust model. The idea of the trust model is to evaluate the trust scores of M2M services and M2M service providers. A special focus is given to the trust score of new M2M entities. The evaluation is handled using the concept proposed in chapter 5 and considers the integration of service testing within the trust evaluation scheme. Another aim is to ensure that the trust data is stored tamper-proof among the nodes. Therefore, the previous section has introduced a blockchain-based approach to ensure data integrity. An M2M service provider can provide many M2M services. Thus, the overall trust score of an M2M peer can be composed by considering service-related and peer-related components. Service-related components consist of the performance of a service in different trust aspects. The peer-related component includes the behaviour and the activities of the M2M peer within the M2M community. Both components are using several metrics and metric parameters to identify the capabilities of the different M2M entities in different situations.

The next two sections will provide more details regarding the service- and peer-related trust evaluation. It will present the different trust metrics and parameters used for the evaluation. Moreover, the relations initial/existing trust score and service/peer trust score will be explained.

6.1.1 Service-Related Trust Evaluation

The M2M community consists of different M2M peers providing services to each other. Thus, M2M services have high relevance in terms of building trust relationships between the nodes. This research proposes to cover several aspects of a service (as stated in section 4.4), starting from its functional behaviour, continuing with performance and monitoring results, and concluding with ratings from other end-users.

Service Testing

Chapter 3 revealed the unsolved initial trust score problem of existing approaches. This research thesis enables the computation of initial trust scores for new M2M services through service testing. However, M2M service testing is not only used for new M2M services and initial trust scores. It is also part of the trust evaluation for ongoing trust scores. Service testing includes the verification of the service functionality and the evaluation of its performance. The presented approach initially considers service acceptance and service response time for the performance evaluation. The trust metric, the sub-metrics, and their corresponding symbols are shown in Table 6.1.

Table 6.1: Trust metric Service Testing

Trust Metric	Trust Sub-Metric	Symbol
Service Testing	Functional testing	S_{ft}
	Response time	S_{ptnrt}
	Service acceptance	S_{ptar}

The functional behaviour of an M2M service is depicted in its service description, which serves as a basis for generating appropriate test cases (see section 4.3). The tests can result in pass or fail verdicts. The test execution report expresses the pass verdicts of all test cases in percentage. These results are used in the proposed model and are scaled (feature scaling) to the value range from 0-1 (worst to best). For standard scaling purposes, the following equation is used:

$$s = \frac{s_i - \min(s_i)}{\max(s_i) - \min(s_i)} \tag{6.1}$$

where: s is the normalised value; s_i non-normalised value (test result); $max(s_i)$ is the maximum value; $min(s_i)$ minimum value;

A transformed form of the previous equation (1) is used to compute the score for functional testing:

$$S_{ft} = \frac{S_{tr} - \min(S_{tr})}{\max(S_{tr}) - \min(S_{tr})} \tag{6.2}$$

where: S_{ft} is the normalised score regarding the functional behaviour; S_{tr} is the test result after functional testing; $max(S_{tr})$ is the maximum possible test score (equal to 100); $min(S_{tr})$ is the minimum possible test score (equal to 0).

The performance capabilities of an M2M service are also included in the service description. One important performance indicator is accessibility which partially can be expressed by the response time (idea adapted from (Nakahira *et al.*, 2015; Ma and Wang, 2016)) in comparison to the maximal response time (defined in the service description of

the service). Another indicator is the service acceptance derived from the relation of positive responses and total requests.

The following equation has been defined for the response time of an M2M service:

for $S_{ptrt} > \max(Resp_{time})$

$$S_{ptnrt} = 1 - \frac{S_{ptrt} - \max(Resp_{time})}{\max(Resp_{time})} \tag{6.3}$$

for $S_{ptrt} < \max(Resp_{time})$ then $S_{ptnrt} = 1$;

for $S_{ptrt} > 2\max(Resp_{time})$ then $S_{ptnrt} = 0$;

where: S_{ptnrt} is the normalised score regarding the response time of the service; S_{ptrt} is the response time of the service; $\max(Resp_{time})$ is the maximal response time.

The following equation has been defined for the service acceptance:

$$S_{ptar} = \frac{Resp_{pos}}{Req} \tag{6.4}$$

where: S_{ptar} is the score regarding the service acceptance; $Resp_{pos}$ are the number of positive responses; Req is the number of requests.

Service testing is done for new and for existing M2M services (see Figure 6.1). For new M2M services, this process forms the initial trust score. For existing ones, it forms the partial trust score (service trust score as stated in section 4.4). The ongoing trust score is computed using the aggregation of partial trust scores for the different trust metrics.

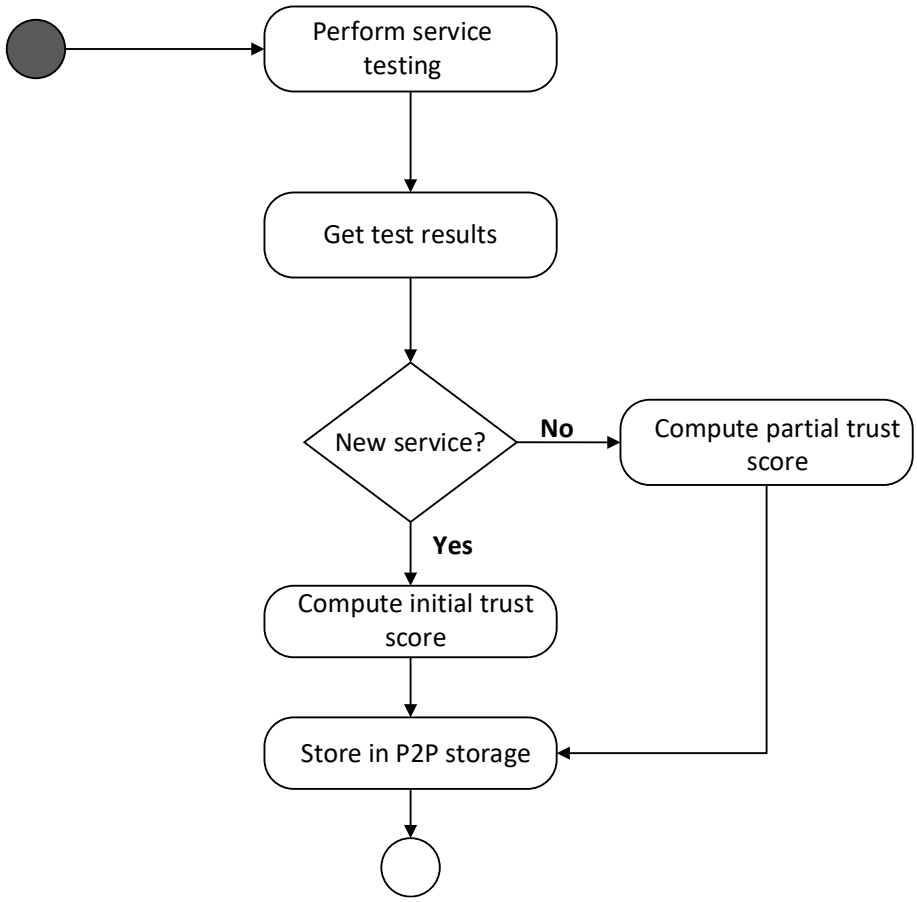

Figure 6.1: Service testing for new and existing M2M service

Service Monitoring

Service monitoring is another important metric used for evaluating the ongoing trust score of services. M2M services are continuously monitored by other participating nodes (acting as test agents). The test agent can directly or indirectly acquire the necessary information for the monitoring. Directly means that the test agent establishes a direct interaction with the service under monitoring and its service provider. Indirectly includes information from the P2P storage or other acting test agents. Figure 6.2 shows a simplified workflow for service monitoring.

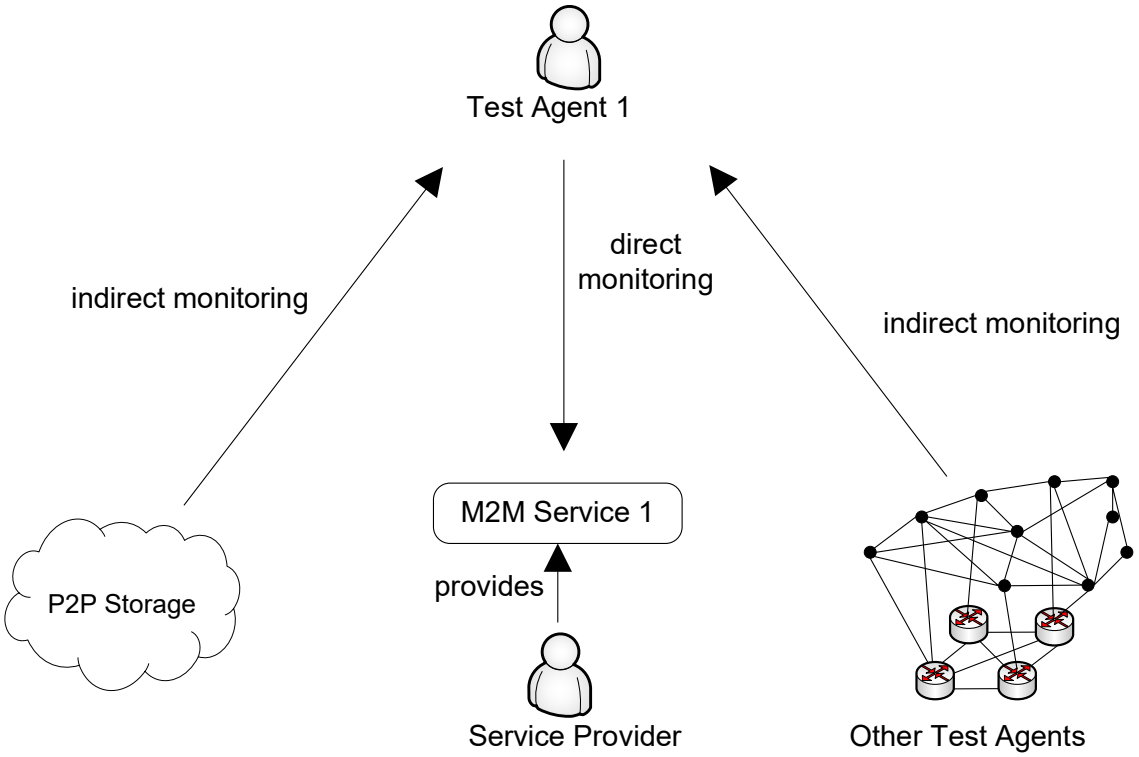

Figure 6.2: Direct and indirect service monitoring

This research proposes a set of sub-metrics for service monitoring that can be divided into two categories: *availability* and *activity*. Table 6.2 shows the trust metric, sub-metrics, and the corresponding symbols.

Table 6.2: Trust metric Service Monitoring

Trust Metric	Trust Sub-Metric	Symbol
Service Monitoring	Service online/offline actions	S_{mtava}
	Service uptime	S_{mtavt}
	Number of times a service is used	S_{mtacn}
	Ratio positive responses	S_{mtar}

The service monitoring category *availability* includes the time a service is online since its deployment and the number of online/offline actions a service takes (idea adapted from (ITU-T, 2015)). The following equation for service online/offline has been defined:

$$S_{mtavt} = \frac{t_{up}}{t_{up} + t_{down}} \tag{6.5}$$

where: S_{mtavt} is the score regarding the service uptime; t_{up} is the uptime; t_{down} is the downtime.

For the online/offline actions, the following equation has been defined:

$$S_{mtava} = \frac{N_{oa}}{M_a} \tag{6.6}$$

where: S_{mtava} is the score regarding the online/offline actions; N_{oa} is the number of online actions; M_a are the monitoring actions.

The service monitoring category *activity* consists of the number of times others use a service for a predefined period and the number of positive responses (idea adapted from (Nakahira *et al.*, 2015; Ma and Wang, 2016)) handled by the service. The following equation is about the number of times a service is used:

for $N_{sa} < N_{saaver}$

$$S_{mtacn} = \frac{\frac{N_{sa}}{t_{mon}}}{\frac{N_{saaver}}{t_{monav}}} \tag{6.7}$$

for $N_{sa} > N_{saaver}$ then $S_{mtacn} = 1$;

where: S_{mtacn} is the score regarding the usability of a service; N_{sa} is the number of service utilisations; N_{saaver} is the average number of service utilisations; t_{mon} is the monitoring time of the service, t_{monav} is the average monitoring time of services.

The following equation represents the number of positive responses handled by a service:

$$S_{mtar} = \frac{Resp_{pos}}{Req} \tag{6.8}$$

where: S_{mtar} is the score regarding the service acceptance; $Resp_{pos}$ are the number of positive responses; Req are the number of requests.

Figure 6.3 summarises the different trust sub-metrics used for service monitoring. The outcomes are monitoring results that will be scaled and aggregated to compute the partial trust score.

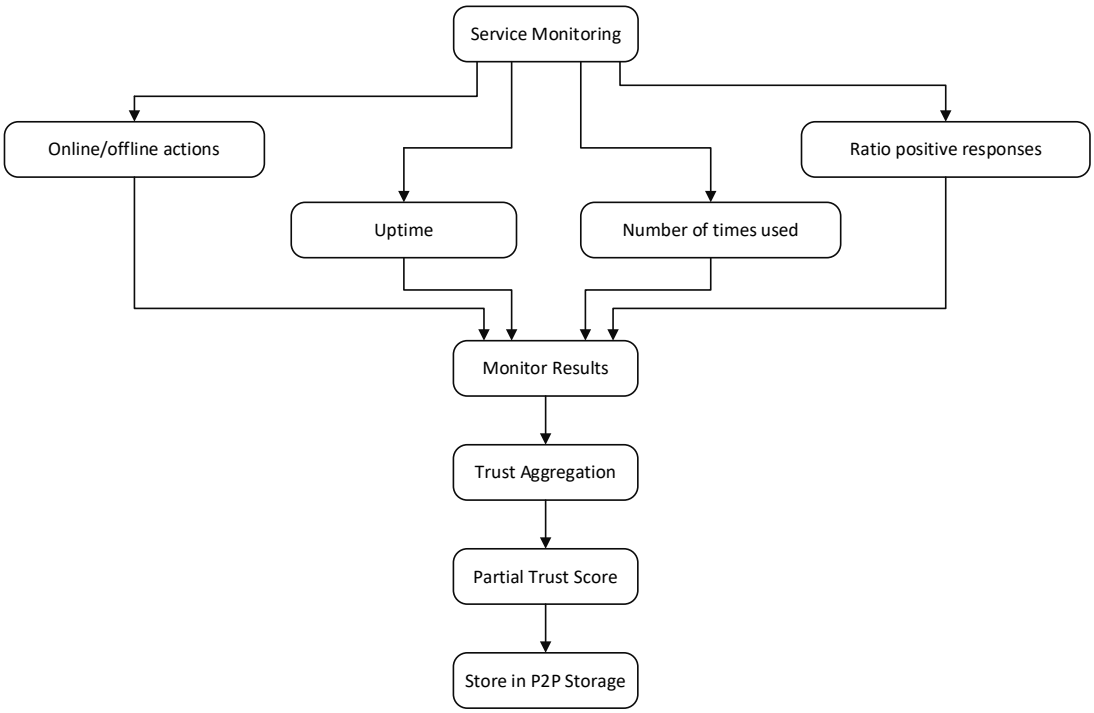

Figure 6.3: Service Monitoring

Service Rating

This research proposes to consider the service experience of end-users for rating an M2M service. Table 6.3 summarises the details regarding the metric, sub-metrics, and the corresponding symbols.

Table 6.3: Trust metric Service Rating

Trust Metric	Trust Sub-Metric	Symbol
Service Rating	Service satisfaction	S_{ratint}
	Number of successful interactions	I_{suc}

First, the rating is done by expressing the level of service satisfaction (S_{ratsat}) for using an M2M service. The range 0 to 1 indicates that level, where 0 stands for not satisfied and 1 for satisfied. Another sub-metric is the number of successful interactions (idea adapted from (Ma and Wang, 2016)) between a service provider and service consumer. The following equation shows this relation:

$$S_{ratint} = \frac{I_{suc}}{I_{tot}} \tag{6.9}$$

where: S_{ratint} is the score regarding the service interactions; $I_{suc}$is the number of successful interactions; I_{tot} is the total number of interactions.

Service rating is considered a subjective metric. Each test agent may have a different view and expectation regarding the use of an M2M service. This fact should be considered when calculating the overall trust score of an M2M service by adjusting the weighting parameter for the scores resulting from the service rating. Besides them, this approach

proposes to combine the trust score of the rater with its rating score. This ensures that test agents with a higher trust score will be weighted more heavily (details on weighting will be provided in section 6.2). It is assumed that nodes with a higher trust score have a better experience and longer lifetime in the M2M community. Therefore, their subjectivity can be considered to be more in line with the objectivities of the whole community members.

6.1.2 Peer-Related Trust Evaluation

The service-related trust score is one element for building the overall trust score of a peer. The other element covers the peer appearance in the M2M community. Thus, this research proposes to focus on two main peer indicators: the behaviour of the peer in terms of data integrity (behaviour trust) and the participation willingness of the peer in different community tasks (task trust). The next subsections will provide more details regarding these two indicators.

Peer Integrity Check

For behaviour trust, this research proposes the so-called peer integrity check as a trust metric. The idea is to check the integrity of different M2M community information and use these outcomes for trust evaluation. As stated in chapter 4, this research uses a P2P storage built by the blockchain and a DHT-based storage. As a result, the level of integrity is increased, decentralisation in the M2M community is maintained, and more flexibility for data lookups is enabled. The different data, service- and trust-related, are stored in the blockchain (on-chain) and also outside the chain (off-chain) in the DHT. The peer integrity check metric catches the functionality of the P2P storage. Peers acting as test agents in the M2M community have the possibility to check the integrity of this data

continuously. This means that data in the off-chain is compared with the same data in the on-chain. Data in the blockchain are tamper-proof stored. If there are differences in this integrity check, it indicates that information has been changed in the DHT storage (off-chain). To determine the offender, the "last edited information"-field in the off-chain is used. As a result, the peer performing the falsification of the data is punished by getting a low trust score.

The sub-metric *service information integrity* is related to the integrity checks for the different service information (service description, service provider contact information). The same also applies to the sub-metric *trust information integrity,* which focuses on the integrity of trust data. Peer integrity checking is not limited to the two proposed sub-metrics. Further sub-metrics under the category of peer integrity can be added to the framework. Table 6.4 summarises the peer integrity check metric with its sub-metrics and corresponding symbols.

Table 6.4: Trust metric Peer Integrity Checking

Trust Metric	Trust Sub-Metric	Symbol
Peer Integrity Checking	Service information integrity	S_{inch}
	Trust information integrity	S_{tinch}

In the following, the equation for the service information integrity is shown.

$$S_{inch} = \frac{M_{corr}}{C_{tot}} \tag{6.10}$$

where: S_{inch} is the score for service information integrity; $M_{corr}$are the correct matches; $C_{tot}$is the total number of checks.

Figure 6.4 shows an overview of the metric *Peer Integrity Checking* and its workflow.

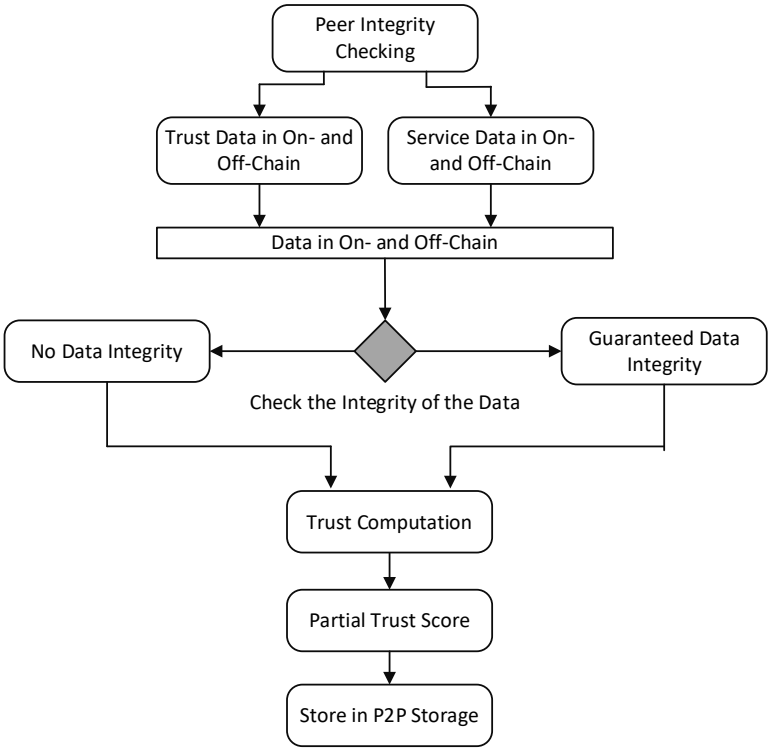

Figure 6.4: Workflow for Peer Integrity Checking

Peer Task Participation

As mentioned, the M2M community consists of M2M services and different M2M peers acting as service providers. Moreover, within the community, there are several actions to be taken by the nodes. Two of them are testing and trust evaluation activities. Moreover, the M2M community has a P2P storage which has to be maintained by the whole network. For instance, in the blockchain, there are several tasks to be done, such as collecting transactions, creating blocks, performing verification and validation, and achieving consensus (Trust-CP) between the nodes. To sum up, a decentralised M2M community is maintained by all its members, and the maintenance includes several tasks. In order to keep the decentralised community alive, the participation of the members is required.

This research proposes distributing different tasks to all the community members to avoid centralised instances within the network. Additionally, this research suggests considering the participation in community task for the trust evaluation of peers. Based on the active involvement in multiple tasks, a peer can increase its trust score. This plays a significant role as each peer aims for a high trust score to make their own services more attractive to others and also to gain access to other services from other M2M service providers. As a result, peers are motivated to participate in different community tasks.

Table 6.5 shows the metric, sub-metrics, and the corresponding symbols used for the trust evaluation process.

Table 6.5: Trust metric Peer Task Participation

Trust Metric	Trust Sub-Metric	Symbol
Peer Task Participation	Test agent activities	S_{tpt}
	Blockchain node activities	S_{tpb}

The peer task participation consists of measuring the effort of the end-user in performing various community tasks. This research considers two sub-metrics in this category of trust evaluation: participation as a test agent for testing and trust activities and participation in different blockchain tasks (as a blockchain node). In the following, the equation for peer task participation (test agent activities) is presented, which can be applied to both test agent and blockchain node activities:

$$S_{tpt} = \frac{\frac{N_{tp}}{t_{mont}}}{\frac{N_{tpaver}}{t_{montav}}} \tag{6.11}$$

where: S_{tpt} is the score for participation in test agent activities; N_{tp} is the number of tasks done; $N_{tpaver}$is the average number of average tasks done; t_{mont} is the monitoring time of a task; t_{montav} is the average monitoring time of tasks.

Figure 6.5 shows a generalised view of the different supporting tasks performed in the M2M community. To sum up, every end-user can act as a test agent and blockchain node. The test agent activities cover the tasks of service testing and trust evaluation. The blockchain node activities cover the tasks of creating, sending, validating, and maintaining transactions and blocks in the network.

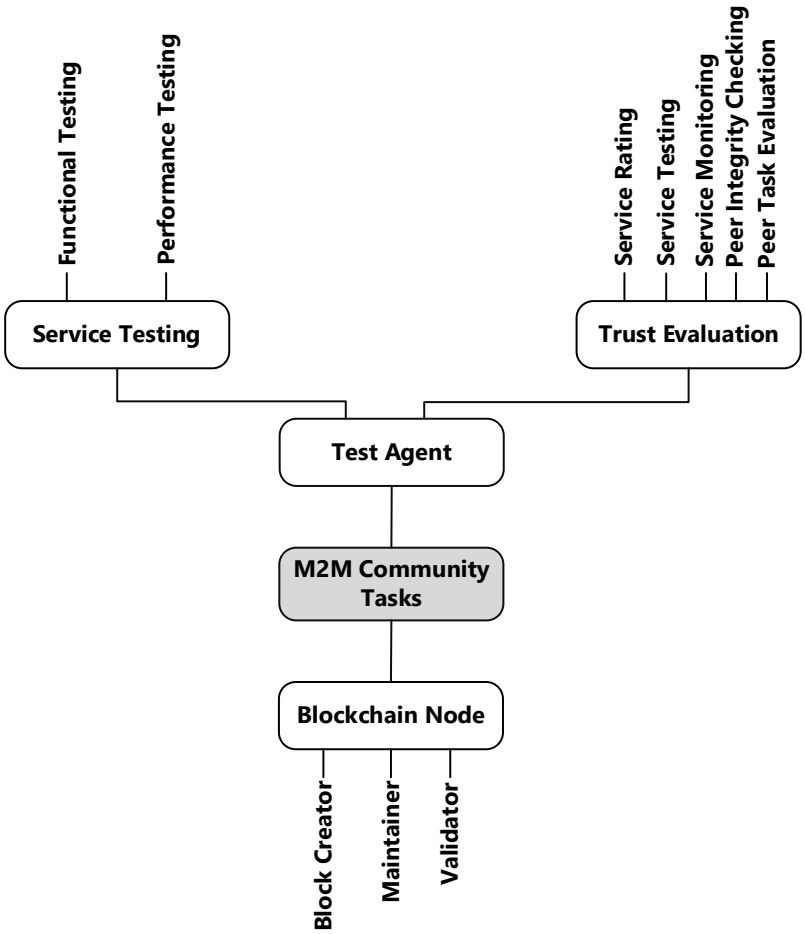

Figure 6.5: Main M2M community tasks

6.2 Trust Evaluation, Computation, and Aggregation

The previous section introduced a number of different trust aspects of a peer. Each M2M service can have its service trust score. An M2M peer can have partial peer trust scores for a specific trust metric (such as peer integrity or peer task participation). The evaluation of the services or peers using one of the proposed trust metrics concludes with a test, monitor, rate, integrity, or task result (based on the considered trust metric, as introduced in section 6.1). These results have to be scaled to the range 0 to 1 to fit into the trust score assignment. All these results are used to compute the overall trust score of a peer.

The next subsections present the methodology for trust evaluation, the initial and ongoing trust score computation, and a trust score aggregation scheme. Furthermore, the blockchain and the proposed Trust-CP are incorporated within the trust evaluation system to optimise the workflow.

6.2.1 Blockchain-Based Trust Evaluation

This research project identifies three ways for performing the trust evaluation in the M2M community:

1. Independent trust evaluation: Each M2M participant performs the trust activities and computes the trust scores of other M2M peers and services. The trust evaluation is done by a peer independently from others.
2. Consensus-based trust evaluation: The proposed Trust Consensus Protocol (T-CP) is used outside the blockchain to achieve consensus among the nodes regarding the computed trust score.

3. Blockchain- and consensus-based trust evaluation: The whole trust evaluation process is done using blockchain principles.

Each of the three possibilities can be used to evaluate the trust score of M2M entities. However, the benefits of blockchain technology and the proposed Trust-CP has been highlighted in the previous sections. Data is stored securely and decisions on trust evaluation outcomes are made in a trustworthy manner. Therefore, this research project follows the third way to evaluate trust and build trust relationships between the nodes.

Each end-user within the M2M community can act as a test agent to perform test and trust activities. The outcomes of these activities are stored in the blockchain to enable a trustworthy storage system for trust data. The trust evaluation of services and peers is done using one of the trust metrics proposed in section 6.1. This research project proposes to use the blockchain workflow for performing trust evaluation activities. The workflow of the proposed approach is shown in Figure 6.6, and the corresponding steps are listed in the following:

1. Every end-user evaluates other services or peers using one of the defined trust metrics (see Figure 6.6, part I).
2. The evaluation results (test, monitor, rate, integrity, or task) are sent as blockchain transactions to the network. These transactions consist of information about the evaluated service/peer and the evaluating end-user (see Figure 6.6, part I).
3. The blockchain transactions are broadcasted to all members and stored with an unconfirmed status in their local memories (also known as transaction pool). These transactions have to be included in a new block by a block creator (as per

the blockchain principles) and confirmed by the community (see Figure 6.6, part I).

4. Over time, many transactions from different test agents are part of the transaction pool, containing relevant trust information about services and service providers. (see Figure 6.6, part II).
5. Each community member can use this information to aggregate an overall trust score for a specific service or peer using one of the following two options (see Figure 6.6, part II):
    a) The unconfirmed transactions are included in a block. The creation of the block and the consensus are done using the proposed Trust-CP (section 5.1.4). The newly created block appears in the blockchain. Each blockchain node can use the trust information stored in that block to aggregate an overall trust score for an M2M entity.
    b) The information from the unconfirmed transactions is taken and used to aggregate an overall trust score. Both unconfirmed transactions and overall trust score are included in a new block. Afterwards, the Trust-CP is applied to confirm that block.
6. Regardless of which one of the two options is chosen (5a or 5b), the collected scores inside the transactions are used to create the overall trust score using a weighting system for each trust metric part of the current calculation (which is further described in the next section).
7. Any member of the M2M community can create a new block containing the current trust score of a service or peer. However, the new block is only positively confirmed by the community if the block creator fulfils the Trust-CP

requirements. The main requirements are that the block creator has a high trust score, and only transactions from high trusted senders are included in that block.

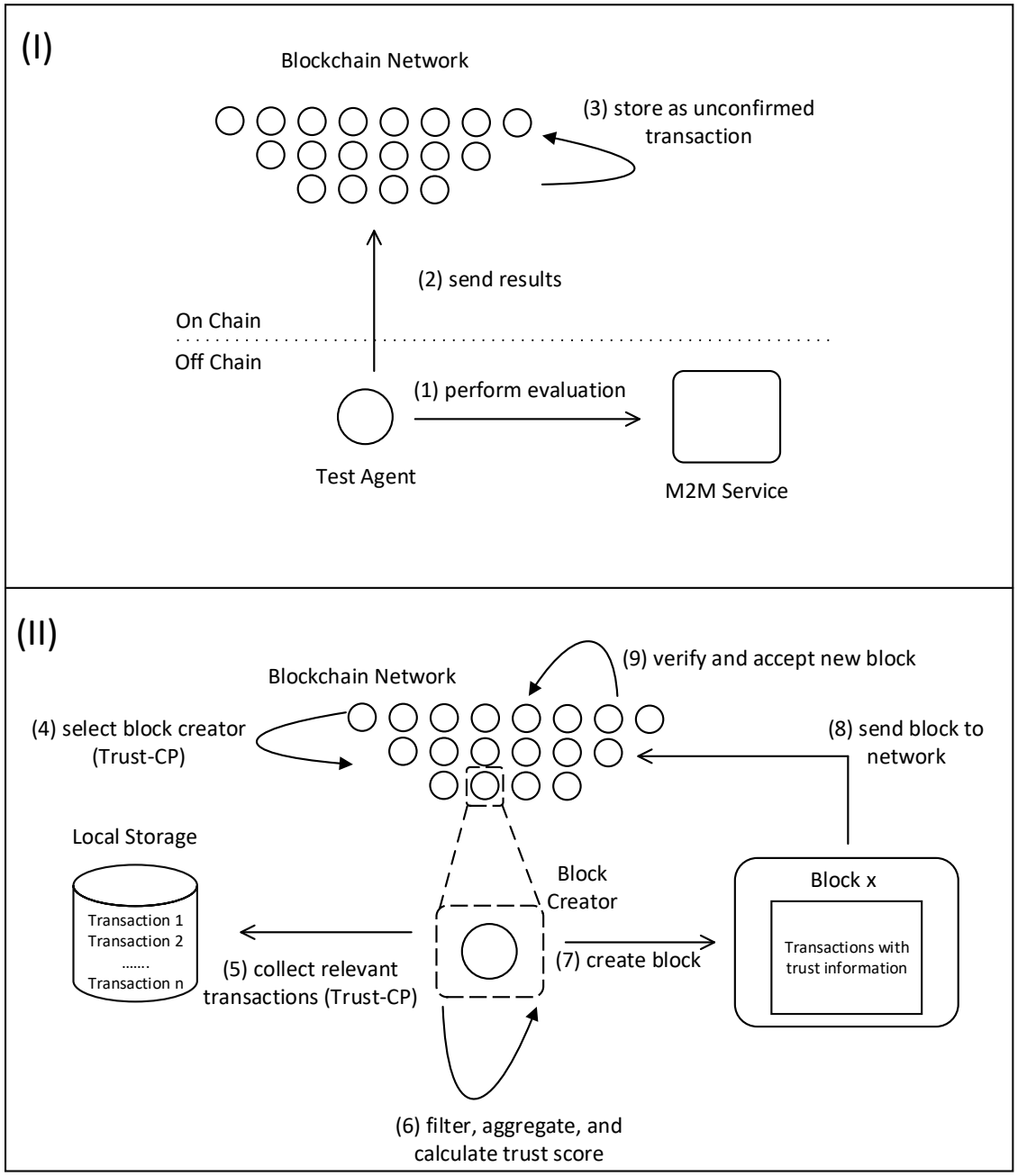

Figure 6.6: Workflow of blockchain-based trust evaluation

8. The aggregation of the overall trust score can be done in predefined timeslots (time-driven) or when a specific number of transactions is reached (event-driven).

9. Afterwards, the current overall trust score of a service/peer is stored securely in the blockchain. Besides them, the previous trust scores are also stored in the blockchain. Each community member can query the blockchain for trust information.
10. Specifically, suppose an end-user or service consumer wants to know the current trust score of a service. In that case, a trust score request is sent to initiate the trust score evaluation (which triggers the whole workflow as described here).

6.2.2 Scheme for Trust Computation and Aggregation

The previous sections presented the structure and the workflow of the proposed trust evaluation approach. This section describes the mathematical model used to compute the trust scores of M2M services and peers. The computation part explained in the following covers the initial and the ongoing trust score as well as the relation of both to get the overall trust score. As stated in section 4.4, the trust evaluation process results in a trust score ranging from 0 to 1, indicating the level of trustworthiness. Therefore, this research initially defines the following levels of trustworthiness (adapted from (Trustpilot, 2021)): 0 to 0.34 (bad trust score - untrustworthy); 0.35 to 0.54 (poor trust score - untrustworthy); 0.55 to 0.74 (average trust score - trustworthy); 0.75 to 0.84 (good trust score - trustworthy); 0.85 to 1 (excellent trust score - trustworthy).

Initial Trust Score

As mentioned, there are two scenarios for new M2M entities. The first one is a new service provider that joins the M2M community and provides a new service. The second scenario is an existing service provider deploying a new M2M service.

The functional behaviour and the service performance are the metrics considered for the trust evaluation when a new service provider launches a new M2M service. The weighting of the two sub-metrics is set according to their importance. It can be argued that the service performance loses its relevance when the service fails to provide its functionality. If the service is not working correctly, the consumer will not pay further attention to other characteristics, such as performance. However, if the service works well, then the performance can affect the satisfaction of the consumer. Performance testing is considered in the weighting because it could be that a service is partially working, or, e.g. 90% of the functionality is working correctly (not all test cases are passed). The following equation shows the calculation of the initial trust score for a new M2M service provided by a new peer:

$$T_{init}^{np} = \mu_{st} S_{st} + \mu_{pt} S_{pt} \tag{6.12}$$

where: T_{init}^{np} is the initial trust score of a new service provided by a new peer; S_{st} is the score for service testing; S_{pt} is the score for performance testing.

The trust evaluation of a new M2M service launched by an existing service provider also includes the current trust score of the service provider. Similar to the case of a new service announced by a new service provider, service functionality has a higher relevance for service consumers compared to performance features or already existing trust scores of service providers. The following equation shows the calculation of the initial trust score for new services provided by existing service providers:

$$T_{init}^{ep} = \mu_{st} S_{st} + \mu_{pt} S_{pt} + \mu_{et} P_{et} \tag{6.13}$$

where: T_{init}^{ep} is the initial trust score of a new service provided by an existing peer; P_{et} is the trust score of the existing peer; μ_{st} is the weight for service testing; μ_{pt} is the weight for performance testing; μ_{et} is the weight for the existing trust score of the service provider.

Ongoing Trust Score

Besides the initial trust score, there is also the ongoing trust score for M2M entities already participating for a while in the M2M community. In order to evaluate the ongoing trust score of an M2M peer (service provider), all the different scores derived from service- and peer-related aspects are to be considered. Each of the trust metrics (service testing, service monitoring, service rating, peer integrity checking, and peer task evaluation) has a different impact under different conditions on the overall trust score of an M2M entity. Therefore, it is required to have an appropriate weighting system. This research suggests combining different aspects to present a dynamic weighting system that enables efficient trust aggregation and automatically weighting adjustment based on the current situation in the M2M community. The blockchain-based trust evaluation workflow remains the same as explained in the previous subsection but includes additionally some steps containing detailed information on the block creation and the calculations:

1. Every peer in the M2M community also behaves as a test agent performing test activities and trust activities. Moreover, every peer part of the blockchain network can participate actively in blockchain activities.

2. Peers continuously act as test agents and perform the tests mentioned above regarding the trust evaluation. The test results are sent to the blockchain as blockchain transactions for tamper-proof storage.
3. A peer is selected as a block creator (called a miner in Bitcoin) using the Trust Consensus Protocol (refer to section 5.1.4) and starts collecting unconfirmed transactions from the transaction pool to form a block. In the collection/selection phase, the block creator considers only transactions from peers with average or high trust scores. Other transactions are not considered (sorted out) – this ensures that malicious or untrustworthy peers cannot impact the trust management system, and hence many attacks are mitigated.
4. Before forming the block, the block creator sorts the filtered transactions based on the trust metric category and starts the trust calculations. This includes the weighted average trust score for a specific trust sub-metric. For instance, the block creator collects three transactions consisting of information about the service testing score. Therefore, the average of these three values is calculated. Moreover, the block creator looks for the trust score of the transaction originator to include it in the trust calculation as well. This process is done for all other trust sub-metrics. In the end, the block creator has a list of parameters that are considered for the next steps of the total trust score computation. The following equation shows the above-described process:

$$S_x^{w_r} = \frac{\sum_{i=1}^{n} T_{p_i} S_{X_i}}{\sum_{i=1}^{n} T_{p_i}} \tag{6.14}$$

where: $S_x^{w_r}$ is the weighted service trust score for a trust sub-metric; T_{p_i} is the trust score of the peer who has evaluated the service; S_{X_i} is the trust score assigned by the peer for one of the specific trust sub-metrics (refer to section 6.1).

5. Another point that has to be considered in the preparation of the trust parameters is the number of tests conducted for a trust evaluation round for a specific trust sub-metric. For instance, service testing is done three times and service rating only one time. The weighted service trust score is calculated using the following equation:

$$S_{nt}^{w_u} = \frac{\sum n_{S_x} S_x}{\sum n_{S_x}} \tag{6.15}$$

where: $S_{nt}^{w_u}$ is the weighted service trust score for the service/peer metrics based on the trust score of each of the considered sub-metrics and their frequency; n_{S_x} is the number of inputs for a specific sub-metric; S_x is the score for the specific sub-metric.

6. The next step is to rank the different parameters from the worst to the best value. According to this ranking, the weighting is assigned to the parameters. Parameters with a bad value should be weighted more heavily to motivate service providers in future rounds to provide better services and participate actively and positively in community activities. The ranking (calculation) is carried out in each round and the weighting of the parameters is adjusted accordingly. One round refers to a new calculation of the overall trust score and the corresponding newly created block. A future step could include the trust score of the block creator in the trust

evaluation process. The following equations are used for ranking the trust parameters (are illustrated with the metric Service Testing):

$$S_{test}^{rk} = 1 - S_{test} \tag{6.16}$$

$$\alpha = \frac{S_{test}^{rk}}{S_{test}^{rk} + S_{mont}^{rk} + S_{rat}^{rk} + S_{inch}^{rk} + S_{tp}^{rk}} \tag{6.17}$$

where: $S_{test}^{rk}, S_{mont}^{rk}, S_{rat}^{rk}, S_{inch}^{rk}, S_{tp}^{rk}$ are the ranking values for the metrics Service Testing S_{test}, Service Monitoring S_{mont}, Service Rating, Peer Integrity Checking, Peer Task Participation; α is the weighting parameter for the metric S_{test}.

Similar calculations are also done for the other weighting parameters: β (for the metric S_{mont}); γ (for the metric S_{rat}); δ (for the metric S_{inch}); ε(for the metric S_{tp}).

7. The overall current trust score of a peer is computed by considering all derived scores from the different trust metrics (and calculated in the previous steps) weighted with the corresponding weighting parameters derived in step 6. The following equation shows the calculation process:

$$T_{total}^{cur} = \alpha S_{test} + \beta S_{mont} + \gamma S_{rat} + \delta S_{inch} + \varepsilon S_{tp} \tag{6.18}$$

where: T_{total}^{cur} is the current total trust score of a peer; α, β, γ, δ, ε are the weighting coefficients for the different metrics (refer to the previous point).

8. In order to compute the overall trust score, the current one should also be combined with the old one. Therefore, both scores are weighted according to the average peer trust score (last block and current block) of each block. The following equation expresses the overall trust score of a peer:

$$T_{total} = \frac{1}{n}\sum_{i=1}^{n} T_{p_i}^{old} \times T_{total}^{old} + \frac{1}{m}\sum_{i=1}^{m} T_{p_i}^{cur} \times T_{total}^{cur} \quad (6.19)$$

where: T_{total} is the overall trust score of a peer; $T_{p_i}^{old}$ is the average peer trust score of the last block; $T_{p_i}^{cur}$ is the average peer trust score of the current block; T_{total}^{old} is the previous trust score of a peer.

6.3 Conclusion

This chapter presented a blockchain-based trust evaluation system that is used to determine the trustworthiness of M2M entities within the decentralised M2M community. The proposed trust approach covers the trust status of existing and new M2M entities. Moreover, it is a complete decentralised trust system relying on the competition-based participation of the community members. The introduced trust model considers several service- and peer-related aspects for trust evaluation activities. Blockchain principles are integrated into different parts of the proposed framework to increase reliability and integrity.

Section 6.1 presented the enhanced trust evaluation system for new and existing M2M entities. The proposed system covers different trust metrics, such as service functionality, service performance, and consumer experience. Moreover, the trust evaluation also

includes peer behaviour and peer participation willingness for several community tasks. The different trust metrics are divided into service- and peer-related ones. This section described each of the different trust metrics and sub-metrics as well as the mathematical model used for trust computation. The use of blockchain technology was extended in this section to store service data in addition to trust data. The proposed trust evaluation system comprising several aspects, such as participation in community tasks, increases the overall trustworthiness of the community. A high trust score enables M2M entities to use or to provide M2M services. Furthermore, the trust score can be boosted up through different activities within the M2M community. Thus, the trust evaluation in the decentralised community is switched to a competition-based challenge among the M2M members. This ensures reliable maintenance of the M2M community and increases the quality of the derived trust scores.

Section 6.2 extended the synergy of blockchain and trust. This section proposed to use the blockchain and the previously presented Trust-CP for trust evaluation as well. The outcomes of the various evaluation activities are sent through blockchain transactions among the network. The agreement and validation of these transactions are confirmed by all participants using the proposed Trust-CP. This process ensures that only trusted transactions from trusted nodes are stored in the blockchain. As a result, low-trusted members are not considered, and fake trust score attacks are mitigated. Thus, the combination of blockchain, Trust-CP, and trust makes the M2M community more resilient against attacks and ensures fair and trusted relationships between the nodes. This section has also presented a new scheme for trust computation and aggregation, including a dynamic weighting system. This scheme includes the different derived metric scores and is used to compute the overall trust score of M2M entities.

Decentralised communities usually lack trust between different participants. However, the combination of blockchain and trust breaks down this limitation and optimises the overall security of the network. Chapters 5 and 6 build the core work of this research thesis. They have presented a novel blockchain-based trust evaluation system which comprises the following main benefits: high data integrity; comprehensive trust model based on various service and peer aspects; trusted decision-making and relation-building processes; initial and ongoing trust score; optimised resilience; optimisation of the M2M community maintenance; increased community competition behaviour. Overall, chapter 4, 5, and 6 presented the framework for trust evaluation and corresponding functional testing of decentralised M2M services. The next chapter (chapter 7) will present a proof of concept implementation and evaluation of the main parts of the proposed framework. Chapter 8 will finalise this research thesis by summarising key findings, limitations, and prospective activities.

7 Framework Evaluation and Research Prototype

This chapter outlines the framework evaluation and prototype implementation. Section 7.1 investigates whether the proposed framework for trust evaluation and corresponding functional testing meets the requirements derived from the limitations of related works. Section 7.2 presents the utilisation of the main parts of the framework consisting of functional testing, trust evaluation, and blockchain activities. It also highlights the resilience of the proposed trust and blockchain approach against malicious activities.

7.1 Evaluation of Framework Requirements

Section 3.3 identified several requirements for an optimised framework for trust evaluation and corresponding functional testing. This section evaluates how the proposed framework fulfils these requirements:

- *Test and trust support* – The proposed framework fulfils this requirement as it enables service testing and trust evaluation activities within the M2M community (as presented in chapter 5 and 6). Moreover, trust is an important factor used for different processes among the participants.

- *Decentralised architecture* – All components of the introduced framework are decentralised. The data storage in the M2M community is implemented using P2P

storage. The P2P storage consists of the combination of DHT and blockchain for integrity reasons. Besides, all trust and test activities are done independently by all M2M community members. There is no centralised authority coordinating or controlling the different processes (refer to chapter 4). The decisions in the community are made using a trust-based consensus protocol (Trust-CP). All nodes are equally maintaining the M2M community.

- *Support decentralised and end-user-based M2M services* – The presented framework provides the possibility to test end-user-based and decentralised M2M services. Moreover, the framework also enables the computation of the trust scores of these services. Overall, the framework supports an end-user-based M2M community without centralised elements.

- *End-user-integration* – All end-users part of the M2M community are actively integrated into different community activities. These include the service provision as well as the aspects of service testing, trust evaluation, and data storage (as introduced in section 2.3, 4.2, 5.1, 5.2, and 6.2).

- *End-user-friendly* – This requirement is fulfilled through the end-user-friendly platform. Every end-user can participate in service testing or trust evaluation with minimal effort and basic software knowledge (as detailed in section 4.3 and 6.2)

- *Incentivisation mechanism* – Incentivisation mechanisms are part of the proposed framework. They ensure that the participants are motivated to perform various tasks in the community (refer to section 6.2). Moreover, they support a

trustworthy environment where nodes with good trust scores receive several benefits.

- *Trust-based service selection & composition* – This requirement is fulfilled as the framework includes trust aspects that can be incorporated in the service selection and composition phase. Only M2M services with a good trust score are considered, whereas low trusted services are ignored. However, the application of the proposed framework to optimise the M2M service provision process is out of the scope (refer to section 8.3).

- *Functional testing of new and existing services* – The framework supports that new M2M services are tested immediately by other community members after being made public. Moreover, the proposed framework also enables service testing of M2M services during operation, thus providing the possibility to have many snapshots of the latest functionality (refer to section 4.3).

- *Test description of M2M services, test automation and test case generation* – These requirements are partially fulfilled by the proposed framework. This research thesis proposed a general workflow for service testing in the M2M community. Further research on the testing part is beyond the scope of this work. The proposed test concept serves as a basis for further research activities of fellows within the research group.

- *Initial trust score* – The introduced framework supports the evaluation of the trustworthiness for new M2M entities (as detailed in section 5.2, 6.1, and 6.2).

After a new M2M service is made available to the M2M community, it is tested by the community members (acting as test agents). The test results will serve as a basis for computing the initial trust score, which supports service consumers in their decisions whether to use that service or not.

- *Ongoing trust score* – The framework also fulfils this requirement, as the trust score of M2M entities is continuously evaluated by other nodes (explained in section 6.1 and 6.2. The aggregation of these scores, taking into account several trust aspects, leads to an overall trust score for M2M services and service providers.

- *Secure trust data storage* – The P2P storage proposed within the framework fulfils this requirement (introduced in section 4.2 and 5.1). The combination of DHT and blockchain technology enables high data integrity. Both storage possibilities are used to enable flexibility and integrity check-ups. The P2P storage is used for storing service, trust and other related data generated in the M2M community.

- *Trust model completeness* – The presented framework fulfils this requirement as it includes a trust model that uses several metrics to evaluate the trustworthiness of M2M entities (as explained in section 6.1). The trust model covers service- and peer-related aspects. It considers the functionality and performance of M2M services. Moreover, it incorporates service satisfactions and runtime performance indicators. Besides, it includes the behaviour of the peers in the trust evaluation

process where the participation willingness in different community tasks and the data integrity are part of it.

- *Trust attack resilience* – Trust is integrated into all parts of the proposed framework. Only highly trusted nodes can perform privileged community tasks. All these tasks and decisions are confirmed by other nodes using a trust-based consensus and decision system. The proposed framework incorporated trust-based incentivisation mechanisms and support a trusted M2M community highly. The integration of trust within the framework enables a high resilience against trust attacks performed by malicious nodes (refer to section 6.2).

Overall, the proposed framework fulfils these requirements conceptually. The next section also exhibits them as proof of concept.

7.2 Proof of Framework Concepts

This chapter demonstrates the essential functionalities of the proposed framework from a practical perspective. The research prototype developed for this reason implements most of the framework components with the required functionality for the proof of concept.

The proof of concept is separated into two main parts: testing and trust. The first section (7.2.1) of this chapter covers service testing. It shows the workflow starting with an M2M service description and concluding with the execution of generated test cases on the SUT. Sections 7.2.2-7.2.4 highlight the trust part of the framework. First, the overall blockchain-based trust management system is shown (section 7.2.2), providing an end-

user-friendly GUI to get or compute trust scores from/for other M2M entities part of the community. Section 7.2.3 evaluates the reliability and resilience of the proposed trust model. Therefore, different scenarios are performed and the outcomes are presented graphically. Section 7.2.4 validates the blockchain- and trust-based consensus protocol Trust-CP in comparison to other existing ones. The validation focuses on the attack resilience of consensus protocols and includes two main attack types to demonstrate their efficiency.

7.2.1 Service Testing

The functional testing concept for decentralised M2M services was presented in section 4.3. The functionality of an M2M service is one of the trust attributes defined for evaluating the overall trust score of services and service providers. This section demonstrates the key aspects of service testing, focusing on the overall workflow of testing an M2M service. The testing scenarios in this section aim to show the feasibility of applying the functional testing concept for decentralised and end-user-based M2M services.

Implementation

The proof of concept implementation of the proposed framework focuses on the test engine (refer to section 4.2) and the SUT. The test engine is used to handle all test activities. The SUT environment is replicated through a simplified M2M service scenario. Some parts in this section were published in (Shala *et al.*, 2017; Le, 2018).

The implemented test engine consists of the Test Generation Unit (TGU) and the Test Execution Unit (TEU). The TGE covers the functionality to receive the model of the system, transform it to a suitable system test model and generate test cases. For test generation purposes, the Tcases tool (*Cornutum*, 2020), a flexible and user-friendly black-box testing tool, was integrated within the TGE. Tcases supports the functional verification of service behaviour and fits well with the end-user-based M2M service environment. For performing the test execution, the Eclipse Titan (*Eclipse Titan*, 2021) toolset was used. It is a powerful TTCN-3-based execution environment and also enables functional testing.

The SUT was implemented using the Java programming language (Oracle, 2021). It consists of three services running on independent nodes and sending status information (based on the specific configuration) to each other. The communication between the nodes is established using the standardised protocol Session Initiation Protocol (SIP). It is integrated into the application using the JAIN SIP (JAIN SIP, 2021) library. The implemented SUT simulates the behaviour of decentralised M2M services and represents the use case of building surveillance. This use case consists of the following services:

- Remote M2M Service (SS1) – monitors smoke/water sensors in a building. Reports possible events to the corresponding M2M service entity (in this case, to the remote building monitoring service).
- Remote Building Monitoring Service (BMS) – receives information from SS1, evaluates them and informs another service (remote alarm service) for further actions.

- Remote Alarm Service (AS) – gets notifications from BMS and alerts the corresponding end-users accordingly for a specific event in their buildings.

Figure 7.1 shows the three decentralised M2M services representing the building surveillance use case. It has to be mentioned that a simplified form of these services with primitive functionality for sending and receiving messages was implemented and used to test corresponding single inputs/outputs.

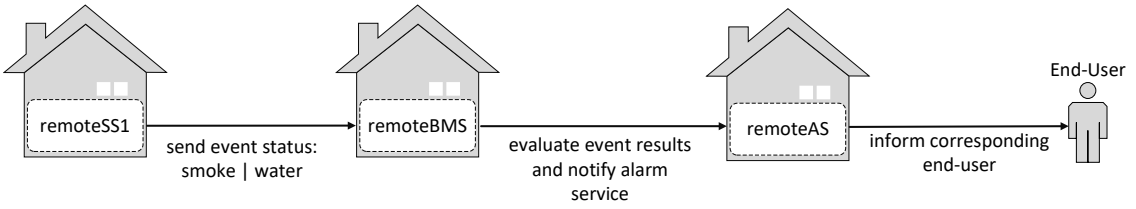

Figure 7.1: Use case Building Surveillance

Section 4.3 described that the decentralised M2M service provision concept considers that end-users can create single and cooperative (composed) M2M services. The building surveillance use case represents a cooperative M2M service consisting of single services. Therefore, the functional testing demonstration considers two main scenarios:

- Testing the whole cooperative M2M service
- Testing one single M2M service (in this case, the service remoteBMS)

The test engine and the three services (representing the SUT) were installed on Linux-based virtual machines. Table 7.1 shows the configuration used to implement the test scenario and enable networking of the test agent and three nodes providing the single services.

Table 7.1: Configuration of nodes for the test scenario

Node	VM IP Address	Port Number
TestModule	192.168.50.21	5060
RemoteSS1	192.168.50.21	5061
RemoteBMS	192.168.50.23	5062
RemoteAS	192.168.50.23	5063

Demonstration

In this scenario, the cooperative M2M service representing the building surveillance use case is tested. The SCE (refer to section 2.3) used to configure M2M services generates an SCXML of the cooperative service (illustrated in Figure 7.2). Additionally, there is a service interface description for every single service (e.g. remote BMS illustrated in Figure 7.3).

The service descriptions are used for the test generation process as system models. Beforehand, these system models are automatically transformed into an appropriate input model for Tcases. The test generation tool uses this model to derive abstract test cases for the SUT. The service interface description is utilised to set up the test configurations necessary to generate executable test cases and enable test executions on Eclipse Titan.

The implemented test system automatically creates executable test cases (TTCN-3-based) from the abstract ones. Figure 7.4 shows an exemplary test case in its abstract and executable version.

```
<?xml version="1.0" encoding="UTF-8"?><scxml xmlns="http://www.w3.org/2005/07/scxml"
datamodel="jexl" initial="remoteSS1" name="cooperativeBuildingMonitoringApp" version="1.0">
        <datamodel>
                <data expr="remoteSS1" id="initial"/>
                <data expr="remoteAS" id="final"/>
        </datamodel>
        <state id="remoteSS1">
                <datamodel>
                        <data expr="" id="remoteSS1.output.event"/>
                        <data expr="Kleiststr.1" id="remoteSS1.output.buildingID"/>
                        <data expr="true" id="initial"/>
                </datamodel>
                <transition cond="$remoteSS1.output.event=smoke OR
                $remoteSS1.output.event=water" target="remoteBMS">
                        <assign expr="$remoteSS1.output.event"location="remoteBMS.input.event"/>
                </transition>
        </state>
        <state id="remoteBMS">
                <datamodel>
                        <data expr="" id="remoteBMS.input.event"/>
                        <data expr="" id="remoteBMS.output.text"/>
                </datamodel>
                <transition cond="$remoteBMS.input.event=water" target="remoteAS">
                        <assign expr="IM" location="remoteAS.config.mode"/>
                        <assign expr="WaterdetectedIMSupporter3"location="remoteAS.input.text"/>
                </transition>
                <transition cond="$remoteBMS.input.event=smoke" target="remoteAS">
                        <assign expr="TTS" location="remoteAS.config.mode"/>
                        <assign expr="SmokedetectedTTSSupporter2"
                        location="remoteAS.input.text"/>
                </transition>
        </state>
        <state id="remoteAS">
                <datamodel>
                        <data expr="" id="remoteAS.input.text"/>
                        <data expr="" id="remoteAS.config.mode"/>
                        <data expr="true" id="final"/>
                </datamodel>
                <final id="remoteASFinal"/>
        </state>
</scxml>
```

Figure 7.2: SCXML service description Building Surveillance

```
<?xml version="1.0" encoding="UTF-8"?>
<AE>
        <appName>Remote Building Monitoring Service</appName>

        <App-ID>remoteBMS</App-ID>

        <pointOfAccess>sip:remoteBMS@192.168.50.23:5062</pointOfAccess>

        <requestReachability>true</requestReachability>

        <creationTime>2020-05-31</creationTime>

        <lastModifiedTime>2020-09-15</lastModifiedTime>

        <contentSerialisation>XML</contentSerialisation>

        <accessControlPolicy>
                <privileges>
                        <accessControlOriginators>all</accessControlOriginators>

                        <accessControlContexts></accessControlContexts >

                        <accessControlOperations>RU</accessControlOperations>
                </privileges>

                <expirationTime></expirationTime>
        </accessControlPolicy>

        <content>
                <input>
                        <inputParameter id="1">
                                <name>remoteBMS.input.event</name>
                                <value></value>
                        </inputParameter>
                </input>

                <output>
                        <outputParameter id="1">
                                <name>remoteBMS.output.text</name>
                                <value></value>
                        </outputParameter>
                </output>

                <config></config>
        </content>

        <description> Manages building events and …  </description>

</AE>
```

Figure 7.3: Service Interface Description Service “remoteBMS”

```
type="text/xsl" ?><TestCases system="testCooperativeService">
  <Function name="FunctionalTest">
    <TestCase id="1">
      <Input type="arg">
        <Var name="Sender" value="remoteSS1">
          <Has name="varName" value="sender"/>
          <Has name="varType" value="String"/>
        </Var>
        <Var name="Output" value="water">
          <Has name="varName" value="output"/>
          <Has name="varType" value="String"/>
        </Var>
        <Var name="To" value="remoteBMS">
          <Has name="varName" value="to"/>
          <Has name="varType" value="String"/>
        </Var>
        <Var name="Notifier" value="remoteAS">
          <Has name="varName" value="notifier"/>
          <Has name="varType" value="String"/>
        </Var>
        <Var name="Status" value="WaterdetectedIMSupporter3">
          <Has name="varName" value="status"/>
          <Has name="varType" value="String"/>
        </Var>
      </Input>
    </TestCase>
```

```
module TestCase2
{
import from SIPmsg_Types all;
import from SIPmsg_PortType all;
import from SIPexampleTemplates all;
        modulepar charstring username := "remoteSS1";
modulepar charstring usernameextra := "\"remoteSS1\"";
modulepar integer sendingport:=5060;
modulepar charstring ipaddress:="192.168.50.21";
    modulepar charstring outputstring := "water";
modulepar integer chieudai1 := 5;
    modulepar charstring usernamedest := "remoteBMS";
modulepar charstring usernamedestextra := "\"remoteBMS\"";
modulepar integer sendingportdest:=5062;
modulepar charstring ipaddressdest:="192.168.50.23";
    modulepar charstring notifier := "remoteAS";
modulepar charstring notifierextra := "\"remoteAS\"";
modulepar integer sendingportnotifier:=5063;
modulepar charstring ipaddressnotifier:="192.168.50.25";
    modulepar charstring statusstring :=
```

Figure 7.4: Abstract versus executable (TTCN-3) test case

In this example, the test generation process produces six test cases indicating the correct functionality of the building surveillance service. These test cases are used to check the inputs/outputs of the whole service chain (*SS1 – remoteBMS - remoteAS*) and pairs of the chain (*SS1 - remoteBMS* and *remoteBMS - remoteAS*). The generated test cases, the test execution of *TestCase2*, and the test report are illustrated in Figure 7.5. The test execution of *TestCase2* receives a "pass" verdict and shows that the cooperative M2M service is fulfilling one of its functional requirements. Based on its configuration and description, the last service of the service chain (remoteAS) reacts with a message `WaterdetectedIMSupporter3` for the event `water` occurring in the building.

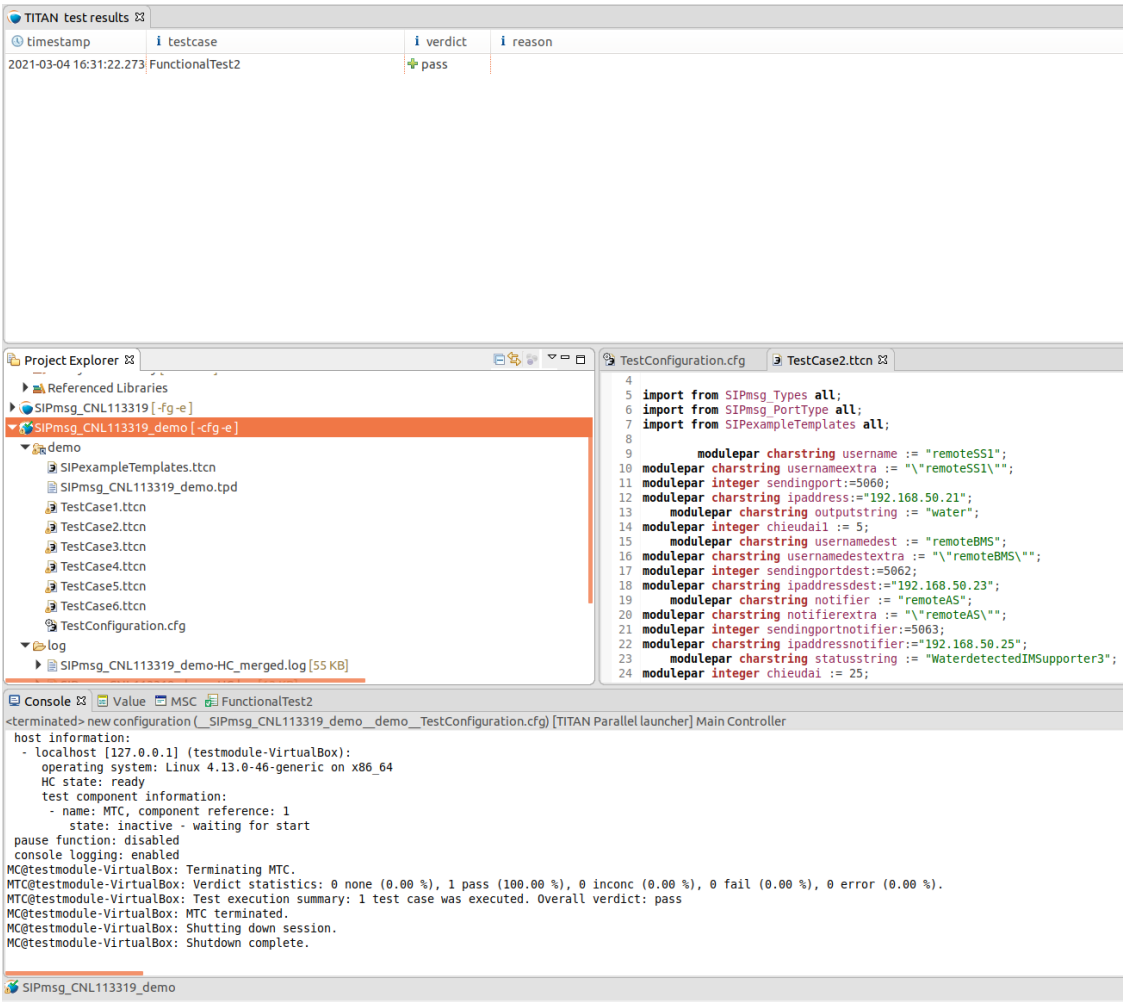

Figure 7.5: Test execution of TestCase2

It is also possible to produce “true” failure test cases with Tcases. These are test cases that the system is not expected to pass during test execution to show its correct functionality. In the following example, twelve test cases were generated in total, where half of them are pass test cases and the others are “true” failure test cases. Figure 7.6 shows the test execution of all of these test cases. It also shows exemplarily one “true” failure test case (*TestCase10*) that the SUT has not passed (as expected).

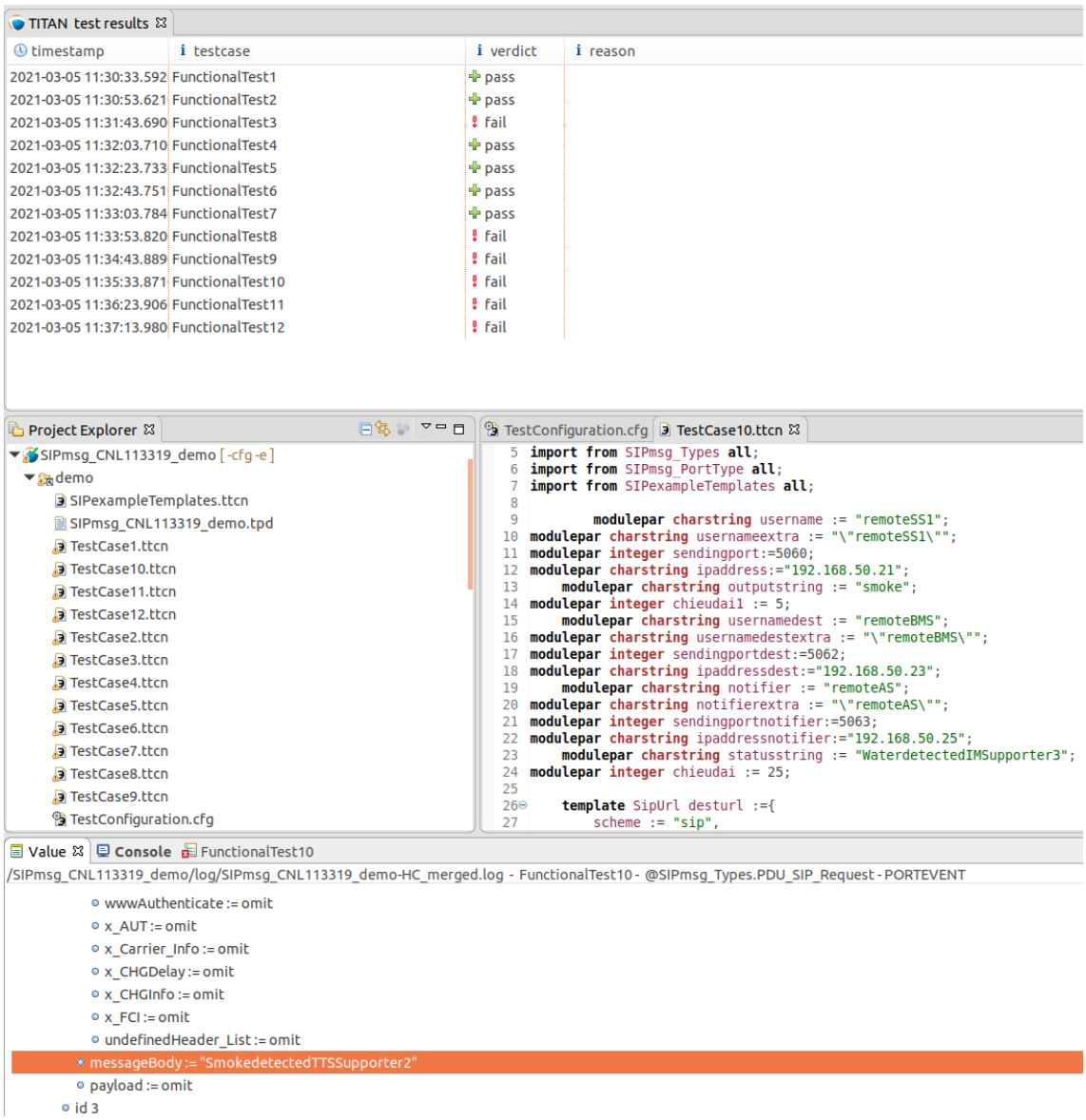

Figure 7.6: Test execution of all test cases with a focus on TestCase10

TestCase10 indicates that the status of *remoteAS* for the event `smoke` should be `WaterdetectedIMSupporter3`. It can be clearly identified that this is not the proper functionality of the service. The value panel in Figure 7.6 also shows the response (`messageBody:= "SmokedetectedTTSSupporter2"`) received from *remoteAS*. To sum up, the test execution outcomes in this example demonstrate that the cooperative M2M service is working correctly.

In the next example, service *remoteAS* was intentionally manipulated to respond with a wrong status for a right input event coming from the two other services (*SS1* and *remoteBMS*). The event is `water`, and the expected status on *remoteAS* is `WaterdetectedIMSupporter3`. However, the manipulated *remoteAS* responds with a `SmokedetectedTTSSupporter`. As a result, the execution of *TestCase2* on the SUT will produce a fail verdict. Figure 7.7 illustrates this scenario.

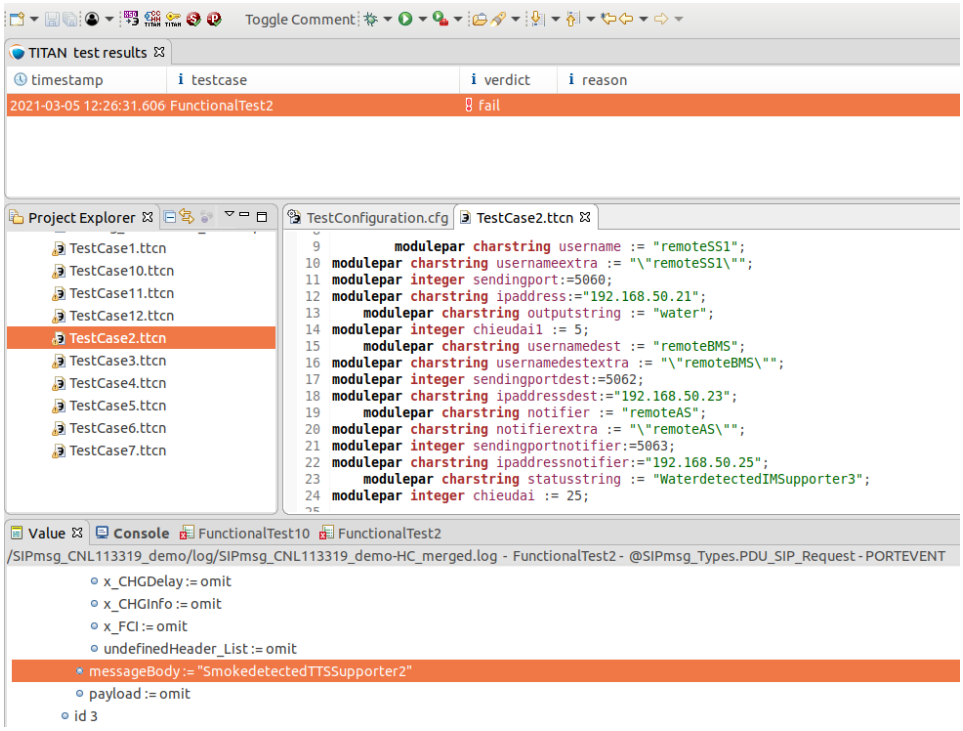

Figure 7.7: Test execution on manipulated M2M service

Besides testing the whole cooperative M2M service, it is also possible to test the single components or services. In the following example, testing the single service *remoteBMS* is demonstrated. The service information coming from the SCXML service description and service interface description outlined in Figure 7.2 and Figure 7.3 was used for the test generation process. As a result, Tcases generated four test cases where two of them expect a pass verdict and the other two a fail one. The following sample in Figure 7.8 shows the test execution on service *remoteBMS* with a focus on *TestCase1*. It is visible that service *remoteBMS* reacts (as expected) with the correct output `SmokedetectedTTSSupporter2` for the input event `smoke`.

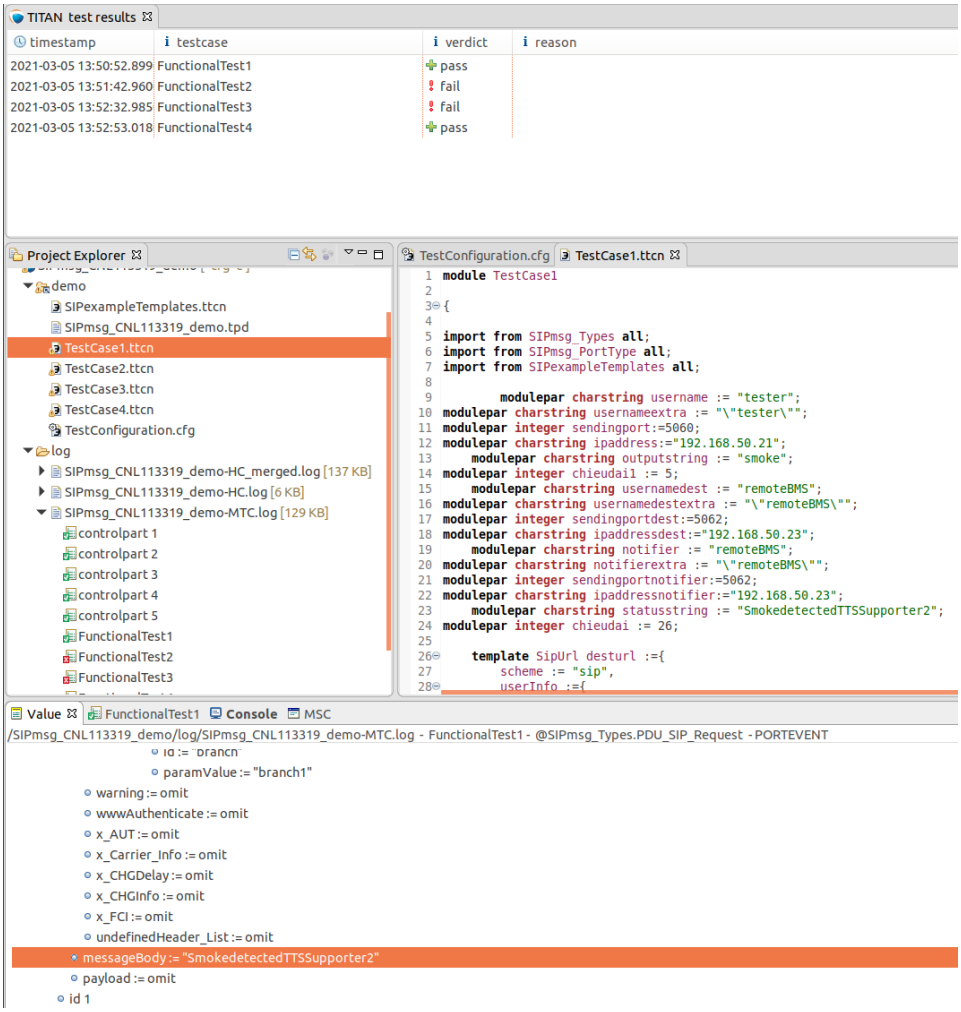

Figure 7.8: Testing single M2M service

Overall, the essential concept for testing decentralised and end-user-based M2M services could be proven by testing a sample cooperative M2M service. As proposed in this research, the test outcomes can be used to create the first trust opinion about a new M2M entity.

7.2.2 Blockchain-Based Trust Management System

Chapter 6 has introduced the blockchain-based trust management system. It uses blockchain-based principles to handle the evaluation activities in the M2M community and store data trustworthy among the nodes. This section shows the prototype architecture and demonstrates the essential functionalities of the system.

Implementation and Architecture

The application was implemented using the Java programming language (Oracle, 2021). Some parts in this section were published in (Shala *et al.*, 2020a; Biswas, 2021). The application includes many features of the proposed framework, such as the:

- G*raphical User Interface (GUI)* – every node can access the trust management platform using the web browser. This platform provides information about service providers, services and their corresponding trust performance.
- *Assessment of different trust attributes* – all participating nodes perform trust evaluation activities using one of the proposed trust metrics (e.g. service testing, service monitoring, or service rating).
- *Aggregation and computation of trust scores* – The outcomes of the trust evaluation activities are sent via transactions through the blockchain network. The

different results are aggregated and computed using the principles introduced in section 6.2.

- *Data storage in the blockchain* – all the data handled among the nodes, such as service information, test and trust results, are stored tamper-proof in the blockchain.
- *Maintenance of the blockchain* – all participating nodes maintain the blockchain equally by performing tasks such as transaction broadcasting, transaction validation, block creation and validation, and consensus-building.
- *Trust-CP* – the proposed trust-based consensus protocol (see section 5.1) is used for consensus building.

The functionality of the application was validated in a decentralised environment through a scenario with many nodes acting as service providers, service consumers, and test agents. The application was set up on different nodes as part of a virtual environment. The nodes were connected using the Common Open Research Emulator (CORE) (*CORE*, 2021). This emulator provides the possibility to create real computer network representations running in real-time. CORE is known as a scalable system and also provides a user-friendly GUI for creating different emulated network scenarios. (*CORE*, 2020) Using CORE, a scenario with up to ten nodes was created to run the blockchain-based trust evaluation application. Figure 7.9 illustrates the created scenario within the emulation environment.

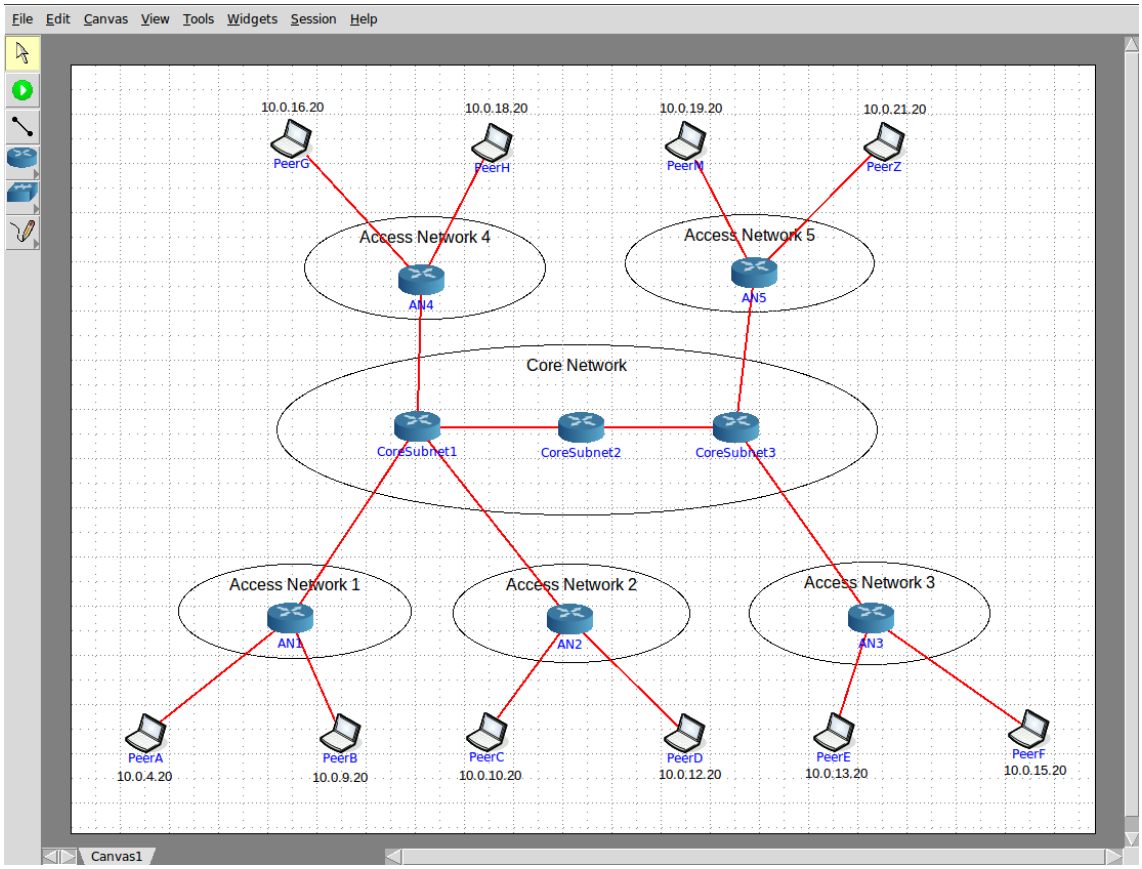

Figure 7.9: The architecture of the created scenario in CORE

Table 7.2 shows the configuration used to implement the networking of peers and running the CORE application. The management network IP addresses are used to create a control network for CORE. The control network enables access (e.g. via a web browser) from the host system to the different running containers.

Table 7.2: Configuration of nodes for CORE and management

Node	CORE IP Address	Management IP Address	Port Number
Peer A	10.0.4.20	20.0.0.6	8081
Peer B	10.0.9.20	20.0.0.11	8082
Peer C	10.0.10.20	20.0.0.12	8083
Peer D	10.0.12.20	20.0.0.14	8084
Peer E	10.0.13.20	20.0.0.15	8085
Peer F	10.0.15.20	20.0.0.17	8086
Peer G	10.0.16.20	20.0.0.18	8087
Peer H	10.0.18.20	20.0.0.20	8088
Peer M	10.0.19.20	20.0.0.21	8089
Peer Z	10.0.21.20	20.0.0.23	8080

In the following some additional details on the implemented application:

- The application is provided as a Spring Boot (Webb *et al.*, 2017) web application.
- The back-end is mainly developed with common Java and Spring Boot libraries such as Thymeleaf (*Thymeleaf*, 2021).
- The front-end, including the graphical evolution of the trust scores, is implemented using an HTML webpage, including JavaScript libraries such as jQuery (*jQuery*, 2021) and Highcharts (*Highcharts*, 2021).
- A simplified abstract representation of core blockchain functionalities and the proposed Trust-CP is developed in Java using its libraries and in-built functions.
- The important contribution is not on the complexity of the implementation but on the produced environment showing the key aspects of the novelties as proof of concept.

Components and Features

One component of the application is the GUI. It serves as a front-end management platform for end-users in the M2M community. The GUI provides a peer and a service part. Figure 7.10. shows the general overview of the webpage, providing trust information about a peer (service provider). Every end-user has the possibility to click on a specific service provider, display last/current trust scores, or trigger trust evaluation activities.

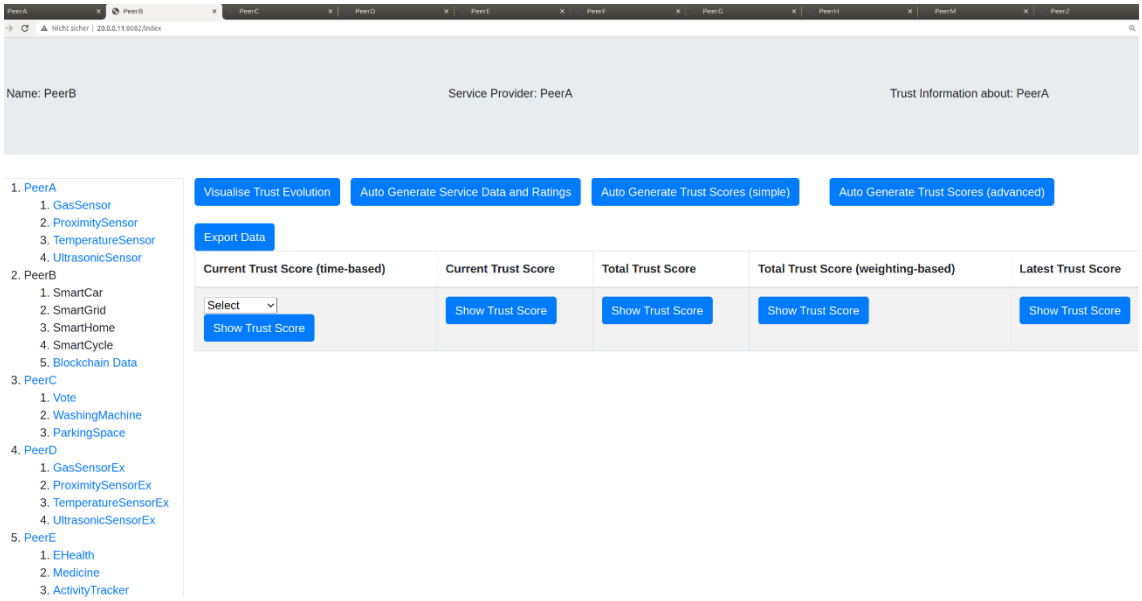

Figure 7.10: GUI – Peer Trust Information (Peer Part)

There is also the possibility to do the same for the different services provided by several service providers. Figure 7.11 shows the general overview of the webpage, providing trust information about individual M2M services.

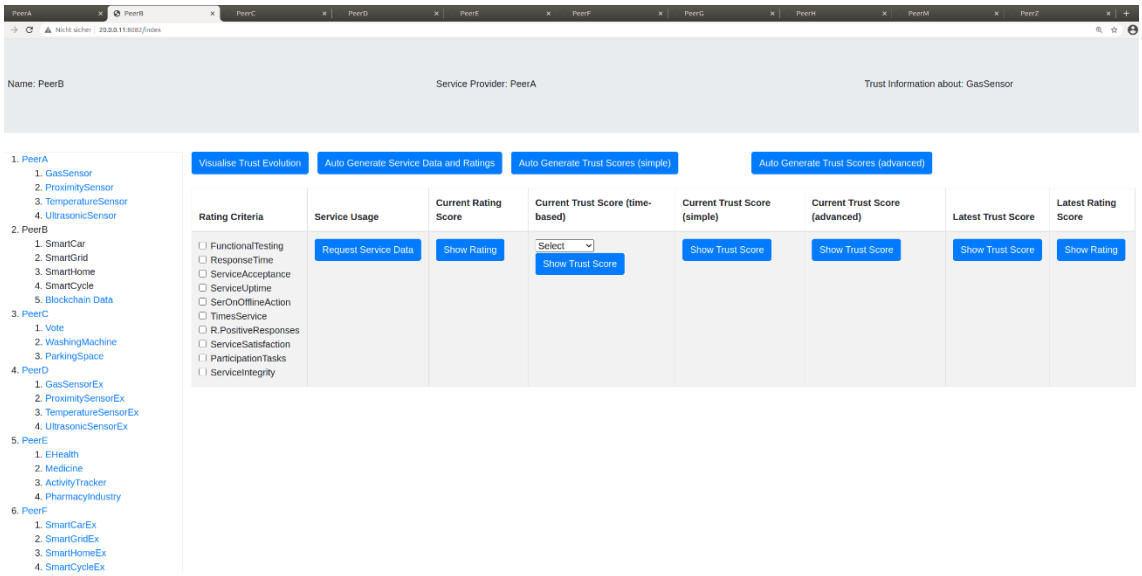

Figure 7.11: GUI – Service Trust Information (Service Part)

The web page is organised as follows: on the left side of the page are the different peers and their services listed. The information about who is using the platform (e.g. *Name:*

PeerB, see Figure 7.11) and about whom trust information is requested (e.g. *Service Provider: PeerA* and *Trust Information about: GasSensor*, see Figure 7.11) is shown at the top. At the centre of the web page are the various buttons used for several purposes, such as frequesting services, rating services, requesting trust information, or visualising the trust evolution of peers and services.

The following will present more details on some main features of the trust management system.

The front-end platform contains several buttons triggering different activities happening on the blockchain (back-end). For example, the request for a current trust score (refer to Figure 7.12) will initiate the introduced Trust-CP process where the most trustworthy node will be selected randomly to generate a new block. Based on previous blocks and the information inside them, the block creator will aggregate and compute the most current trust score, create the new block with this information, and broadcast it to others for validation. The back-end activities after requesting the current trust score are shown in Figure 7.13.

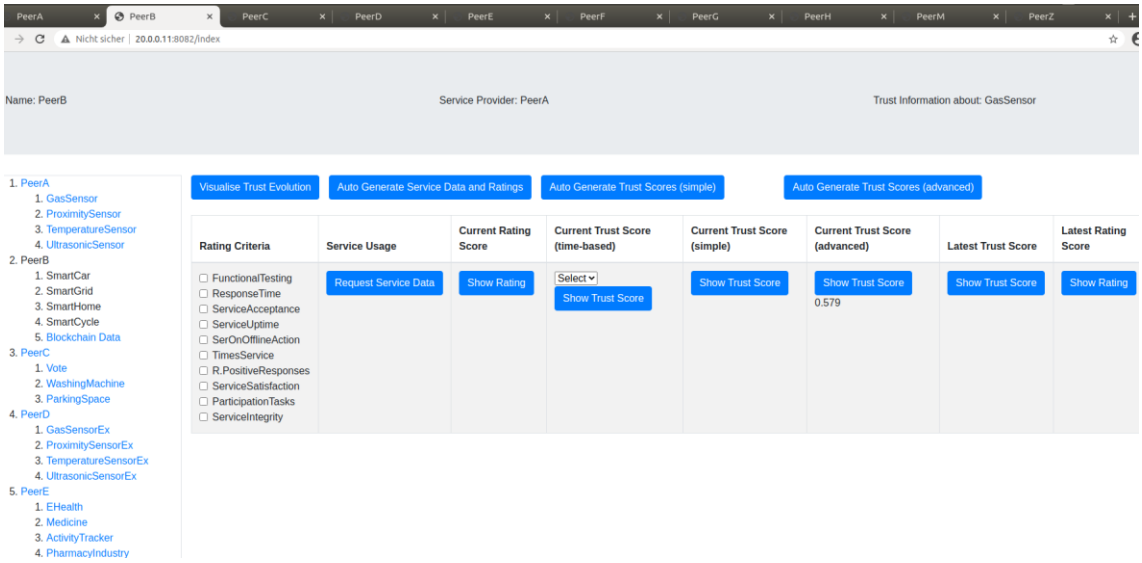

Figure 7.12: Checking the current trust score of an M2M service

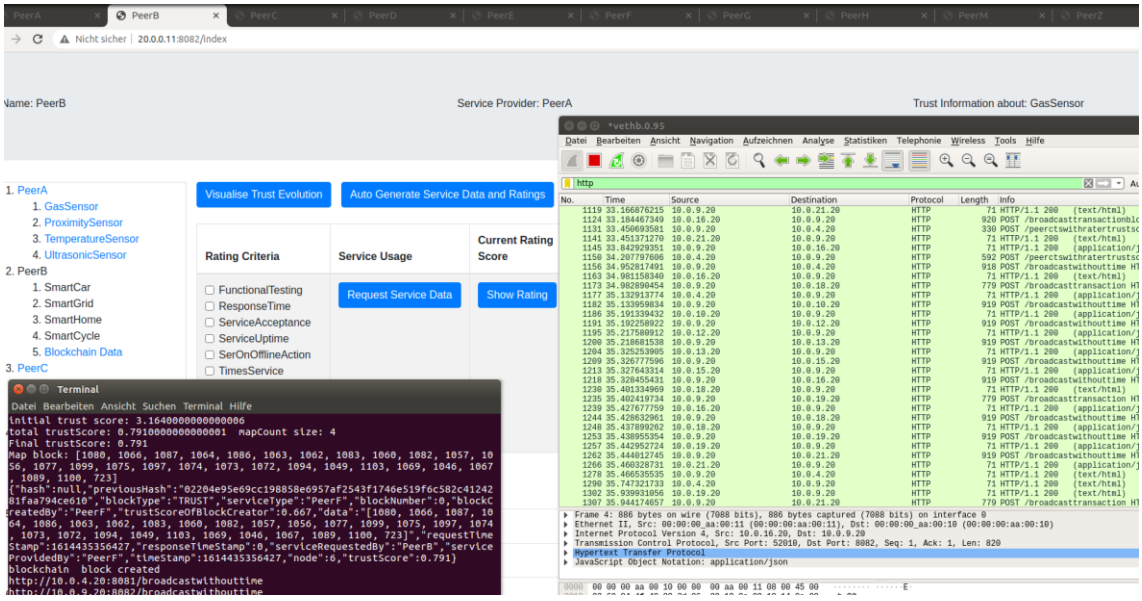

Figure 7.13: Backend activities after requesting the trust score of a service (terminal and wireshark trace)

Table 7.3 shows a brief description of the different features and fields on the web application.

Table 7.3: Description of different features and fields on the web application

	Field/Features	Description
General	Visualise Trust Evolution	Presents a graphical representation of the trust evolution
	Export Data	Export the current generated blockchain data
	Auto Generate	For testing purposes: auto-generate service data, ratings and trust scores
	Blockchain Data	Display the current blockchain
Service Part	Rating Criteria	Used to select the rating criteria used for evaluation
	Action	Used to request service data
	Current Rating Score	Displays the current rating score based on the selected criteria
	Current Trust Score (time-based)	Displays current trust score of the service considering previous trust information for a selected period
	Current Trust Score (simple)	Displays current trust score using a simple calculation
	Current Trust Score (advanced)	Displays current trust score using the proposed calculation
	Latest Trust Score	Displays the latest computed trust score
	Latest Rating Score	Displays the latest rating score
Peer Part	Current Trust Score (time-based)	Displays current trust score using a simple calculation
	Current Trust Score	Displays current trust score using a simple calculation
	Total Trust Score	Displays current trust score using the proposed concept
	Total Trust Score (weighting-based)	Displays current trust score using the proposed concept plus weighting parameters
	Latest Trust Score	Displays the latest computed trust score

Figure 7.14 shows exemplarily the current trust score of *PeerA* and the corresponding Wireshark trace indicating that *PeerM* has performed the trust computation.

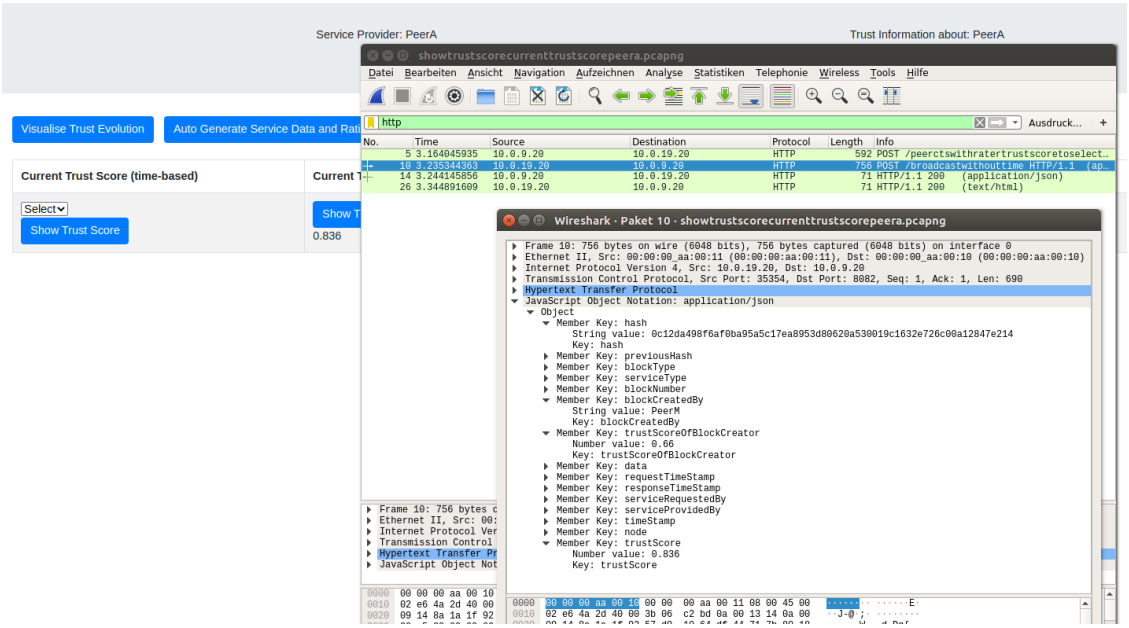

Figure 7.14: Wireshark trace on the current trust score of *PeerA*

One feature of the web application is to show the trust evolution about a peer or its services. Figure 7.15 illustrates exemplarily the visualised trust information about service provider *PeerA*.

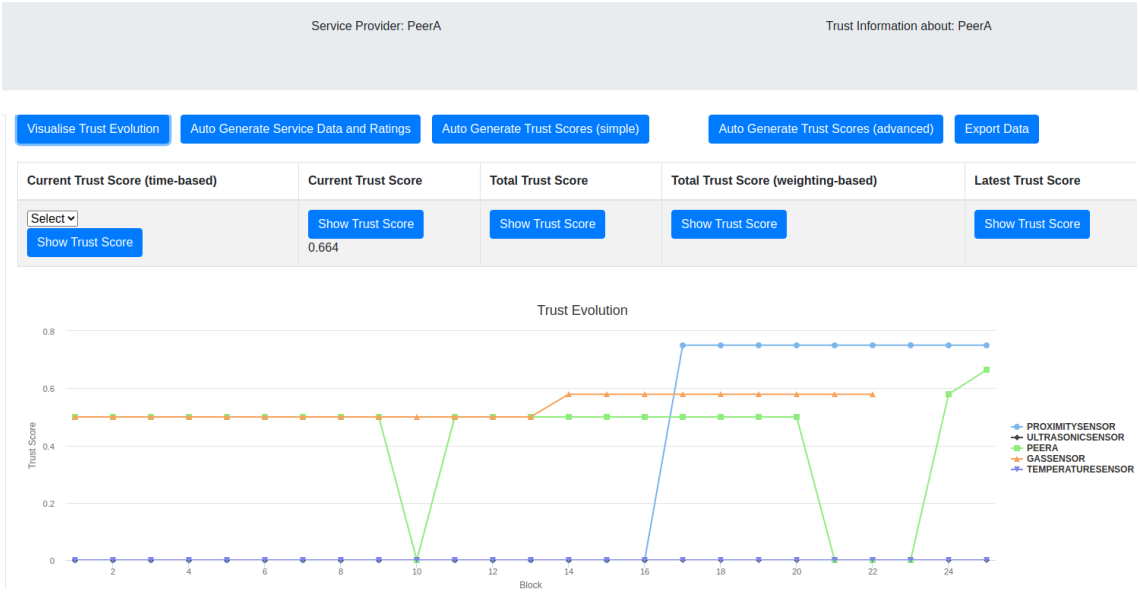

Figure 7.15: Visualise the trust evolution of a service provider and its services

Another feature is displaying the current blockchain (Figure 7.16 illustrates exemplarily the block with number 1131). It has to be mentioned that there are three block types used in the implementation. One is for service information provided by related services, one is for service ratings, and the other for trust scores.

Service information is used for performing the service ratings based on several criteria. Service rating is used here to represent all the service evaluation activities described in section 6.1. On the other side, the different rating results/scores are used to compute the trust scores of the services. The aggregation of these scores using the proposed mathematical trust model (as introduced in section 6.2) forms the overall trust score of the service provider (peer).

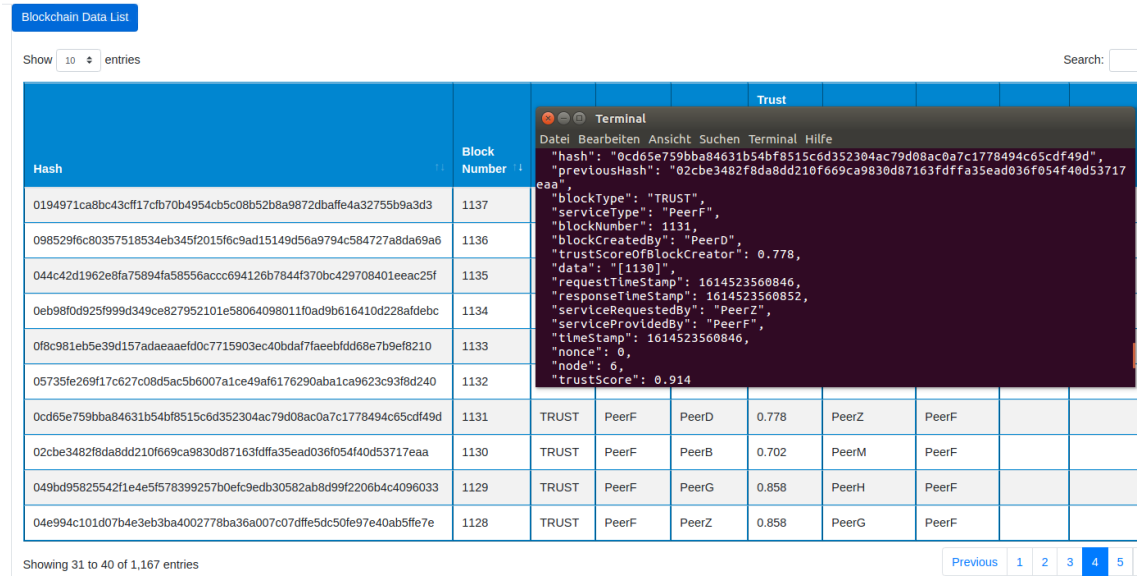

Figure 7.16: Representation of the block with number 1131 on the GUI and terminal

Overall, the application implements the main principles introduced in the previous chapters. The next two sections will practically evaluate the proposed trust model and the Trust-CP.

7.2.3 Trust Model Evaluation

The implementation and architecture of the blockchain-based trust management system was introduced in the previous section. The comprehensive trust model, with its metrics and calculation principles, was presented in chapter 6. The proposed trust model includes features such as initial trust score considerations, different trust metrics, trustworthy consensus protocol, trust aggregation concept, and synergy of blockchain, smart contracts, and trust. This section conducts different scenarios and statistical analysis to show the reliability and resilience of the trust model.

Some parts in this section were published in (Shala *et al.*, 2020a).

Scenario Settings

The resilience of the trust model against different attacks was tested using the following scenarios:

- Increasing malicious population (nodes providing false trust information)
- Impact and evolution of initial trust scores
- Static vs dynamic weighting
- Bad-mouthing attack (malicious nodes providing bad recommendations to good nodes) (ITU-T, 2017b)
- Ballot-stuffing attack (malicious nodes providing good recommendations to bad nodes) (ITU-T, 2017b)
- Comparative analysis with other existing trust models

This section also includes scenarios where all peers are trustworthy (trust score is above equal to 0.55) and scenarios where the percentage of malicious peers trying to manipulate the overall trust score of the evaluated peer by sending false trust scores varies from 20% to 80%. In the scenarios, the ability of the introduced trust model to provide initial trust scores is compared to other approaches that only provide default values or no initial trust scores. Moreover, the introduced dynamic weighting system is compared against other approaches with no or static weighting. In order to highlight the resilience of the proposed trust model, a comparison with other trust models is made to identify the performance differences when attacked by other nodes under an increasing malicious population. Finally, a relative trust score is derived and used to assess the accuracy of the proposed trust model compared to existing ones in a comparative analysis.

All scenarios are conducted under the same general conditions presented in the following:

- Ten to fifty transactions per block.
- Each transaction consists of evaluation results regarding different metrics and sub-metrics.
- The evaluation activities are performed by different nodes part of the network.
- Five to ten blockchain circles (blocks) are executed.
- Each block consists of the overall computed trust score of the evaluated M2M entity.

Further details on the experimental setups are presented in the following:

- Simple equations from different models used for comparison throughout the scenarios are derived, reproducing their key ideas. The different trust models represent the proposed trust model and related approaches.
- Table with different trust metrics and sub-metrics is created.
- Each transaction consists of evaluation results regarding these metrics. These results are randomly generated values (trust scores) for the experiments.
- Each transaction represents the evaluation done by one node for an M2M entity.
- Ten to fifty transactions are included for the calculation of the overall trust score.
- The randomly generated values (trust scores) within these transactions are used to execute the reproduced equations.
- Each block represents the calculation of the overall trust score. Five to ten blockchain circles (blocks) are executed.
- The reproduced equations are executed with randomly generated values (trust scores).

- The outcomes are graphically presented via diagrams.
- A spreadsheet application is used for graphical representation of the values and analysis.

Scenario 1: Impact of Initial Trust Score on Trust Evolution

Existing trust approaches in M2M/IoT are not considering the trust score of new M2M services (as reviewed in section 3.2). Scenario 1 shows the trust score evolution when using the proposed trust model with its initial trust score strategy compared to trust models assigning only default scores or even not considering any score.

The scenario starts with a trustworthy service provider (trust score 0.8), which provides five trustworthy services. At a given time, five additional new services are made available by the service provider. It is considered that these new M2M services are performing badly in the M2M community (trust score 0.2). Moreover, it is also assumed that the existing ones are also slightly decreasing their trust scores (from 0.8 to 0.55) throughout a given time. Afterwards, the evolution of the trust score is analysed in relation to the proposed trust approach and two other approaches (with a default initial trust score of 0.5 and without an initial trust score).

Figure 7.17 shows the outcomes of the scenario. It depicts the trust score evolution when the service provider switches to a malicious node and provides untrustworthy services. With its initial trust score consideration, the proposed trust model enables quick identification of untrustworthy behaviour compared to alternative methods. As a result, service consumers can faster decide whether to use or not these services. Overall, the

reaction times are optimised to avoid or neglect untrustworthy nodes (with past good trust scores) for service provision or other community activities.

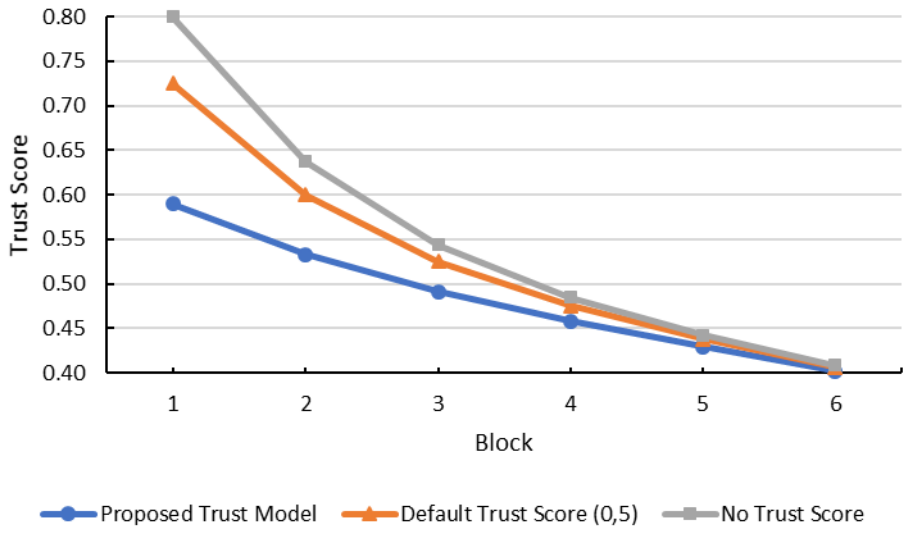

Figure 7.17: Trust evolution with new services (good to bad) (Shala *et al.*, 2020a)

Scenario 2: Static vs Dynamic Weighting

Section 6.2 introduced the dynamic weighting system of the proposed trust model. The weighting system enables efficient trust aggregation and dynamic weighting adjustment based on the current situation in the M2M community. Scenario 2 shows the trust evolution when using the proposed dynamic weighting system compared with static and no weighting approaches. Two cases are used to highlight the benefits of dynamic weighting: behaviour worsening and behaviour improving.

The first case starts with a service provider offering various good services. Then, the number of low trusted services (up to 60% of the services) is intentionally increased during a predefined period (eight block cycles). Through the proposed dynamic weighting scheme, each service gets a weight based on the current situation. The weights are adapted

in future rounds of the trust evaluation process. This workflow motivates participating nodes to remain active in all steps of service provision. In contrast, a static weighting scheme considers only predefined service weights without including the most recent behaviour situation.

Figure 7.18 shows the outcomes of the first case. It highlights that the changing behaviour of the nodes is detected faster using the proposed model with its dynamic weighting system. Thus, the model enables the identification of dishonest behaviour and demotivates the involved parties to maintain these activities. Such recurring behaviour leads to a lower trust score and less acceptance in the M2M community. In contrast, static weighting supports the passivity of service providers as trust weights of the various services are known and static. As a result, service providers may neglect one of these services leading to a non-improving service community depicted with less service quality.

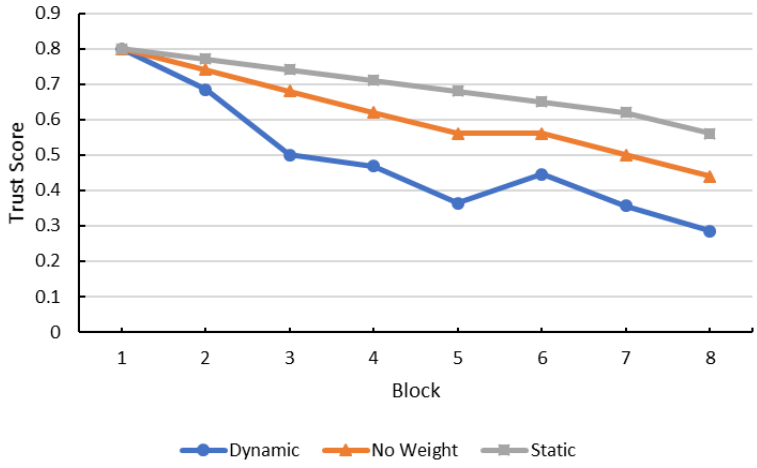

Figure 7.18: Trust evolution – behaviour worsening (Shala *et al.*, 2020a)

The second case considers a low trust service provider providing bad services (also with low trust scores). At a given time, the service provider tries to attack the system. Therefore, it increases its overall trust score by seemingly providing some good services.

The outcomes of the second case are shown in Figure 7.19. They illustrate that the trust evolution with dynamic weighting increases more slowly than static or without weighting approaches. The proposed trust model mitigates bad service providers because they cannot increase their trust scores quickly. Moreover, bad service providers are demotivated to remain passive and untrustworthy in the M2M community.

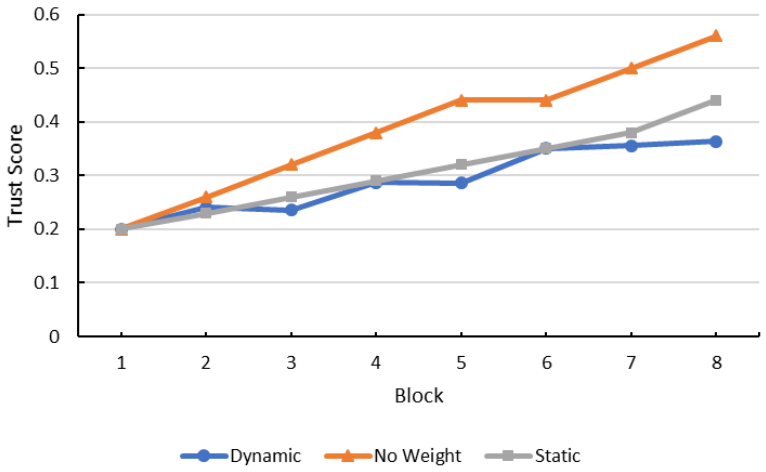

Figure 7.19: Trust evolution – behaviour improving (Shala *et al.*, 2020a)

Other cases where a service provider offers only one service and changes its characteristics lead to similar outcomes as in scenario 2.

Scenario 3: Proposed Trust Model vs Simple Trust Model – Bad-Mouthing Attack

This scenario compares the proposed trust model with a simple one (with basic average calculations) in terms of resilience against the so-called bad-mouthing attack (ITU-T, 2017b). In this kind of attack, bad or malicious nodes provide bad recommendations for good nodes. The aim is to show the performance and reaction differences between these two trust models.

Scenario 3 considers a bad-mouthing attack performed against one service. Both service and service provider are considered trustworthy M2M entities. During this scenario, the percentage of malicious nodes in the network is increased. They are attacking the service provider by providing fake recommendations consisting of low trust scores about its service. This scenario aims to identify the changes in the trust score evolution during different block cycles and under the impact of bad-mouthing transactions. This scenario also shows the resilience differences between the proposed trust model and a simple trust model in different scenarios.

Many trustworthy nodes (trust score 0.55 or higher) evaluate a service provider and its service in the first case. They are sending different trust scores for these two M2M entities. Figure 7.20 shows the outcomes of the first case. When using a simple trust model, the overall trust score of the service provider is marginally better than with the proposed one. However, this result does not reflect the real truth as the simple trust model fails to consider the trust scores of the evaluating entities.

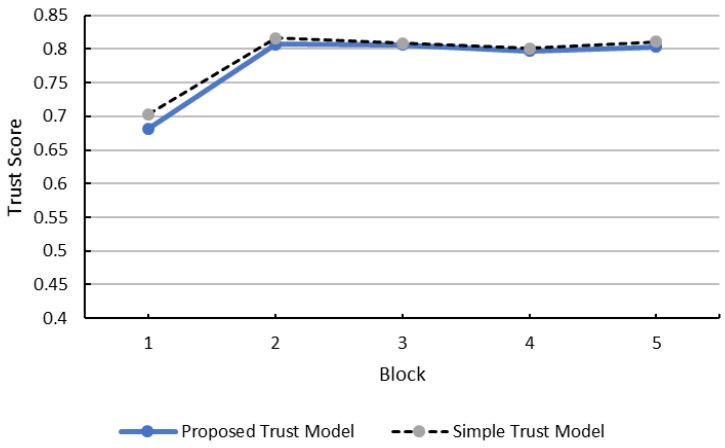

Figure 7.20: Trustworthy peers (Shala *et al.*, 2020a)

The other cases consider the presence of malicious nodes (trust score 0.2). They are continuously sending fake transactions with low trust scores to influence the trust

evolution of service provider and service. Figures 7.21 – 7.24 show the outcomes of these cases. They depict that, when using the simple trust model with an increasing number of malicious nodes, the trust resilience and the trust scores decrease. They also show that the higher the number of malicious nodes, the higher the trust evolution difference between the proposed trust model and a simple one. Overall, when using the new proposed trust model, the trust score stays stable with only minor changes in contrast to the case without malicious nodes.

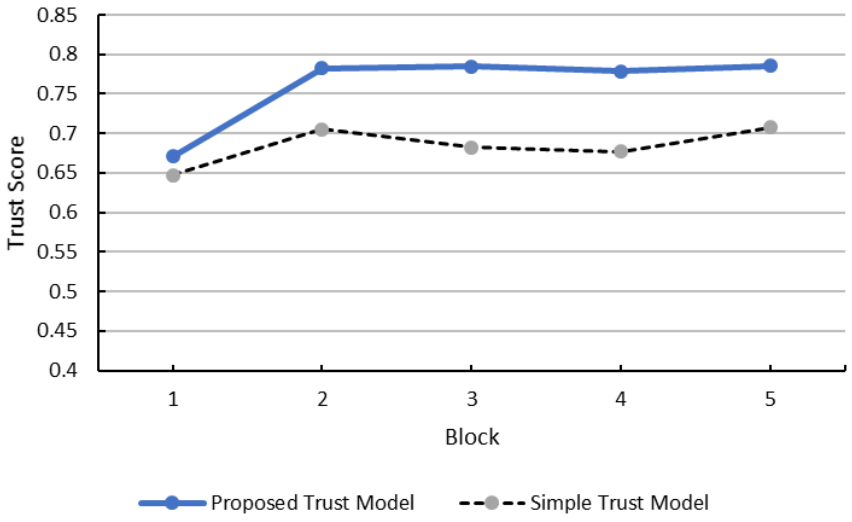

Figure 7.21: 20% of malicious peers (Shala *et al.*, 2020a)

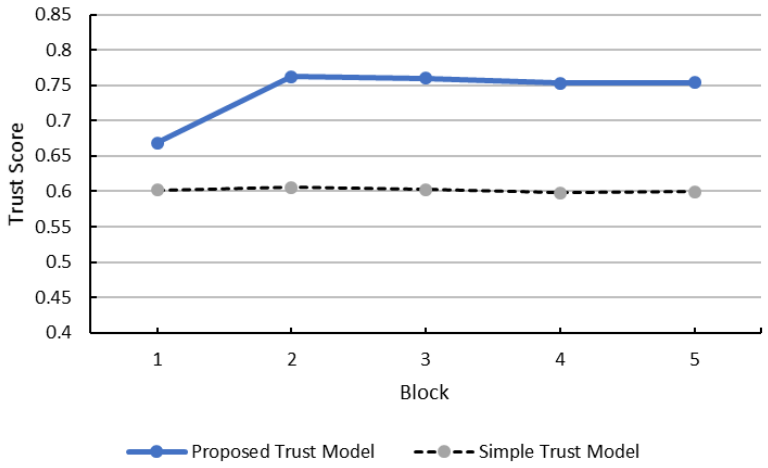

Figure 7.22: 40% of malicious peers (Shala *et al.*, 2020a)

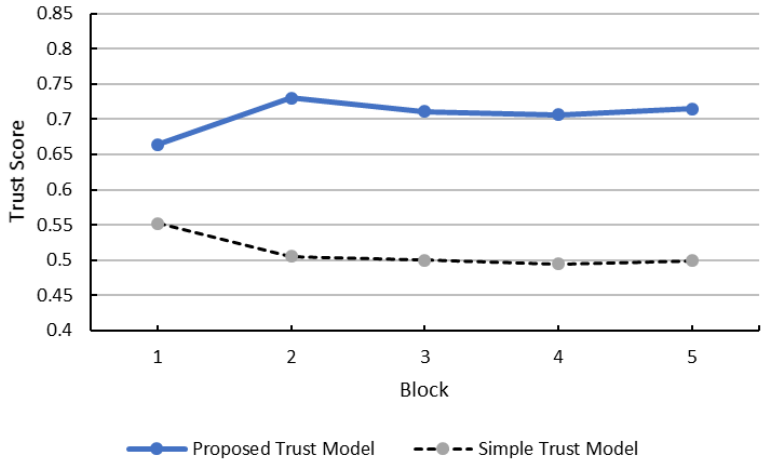

Figure 7.23: 60% of malicious peers (Shala *et al.*, 2020a)

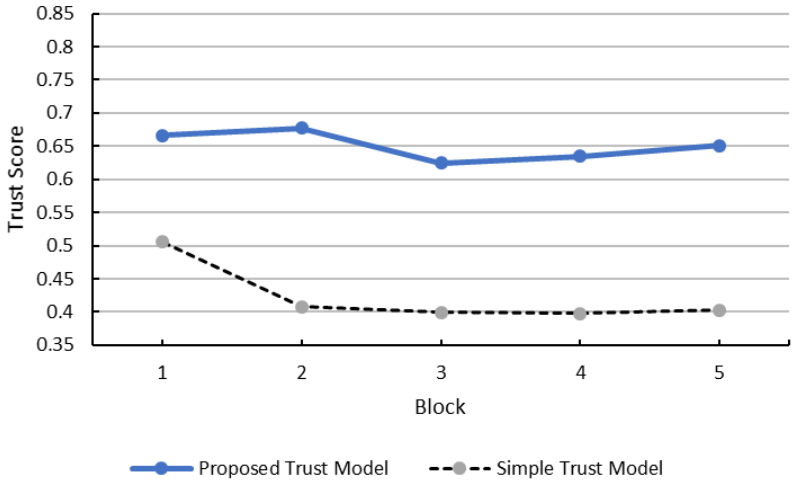

Figure 7.24: 80% of malicious peers (Shala *et al.*, 2020a)

Figure 7.25 highlights the trust evolution towards the increasing percentage of malicious nodes in the community. The outcome shows that the proposed trust model provides good resilience against attacks even if they increase to an 80% population. The resilience is a result of ignoring untrustworthy nodes in the proposed block building and trust aggregation process.

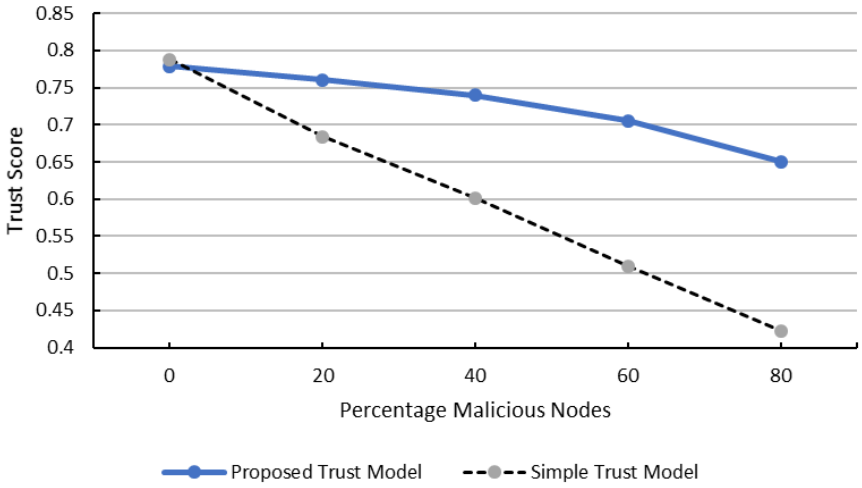

Figure 7.25: Trust evolution in relation to malicious population (Shala *et al.*, 2020a)

Scenario 4: Proposed Trust Model vs Simple Trust Model – Ballot-Stuffing Attack

This scenario considers malicious nodes performing the ballot-stuffing attack (ITU-T, 2017b), where good recommendations are assigned for other malicious nodes. The aim of this scenario is to provide a comparison in terms of attack resilience between the proposed trust model and a simple one (with basic average calculations).

The ballot-stuffing attack is performed against one M2M service. Both service and service provider have a low trust score. During this case, the percentage of malicious nodes in the network is increased. They are trying to increase the overall trust score of the service provider. This case aims to identify the changes in the trust score evolution during different block cycles and under the impact of ballot-stuffing transactions. It also shows the resilience differences between the proposed trust model and a simple trust model in different scenarios.

Under normal conditions without malicious peers in the network, the performance of the two models is almost the same (shown in Figure 7.26), but the differences appear when starting the attacks.

Figures 7.27 – 7.30 show that the proposed trust model almost stays resilient when the percentage of malicious nodes sending good trust scores to untrustworthy service is increased. In contrast, the simple trust model fails to deal with that attack. The trust score of the untrustworthy service provider increases up to 100% in relation to the starting score (shown in Figure 7.31).

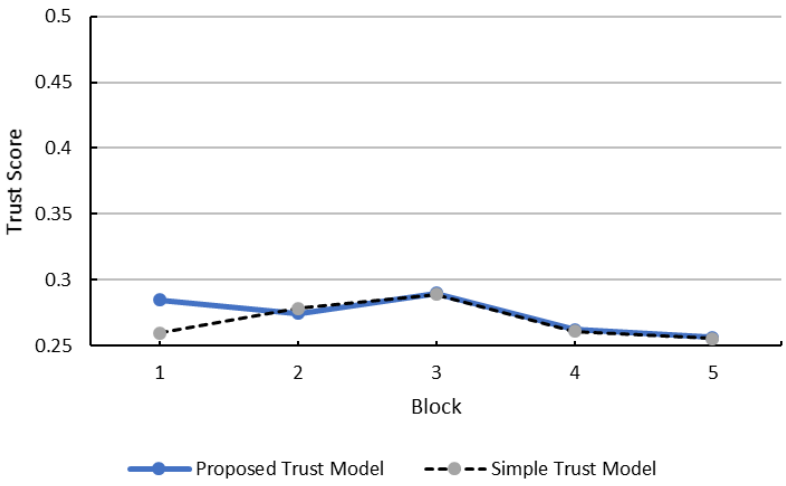

Figure 7.26: Trustworthy peers (Shala *et al.*, 2020a)

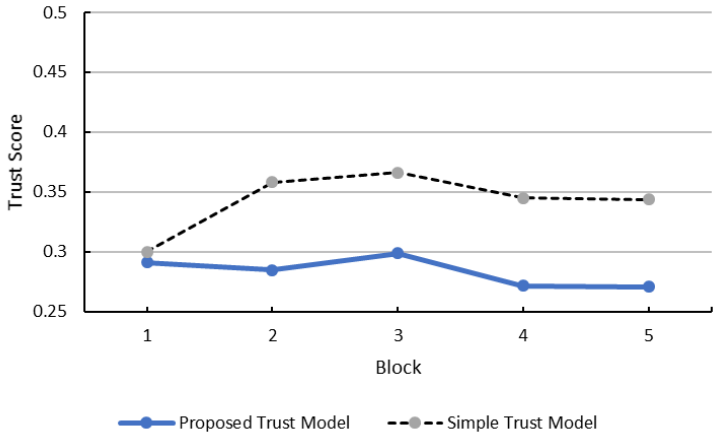

Figure 7.27: 20% of malicious peers (Shala *et al.*, 2020a)

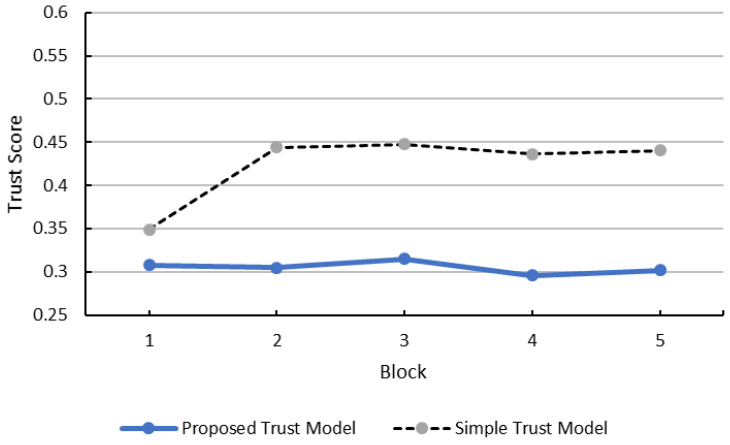

Figure 7.28: 40% of malicious peers (Shala *et al.*, 2020a)

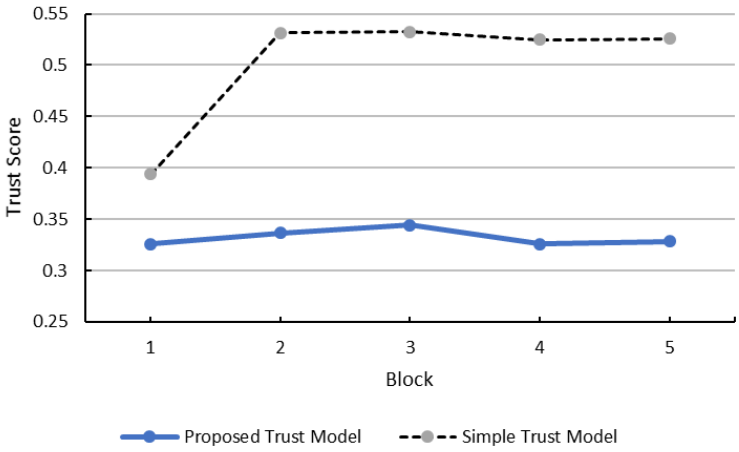

Figure 7.29: 60% of malicious peers (Shala *et al.*, 2020a)

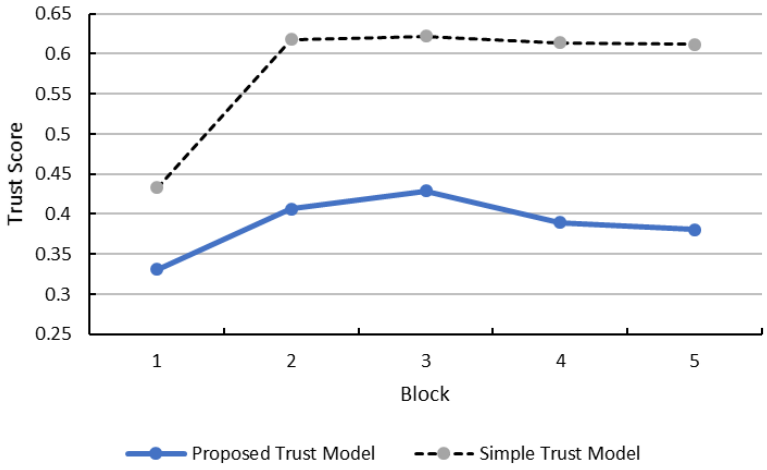

Figure 7.30: 80% of malicious peers (Shala *et al.*, 2020a)

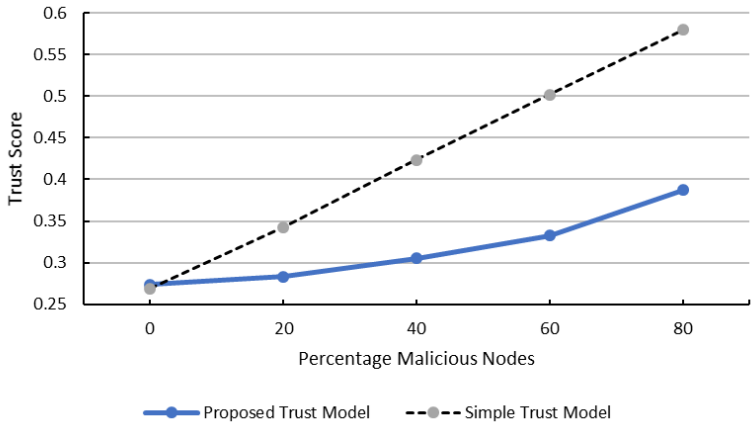

Figure 7.31: Trust evolution in relation to malicious population (Shala *et al.*, 2020a)

Scenario 5: Comparative Analysis

This scenario presents a comparative analysis of the proposed trust model against other relevant trust approaches already evaluated theoretically in the previous chapters: BlockTIoT (Lahbib *et al.*, 2019), HierSysT (Kouicem *et al.*, 2018), TrustChain (Malik *et al.*, 2019b), SybRet (Asiri and Miri, 2018), and TArChain (Dedeoglu *et al.*, 2019). The aim is to show the performance of these approaches when being attacked by an increasing

population of malicious nodes. Beforehand, this scenario defines a relative trust score necessary for the assessment. The relative trust score is calculated based on the average of all trust scores conducted by different trust models under normal conditions.

In the first case, the bad-mouthing attack is performed by malicious nodes for different block rounds. During the first case, the percentage of malicious nodes is increased. Figures 7.32 – 7.35 show the outcomes when using the different trust models. They also highlight the strengths of the proposed trust model, which stays quite stable and resilient throughout the attacks. The outperformance results from ignoring the increasing percentage of false trust information coming from untrustworthy nodes in the block building process. The initial trust score feature of the proposed trust model also enables a true start in building trust among the nodes. Related approaches do not ensure this. The main limitations of existing trust models are the uncomplete trust metric lists, susceptible weighting systems, and missing trust entity considerations. Figure 7.36 confirms the trust resilience of the proposed trust model in comparison to BlockTIoT (Lahbib *et al.*, 2019), HierSysT (Kouicem *et al.*, 2018), TrustChain (Malik *et al.*, 2019b), SybRet (Asiri and Miri, 2018), and TArChain (Dedeoglu *et al.*, 2019) under increasing malicious nodes population in the network. As a result, bad-mouthing attacks, where nodes try to downrate a good performing node, are almost mitigated by the proposed trust model.

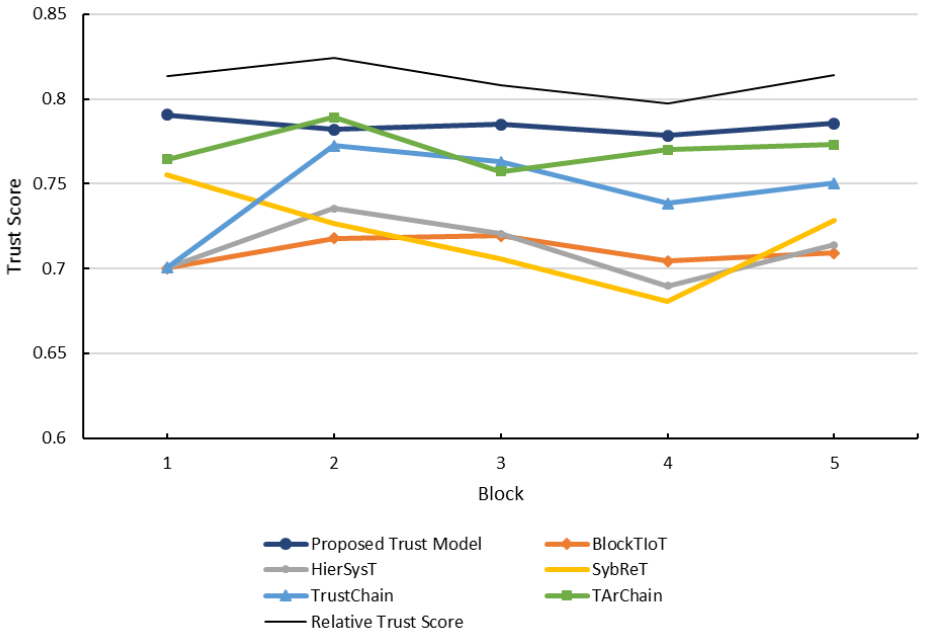

Figure 7.32: 20% of malicious peers (bad-mouthing) (Shala *et al.*, 2020a)

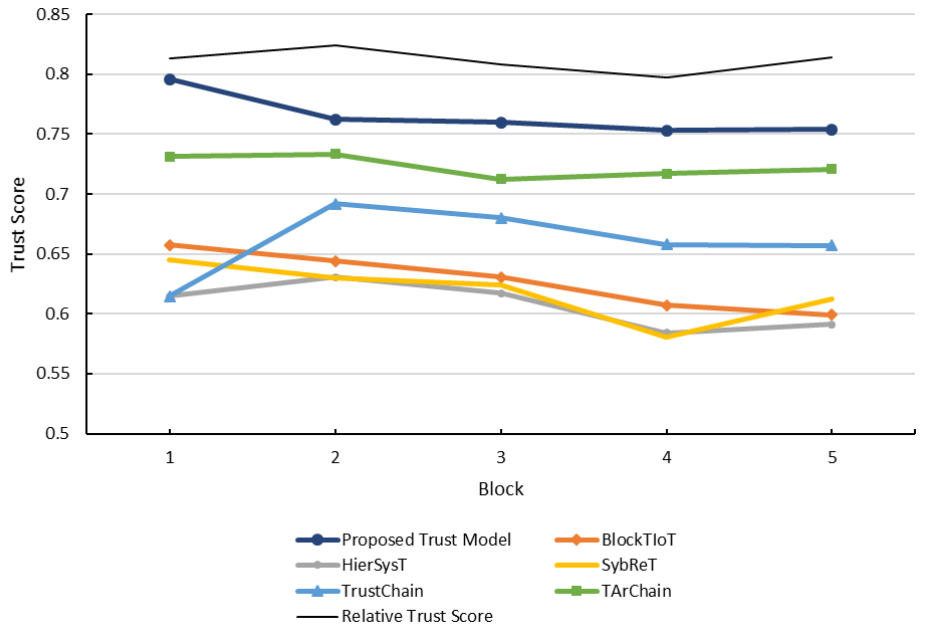

Figure 7.33: 40% of malicious peers (bad-mouthing) (Shala *et al.*, 2020a)

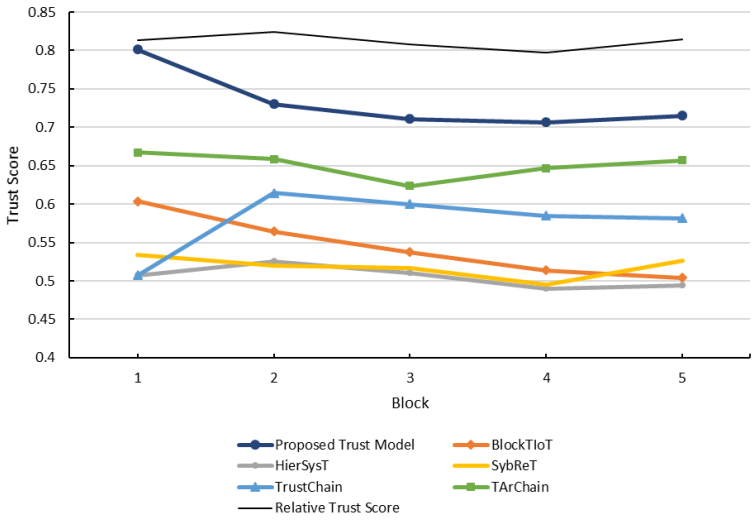

Figure 7.34: 60% of malicious peers (bad-mouthing) (Shala *et al.*, 2020a)

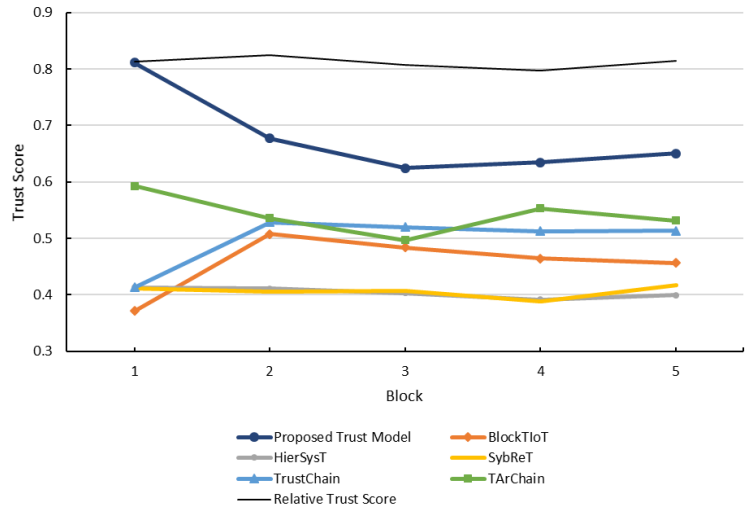

Figure 7.35: 80% of malicious peers (bad-mouthing) (Shala *et al.*, 2020a)

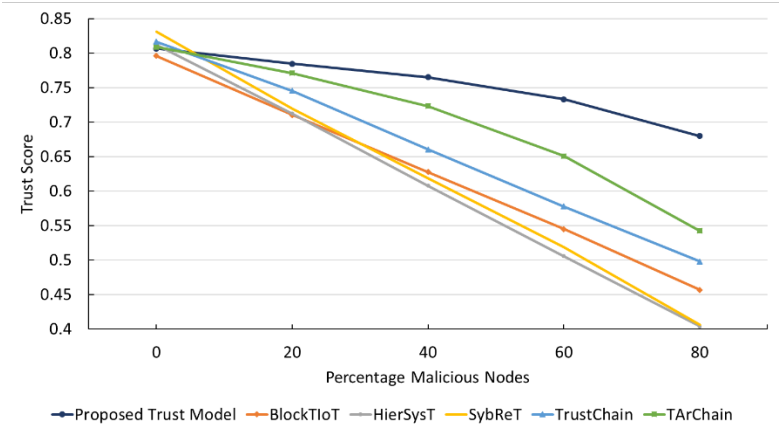

Figure 7.36: Trust evolution (bad-mouthing) (Shala *et al.*, 2020a)

Similarly, the second case presents a comparison of different trust models under the influence of the ballot-stuffing attack (shown in Figures 7.37 – 7.40). The goal of the malicious nodes is to increase the trust score of one of their friends to harm the system. The figures illustrate that this attack only slightly impacts the proposed trust model. The global view about the bad node remains untrustworthy throughout the different block rounds. Figure 7.41 shows the trust evolution under the increasing malicious population. It demonstrates the reliability and stability of the proposed trust model again. Most of the existing trust models are quickly impacted by the attack, giving the malicious nodes the possibility to change the overall opinion of the M2M community. By doing so, malicious nodes take advantage to control the network and to perform further attacks.

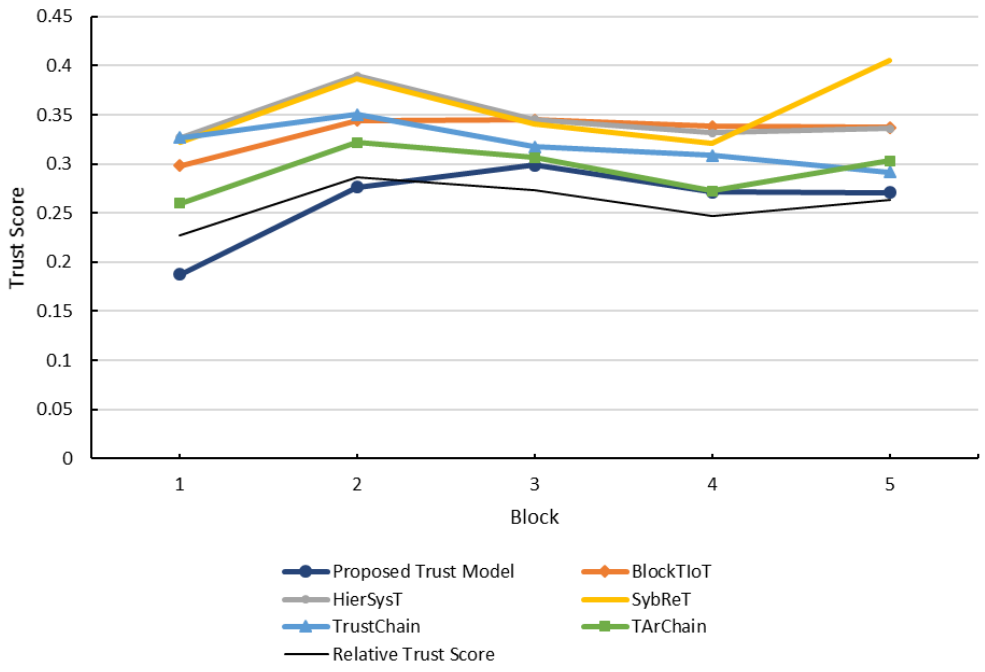

Figure 7.37: 20% of malicious peers (ballot-stuff) (Shala *et al.*, 2020a)

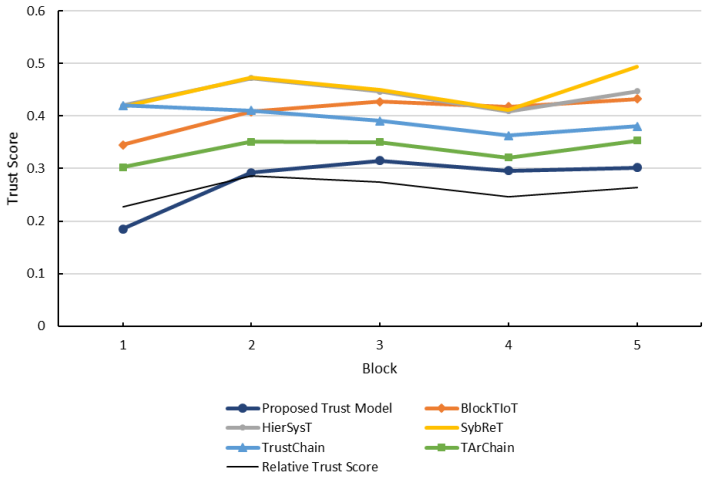

Figure 7.38: 40% of malicious peers (ballot-stuff) (Shala *et al.*, 2020a)

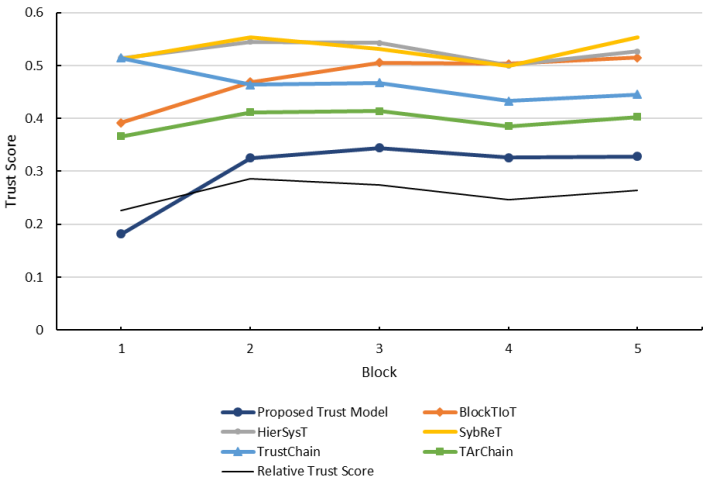

Figure 7.39: 60% of malicious peers (ballot-stuff) (Shala *et al.*, 2020a)

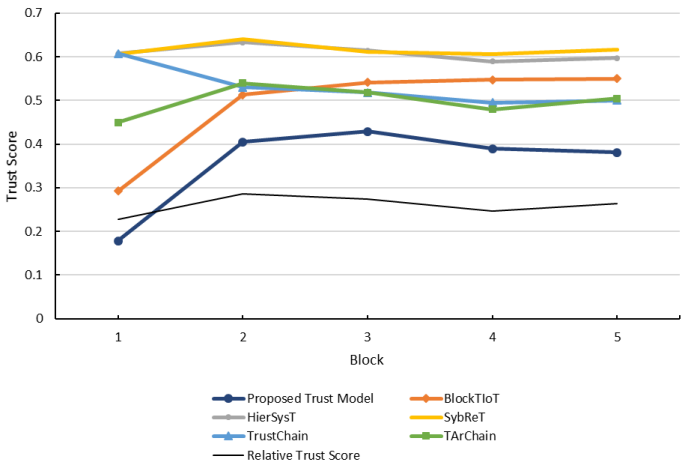

Figure 7.40: 80% of malicious peers (ballot-stuff) (Shala *et al.*, 2020a)

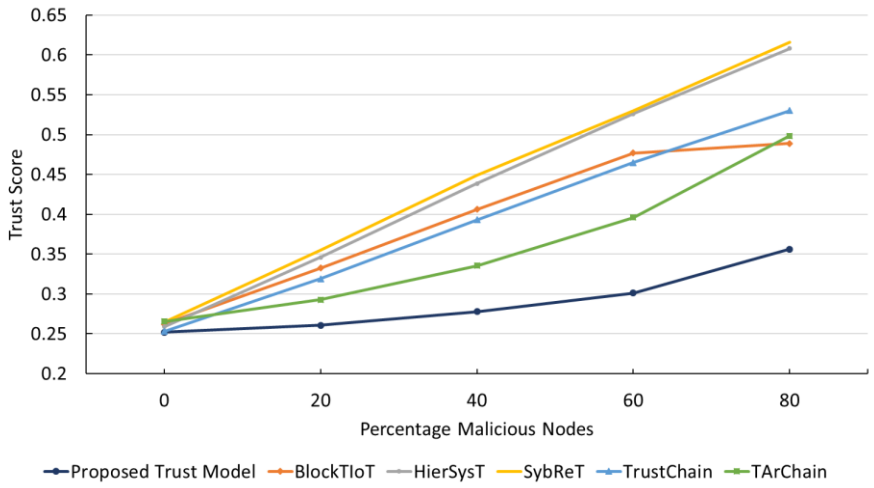

Figure 7.41: Trust evolution (ballot-stuff) (Shala *et al.*, 2020a)

To sum up, the comparative analysis demonstrates the high resilience of the proposed trust model against various attacks (bad-mouthing and ballot-stuffing). The increasing number of malicious nodes in the network also has a low impact on its performance. The different scenarios highlight the advantages of the proposed trust model. The initial trust score scheme, the dynamic weighting system, the integration of trust in all trust evaluation steps, and the blockchain technology are key elements to enable a trustworthy and reliable M2M environment.

7.2.4 Trust-Consensus Protocol Evaluation

Section 5.1 proposed a novel trust-based consensus protocol called Trust-CP. This protocol includes trust elements in all steps of its workflow, starting from the selection of the block creator until the validation of new blocks. The Trust-CP is an important aspect of this research used not only for the blockchain activities but also for several other M2M community activities. Therefore, this section runs different scenarios to evaluate the attack robustness of the Trust-CP in comparison with other existing consensus protocols.

Some parts in this section were published in (Nguyen, 2019; Shala *et al.*, 2019c).

Scenario Settings

The Trust-CP and other consensus protocols are evaluated under the influence of several attack types. These attacks are explained in the following:

- Attack 1: One or more nodes try to spam the local memory pools of other nodes by broadcasting fake transactions. Afterwards, these transactions can be added by the block creator to a new block. This block is distributed to other nodes for validation and inclusion in the blockchain.
- Attack 2: The block creator tries to harm the system by including fake transactions in a new block and broadcasting it to validation nodes. The process results in a new block accepted to the blockchain. The transactions inside that block contain fake low trust scores about trustworthy nodes.

The scenarios are running on several virtual machines on the Core Emulator (as introduced in section 7.2.2). The essential functionalities of the different consensus protocols are implemented prototypically in Java. Besides the Trust-CP, the evaluation includes traditional consensus protocols (PoW, PoS, DPoS, Nano, Ripple, and Tangle), theoretically evaluated in section 5.1.

In the following some additional details on the implemented application:

- The prototypical implementation consists of a simplified approach of the core functionalities of the different consensus protocols. Only the key ideas and concepts of the consensus protocols presented in section 5.1 were considered and implemented (with a focus on the consensus workflow).

- Implementation was carried out with Java without specific consensus protocol libraries. Thus, the prototypical implementation does not consist of the official implementation of the protocols, which might be used in a real-world environment.
- The important contribution is not on the complexity of the implementation but on the produced environment showing the key aspects of the novelties as proof of concept. The value is not in the code but on the presented trust model and its resilience compared to related approaches.

The following settings and conditions are used to assess the robustness of the different consensus protocols:

- Every participating blockchain node has 1000 transactions in its local memory pool.
- The percentage of malicious nodes changes throughout the scenarios.
- The number of fake transactions within the 1000 transactions is increased through the scenarios from 100 to 900 fake transactions.
- Every block includes one transaction.

Results

The aim of the scenarios is to compare the performance of the different consensus protocols when running attacks against them. The ratio between correct and fake blocks is used as an indicator for the assessment. Throughout the scenarios, the number of fake transactions and the percentage of malicious nodes are continuously increased.

The first scenario assumes that all participating nodes are trustworthy. Moreover, it considers that there are fake transactions in the local memory pools of the nodes. Figure 7.42 shows the outcomes of the first scenario. It highlights that fake transactions do not influence Trust-CP. The Trust-CP includes only correct and high trusted transactions in a new block, even if there are many fake ones. Another feature of the Trust-CP is that it assigns trust scores for each transaction based on its originator. The functionality of the Trust-CP enables it to filter out fake transactions and to include only good ones. Figure 7.42 also shows that traditional consensus protocols do not perform well under the increasing number of fake transactions in the network. The figure also reveals that traditional consensus protocols do not have a mechanism to distinguish between trusted or untrusted transactions. Only the Ripple consensus protocol relies on transaction similarity. With an increasing number of fake transactions, the probability of passing the Ripple similarity check is high.

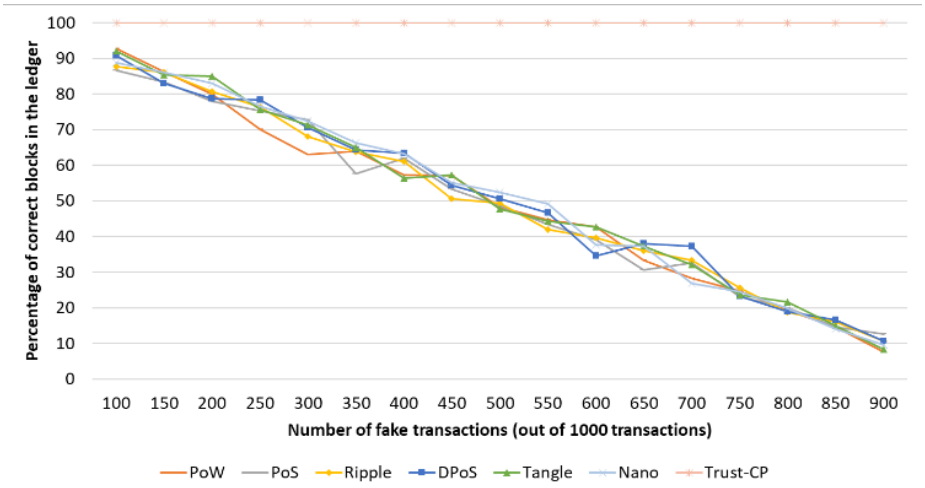

Figure 7.42: Fake transaction in various protocols (0% malicious nodes) (Shala *et al.*, 2019c)

The next scenarios include malicious nodes controlling the block creation and validation part. For these scenarios, the percentage of malicious nodes is continuously increased.

Figures 7.43 – 7.47 show the performance of the different consensus protocols under these conditions. The results show that except for the proposed Trust-CP, all other traditional protocols have a decreasing resilience with the increasing number of malicious nodes and the number of fake transactions. The bad performance is due to the fact that traditional consensus protocols rely on computing power or simple selection algorithms to perform block creation activities and do not include trust in any part. Thus, if a malicious node is selected as a block creator, it creates a block, solves the cryptographical puzzle, and broadcasts the block to others for validation. The other nodes verify the hash values and the keys of the created block and accept them. They do not consider the trustworthiness of the block, block creator, or the transactions part of the block. The higher the percentage of malicious nodes in the network, the higher the probability of selecting one malicious block creator. Overall, the performance of the reviewed traditional consensus protocol is continually decreasing. One example is the Ripple consensus protocol. Without malicious nodes in the network, it performs like the other ones. When adding malicious nodes, its overall security breaks down because of the similarity check feature of Ripple. By increasing the number of malicious nodes, the probability that the malicious nodes will have the same fake transactions for the similarity check is high. As a result, the number of fake transactions in the ledger is also high. The DPoS consensus protocol performs better than the others in the beginning (see Figure 7.43 and Figure 7.44). It can be argued by the fact that the DPoS uses a voting mechanism to select the block creation leader. As long as the number of malicious nodes in the network is not equal to or higher than 50%, the other nodes are more likely to elect a good node as a leader, resulting in a higher number of correct blocks in the ledger.

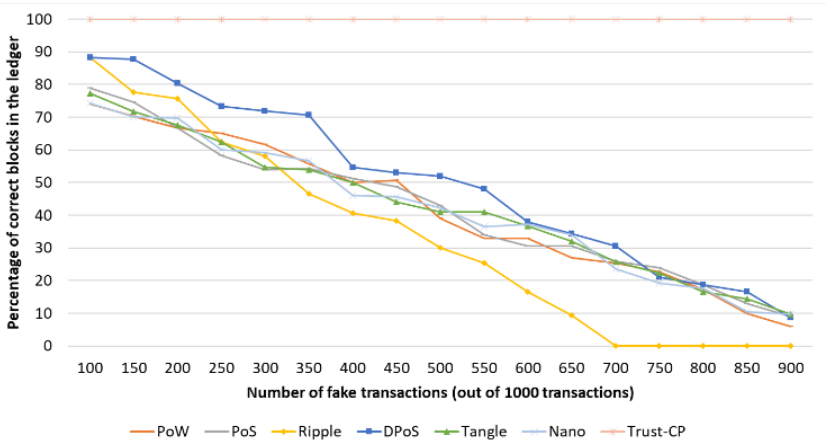

Figure 7.43: Fake transaction in various protocols (16% malicious nodes) (Shala *et al.*, 2019c)

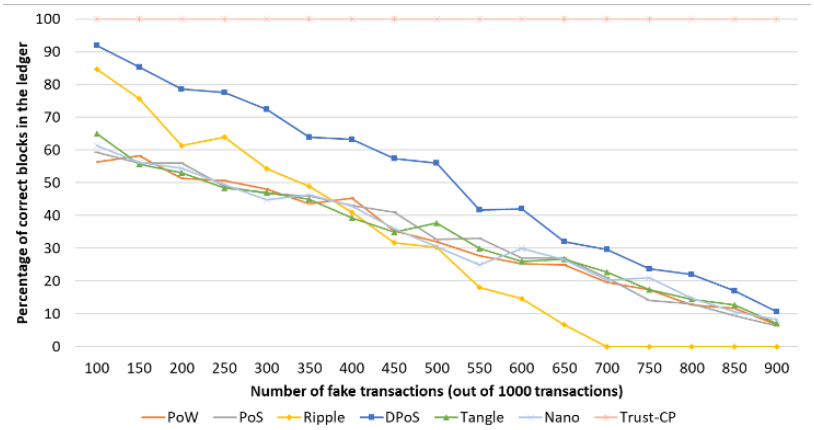

Figure 7.44: Fake transaction in various protocols (32% malicious nodes) (Shala *et al.*, 2019c)

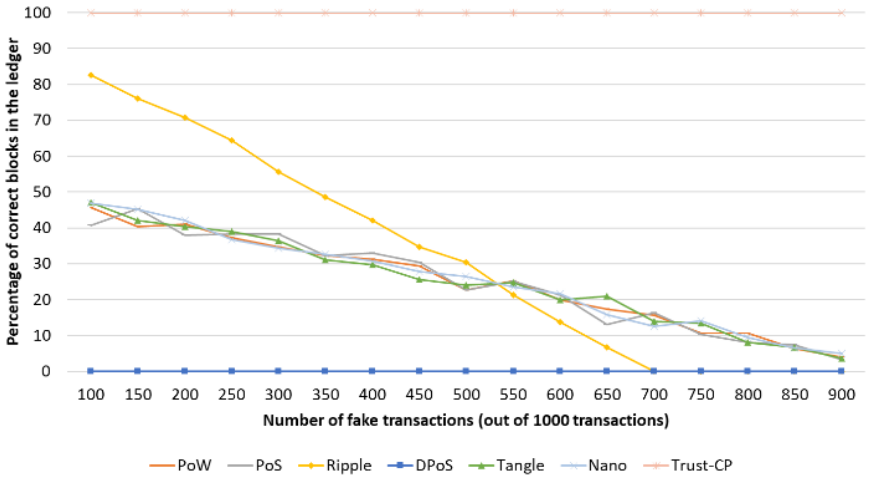

Figure 7.45: Fake transaction in various protocols (50% malicious nodes) (Shala *et al.*, 2019c)

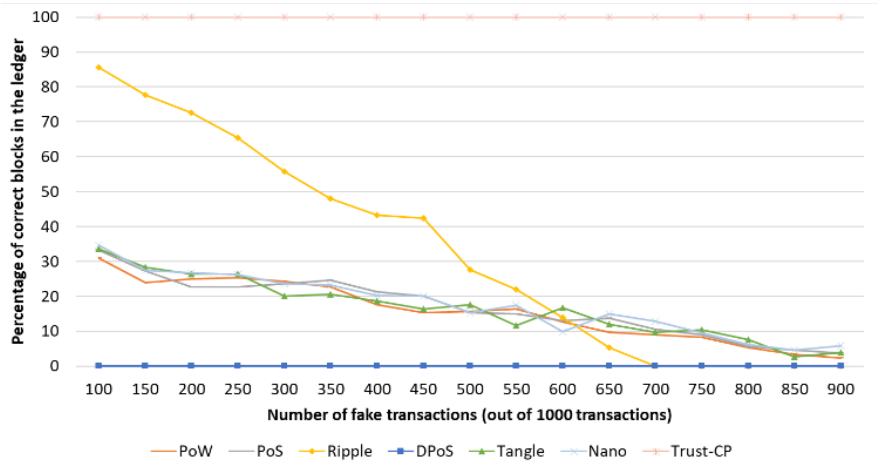

Figure 7.46: Fake transaction in various protocols (66% malicious nodes) (Shala *et al.*, 2019c)

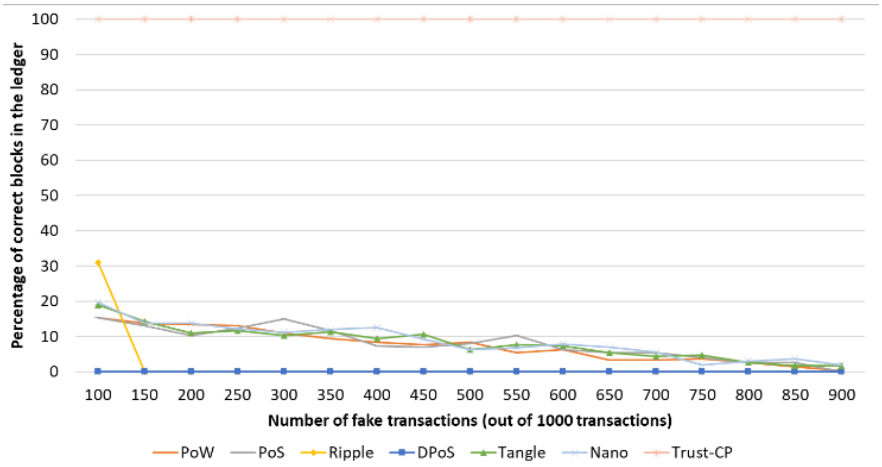

Figure 7.47: Fake transaction in various protocols (84% malicious nodes) (Shala *et al.*, 2019c)

7.3 Conclusion

This chapter presented the research prototype and the framework evaluation. Section 7.1 has assessed whether the framework meets the different requirements derived in section 3.3. The outcomes of the evaluation show that the proposed framework fulfils these requirements. Section 7.2 has presented the implemented prototype used to prove the concepts introduced within these theses. The prototype separated the essential

functionalities of the proposed framework into two main parts consisting of functional testing and trust evaluation.

The proof of concept for functional testing was demonstrated using a use case with several nodes providing different services and combining these to create a cooperative M2M service. The proof of concept illustrated the whole workflow of testing the functionality of single and cooperative M2M services. It has shown the transformation of the existing SCXML service description to a test model used by a test generation tool to derive appropriate test cases. Finally, these test cases are executed on the SUTs using the execution environment of Eclipse Titan.

The proof of concept for the proposed trust evaluation system focused on illustrating the core functionality of the blockchain-based trust management system. All the nodes are using a blockchain network to store the trust information. Moreover, the Trust-CP is used to achieve consensus for new blocks. A graphical user interface enables end-users to control and monitor the trust status of their own or third-party M2M services. The presented GUI also provides the possibility to show the trust evolution for a specific M2M entity.

Another focus of the proof of concept for the trust evaluation system was its resilience against different attacks. The results of the various scenarios show that the introduced trust model is less affected by various trust attacks. Combining the trust model with the blockchain and the Trust-CP makes the approach more resilient than other models when the number of malicious nodes providing false trust information is increased. Trust scores are securely stored in the blockchain, and only trustworthy nodes perform the trust evaluation activities. The Trust-CP ensures that only trusted transactions coming from

trusted sources are included in the blockchain. In addition, the initial trust score of the proposed approach supports earlier detection of malicious behaviour in the community. Finally, section 7.2 included the evaluation of the novel Trust-CP in relation to traditional blockchain-based consensus protocols. Therefore, the percentage of fake transactions and malicious nodes was manipulated, and the performance of the protocols measured. The scenarios concluded with outstanding results of the Trust-CP.

Overall, the evaluation and outcomes in this chapter have demonstrated the main functionality of the proposed framework and proved its applicability and resilience.

8 Conclusion and Future Work

This chapter concludes the thesis by summarising the main achievements of the research work (section 8.1). Identified limitations of the research are discussed in section 8.2. Finally, section 8.3 suggests potential ideas for further research.

8.1 Achievements of the Research

This research work was dedicated to the proposed and benchmarked novel approach for trust evaluation and corresponding functional testing of decentralised M2M services. The integration of end-users and their personal environment devices into the M2M service provision process is an emerging and promising approach for the community. However, chapter 2 identified that the lack of trust within the entities and unreliable M2M services are the main deficits harming the success of decentralised and end-user-based M2M service marketplaces.

Based on these findings, chapter 3 performed an exhaustive literature review of relevant and most current works in the field of testing and trust in M2M/IoT. Current trust approaches mainly suffer from insecure trust data storage, missing trust information on new M2M entities, and incomplete trust models. Existing test approaches are not end-user-friendly, rely on standard service development lifecycles, and provide inappropriate test development strategies. The analyses of these projects lead to several requirements relevant for a novel framework addressing trust and test issues. The assessment of the related projects illustrated that none of them fulfilled these requirements and reinforced

the necessity for an optimised approach for ensuring trust and correct working M2M services within the M2M community.

A novel framework has been designed based on the identified strengths and limitations of existing works (refer to chapter 4). The three main components of the proposed framework architecture are the trust engine, test engine, and P2P storage engine. The P2P storage engine was introduced to ensure the high integrity of all relevant data in the M2M community. It consists of the DHT and the Blockchain storage, which provide both a decentralised architecture decoupled from centralised operators. The DHT is used for the most recent data, whereas the blockchain for storing the history of all transactions. End-users also have the possibility to perform integrity check-ups by comparing the data from both storage types. The P2P storage engine also serves as a data source to support the trust evaluation and functional testing processes. The trust engine is the core element of the novel framework and covers all trust evaluation activities within the M2M community. It enables the evaluation of new and existing M2M entities. Moreover, it uses a comprehensive trust model depicted with several trust metrics to assess trust in the community. The test engine enables end-user-friendly functional testing of services and supports the trust engine for trust evaluation activities. Blockchain principles were also integrated into the different components of the framework to support the maintenance of the decentralised network and to optimise the different workflows through trust-based consensus-building schemes. The proposed framework can be included and operated within the service creation environment located in the environment of end-users. The combination of trust, testing, and blockchain to optimise decentralised M2M services forms the general novelty of this research.

Trust information is sensitive data and essential to ensure the right trust in a decentralised network (refer to chapter 5). Therefore, this research proposed to store trust data in a blockchain. Blockchain technology is known as a tamper-proof ledger and decentralised consensus-building system. In the context of blockchain, a deep analysis of existing consensus protocols was done, and the main limitations identified. Based on this evaluation, a novel consensus protocol called Trust-CP was proposed for the blockchain and also used to optimise other parts of the framework. The integration of blockchain within the proposed framework and the trust-based consensus protocol build two further novelties in this research.

Another novelty of this research is the synergy of testing and trust evaluation activities. This combination enables the trust assessment for new M2M entities joining the community (see chapter 5). Service testing is also part of the trust model used to evaluate existing M2M entities. It forms an important pillar within the different trust metrics. It confirms the functional behaviour of different services and is a good starting point to build trust relationships among the nodes.

Chapter 6 introduced a novel trust evaluation system covering different aspects of an M2M entity. It comprises service- and peer-related attributes such as service functionality, service performance, consumer experience, or participation willingness in various community tasks. These attributes are assessed through different community activities. Two of these activities are maintaining the blockchain network and acting as a test agent in the M2M community. Incorporating several elements into the trust evaluation process ensures a competition-based challenge among the M2M members, enhancing the quality of the computed trust score. The competition character of the

proposed approach incentivises the nodes to perform various necessary activities to boost up their trust scores.

Another key novelty of this research is the synergy of blockchain and trust to form a blockchain-based trust evaluation system. The trust evaluation activities were merged with the blockchain ones. Thus, blockchain transactions containing trust information are sent among the nodes and confirmed using the newly introduced Trust-CP. Additionally, a trust computation and aggregation scheme, including a dynamic weighting system, was introduced. These proposals, resulting in the combination of blockchain principles, Trust-CP, and trust evaluation, enable a more resilient M2M community and ensure trustworthy trust relationships among the nodes.

The proposed framework was theoretically and practically evaluated in chapter 7. The theoretical evaluation concluded that the framework fulfils all of the identified requirements. The practical evaluation included aspects of service testing, trust management, consensus-building, and attack resilience. The workflow of the proposed service testing approach was shown through a use case scenario. The trust management was illustrated with an end-user-friendly user interface and included trust evaluation activities, blockchain maintenance, and trust-based consensus protocol. The performance of the proposed trust model and the novel Trust-CP were evaluated using several scenarios and attack types. The evaluation results highlighted the outstanding performance of the proposed framework in comparison with other existing approaches.

In summary, this research work has met all of the objectives outlined in chapter 1. It has resulted in the design of a framework for trust evaluation and corresponding functional testing of decentralised M2M services. Beyond, the proposed framework can be

incorporated in different application fields and serves as a strong basis for further research activities. The research outcomes of this thesis were presented at related conferences and published in several conferences and internationally recognised journals. The publications received positive comments from reviewers and delegates. Moreover, two of the research papers were awarded the best paper prize.

8.2 Limitations of the research

This research project has met all objectives and presented several novelties to enable trust in decentralised M2M communities. However, some limitations were unavoidable as they are outside the scope of this research and require an extraordinary expenditure in areas where no novel findings were expected. The limitations are summarised below:

1. As part of the trust approach, this research has presented an end-user-friendly service testing concept where every node part in the M2M community has the possibility to act as a test agent. The focus was on optimising the existing system models and creating test models which can be used for test case generation. However, further investigations on test case techniques and execution are not conducted as the value of knowledge would be limited. There are already several well-established approaches regarding test generation.
2. Although they are part of the service testing and trust evaluation concept, the integration of test data and performance testing are not finally investigated in this research. Test data coming from the blockchain storage is used in combination with abstract test cases to perform test executions. Performance testing is proposed to be

used in addition to functional testing for evaluating the initial trust score of M2M entities. Both concepts in this research can be further expanded.

3. Some trust metrics, such as service monitoring or rating, are not incorporated within the trust evaluation prototype as they are not key novelties in this research. Nevertheless, the research prototype aimed to show that the proposed approach is viable and also that it is resilient against different trust attacks.
4. The presented prototype for evaluating functional testing is used to show the general workflow for testing a cooperative M2M service. However, the service testing component is implemented as a single system outside the trust evaluation. This aspect is not considered in the implementation as it would not have brought much added value.
5. The application of the proposed framework to optimise the M2M service provision approach and the integration of smart contracts are not considered in the prototype evaluation as they present prospective works and are outside of the scope of this research thesis.

Despite these limitations, the research project has made valid contributions to the knowledge and research community.

8.3 Future Work

The synergy of blockchain technology and trust has powerful attributions to effectively counteract security and trust-related issues present in decentralised communities. As such, it enables a strong perspective for end-user-based M2M service provision and beyond it. This research has advanced the field for trust evaluation of decentralised M2M services.

Nevertheless, some suggestions and areas for future work have been identified and are summarised below.

1. This research gives other researchers a comprehensive overview of trust evaluation and blockchain within the domain of M2M/IoT. The proposed blockchain-based trust model can serve as a good basis for further research on increasing the trustworthiness in M2M communities by incorporating blockchain. The combination of blockchain and trust opens new research questions and directions. Further research may use this to identify open issues in other fields. Other fields of interest could include (but are not limited to) VANETs, Flying Ad-Hoc Networks (FANETs) or the Internet of Everything (IoE).
2. Another interesting aspect is the incorporation of the proposed concepts within the workflow of M2M service provision. Further work should investigate the admission of new M2M entities in the M2M community, the registration of M2M services, the creation of cooperative M2M services, and the data management in the M2M community, and how the proposed framework can be used to increase the overall confidence of the M2M service provision workflow.
3. Further improvements could include the application of smart contracts to enable trustless agreements between entities and the automation of different processes without relying on third parties. The integration of trust concepts within smart contracts provides new avenues for research, such as automated and trusted service provision or for performing service tests and trust evaluations.
4. Further research could be carried out to review other trust metric parameters and attributes that can be used to extend the trust model. One aim should be to improve the capabilities of the trust model and apply it in different use cases to achieve a

considerable generalisation level. Also, potential scalability issues could be identified and addressed.

5. Recently, the next generation of the Internet called Web3 is rising up. The Web3 is characterised by incorporating blockchain for information storage, smart contracts for agreement executions, end-users serving as maintenance entities, and the development of decentralised application within this network. The trustworthiness of the Web3 is still an open research issue where the introduced blockchain-based trust model fits very well to the Web3 architecture as it considers, similar to the Web3, completely decentralised networks, blockchain for integrity reasons, and the integration of end-users in decision-making processes. Incorporating the presented approach within the next generation Internet is also part of future work and can contribute to further research in academia.
6. The ideas regarding the combination of trust and smart contracts for trust automation within the decentralised M2M community can be further developed, e.g., through several other scenarios in M2M and beyond.
7. Future work could include further considerations on different testing aspects such as the methodology for describing end-user-based M2M services, the generation of reusable test modules and test data, the derivation of extended test suites for complex M2M service functionality, and the process for testing cooperative M2M services.

References

Adewuyi, A. A., Cheng, H., Shi, Q., Cao, J., MacDermott, A. and Wang, X. (2019) 'CTRUST: A Dynamic Trust Model for Collaborative Applications in the Internet of Things', *IEEE Internet of Things Journal*, 6(3), pp. 5432–5445. doi: 10.1109/JIOT.2019.2902022.

Ahmad, A., Bouquet, F., Fourneret, E., Le Gall, F. and Legeard, B. (2016) 'Model-Based Testing as a Service for IoT Platforms', in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 727–742. doi: 10.1007/978-3-319-47169-3_55.

Ali, J., Ali, T., Alsaawy, Y., Khalid, A. S. and Musa, S. (2019) 'Blockchain-based Smart-IoT Trust Zone Measurement Architecture', in *Proceedings of the International Conference on Omni-Layer Intelligent Systems*. New York, NY, USA: ACM, pp. 152–157. doi: 10.1145/3312614.3312646.

Aljazzaf, Z. M., Capretz, M. A. M. and Perry, M. (2011) 'Trust bootstrapping services and service providers', in *2011 Ninth Annual International Conference on Privacy, Security and Trust*. IEEE, pp. 7–15. doi: 10.1109/PST.2011.5971957.

Alowayed, Y., Canini, M., Marcos, P., Chiesa, M. and Barcellos, M. (2018) 'Picking a partner: A fair blockchain based scoring protocol for autonomous systems', in *ANRW 2018 - Proceedings of the 2018 Applied Networking Research Workshop*, pp. 33–39. doi: 10.1145/3232755.3232785.

Amalfitano, D., Amatucci, N., De Simone, V., Riccio, V. and Rita, F. A. (2017) 'Towards a Thing-In-the-Loop approach for the Verification and Validation of IoT systems', in *Proceedings of the 1st ACM Workshop on the Internet of Safe Things*. New York, NY, USA: ACM, pp. 57–63. doi: 10.1145/3137003.3137007.

Asemani, M., Abdollahei, F. and Jabbari, F. (2019) 'Understanding IoT Platforms: Towards a comprehensive definition and main characteristic description', in *2019 5th International Conference on Web Research, ICWR 2019*. IEEE, pp. 172–177. doi: 10.1109/ICWR.2019.8765259.

Asiri, S. and Miri, A. (2016) 'An IoT trust and reputation model based on recommender systems', in *2016 14th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, pp. 561–568. doi: 10.1109/PST.2016.7907017.

Asiri, S. and Miri, A. (2018) 'A Sybil Resistant IoT Trust Model Using Blockchains', in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, pp. 1017–1026. doi: 10.1109/Cybermatics_2018.2018.00190.

Awan, K. A., Din, I. U., Almogren, A., Guizani, M., Altameem, A. and Jadoon, S. U. (2019a) 'RobustTrust – A Pro-Privacy Robust Distributed Trust Management Mechanism for Internet of Things', *IEEE Access*, 7, pp. 62095–62106. doi: 10.1109/ACCESS.2019.2916340.

Awan, K. A., Din, I. U., Zareei, M., Talha, M., Guizani, M. and Jadoon, S. U. (2019b) 'HoliTrust-A Holistic Cross-Domain Trust Management Mechanism for Service-Centric Internet of Things', *IEEE Access*, 7, pp. 52191–52201. doi: 10.1109/ACCESS.2019.2912469.

Bahri, L. and Girdzijauskas, S. (2018) 'When Trust Saves Energy', in *Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18*. New York, New York, USA: ACM Press, pp. 1165–1169. doi: 10.1145/3184558.3191553.

Bahutair, M., Bougeuttaya, A. and Ghari Neiat, A. (2019) 'Adaptive Trust: Usage-Based Trust in Crowdsourced IoT Services', in *2019 IEEE International Conference on Web Services (ICWS)*. IEEE, pp. 172–179. doi: 10.1109/ICWS.2019.00038.

Baker, P., Dai, Z. R., Grabowski, J., Haugen, Ø., Schieferdecker, I. and Williams, C. (2008) *Model-driven Testing: Using the UML testing profile*, *Model-Driven Testing: Using the UML Testing Profile*. Berlin, Heidelberg: Springer-Verlag. doi: 10.1007/978-3-540-72563-3.

Bashir, I. (2017) *Mastering Blockchain*. Birmingham: Packt Publishing.

Benkerrou, H., Heddad, S. and Omar, M. (2016) 'Credit and honesty-based trust assessment for hierarchical collaborative IoT systems', in *2016 7th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)*. IEEE, pp. 295–299. doi: 10.1109/SETIT.2016.7939883.

Biswas, P. (2021) *Development and realisation of blockchain-based trust evaluation system for IoT services and service providers*. Master Thesis, Frankfurt University of Applied Sciences.

Bitcoin - Developer Guides (2021). Available at: https://developer.bitcoin.org/devguide/ (Accessed: 16 March 2021).

Bosmans, S., Mercelis, S., Denil, J. and Hellinckx, P. (2019) 'Testing IoT systems using a hybrid simulation based testing approach', *Computing*, 101(7), pp. 857–872. doi: 10.1007/s00607-018-0650-5.

Boswarthick, D., Elloumi, O. and Hersent, O. (2012) *M2M Communications*, *M2M Communications: A Systems Approach*. Edited by D. Boswarthick, O. Elloumi, and O. Hersent. Chichester, UK: John Wiley & Sons, Ltd. doi: 10.1002/9781119974031.

Boussard, M., Papillon, S., Peloso, P., Signorini, M. and Waisbard, E. (2019) 'STewARD:SDN and blockchain-based Trust evaluation for Automated Risk management on IoT Devices', in *INFOCOM 2019 - IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2019*, pp. 841–846. doi: 10.1109/INFCOMW.2019.8845126.

Boustanifar, F. and Movahedi, Z. (2016) 'A Trust-Based Offloading for Mobile M2M Communications', in *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*. IEEE, pp. 1139–1143. doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0175.

Castro, M., Jara, A. J. and Skarmeta, A. F. (2012) 'An Analysis of M2M Platforms:

Challenges and Opportunities for the Internet of Things', in *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. IEEE, pp. 757–762. doi: 10.1109/IMIS.2012.184.

Chalaemwongwan, N. and Kurutach, W. (2018) 'State of the art and challenges facing consensus protocols on blockchain', in *International Conference on Information Networking*, pp. 957–962. doi: 10.1109/ICOIN.2018.8343266.

Chen, I.-R., Guo, J. and Bao, F. (2016) 'Trust Management for SOA-Based IoT and Its Application to Service Composition', *IEEE Transactions on Services Computing*, 9(3), pp. 482–495. doi: 10.1109/TSC.2014.2365797.

Christidis, K. and Devetsikiotis, M. (2016) 'Blockchains and Smart Contracts for the Internet of Things', *IEEE Access*, 4, pp. 2292–2303. doi: 10.1109/ACCESS.2016.2566339.

Cirani, S., Ferrari, G., Picone, M. and Veltri, L. (2018) *Internet of Things*. Chichester, UK: John Wiley & Sons, Ltd. doi: 10.1002/9781119359715.

CORE (2020) *Common Open Research Emulator (CORE) Documentation*. Available at: http://coreemu.github.io/core/ (Accessed: 16 March 2021).

CORE (2021) *Common Open Research Emulator (CORE), U.S. Naval Research Laboratory*. Available at: https://www.nrl.navy.mil/Our-Work/Areas-of-Research/Information-Technology/NCS/CORE/ (Accessed: 16 March 2021).

Cornutum (2020) *Tcases: The Complete Guide Version 3.5.0*. Available at: http://www.cornutum.org/tcases/docs/Tcases-Guide.htm (Accessed: 16 March 2021).

COTI (2018) *The Trust Chain Consensus, COTI: a Decentralized, High Performance Cryptocurrency Ecosystem Optimized for Creating Digital Payment Networks and Stable Coins, White Paper*. Available at: https://coti.io/files/COTI-technical-whitepaper.pdf (Accessed: 16 March 2021).

COTI (2021). Available at: https://medium.com/cotinetwork (Accessed: 16 March 2021).

Dedeoglu, V., Jurdak, R., Putra, G. D., Dorri, A. and Kanhere, S. S. (2019) 'A trust architecture for blockchain in IoT', in *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. New York, NY, USA: ACM, pp. 190–199. doi: 10.1145/3360774.3360822.

Djamaludin, C. I., Foo, E. and Corke, P. (2013) 'Establishing initial trust in autonomous Delay Tolerant Networks without centralised PKI', *Computers & Security*, 39(PART B), pp. 299–314. doi: 10.1016/j.cose.2013.08.007.

Doukas, C. and Antonelli, F. (2013) 'COMPOSE: Building smart & context-aware mobile applications utilizing IoT technologies', in *Global Information Infrastructure Symposium - GIIS 2013*. IEEE, pp. 1–6. doi: 10.1109/GIIS.2013.6684373.

Eclipse Titan (2021). Available at: https://projects.eclipse.org/projects/tools.titan (Accessed: 16 March 2021).

ETSI (2010) *Technical Specification, ETSI TS 102 689 v1.1.1 2010-08, Machine-to-Machine communications (M2M); M2M service requirements*. Available at: https://www.etsi.org/deliver/etsi_ts/102600_102699/102689/01.01.01_60/ts_102689v01

0101p.pdf (Accessed: 16 March 2021).

ETSI (2013a) *Technical Report, ETSI TR 102 725 v1.1.1 2013-06, Machine-to-Machine communications (M2M); Definitions*. Available at: http://www.etsi.org/deliver/etsi_tr/102700_102799/102725/01.01.01_60/tr_102725v010101p.pdf (Accessed: 16 March 2021).

ETSI (2013b) *Technical Specification ETSI TS 102 690 v2.1.1 2013-10, Machine-to-Machine Communications: Functional Architecture*. Available at: https://www.etsi.org/deliver/etsi_ts/102600_102699/102690/02.01.01_60/ts_102690v020101p.pdf (Accessed: 16 March 2021).

ETSI (2015) *Technical Report ETSI TR 101 583 v1.1.1 2015-03, Methods for Testing and Specification (MTS); Security Testing; Basic Terminology*. Available at: https://www.etsi.org/deliver/etsi_tr/101500_101599/101583/01.01.01_60/tr_101583v010101p.pdf (Accessed: 16 March 2021).

Felderer, M., Zech, P., Breu, R., Büchler, M. and Pretschner, A. (2016) 'Model-based security testing: a taxonomy and systematic classification', *Software Testing, Verification and Reliability*, 26(2), pp. 119–148. doi: 10.1002/stvr.1580.

Fridelin Panduman, Y. Y., Sukaridhoto, S. and Tjahjono, A. (2019) 'A Survey of IoT Platform Comparison for Building Cyber-Physical System Architecture', in *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*. IEEE, pp. 238–243. doi: 10.1109/ISRITI48646.2019.9034650.

Gao, Z., Zhao, W., Xia, C., Xiao, K., Mo, Z., Wang, Q. and Yang, Y. (2019) 'A Credible and Lightweight Multidimensional Trust Evaluation Mechanism for Service-Oriented IoT Edge Computing Environment', in *2019 IEEE International Congress on Internet of Things (ICIOT)*. IEEE, pp. 156–164. doi: 10.1109/ICIOT.2019.00035.

Gattermayer, J. and Tvrdik, P. (2017) 'Blockchain-Based Multi-Level Scoring System for P2P Clusters', in *2017 46th International Conference on Parallel Processing Workshops (ICPPW)*. IEEE, pp. 301–308. doi: 10.1109/ICPPW.2017.50.

German Testing Board (2014) 'ISTQB/GTB Standardglossar der Testbegriffe Version 2.3', (April). Available at: http://www.german-testing-board.info/fileadmin/gtb_repository/downloads/pdf/glossar/CT_Glossar_DE_EN_V23.pdf.

Goka, S. and Shigeno, H. (2018) 'Distributed management system for trust and reward in mobile ad hoc networks', in *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, pp. 1–6. doi: 10.1109/CCNC.2018.8319278.

Guth, J., Breitenbücher, U., Falkenthal, M., Fremantle, P., Kopp, O., Leymann, F. and Reinfurt, L. (2018) 'A Detailed Analysis of IoT Platform Architectures: Concepts, Similarities, and Differences', in Di Martino, B., Li, K.-C., Yang, L. T., and Esposito, A. (eds). Singapore: Springer Singapore, pp. 81–101. doi: 10.1007/978-981-10-5861-5_4.

Hassan, Q. (2018) *Internet of Things A to Z*. Edited by Q. Hassan. Hoboken, NJ, USA: John Wiley & Sons, Inc. doi: 10.1002/9781119456735.

Hejazi, H., Rajab, H., Cinkler, T. and Lengyel, L. (2018) 'Survey of platforms for massive IoT', in *2018 IEEE International Conference on Future IoT Technologies (Future IoT)*.

IEEE, pp. 1–8. doi: 10.1109/FIOT.2018.8325598.

Highcharts (2021) *Highcharts - Interactive javascript charts library*. Available at: https://www.highcharts.com/ (Accessed: 11 October 2021).

Holler, J., Tsiatsis, V., Mulligan, C., Avesand, S., Karnouskos, S. and Boyle, D. (2014) *From Machine-To-Machine to the Internet of Things*, *From Machine-To-Machine to the Internet of Things*. Elsevier. doi: 10.1016/C2012-0-03263-2.

ICASH (2018) *Whitepaper, A Smart Contract origination and settlement platform leveraging the Proof of Trust protocol*. Available at: https://cdn2.hubspot.net/hubfs/4276960/ICASH whitepaper.pdf (Accessed: 12 April 2019).

IEEE (1990) 'IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology', *IEEE Std 610.12-1990*, pp. 1–84. doi: 10.1109/IEEESTD.1990.101064.

ITU-T (2012) *Recommendation ITU-T Y.2060 06/2012, Next Generation Networks - Frameworks and functional architecture models, Overview of the Internet of things*. Available at: https://www.itu.int/rec/T-REC-Y.2060-201206-I (Accessed: 16 March 2021).

ITU-T (2014) *Recommendation ITU-T Y.2066 06/14, Common requirements of the Internet of things*. Available at: https://www.itu.int/rec/T-REC-Y.2066-201406-I/en (Accessed: 16 March 2021).

ITU-T (2015) *Recommendation ITU-T X.1163, Security requirements and mechanisms of peer-to-peer-based telecommunication networks*. Available at: https://www.itu.int/rec/T-REC-X.1163-201505-I (Accessed: 16 March 2021).

ITU-T (2016) *ITU-T Technical Report, Trust Provisioning for future ICT infrastructures and services*. Available at: https://www.itu.int/en/publications/Pages/publications.aspx?lang=en&media=electronic&parent=T-TUT-TRUST-2016-1 (Accessed: 16 March 2021).

ITU-T (2017a) *ITU-T FG-DFS 03/2017, ITU-T Focus Group Digital Financial Services, Distributed Ledger Technologies and Financial Inclusion*. Available at: https://itu.int/en/ITU-T/focusgroups/dfs/Documents/201703/ITU_FGDFS_Report-on-DLT-and-Financial-Inclusion.pdf (Accessed: 16 March 2021).

ITU-T (2017b) *Recommendation ITU-T Y.3052 03/2017, Overview of trust provisioning in information and communication technology infrastructures and services*. Available at: https://www.itu.int/rec/T-REC-Y.3052/en (Accessed: 16 March 2021).

ITU-T (2019a) *ITU-T, ITU Focus Group on Application of Distributed Ledger Technology (FG DLT), Technical Report FG DLT D2.1, Distributed ledger technology use cases*. Available at: https://www.itu.int/en/ITU-T/focusgroups/dlt/Documents/d21.pdf (Accessed: 16 March 2021).

ITU-T (2019b) *ITU-T Technical Report, ITU-T Focus Group on Application of Distributed Ledger Technology (FG DLT), Technical Report FG DLT D1.2, Distributed ledger technology overview, concepts, ecosystem*. Available at: https://www.itu.int/en/ITU-T/focusgroups/dlt/Documents/d12.pdf (Accessed: 16 March 2021).

ITU-T (2019c) *ITU-T Technical Report, ITU-T Focus Group on Data Processing and Management to support IoT and Smart Cities & Communities, Technical Report D3.5, Overview of blockchain for supporting IoT and SmartCity&Communities (SC&C) in Data Processing Management (DPM*. Available at: https://www.itu.int/dms_pub/itu-t/opb/fg/T-FG-DPM-2019-3.5-PDF-E.pdf (Accessed: 16 March 2021).

ITU-T (2019d) *ITU-T Technical Specification, ITU-T Focus Group on Application of Distributed Ledger Technology (FG DLT), Technical Specification FG DLT D1.1, Distributed ledger technology terms and definitions*. Available at: https://www.itu.int/en/ITU-T/focusgroups/dlt/Documents/d11.pdf (Accessed: 16 March 2021).

JAIN SIP (2021) *JSR 32: JAIN SIP API Specification*. Available at: https://jcp.org/en/jsr/detail?id=32 (Accessed: 16 March 2021).

Jayasinghe, U., Truong, N. B., Lee, G. M. and Um, T.-W. (2016) 'RpR: A Trust Computation Model for Social Internet of Things', in *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*. IEEE, pp. 930–937. doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0146.

jQuery (2021). Available at: https://jquery.com/ (Accessed: 11 October 2021).

Kandah, F., Huber, B., Skjellum, A. and Altarawneh, A. (2019) 'A Blockchain-based Trust Management Approach for Connected Autonomous Vehicles in Smart Cities', in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, pp. 0544–0549. doi: 10.1109/CCWC.2019.8666505.

Kim, J., Lee, J., Kim, J. and Yun, J. (2014) 'M2M service platforms: Survey, issues, and enabling technologies', *IEEE Communications Surveys and Tutorials*, 16(1), pp. 61–76. doi: 10.1109/SURV.2013.100713.00203.

Kim, Y. J., Kim, E. K., Nam, B. W. and Chong, I. (2012) 'Service composition using new DSON platform architecture for M2M service', in *The International Conference on Information Network 2012*. IEEE, pp. 114–119. doi: 10.1109/ICOIN.2012.6164360.

King, S. and Nadal, S. (2012) *PPCoin: Peer-to-Peer Kryptowährung mit Proof-of-Stake*. Available at: https://www.peercoin.net/whitepapers/peercoin-paper-de.pdf (Accessed: 16 March 2021).

Kitagami, S., Miyanishi, Y., Urano, Y. and Shiratori, N. (2014) 'Proposal of a Distributed Cooperative M2M System for Flood Disaster Prevention', in *2014 IEEE 11th Intl Conf on Ubiquitous Intelligence and Computing and 2014 IEEE 11th Intl Conf on Autonomic and Trusted Computing and 2014 IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops*. IEEE, pp. 637–642. doi: 10.1109/UIC-ATC-ScalCom.2014.51.

Kouicem, D. E., Bouabdallah, A. and Lakhlef, H. (2018) 'An Efficient Architecture for Trust Management in IoE Based Systems of Systems', in *2018 13th Annual Conference on System of Systems Engineering (SoSE)*. IEEE, pp. 138–143. doi: 10.1109/SYSOSE.2018.8428732.

Kuroiwa, T., Aoyama, Y. and Kushiro, N. (2019) 'A Hybrid Testing Environment between Execution Test and Model Checking for IoT System', in *2019 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, pp. 1–2. doi: 10.1109/ICCE.2019.8661998.

Lahbib, A., Toumi, K., Laouiti, A., Laube, A. and Martin, S. (2019) 'Blockchain based trust management mechanism for IoT', in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, pp. 1–8. doi: 10.1109/WCNC.2019.8885994.

Le, V. T. (2018) *Development and realisation of tests for decentralised M2M applications*. Master Thesis, Frankfurt University of Applied Sciences.

LeMahieu, C. (2018) *Nano: A Feeless Distributed Cryptocurrency Network*, *White paper*. Available at: https://content.nano.org/whitepaper/Nano_Whitepaper_en.pdf (Accessed: 16 March 2021).

Leotta, M., Clerissi, D., Olianas, D., Ricca, F., Ancona, D., Delzanno, G., Franceschini, L. and Ribaudo, M. (2018) 'An acceptance testing approach for Internet of Things systems', *IET Software*, 12(5), pp. 430–436. doi: 10.1049/iet-sen.2017.0344.

Li, N., Varadharajan, V. and Nepal, S. (2019) 'Context-Aware Trust Management System for IoT Applications with Multiple Domains', in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, pp. 1138–1148. doi: 10.1109/ICDCS.2019.00116.

Lin, K.-J., Reijers, N., Wang, Y.-C., Shih, C.-S. and Hsu, J. Y. (2013) 'Building Smart M2M Applications Using the WuKong Profile Framework', in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*. IEEE, pp. 1175–1180. doi: 10.1109/GreenCom-iThings-CPSCom.2013.204.

Lopez, G., Moura, P., Moreno, J. I. and de Almeida, A. (2011) 'ENERsip: M2M-based platform to enable energy efficiency within energy-positive neighbourhoods', in *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, pp. 217–222. doi: 10.1109/INFCOMW.2011.5928812.

Ma, Y. and Wang, D. (2016) 'A novel trust model for P2P networks', in *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*. IEEE, pp. 1969–1973. doi: 10.1109/FSKD.2016.7603482.

Malik, B. H., Khalid, M., Maryam, M., Nauman, M., Yousaf, S., Mehmood, M. and Saleem, H. (2019a) 'IoT Testing-as-a-Service: A New Dimension of Automation', *International Journal of Advanced Computer Science and Applications*, 10(5), pp. 364–371. doi: 10.14569/IJACSA.2019.0100545.

Malik, S., Dedeoglu, V., Kanhere, S. S. and Jurdak, R. (2019b) 'TrustChain: Trust Management in Blockchain and IoT Supported Supply Chains', in *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, pp. 184–193. doi: 10.1109/Blockchain.2019.00032.

Mendoza, C. V. L. and Kleinschmidt, J. H. (2015) 'Mitigating On-Off Attacks in the Internet of Things Using a Distributed Trust Management Scheme', *International Journal of Distributed Sensor Networks*, 11(11), p. 859731. doi: 10.1155/2015/859731.

Mendoza, C. V. L. and Kleinschmidt, J. H. (2016) 'Defense for selective attacks in the IoT with a distributed trust management scheme', in *2016 IEEE International Symposium on Consumer Electronics (ISCE)*. IEEE, pp. 53–54. doi: 10.1109/ISCE.2016.7797367.

METI (2016) *Survey on Blockchain Technologies and Related Services FY2015 Report. Nomura Research Institute*. Available at: https://www.meti.go.jp/english/press/2016/pdf/0531_01f.pdf (Accessed: 16 March 2021).

Mocnej, J., Seah, W. K. G., Pekar, A. and Zolotova, I. (2018) 'Decentralised IoT Architecture for Efficient Resources Utilisation', *IFAC-PapersOnLine*, 51(6), pp. 168–173. doi: 10.1016/j.ifacol.2018.07.148.

Nakahira, S., Nakamura, S., Enokido, T. and Takizawa, M. (2015) 'Trustworthiness in Peer-to-Peer Systems', in *2015 18th International Conference on Network-Based Information Systems*. IEEE, pp. 652–657. doi: 10.1109/NBiS.2015.97.

Nakamoto, S. (2008) *Bitcoin: A Peer-to-Peer Electronic Cash System*. Available at: https://bitcoin.org/bitcoin.pdf (Accessed: 16 March 2021).

Natoli, C. and Gramoli, V. (2016) 'The Blockchain Anomaly', in *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*. IEEE, pp. 310–317. doi: 10.1109/NCA.2016.7778635.

Nguyen, D. H. (2019) *Evaluation and Implementation of Consensus Mechanisms used in Distributed Ledger Approaches*. Master Thesis, Frankfurt University of Applied Sciences.

Nguyen, H. T., Yang, J. and Zhao, W. (2012) 'Bootstrapping Trust and Reputation for Web Services', in *2012 IEEE 14th International Conference on Commerce and Enterprise Computing*. IEEE, pp. 41–48. doi: 10.1109/CEC.2012.16.

Nguyen, T., Hoang, D., Nguyen, D. and Seneviratne, A. (2017) 'Initial trust establishment for personal space IoT systems', in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, pp. 784–789. doi: 10.1109/INFCOMW.2017.8116476.

oneM2M (2016a) *oneM2M Technical Specification TS-0001-V1.13.1, Functional Architecture*. Available at: https://www.onem2m.org/images/files/deliverables/TS-0001-Functional_Architecture-V1_13_1.pdf (Accessed: 16 March 2021).

oneM2M (2016b) *oneM2M Technical Specification TS-0001-V2.10.0, Functional Architechture*. Available at: https://www.onem2m.org/images/files/deliverables/Release2/TS-0001-Functional_Architecture-V2_10_0.pdf (Accessed: 16 March 2021).

oneM2M (2016c) *oneM2M Technical Specification TS-0002-V2.7.1, Requirements*. Available at: https://www.onem2m.org/images/files/deliverables/release2/ts-0002-requirements-v2_7_1.pdf (Accessed: 16 March 2021).

oneM2M (2016d) *oneM2M Technical Specification TS-0007-V2.0.0, Service Components*. Available at: https://www.onem2m.org/images/files/deliverables/Release2/TS-0007-Service_Components-V2_0_0.pdf (Accessed: 16 March 2021).

oneM2M (2019) *oneM2M Technical Specification TS-0011-V3.0.1, Common Terminology*. Available at: https://www.onem2m.org/images/files/deliverables/Release3/TS-0011-Common_Terminology-V3_0_1.pdf (Accessed: 16 March 2021).

Oracle (2021) *Java Platform, Standard Edition*. Available at: https://www.oracle.com/java/technologies/javase-downloads.html (Accessed: 16 March 2021).

Padilla, J. E. V., Lee, J. O. and Kim, J. H. (2013) 'A M2M horizontal services platform implementation over IP multimedia subsystem (IMS)', in *15th Asia-Pacific Network Operations and Management Symposium: 'Integrated Management of Network Virtualization', APNOMS 2013*.

Pal, S., Hitchens, M. and Varadharajan, V. (2019) 'Towards the Design of a Trust Management Framework for the Internet of Things', in *2019 13th International Conference on Sensing Technology (ICST)*. IEEE, pp. 1–7. doi: 10.1109/ICST46873.2019.9047734.

Pezzè, M. and Young, M. (2009) *Software testen und analysieren*. Oldenbourg, Munich: De Gruyter.

Di Pietro, R., Salleras, X., Signorini, M. and Waisbard, E. (2018) 'A blockchain-based Trust System for the Internet of Things', in *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies*. New York, NY, USA: ACM, pp. 77–83. doi: 10.1145/3205977.3205993.

Pontes, P. M., Lima, B. and Faria, J. P. (2018) 'Izinto: A pattern-based IoT testing framework', in *Companion Proceedings for the ISSTA/ECOOP 2018 Workshops*, pp. 125–131. doi: 10.1145/3236454.3236511.

Popov, S. (2018) *The Tangle, IOTA Whitepaper v1.4.3*. Available at: https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf (Accessed: 16 March 2021).

Reetz, E. S. (2016) *Service Testing for the Internet of Things*. PhD Thesis, University of Surrey.

Reetz, E. S., Kümper, D., Tönjes, R. and Lehmann, A. (2012) 'Test driven life cycle management for internet of things based services: A semantic approach', in *VALID 2012 - 4th International Conference on Advances in System Testing and Validation Lifecycle*, pp. 21–27.

Rosenkranz, P., Wählisch, M., Baccelli, E. and Ortmann, L. (2015) 'A Distributed Test System Architecture for Open-source IoT Software', in *Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems*. New York, NY, USA: ACM, pp. 43–48. doi: 10.1145/2753476.2753481.

Sagar, S., Mahmood, A., Sheng, Q. Z. and Zhang, W. E. (2020) 'Trust Computational Heuristic for Social Internet of Things: A Machine Learning-based Approach', in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. IEEE, pp. 1–6. doi: 10.1109/ICC40277.2020.9148767.

Saied, Y. Ben, Olivereau, A., Zeghlache, D. and Laurent, M. (2013) 'Trust management

system design for the Internet of Things: A context-aware and multi-service approach', *Computers & Security*, 39(PART B), pp. 351–365. doi: 10.1016/j.cose.2013.09.001.

Schwartz, D., Youngs, N. and Britto, A. (2014) *The Ripple protocol consensus algorithm*, *Ripple Labs Inc White Paper*. Available at: https://ripple.com/files/ripple_consensus_whitepaper.pdf (Accessed: 16 March 2021).

Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2019a) 'Blockchain-Based Trust Communities for Decentralized M2M Application Services', in *Lecture Notes on Data Engineering and Communications Technologies*, pp. 62–73. doi: 10.1007/978-3-030-02607-3_6.

Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2019b) 'Distributed Ledger Technology for Trust Management Optimisation in M2M', in *Mobile Communication - Technologies and Applications; 24. ITG-Symposium*, pp. 1–6.

Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2019c) 'Novel trust consensus protocol and blockchain-based trust evaluation system for M2M application services', *Internet of Things*, 7, p. 100058. doi: 10.1016/j.iot.2019.100058.

Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2020a) 'Blockchain and Trust for Secure, End-User-Based and Decentralized IoT Service Provision', *IEEE Access*, 8, pp. 119961–119979. doi: 10.1109/ACCESS.2020.3005541.

Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2020b) 'Synergy of Trust, Blockchain and Smart Contracts for Optimization of Decentralized IoT Service Platforms', in *Advances in Intelligent Systems and Computing*, pp. 547–558. doi: 10.1007/978-3-030-44041-1_49.

Shala, B., Trick, U., Lehmann, A., Shala, B., Ghita, B. and Shiaeles, S. (2018) 'Trust-Based Composition of M2M Application Services', in *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, pp. 250–255. doi: 10.1109/ICUFN.2018.8436992.

Shala, B., Wacht, P., Trick, U., Lehmann, A., Shala, B., Ghita, B. and Shiaeles, S. (2017) 'Framework for automated functional testing of P2P-based M2M applications', in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, pp. 916–921. doi: 10.1109/ICUFN.2017.7993932.

Shih, C.-S., Lin, K.-J., Chou, J.-J. and Chuang, C.-C. (2014) 'Autonomous Service Management for Location and Context Aware Service', in *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*. IEEE, pp. 246–251. doi: 10.1109/SOCA.2014.10.

Snider, M., Samani, K. and Jain, T. (2018) *Delegated Proof of Stake: Features and Tradeoffs*, *Multicoin Capital*. Available at: https://multicoin.capital/2018/03/02/delegated-proof-stake-features-tradeoffs/ (Accessed: 16 March 2021).

Steinheimer, M. (2018) *Autonomous decentralised M2M Application Service Provision*. PhD Thesis, University of Plymouth. Available at: https://pearl.plymouth.ac.uk (Accessed: 16 March 2021).

Steinheimer, M., Trick, U., Fuhrmann, W., Ghita, B. and Frick, G. (2017a) 'M2M

Application Service Provision: An Autonomous and Decentralised Approach', *Journal of Communications*, pp. 489–499. doi: 10.12720/jcm.12.9.489-499.

Steinheimer, M., Trick, U., Fuhrmann, W., Steinheimer, M. and Ghita, B. (2013) 'P2p-based community concept for M2M applications', in *Second International Conference on Future Generation Communication Technologies (FGCT 2013)*. IEEE, pp. 114–119. doi: 10.1109/FGCT.2013.6767198.

Steinheimer, M., Trick, U. and Ghita, B. (2019) 'Formal Application Description of Autonomous and Cooperative M2M Application Services', *Journal of Communications*, 14, pp. 33–39. doi: 10.12720/jcm.14.1.33-39.

Steinheimer, M., Trick, U., Steinheimer, M., Ghita, B. and Fuhrmann, W. (2017b) 'Autonomous decentralised M2M application service provision', in *2017 Internet Technologies and Applications (ITA)*. IEEE, pp. 18–23. doi: 10.1109/ITECHA.2017.8101904.

Strobel, V. and Dorigo, M. (2018) 'Blockchain technology for robot swarms: A shared knowledge and reputation management system for collective estimation', *IRIDIA – Technical Report Series, Technical Report No. TR/IRIDIA/2018-009*. Available at: https://iridia.ulb.ac.be/IridiaTrSeries/link/IridiaTr2018-009.pdf (Accessed: 21 April 2021).

Talbi, S. and Bouabdallah, A. (2020) 'Interest-based trust management scheme for social internet of things', *Journal of Ambient Intelligence and Humanized Computing*, 11(3), pp. 1129–1140. doi: 10.1007/s12652-019-01256-8.

Thymeleaf (2021). Available at: https://www.thymeleaf.org/ (Accessed: 11 October 2021).

Tibermacine, O., Tibermacine, C. and Cherif, F. (2015) 'Regression-Based Bootstrapping of Web Service Reputation Measurement', in *2015 IEEE International Conference on Web Services*. IEEE, pp. 377–384. doi: 10.1109/ICWS.2015.57.

Truong, N. B., Um, T.-W. and Lee, G. M. (2016) 'A Reputation and Knowledge Based Trust Service Platform for Trustworthy Social Internet of Things', in *19th International ICIN Conference - Innovations in Clouds, Internet and Networks*, pp. 104–111.

Trustpilot (2021) *TrustScore and star rating explained – Trustpilot Support Center*. Available at: https://support.trustpilot.com/hc/en-us/articles/201748946-TrustScore-and-star-rating-explained (Accessed: 18 March 2021).

W3C (2015) *Recommendation, State Chart XML (SCXML): State Machine Notation for Control Abstraction*. Available at: https://www.w3.org/TR/scxml/ (Accessed: 13 April 2021).

Wacht, P. (2017) *Framework for Automated Functional Tests within Value-Added Service Environments*. PhD Thesis, University of Plymouth. Available at: http://hdl.handle.net/10026.1/5335 (Accessed: 16 March 2021).

Wacht, P. and Trick, U. (2016) 'A novel test creation framework for value-added services', in *2016 24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE, pp. 1–6. doi: 10.1109/SOFTCOM.2016.7772108.

Wahab, O. A., Cohen, R., Bentahar, J., Otrok, H., Mourad, A. and Rjoub, G. (2020) 'An endorsement-based trust bootstrapping approach for newcomer cloud services', *Information Sciences*, 527, pp. 159–175. doi: 10.1016/j.ins.2020.03.102.

Wall, E. (2017) *IOTA is centralized*. Available at: https://medium.com/@ercwl/iota-is-centralized-6289246e7b4d (Accessed: 16 March 2021).

Wang, Y. (2020) 'A Trust Management Model for Internet of Vehicles', in *Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy*. New York, NY, USA: ACM, pp. 136–140. doi: 10.1145/3377644.3377664.

Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C., Deleuze, S. and Simons, M. (2017) *Spring Boot Reference Guide 1.5.8.RELEASE*.

Xiaocong, Q. and Jidong, Z. (2010) 'Study on the structure of “Internet of Things(IOT)” business operation support platform', in *2010 IEEE 12th International Conference on Communication Technology*. IEEE, pp. 1068–1071. doi: 10.1109/ICCT.2010.5688537.

Yang, Z., Yang, K., Lei, L., Zheng, K. and Leung, V. C. M. (2019) 'Blockchain-Based Decentralized Trust Management in Vehicular Networks', *IEEE Internet of Things Journal*, 6(2), pp. 1495–1505. doi: 10.1109/JIOT.2018.2836144.

Yu, Y., Xia, C., Li, S. and Li, Z. (2015) 'Trust type based trust bootstrapping model of computer network collaborative defense', *China Communications*, 12(12), pp. 133–146. doi: 10.1109/CC.2015.7385521.

Zou, J., Ye, B., Qu, L., Wang, Y., Orgun, M. A. and Li, L. (2019) 'A Proof-of-Trust Consensus Protocol for Enhancing Accountability in Crowdsourcing Services', *IEEE Transactions on Services Computing*, 12(3), pp. 429–445. doi: 10.1109/TSC.2018.2823705.

Appendix A – Abbreviations

A

AI	Artificial Intelligence
API	Application Programming Interface
AS	Remote Alarm Service
ATM	Automated Teller Machine

B

BMS	Remote Building Monitoring Service

C

CORE	Common Open Research Emulator

D

DAG	Directed Acyclic Graph
DHT	Distributed Hash Table
DLT	Distributed Ledger Technology
DPoS	Delegated Proof of Stake
DTN	Delay Tolerant Network

E

e-DSON	Enhanced Dynamic Service Overlay Network
ETSI	European Telecommunications Standards Institute

F

FANET	Flying Ad-Hoc Network

G

GUI	Graphical User Interface

H

HiL	Hardware in the Loop

I

IFD	Interface Description
IoE	Internet of Everything
IoT	Internet of Things
IoV	Internet of Vehicles
ITU	International Telecommunication Union

L

LCF	Leverage of Common Friends

M

M2M	Machine-to-Machine Communication
MBT	Model-based Testing
MBTAAS	Model-based Testing as a Service
MiL	Model in the Loop

N

NEPI	Network Experiment Programming Interface

P

P2P	Peer-to-Peer
PIM	Platform Independent Model
PKI	Public Key Infrastructure
PoA	Proof of Authority
PoS	Proof of Stake
POS	Point of Sales
PoW	Proof of Work

Q

QoS	Quality of Service

S

SAR	Service/Application Registry
SC	Service Consumer
SCE	Service Creation Environment
SCXML	State Chart extensible Markup Language
SDP	Service Delivery Platform
SiL	Software in the Loop
SIoT	Social Internet of Things
SIP	Session Initiation Protocol
SOA	Service Oriented Architecture
SS1	Remote M2M Service
SUT	System under Test

T

TERMS	Terms of Use
TEU	Test Execution Unit
TGU	Test Generation Unit
TiL	Thing in the Loop
Trust-CP	Trust Consensus Protocol
TTCN-3	Testing and Test Control Notation Version 3
TTE	Test Engine
TUE	Trust Engine

U

UI	User Interface
UML	Unified Modelling Language
UNL	Unique Node List

V

VANET	Vehicular Ad-Hoc Network

W

WSDL	Web Services Description Language

X

xIL	x in the Loop
XML	Extensible Markup Language

Appendix B – Own Publications

The following list includes publications related to the area of this research, to which the author of this thesis has contributed during the course of research.

- Shala, B., Wacht, P., Trick, U., Lehmann, A. (2017), "Optimised Test and Security Solution for P2P-based M2M Applications", Mobilkommunikation: Technologien und Anwendungen, ITG-Fachtagung, pp. 105-110, Osnabrück, Germany.
  VDE: https://www.vde-verlag.de/proceedings-en/454408016.html

- Shala, B., Wacht, P., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2017), "Framework for Automated Functional Testing of P2P-based M2M Applications", IEEE International Conference on Ubiquitous and Future Networks (ICUFN 2017), pp. 916-921, Milan, Italy.
  DOI: https://doi.org/10.1109/ICUFN.2017.7993932
  IEEE Xplore: https://ieeexplore.ieee.org/abstract/document/7993932

- Shala, B., Wacht, P., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2017), „Trust Integration for Security Optimisation in P2P-based M2M Applications", IEEE Trustcom/BigDataSE/ICESS, pp. 949-954, Sydney, NSW, Australia.
  DOI: https://doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.335
  IEEE Xplore: https://ieeexplore.ieee.org/abstract/document/8029538

- Shala, B., Wacht, P., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2017), "Automated Functional Testing of P2P-based M2M Applications", IEEE Internet Technologies and Applications (ITA 17), pp. 29-34, Wrexham, UK.
  DOI: https://doi.org/10.1109/ITECHA.2017.8101906
  IEEE Xplore: https://ieeexplore.ieee.org/abstract/document/8101906

- Shala, B., Wacht, P., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2017), "Ensuring Trustworthiness for P2P-based M2M Applications", IEEE Internet Technologies and Applications (ITA 17), pp. 58-63, Wrexham, UK.
  DOI: https://doi.org/10.1109/ITECHA.2017.8101911
  IEEE Xplore: https://ieeexplore.ieee.org/abstract/document/8101911

- Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2018), "Trust-based Composition of M2M Application Services", IEEE International Conference on Ubiquitous and Future Networks (ICUFN 2018), pp. 250-255, Prague, Czech Republic.
  DOI: https://doi.org/10.1109/ICUFN.2018.8436992
  IEEE Xplore: https://ieeexplore.ieee.org/abstract/document/8436992

- Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2019), "Blockchain-based Trust Communities for Decentralized M2M Application Services", Advances on P2P, Parallel, Grid, Cloud and Internet Computing: Proceedings of the 13th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2018), Taichung, Taiwan. ed. / F. Xhafa; F.-Y. Leu; M. Ficco; C.-T. Yang. Springer, 2019. p. 62-73 (Lecture Notes on Data Engineering and Communications Technologies; Vol. 24).
  DOI: https://doi.org/10.1007/978-3-030-02607-3_6
  Springer: https://link.springer.com/chapter/10.1007/978-3-030-02607-3_6

- Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2019), "Distributed Ledger Technology for Trust Management Optimisation in M2M", Mobile Communication - Technologies and Applications, ITG-Symposium, pp. 1-6, Osnabrueck, Germany.
  IEEE Xplore: https://ieeexplore.ieee.org/abstract/document/8731781

- Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2019), "Novel Trust Consensus Protocol and Blockchain-based Trust Evaluation System for M2M Application Services", Internet of Things: Engineering Cyber Physical Human Systems, Elsevier Journal, Vol. 7, 100058.
  DOI: https://doi.org/10.1016/j.iot.2019.100058
  ScienceDirect:
  https://www.sciencedirect.com/science/article/pii/S2542660519301234

- Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2020), "Trusted, Decentralised and Blockchain-based M2M Application Service Provision". Proceedings of the 14th International Conference on Broad-Band Wireless Computing, Communication and Applications (BWCCA 2019), Antwerp, Belgium. ed. / Leonard Barolli; Peter Hellinckx; Tomoya Enokido. Springer, 2020. p. 210-221 (Lecture Notes in Networks and Systems; Vol. 97).
  DOI: https://doi.org/10.1007/978-3-030-33506-9_19
  Springer: https://link.springer.com/chapter/10.1007/978-3-030-33506-9_19

- Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2020), "Synergy of Trust, Blockchain and Smart Contracts for Optimisation of Decentralized IoT Service Platforms". Proceedings of the 34th International Conference on Advanced Information Networking and Applications, AINA 2020, Caserta, Italy. ed. / Leonard Barolli; Flora Amato; Francesco Moscato; Tomoya Enokido; Makoto Takizawa. Springer, 2020. p. 547-558 (Advances in Intelligent Systems and Computing; Vol. 1151).
  DOI: https://doi.org/10.1007/978-3-030-44041-1_49
  Springer: https://link.springer.com/chapter/10.1007/978-3-030-44041-1_49

- Shala, B., Trick, U., Lehmann, A., Ghita, B. and Shiaeles, S. (2020), "Blockchain and Trust for Secure, End-User-Based and Decentralised IoT Service Provision", in IEEE Access, vol. 8, pp. 119961-119979.
  DOI: https://doi.org/10.1109/ACCESS.2020.3005541
  IEEE Xplore: https://ieeexplore.ieee.org/abstract/document/9127445