2024-02

An In-Depth Analysis on Efficiency and Vulnerabilities on a Cloud-Based Searchable Symmetric Encryption Solution

Chin, J-J

10.33093/jiwe.2024.3.1.19
Journal of Informatics and Web Engineering
Multimedia University

Journal of Informatics and Web Engineering
Vol. 3 No. 1 (February 2024) eISSN: 2821-370X

An In-Depth Analysis on Efficiency and Vulnerabilities on a Cloud-Based Searchable Symmetric Encryption Solution

Prithvi Chaudhari[1], Ji-Jian Chin[1*], Soeheila Moesfa bt Mohamad[2]

[1]School of Computing, Engineering and Mathematics, University of Plymouth, United Kingdom
[2]Faculty of Computing and Informatics, Multimedia University, Malaysia
**corresponding author: (ji-jian.chin@plymouth.ac.uk; ORCiD: 0000-0001-9809-6976)*

Abstract - Searchable Symmetric Encryption (SSE) has come to be as an integral cryptographic approach in a world where digital privacy is essential. The capacity to search through encrypted data whilst maintaining its integrity meets the most important demand for security and confidentiality in a society that is increasingly dependent on cloud-based services and data storage. SSE offers efficient processing of queries over encrypted datasets, allowing entities to comply with data privacy rules while preserving database usability. Our research goes into this need, concentrating on the development and thorough testing of an SSE system based on Curtmola's ~~Curtmola et al. [12]~~ architecture and employing Advanced Encryption Standard (AES) in Cypher Block Chaining (CBC) mode. A primary goal of the research is to conduct a thorough evaluation of the security and performance of the system. In order to assess search performance, a variety of database settings were extensively tested, and the system's security was tested by simulating intricate threat scenarios such as count attacks and leakage abuse. The efficiency of operation and cryptographic robustness of the SSE system are critically examined by these reviews.

Keywords— Secure *Cloud Storage, Searchable Symmetric Encryption, Searchable Encryption, Leakage Abuse Attack, Count Attack*

I. INTRODUCTION

SSE (Searchable Symmetric Encryption) is a form of encryption that allows one party to safely grant another party access to data storage while still allowing searches on the data [1]. SSE lets users search for keywords inside documents even when they are saved in encrypted text format. SSE's goal is to stop any well-defined leaks while also stopping an untrusted server from picking up on any missing details regarding the search terms or documents. Information leakage, when referring to a search on encrypted data, is the accidental disclosure of private information or metadata. A basic structure of how SSE works is shown in Figure 1. Query and access patterns, result sizes, timing features, and other factors might all be contributing factors to this leakage. The main objective of safeguarding data privacy can be compromised by information leakage in SSE systems, which is a serious problem

Commented [d1]: Abstract cannot have citation. Please remove

Commented [JC2R1]: done

Commented [d3]: Should start with [1],[2],[3] in the sequence. Please change for the rest.

Commented [JC4R3]: done

Journal of Informatics and Web Engineering
https://doi.org/10.33093/jiwe.2024.3.1.19

URL: https://journals.mmupress.com/jiwe

when search is provided in a cloud environment [2]. A leakage profile, which may include several leaking patterns, is used to specifically express the leakage of an SSE system [3].

The foundation for creating a Searchable Symmetric Encryption (SSE) was laid by Curtmola et. al. in 2006 [1]. Practical and secure ways for executing search queries over encrypted data that only reveals the search results were proposed by the authors. Strict security specifications and organized SSE techniques provided by this research minimized the information loss during searches. Further SSE investigations that examined the differences between search efficiency and privacy were backed by the groundwork done by the authors and are still relevant for ongoing SSE research [4].

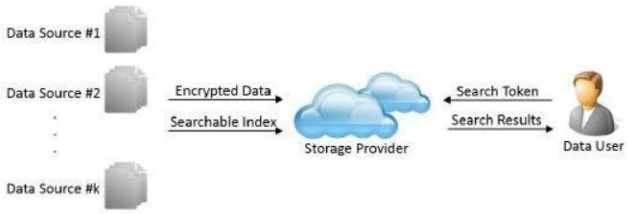

Figure 1: General SSE architecture where encrypted data is stored in the cloud and returns data when user queries after searching on encrypted data.

As part of this research, we have conducted an in-depth analysis of academic literature from the year 2014 to 2023. The materials, which focused on the fields of searchable encryption, searchable symmetric encryption (SSE), and secure cloud storage, were carefully chosen from trusted academic journals and conferences. The paper was drawn up to first describe the approach, namely how the Curtmola SSE scheme was implemented for practical research. In the next section, we have tested the search efficiency of the system and also tried to test the security of our system by subjecting it to Leakage Abuse Attacks. The results and their wider implications are then discussed and summarized in the conclusion. Finally, the article offers suggestions for future lines of research in this area.

Lastly it is interesting to note that unlike asymmetric key alternatives which can be broken by Shor's algorithm, SSE only has to contend with Grover algorithm searches, but that is still quite a long way from being realized with only small implementation examples as illustrated by [5].

II. LITERATURE REVIEW

A. Review of SSE

Searchable Symmetric Encryption (SSE) has swiftly gained essential importance in the data security environment, particularly in this period of increasing cloud computing use. This innovative field has advanced through a series of significant milestones, demonstrating an ever-growing response to the increasingly complicated restrictions of data privacy and security in the digital era. SSE may be dated back to early efforts to balance the requirements of data availability and integrity [6]. While data is safely encrypted in a standard encryption framework, looking through such data needs either full decryption or a less secure technique, both of which offer significant security issues. SSE solves this problem by allowing encrypted data to be searched without first decrypting it, preserving confidentiality while assuring quick data retrieval. SSE has grown in popularity as cloud-based services have become more common, given its ability to provide safe, efficient search capabilities across encrypted data stored in cloud settings. This is especially important in light of the massive amounts of sensitive data handled and stored remotely in today's digital world. SSE's guarantees of security are crucial in limiting the risks connected with unauthorized data access and breaches, which have grown more regular and serious in the digital world [7]. In this literature review, we will

be studying multiple papers from 2014 to 2023 and try to explain the advancements made, trends and limitations of these papers. Also, topics like need of SSE, and its history and background will be discussed further.

B. Advantages and Need of SSE

The core principle of SSE is striking a balance between protecting data privacy and supporting effective access and use. Traditional encryption techniques safeguard data adequately, but they also render it inaccessible without decryption, making it useless for any real use. SSE overcomes this restriction by maintaining the functionality of encrypted data, especially with regard to search capability [8]. Functional encryption has made it possible for users to query encrypted databases without revealing their contents, which is a significant shift. An unusual amount of data, most of it private and sensitive, has been collected since the arrival of the digital age. Alongside this data collection, there has been a growing demand for efficient and safe data storage options, especially cloud-based services. These demands led to the development of SSE, which offers a method for storing data in an encrypted format while preserving searchability. Businesses, organizations, and people who depend on speedy data retrieval without sacrificing security must have this capacity [4].

Although cloud computing has completely changed the way that data is accessed and stored, it has also brought up new security issues. SSE agrees ideologically that cloud computing security has to be improved. It lets customers take use of cloud storage features like scalability and accessibility while making sure their data is safe from online dangers, including those that come from cloud service providers [30]. The philosophy behind SSE is dynamic and changes to reflect the way technology and data consumption are developing. Safe and easily available data is becoming more and more important as we head towards a future filled with big data, AI, and machine learning. SSE stands for a flexible, forward-thinking approach to data security that acknowledges the evolving nature of data usage and the accompanying security needs [7].

C. History and Background

The evolution of Searchable Symmetric Encryption (SSE) is founded in the history of encryption and the changing demands for data security, particularly in cloud computing contexts. Encryption techniques have a lengthy history, beginning with ancient systems such as the Caesar cipher, which required changing the alphabet to encrypt information. As the demand for secure communication expanded, these early approaches developed into increasingly complicated systems [9]. Modern encryption techniques, such as symmetric and asymmetric encryption algorithms, were developed as a result of this change. Asymmetric encryption utilizes two keys (public and private), whereas symmetric encryption uses one key for encryption and decryption [10]. The capacity to effectively look through data is restricted when it is encrypted, even while it guarantees secrecy when stored on cloud servers. There is a major problem with searching encrypted data without first decrypting it since the server cannot access the plaintext to search for keywords. SSE was created as a result, enabling secret data to be searched over encrypted channels [11].

The data owner, data consumer, cloud service provider, and key generator are the four entities that usually make up the architecture of SSE systems. Using cryptographic algorithms that allow for searching, the data owner encrypts the data and metadata. To find particular encrypted data, the data user who may also be the data owner, sends encrypted queries to the cloud service provider. The cloud service provider manages and stores data, running encrypted queries on the stored data without gaining any knowledge about it. It is composed of a data server and a service manager. The last party in charge of creating and maintaining encryption and decryption keys is the key generator, a reliable management [9]. Large datasets may require more time to process since the non-keyword-based method searches through whole documents to identify terms of interest. The index/keyword-based technique, on the other hand, provides speedier search operations by creating an index for each term and listing the associated documents comprising it. Public key cryptography and private key cryptography are two types of cryptographic models that may be used in SSE methods. Multi-user data encryption using a public key is possible with asymmetric SSE, but token generation and search operations are restricted to the owner holding the matching private key [7].

D. Evolution of SSE over the years

An innovative approach was presented by Song et. al. in 2000 [12] which addressed the rising concern in the digital era to protect data confidentiality while enabling effective search capabilities. Previously there was a widespread opinion that encryption made data unsearchable until it was decrypted which presented a serious issue in situations when both data secrecy and searchability were essential. According to Song et al.'s [12] technique every word in a document was encrypted individually and searches were conducted by encrypting the search term and comparing it

with the encrypted words. This method was groundbreaking for a number of reasons. First of all, it made it possible to safely store private information in settings like cloud services, where the security of the storage provider is uncertain. Second, it introduced the idea of manipulating encrypted data without first having to decode it, which put established cryptographic concepts to the test. The approach of Song et al. [12] was not very practical as there were two primary challenges: computational burden of encrypting each word individually and the inefficiency of searching through big databases. Furthermore, the technique was mostly restricted to straightforward keyword searches, which limited its use for more intricate queries. Another huge advancement in SSE world was made by Curtmola et al. in 2006 [1]. His fundamental contribution was the development of two effective SSE systems based on searchable encryption and inverted index structures. By building an index of keywords and the documents that contain them, this method significantly increased search efficiency, especially for huge datasets. Stronger security specifications for SSE, including indistinguishability and non-adaptivity, were also proposed by Curtmola et al. [1] in their work. These improved security features made sure that the encrypted data did not disclose plaintext information and that prior searches did not threaten the security of subsequent queries.

In 2014, the concept of DSSE (Distributed Searchable Symmetric Encryption) was introduced by Bosch et al [13]. A significant limitation of earlier SSE methods was that disclosing the search pattern resulted in security flaws. This is why the DSSE strategy was introduced by Bosch et al [13]. to conceal the search pattern. They used a scrambling function for security in private information retrieval (PIR) together with an inverted index strategy for efficiency. The whole technique was built on pseudorandom functions and XOR operations. They developed the idea of a query proxy, which got a new (shuffled and re-encrypted) index for each query. However, this concept exhibits promise in controlled experimental and theoretical settings, but the research doesn't go into great detail about how difficult it would be to implement the system in reality. Subsequently, Cash et al.'s [14] main contribution in 2015 was identifying and demonstrating how information leakage in SSE may be exploited. The study established the theory of 'leakage-abuse' attacks, in which an adversary might utilize leaked information to discover more about the encrypted material than was intended. These attacks took use of access pattern leakage and search pattern leaking. New types of attacks like the count attacks and IKK (Islam, Kuzu, and Kantarcioglu) [15] were tried and tested on the existing schemes [1], [12] to study about their security and reliability. It became evident that simply protecting the privacy of the search terms and the data was insufficient, it was also necessary to address and reduce the leakage patterns themselves. It forced researchers and developers to reevaluate the security concepts used in various SSE implementations and to include extra protections against leakage-abuse attacks.

A major gap in cloud data management was filled in 2015 by Salam [7] et al. by making it possible to do quick and safe keyword searches on encrypted data. The method they used to enable keyword searches on encrypted cloud-stored data was the highlight of it. In order to generate an index of keywords and their corresponding places in the encrypted files, the authors used an inverted index structure. Scalability was one of the main issues. There was no research done on how well the suggested SSE technique performs in expansive, complicated cloud systems, particularly ones that handle huge amounts of data. The static nature of the index update procedure prevents new file additions and updates to already-existing files. Furthermore, there was no discussion of how encryption affects other cloud services, such as data sharing and synchronization, which is essential to knowing the SSE scheme's wider application. In the next year, Chen et al. (2016) [16] investigated the relationship between secure network coding and secure cloud storage. In order to strengthen cloud storage security, this research presents an inventive combination of network coding with encryption approaches. The process of combining and transforming data packets as they go over a network is known as network coding, and it offers built-in security against data loss and illegal access. The system may save storage space and accelerate data retrieval procedures by merging and coding data packets. Although the paper by Chen et al. [16] has inventive methodology, the paper presents a few limitations. It can be difficult and resource-intensive to manage and maintain the coding techniques, particularly in a highly scattered cloud system. But it can also result in performance overheads, especially when it comes to the processing time required to code and decode data packets. There are still concerns about the proposed network coding system's scalability in large-scale cloud systems, where millions of data packets may be sent and stored.

Multi-functional searchable symmetric encryption systems were introduced by Hu et al., in 2016 [10] which was their main contribution. In contrast to traditional SSE systems, which were restricted to keyword search, their methodology enabled a range of search features, including Boolean, wildcard, and range searches. Users may search for data inside a specified range of values using range queries, or they could use wildcard queries for a more flexible searches with incomplete data. Searches based on logical phrase combinations were made possible by Boolean queries. The dynamic operations, such as adding and removing data files, might be supported by the suggested SSE system. By using arrays in place of matrices and Bloom filters, the approach was also effective in terms of

computational and relative complexity. A significant barrier of Hu et al.'s [10] proposition was the level of complexity associated in executing multi-functional SSE methods. Improved functionality frequently carries the risk of worse performance. When working with huge datasets, the sophisticated search capabilities offered by Hu et al. [10] may cause greater processing cost or longer search times. In the same year, 2016, Babitha et al. [6] concentrated on using Advanced Encryption Standard (AES) encryption to improve cloud storage security. Just before the data is sent to the cloud, it is encrypted at the client end using AES. In addition to securing the data during transmission, this encryption procedure guarantees that the data in the cloud storage stays secured at all times. This encryption procedure guarantees that the data is encrypted both during transmission and while it is stored in the cloud. The integrity and confidentiality of the encryption and decryption keys have a major impact on AES security. In order to improve user awareness and data security. This paper also includes an SMS warning system that notifies users of attempted illegal access. According to Babitha et al., [6] one of the main drawbacks of using 128-bit AES encryption is the possible speed penalty. In certain situations, depending solely on SMS alerts for security notifications could not be totally dependable. Scalability issues arise when implementing AES encryption in large-scale cloud settings.

Poh et al.'s [17] work gives a comprehensive overview and evaluation of Searchable Symmetric Encryption (SSE). It recognizes and expands on four major SSE design approaches: construction without index, direct index, inverted index, and tree structures. The study investigates several security models and the leakages related with various SSE systems. The research also looks into dynamic and verifiable SSE methods, which are important for practical implementations. It compares the performance of several SSE structures in terms of search efficiency and storage needs. While the article gives an overview of the status of SSE in 2017, further advances and breakthroughs in the area may have generated new designs, issues, and solutions not mentioned in the study. As a result, it may provide only a limited understanding of the practical implementation issues and real-world uses of SSE. This covers concerns about scalability, interaction with current systems, and user experience. Kim et al.'s [11] study effectively implements forward security in DSSE, preventing information about newly uploaded files matching earlier search queries from being leaked. Dynamic SSE enables changes (updates and deletions) to the encrypted dataset without re-encrypting the entire dataset. Forward security, on the other hand, ensures that any previously obtained search tokens cannot be used to obtain information on newly uploaded documents. The approach devised by Kim et al. [11] was intended to reduce information leakage during search and update activities. For both search and update activities, the plan achieves the ideal level of computational and communication complexity. Additionally, it provides a unique data structure that makes efficient search and update operations possible by combining forward and inverted indexes. The plan is realistic and simple to apply, as evidenced by trials conducted on real-world datasets such as Wikipedia and Enron emails.

Song et al.'s made a major contribution in 2018 [18] with merging forward privacy with optimized I/O efficiency in SSE. As previously explained, forward privacy ensures that search tokens from the past do not undermine the privacy of freshly uploaded documents. FASTIO, in particular, boosts I/O efficiency, which is essential for scaling SSE algorithms to huge data sets. Both systems use symmetric key cryptography exclusively, eliminating the performance limitations associated with public key operations. The techniques are intended to prevent attacks that take use of SSE leaks, such as file-injection attacks. These approaches may be more scalable and efficient for even bigger datasets if they are used in more diverse and extensive real-world contexts. The techniques demonstrate great I/O and computational efficiency, particularly FASTIO. The encrypted index's states were managed and search tokens were generated using pseudorandom functions (PRFs). By prioritizing forward privacy, the risk of leakage-abuse attacks which take advantage of leaks during search and update operation is greatly decreased. Further, in 2018, Chen et al. [19] concentrated on tackling access pattern leakage in order to strengthen the security of Searchable Symmetric Encryption (SSE). They created a technique to hide SSE access patterns, which lessened the possibility of query recovery attacks. In order to give verifiable security assurances against attackers with any background information, d-privacy: a generic kind of differential privacy was utilized. Through their framework, they addressed the issues with search results by handling False Positives and Negatives and implementing a prototype of the suggested obfuscation system. Assumptions made on the attacker's capabilities determine how successful the suggested solution will be. The success of the differential privacy implementation may be risked if these assumptions are not true in practical situations. The usefulness of the SSE system may be limited by excessive obfuscation, which can produce less significant or helpful search results.

With their article "SAP-SSE: Protecting Search Patterns and Access Patterns in Searchable Symmetric Encryption," Song et al. in 2020 [8] introduced a new method in the field of SSE. An attacker may be able find out whether the exact same search request has been issued more than once thanks to a vulnerability known as search pattern leakage. Leakage of access patterns concerns the potential for an attacker to determine which data items are being retrieved

or accessed in response to a request. Even if the dataset is encrypted, this can still provide important insights about the dataset. It has innovative redistribution and index shuffle methods that help regularly reorganize cloud-side indexes for increased security. With the scheme's adjustable security policy, customers can modify the scheme to their own needs by balancing security and efficiency. SAP-SSE offers great confidentiality, avoids query imitation, and achieves sublinear search complexity. Experimental evaluation of the technique shows that it can successfully avoid keyword recovery attempts with minimal cost. After that, the CLOSE framework, which consists of two schemes: CLOSE-F (forward secure DSSE scheme) and CLOSE-FB (forward and backward secure DSSE scheme) was introduced in a study by He et al. in 2020 [20]. The increasing expense of client-side storage is a major drawback for many SSE systems, particularly in dynamic circumstances. The work of He et al. [20] suggested a way to maintain this cost independent of the quantity of data or the number of updates. The fish-bone chain, a unique two-level structure made up of the Document Index Chain (DIC) and Logical Keyword Index Chain (LoKIC), was first shown which can manage and arrange the encrypted data. In terms of updating and searching, CLOSE-F and CLOSE-FB both show efficient computing costs that are either on level with or better than those of other innovative methods. Good efficiency is indicated by the fact that the search time in CLOSE-F and CLOSE-FB rises slowly with the number of matched items. The forward and backward security included into the schemes greatly lessens their susceptibility to leakage abuse attacks.

In 2021, Li et al. [21] created Khons, an innovative SSE technique that meets forward search privacy as well as forward update privacy. Additionally, they showed that Khons outperforms other forward-privilege SSE techniques, especially when used to big datasets like Wikipedia. The plan is intended to survive modern attacks, such as adaptive file-injection attacks. This may be more useful in industries like healthcare and banking, where maintaining confidence and sensitive data are crucial. They tested and put the Khons plan into practice, demonstrating its usefulness in practical situations. Computational complexity may rise in SSE systems when forward search privacy is implemented. Slower search performance might result from this complexity, which is important for large-scale systems or applications that need fast data retrieval. Like with many security solutions, improving privacy through forward search privacy may increase system complexity and negatively affect user experience with the SSE system.

One of the major advancements given that most of the earlier methods concentrated on a single-client context happened in 2021 by Gan et al. [22] when their forward private SSE approach presented allowed multiple clients in a cloud computing context. This approach makes use of XOR-homomorphic functions and adds two new data structures: the public search tree and the private link. The system uses solely symmetric cryptographic primitives to maximize the efficiency of search and updating. Forward privacy protects search query confidentiality from unauthorized access in the future. Stated otherwise, if an encryption key is ever compromised, the searches carried out prior to the system continue to be safe. The special concentration of multi-client SSE is on situations in which several clients access and query a single SSE database. Furthermore In 2021, the authors Asharov et al. [23] presented two frameworks for SSE schemes: the statistical-independence framework and the pad-and-split framework. By hiding the same current SSE techniques, this framework improves upon Cash and Tessaro's non-overlapping reads framework. The authors demonstrated that, for databases of size N, every pad-and-split strategy with locality L has to consume space $O(NlogN/logL)$, offering a tight lower bound. They created a scheme that, when included in this framework, matches their lower limit for constant locality within an additive $O(log\ N)$ factor in terms of read efficiency. This approach provides near-optimal read efficiency together with optimal space and locality. Better read efficiency is achieved for less often occurring keywords via their statistical-independence approach, which is dependent on the frequency of the keywords searched. A solid theoretical basis was presented in the research, however there may be more difficulties when putting the. A solid theoretical basis was presented in the research, however there may be more difficulties when putting these trade-offs into practice in real-world systems. Research on this topic is still ongoing. The study focused on how difficult it is to get the right balance between functionality, security, and efficiency in SSE systems.

Subsequently, Wang et al.'s 2022 [24] study focuses on improving dynamic searchable symmetric encryption's (DSSE) security, paying particular attention to range queries. The research presents innovative approaches that accomplish both forward and backward security, which are critical for safeguarding the privacy of erased data and preventing file injection attacks, respectively. Forward security makes ensuring that previous searches and data are safe and unavailable to the attacker, even in the event that future keys or encryption states are stolen. On the other hand, backward security ensures that the security of subsequent searches and data is not compromised in the event that previous keys or states are compromised. With dynamic datasets, where data may be added or removed over time, a Binary-Tree-Like index can manage them well without materially affecting the security or effectiveness of the SSE system. Although binary trees improve search efficiency, there may be performance overheads due to the

extra security requirements (such as encrypting nodes and hiding access patterns). Thus, it may impact both speed and security, the scalability of such an index in extremely big datasets or highly dynamic contexts.

In 2022, Chen et al. [4] introduced the CASE-SSE system, which is context-sensitive. This indicates that the search queries might be understood and interpreted by the system aligning with the particular scenario or setting in which they are used. Since the model was intended to be conceptually expandable, it should be able to manage and comprehend the connections and meanings of the terms found in the search queries. In encrypted datasets, this made it possible to produce more insightful and relevant search results. The strategy, which concentrated on encrypted cloud data, sought to provide sophisticated search features without compromising data security and privacy which is an essential need in cloud computing settings. Using the Word2vec architecture, CASE-SSE extends query keywords analytically depending on the context of the content. This overcomes the drawbacks of conventional SSE systems, which frequently produce insufficient search results because they lack semantic knowledge. A distributed word representation of the dataset is produced by using the Word2vec Model. This enhances the relevancy and precision of the search results by precisely calculating the semantic distance between terms. The Two-Layer Index Structure is a proposal that combines inverted index constructions with AVL-tree and K-means clustering into a two-layer index structure. By enhancing query processing and decreasing the dimensionality of index nodes, this structure improves search efficiency. Khosro Salmani's work in 2022 [25] presents a new approach in the field of Dynamic Searchable Symmetric Encryption (DSSE). This research makes a significant contribution by integrating result verifiability into the DSSE methodology. Users may check the accuracy and completeness of search results thanks to this. The research suggests a brand-new, lightweight index structure that may be implemented on devices with limited resources. The study provides an experimental demonstration and a theoretical framework. However, it falls short of addressing the difficulties and complexities of implementing the system in a variety of real-world settings. Similar to other DSSE techniques, Salmani's [25] method nonetheless permits certain information leakage (such search patterns) for efficiency which could be improved in the future.

In 2023, Chen et al. [26] presented a study introducing a DSSE technique that provides backward and forward privacy support. While backward privacy safeguards the privacy of deleted data, forward privacy guarantees that newly uploaded files stay hidden from earlier search searches. The suggested scheme's support for data deduplication is one of its main characteristics. This reduces redundant data inputs, which helps prevent the waste of cloud storage resources. Effective conjunctive searches are made possible by the technique, opening up more sophisticated and focused search options inside the encrypted material. To improve the performance of search and update operations, the system manages keyword/file-ID pairings using a bitmap index structure. The study claims that its unique encryption design and indexing method have enhanced search efficiency. Although the technique handles both forward and backward privacy, it does not thoroughly investigate its possible susceptibility to leakage abuse attacks, especially when it comes to search pattern leaking.

We summarise the papers mentioned about the developments in SSE over the years above in Table 1, providing a chronological overview of the various works with mention of their innovations and algorithms used.

Table 1: Summary of all the papers mentioned in the literature review.

Year of Publication	**Name of the Paper**	**Name of the Author**	**Innovations**	**Algorithms Used**
2000	Practical Techniques for Searches on Encrypted data	Song et al. [12]	Encrypted individual words for data confidentiality while enabling search	Encryption of individual words and comparison with encrypted search terms
2006	Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions	Curtmola et al. [1]	Created two efficient SSE systems using inverted index structures and searchable encryption	Inverted index structures and searchable encryption
2012	Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation	Islam et al. [15]	The original attack paper on SSE.	An empirical analysis with a real-world dataset shows that the proposed attack is able to disclose sensitive information with a very high accuracy

2014	Distributed Searchable Symmetric Encryption	Bosch et al. [13]	DSSE was implemented to hide search patterns	Inverted index technique, pseudorandom functions, and scrambling function for security
2015	Leakage-Abuse Attacks Against Searchable Encryption	Cash et al. [14]	Categorical types of attacks such as count attack and IKK	An examination of the disclosure of search and access patterns
2015	Implementation of Searchable Symmetric Encryption for Privacy-Preserving keyword search on Cloud Storage	Salam et al. [7]	Quick and safe keyword searches	Used an inverted index structure to generate an index of keywords and their corresponding places in the encrypted files
2016	Secure Cloud Storage Meets with Secure Network Coding	Chen et al. [16]	Invented combination of network coding with encryption approaches	Transformed data packets while sending it over the network
2016	Efficient and secure multi-functional searchable symmetric encryption schemes	Hu et al. [10]	Multi-functional SSE systems allowing a range of search features	Used Arrays instead of matrices and Bloom filters
2016	Secure Cloud Storage using AES Encryption	Babitha et al. [6]	AES Encryption to make data encrypted more secure and SMS warning system	Used Advanced Encryption Standard (AES) encryption to improve cloud storage security
2016	A Client-Server Prototype of a Symmetric Key Searchable Encryption Scheme using Open Source Applications	Tan et al. [27]	Utilises open-source libraries for a mobile app SSE application.	A cost effective implementation of Curtmola's scheme using open source tools and libraries.
2017	Searchable Symmetric Encryption: Designs and Challenges	Poh et al. [17]	Comprehensive overview and evaluation of SSE	Analysis of SSE design approaches and security models
2017	Forward Secure Dynamic Searchable Symmetric Encryption with Efficient Updates	Kim et al. [11]	Implementation of forward security in DSSE	Forward and inverted indexes for dynamic SSE
2018	Forward Private Searchable Symmetric Encryption with Optimized I/O Efficiency	Song et al. [18]	Merging forward privacy with optimized I/O efficiency in SSE	Symmetric key cryptography, pseudorandom functions (PRFs)
2018	Differentially Private Access Patterns for Searchable Symmetric Encryption	Chen at al. [19]	D-privacy: a generic kind of differential privacy	By handling False Positives and Negatives
2020	SAP-SSE: Protecting Search Patterns and Access Patterns in Searchable Symmetric Encryption	Song et al. [8]	SAP-SSE: Protecting Search and Access Patterns in SSE	Redistribution and index shuffle methods
2020	Secure Dynamic Searchable Symmetric Encryption with Constant Client Storage Cost	He et al. [20]	CLOSE framework for reducing client-side storage costs in SSE	Document Index Chain (DIC) and Logical Keyword Index Chain (LoKIC)

2021	Towards Multi-Client Forward Private Searchable Symmetric Encryption in Cloud Computing	Gan et al. [22]	Forward private SSE approach for multi-client cloud computing	XOR-homomorphic functions, public search tree, private link
2021	Tight Tradeoffs in Searchable Symmetric Encryption	Asharov et al. [23]	Frameworks for SSE schemes focusing on statistical-independence and pad-and-split	Statistical-independence and pad-and-split frameworks
2021	Searchable Symmetric Encryption with Forward Search Privacy	Li et al. [21]	Creation of Khons, an SSE technique with forward search and update privacy	Techniques for forward-privilege SSE
2022	Forward and Backward-Secure Range-Searchable Symmetric Encryption	Wang et al. [24]	Improving DSSE security with attention to range queries	Binary-Tree-Like Index, forward and backward security
2022	CASE-SSE: Context-Aware Semantically Extensible Searchable Symmetric Encryption for Encrypted Cloud Data	Chen et al. [4]	CASE-SSE system, context-sensitive SSE	Word2vec architecture, Two-Layer Index Structure (inverted index, AVL-tree, K-means clustering)
2022	An Efficient, Verifiable, and Dynamic Searchable Symmetric Encryption with Forward Privacy	Salmani et al. [25]	Integrating result verifiability in DSSE	Lightweight index structure
2023	Toward Forward and Backward Private Dynamic Searchable Symmetric Encryption Supporting Data Deduplication and Conjunctive Queries	Chen et al. [26]	DSSE technique with backward and forward privacy support	Bitmap index structure

III. RESEARCH METHODOLOGY

In alignment with the project scope, this dissertation method was designed to cover the creation, assessment, and analysis of a Searchable Symmetric Encryption (SSE) system that is built upon the Curtmola et al. [1] framework. Specifically, the integration of AES in Cypher Block Chaining (CBC) mode whose working is seen in Figure 2 was the main focus of this method. In our system, parallel processing was neither practical nor economical, hence the sequential CBC mode was advantageous. Furthermore, AES-GCM may be susceptible if the same initialization vector (IV) is used twice, which might compromise the key owing to the usage of (Galois Message Authentication Code) GMAC. In contrast, CBC mode does not have this particular vulnerability. Also, CBC mode was chosen because of its simplicity and compatibility with our existing system. Building an SSE system from the ground and putting it through a series of actual practical tests to test its search effectiveness in various database contexts was a key component of our methodology. These assessments were thoroughly set up to assess performance metrics in databases with different sizes and configurations, such as accuracy and response time. Additionally, we conducted a thorough security research on showing the system's vulnerability to various attack channels and leakage abuse attacks. Also, the project focuses on the analysis of the widely recognized Curtmola et al. [1] model rather than on the development of new cryptographic algorithms or changes to SSE frameworks. This methodology section explains the actual steps that were followed for implementing it and then the things that were done to carry out these experiments are explained.

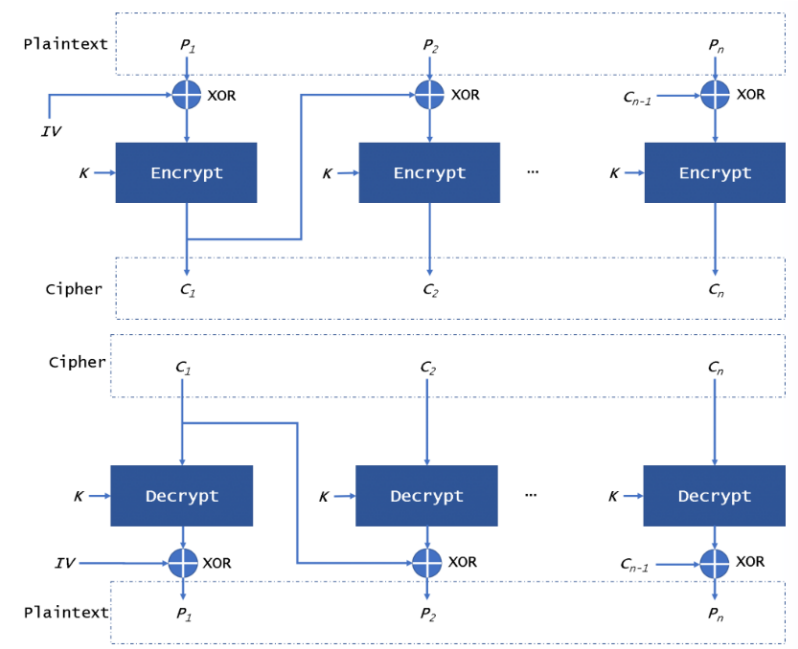

Figure 2: Working of AES in CBC mode

A. Building the SSE step by step.

The detailed visual process of these steps can be seen in Figure 3:

STEP I - Encryption Key Generation:

Generating a safe encryption key is the first stage. Encryption of the system's documents and index will be done with this key. In order to ensure that the key is long enough to offer strong security, it is produced at random using the 'secret.token_bytes' Python function, which creates a random byte string every time.

STEP II - Encrypting Documents:

Using the created key, every document to be uploaded into the SSE system is encrypted. To protect the documents' secrecy and integrity, this procedure uses a symmetric encryption algorithm which in our instance is AES in Cypher Block Chaining (CBC) mode. After that, the encrypted papers are kept in a safe database.

STEP III - Creating the encrypted index generation:

Creating an encrypted index is the next essential step, which enables effective searching via the encrypted documents. This entails processing every document to extract keywords and map those keywords to the associated document IDs.

Index Encryption: The created index is then encrypted with the same encryption key. It is important to guarantee that the index structure facilitates effective search functions. Each index entry leads to a collection of documents that include the keyword, according to the inverted index structure approach.

STEP IV - Utilizing XAMPP to host the system:

Configuring XAMPP: A local server environment suitable for hosting the SSE system is offered by XAMPP. To establish a local hosting environment, we have installed XAMPP and set up its components, such as the web server Apache and the database MySQL.

Implementing the SSE Framework: In order to ensure that the encrypted documents and index are correctly incorporated into the system, we have implemented the SSE system on XAMPP. This entails specifying file storage directories or database schemas.

STEP V - Web Interface Development:

To enable user interaction with the SSE system, we have created a web interface. Users may enter their search queries and examine the results using this interface's functions.

Search Functionality Integration: The online interface incorporates search functionality. The system returns matched encrypted documents when a user enters a search query and searches the encrypted index.

STEP VI - Showing Time Metrics and Search Results:

Results Displayed: The online interface displays the results once the system has retrieved the encrypted documents that correspond with the search query.

Measuring Search Time: A feature to check how long a search takes to complete has been introduced. This is essential for assessing the SSE system's search efficiency. On the web interface, the time metrics are shown next to the search results.

Key details of SSE implementation:

Cryptographic Specifics:

Encryption: AES in CBC mode with Initialization Vector (IV) and a 256-bit key, following the 256-bit AES block size.

Padding: PKCS7 padding makes ensuring that data fits inside the block size of AES.

For data integrity checks, use SHA-256 hashing.

Libraries:

Flask: Helps with web server configuration and API endpoints for the SSE service.

PyCryptodome: Provides cryptographic primitives, such as AES encryption and SHA-256 hashing.

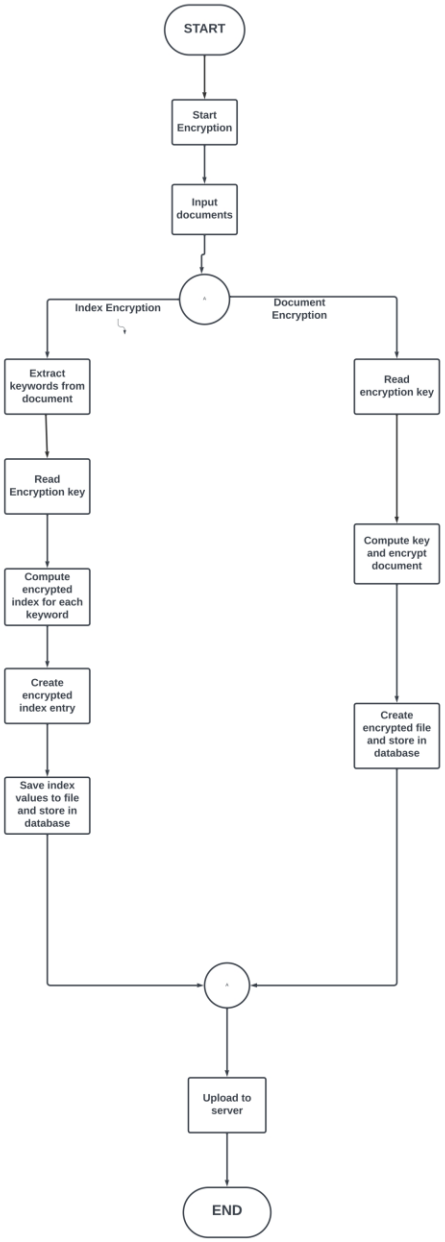

Figure 3: Flowchart explaining the index and document encryption process.

B. Testing Search efficiency of the system.

Evaluating the search function's performance is essential to determining its scalability and usefulness. The goal is to measure how quickly the system responds and how well it uses its resources [28]. The same large Enron mail database is used as it has a lot of entries and unique searchable keywords, which makes the document's index size larger. In order to simulate real-world search behavior, a wide variety of search queries including both common and uncommon terms were created. The purpose of these queries was to assess the system's capacity to manage standard user inquiries. The SSE search function was created in such a manner that it returned records related with the keyword together with the duration of the query search across the database.

While testing, databases of 100 to 100,000 entries were used to assess our system's performance and security. This selection allowed for an in-depth look across a wide range of data volumes, with an average dataset size falling within this range, serving as an acceptable basis for system analysis. This revealed crucial insights into the SSE system's scalability and efficiency, as well as its level of security revealing future improvement tactics for data processing at large scales. Also, the initial duration from the start of the search query and the retrieval of the results was noted. The system's capacity was measured by keeping track of how many requests it could execute in a given amount of time. Our findings provide the relationship between index density and search response times by correlating variations in search efficiency with index size. We expected that more items would mean more extensive index traversals, which would lead to longer search times. Through performance analysis of the system against a big database, we were able to obtain important insights on the system's suitability for practical use.

C. Testing of the system against Leakage Abuse Attacks.

In this research, we have used count attack to test the security of our Searchable Symmetric Encryption (SSE). Count attack uses the number of items returned by a query's crucial metadata leaking to its advantage in order to obtain information about the encrypted dataset. This count attack simulation's goals are to identify any potential weaknesses in the SSE system that might allow the count of documents for each search query to be compromised and to assess the amount of data that could be deduced from those breaches. Utilizing the stability of AES in CBC mode encryption, the SSE system was set up in line with the Curtmola et al. [1] approach. The count attack relies on where the attacker has access permissions to the SSE's search operation and seeks to extract extra data beyond what is meant to be used. We have selected a set of keywords from our database to search for their frequency without searching on the SSE. Using the chosen keywords, an automated script in Python was created to run restricted search queries on the SSE system. The number of items that matched the keyword was shown for each search query. The script captured not just the count but also the total amount of time spent searching for every keyword across several rounds.

IV. RESULTS AND DISCUSSIONS

A. Search Efficiency

We conducted a thorough analysis of the Searchable Symmetric Encryption (SSE) system, with particular focus on assessing its search time efficiency. Because it directly affects both the user experience and the overall health of the SSE implementation in real-world circumstances, this aspect of the system's performance is vital. In order to precisely record and analyze how the system's efficiency varied under various situations, we carefully constructed our assessment approach. Specifically, we looked at how the quantity of entries associated with each search phrase influenced the search time.

We created a number of datasets with differing amounts of entries related to particular keywords as part of our experimental setup to evaluate search time efficiency. With this method, we were able to model a wide range of actual search situations, from those that would produce a huge number of documents to those that would return a small number of results. Our SSE system was utilized to process and index the datasets. The appropriate records for every keyword were saved in a manner that followed to the rules of symmetric searchable encryption and were encrypted for security. We searched for a variety of keywords, carefully documenting the amount of time it took from the start of the query to the return of the results. With great accuracy, the response time for every query was logged, giving a clear indication of the system's effectiveness in various search settings.

These tests produced plenty of data, which were then thoroughly statistically analyzed. The detailed data points are tabled in Tables 2 and 3 for the number of entries and the number of keyword matches respectively, with corresponding line graphs generated in Figures 4 and 5. As seen in Figure 4 and Figure 5: Our research showed an interesting trend: rather than the total size of the database, the number of items linked to each keyword had a significant impact on the search time efficiency of the SSE system. When there were less items related with the term, the system performed extremely well, providing the search results rapidly. This efficiency is explained by the fact that collecting and decrypting a smaller subset of document IDs from the encrypted index requires less computing power. However, we found that the search time rose proportionally with the number of entries for a specific term. Longer response times resulted from the need for more complex data retrieval processes for keywords linked to higher volumes of entries. This pattern held true for all datasets, highlighting a crucial component of the SSE system's performance characteristics: while the system performs exceptionally well for queries that yield fewer results, the number of entries per keyword rises gradually taxing the system's capacity.

Commented [d5]: Tables 2 and 3?

Commented [JC6R5]: Corrected

Table 2: Data Points gathered and used to make the Graph Size vs Search Time.

Number of entries in a database	Search Time (in ms)
17	11
131	14
541	14
1114	17
2824	12
5315	13
7812	18
18296	12
28612	15
64923	14

Table 3: Data points gathered and used to make the graph: Number of matching entries vs Search Time.

Number of entries related to the keyword	Search Time (in ms)
9	14
112	21
523	25
1005	27
2517	32
5089	38
7348	42
12468	46
25781	47
54692	51
75201	55
108164	59

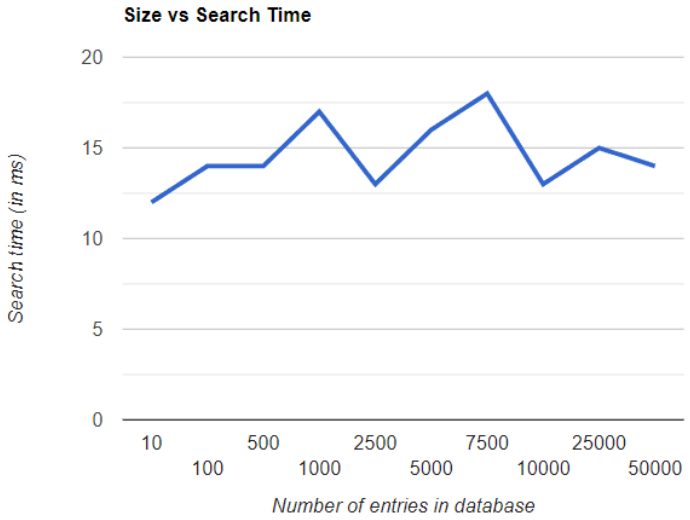

Figure 4: Graph showing how number of entries in a database affect the search time.

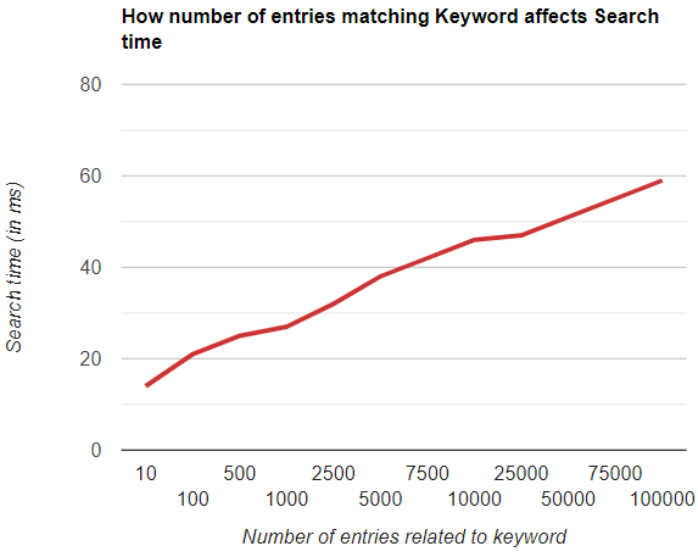

Figure 5: Graph explaining how the number of entries that match the keyword affects the search time.

We found that the number of entries for every keyword has a greater impact on search performance in our SSE model than does the size of the database as a whole. This result is important because it highlights a crucial component of SSE performance that has to be properly taken into account in real-world applications. The usability and operational success of an SSE system are critically dependent on its capacity to sustain high search efficiency, particularly in real-world circumstances where databases frequently comprise an extensive and complex range of items [20]. For example, in corporate settings, where it is often necessary to have rapid access to large databases, a decrease in search performance caused by high keyword density can result in delays and inefficiencies, which can affect decision-making procedures and overall productivity. Furthermore, the effects of this efficiency feature are especially significant in light of the increasing dependence of several industries on large-scale data analytics. Dense keyword density may be related with greater time complexity, which might provide significant issues in industries like healthcare, banking, and legal services where sensitive data is often analyzed and searched [29].

B. Leakage Abuse Attacks (LAA)

Our study's most notable finding was that the SSE's vulnerability to count attacks. This kind of leakage-abuse attack takes advantage of the unintentional disclosure of confidential metadata, and it is a major security risk in the field of encrypted search methods. In particular, it takes advantage of a feature of the SSE system that shows how many documents are related to a certain search query. Despite not being immediately obvious, this vulnerability is quite dangerous since it has the potential to indirectly reveal information about the contents of the encrypted dataset. We conducted extensive testing and analysis on this vulnerability inside our SSE implementation as a reaction to this possible threat.

A customized Python script was created and implemented for this reason. This script was created to do a number of search queries using a wide range of terms on the SSE system, much like a typical user would. Every term was carefully selected to cover a wide range of possible usage scenarios, from often occurring terms to uncommon and specialized phrases. The script conducted many searches on the SSE system during the testing runs in order to get detailed information about the quantity of items linked to each term. In order to comprehend the trends and effects of the count information compromise, the recorded data was examined further. The main goal of our study was to establish a correlation between the count data and the dataset's known keyword distribution. This stage was essential to ascertain if the count data alone could be used by an outside observer to deduce particular features of the dataset. For instance, a particularly high count for a generally uncommon term may indicate a concentrated concentration of papers on a certain subject, thereby disclosing private topic information.

Our thorough investigation and testing resulted in a certain conclusion: our SSE system was vulnerable to count attacks. This vulnerability was not only theoretical but also clearly visible in the real-world testing environment. Every query that the script ran produced a count of the number of pages that matched the searched term in addition to the encrypted results. One interesting weakness in our SSE system's defense was the ability to view the number of documents returned outside for each keyword without breaking the underlying encryption. It displayed that even with the papers' content encrypted and safe, patterns, trends, and potentially sensitive information insights could still be found in the metadata linked to search results.

When comparing the performance of our SSE system to other available encrypted search systems we observe that there are compromises to take into account. Options like Fully Homomorphic Encryption (FHE) and Oblivious RAM (ORAM) provide better security guarantees, especially against side-channel attacks like access pattern leaks. These techniques can, however, have a serious negative influence on search performance and usually have larger computing overheads. On the other hand, simpler encrypted search techniques may not give the same level of protection against advanced attacks even though they are more efficient. The necessity to balance the need for

effective search capabilities with strong security and privacy controls is a major difficulty in the development of SSE systems, as this uncertainty demonstrates [30], [31].

The analysis also reveals a serious flaw in many SSE systems, which is the capacity of external actors to detect the quantity of documents linked to particular keywords. It has come to light that our SSE system exposes exact counts of document occurrences for every query. This information can be dangerous, particularly in situations where it might be used to deduce private or sensitive information. This vulnerability has wide-ranging effects in real-world situations. Imagine working in a corporate setting where private papers are managed and searched using an SSE system. Even in the absence of direct content access, an adversary might nevertheless identify organizational trends or focal points by examining the frequency of documents associated with particular keywords. of a legal context, for instance, an unexpected increase of paperwork including phrases like "merger" or "layoff" may reveal important business choices. Even in cases when the contents of the documents are not immediately disclosed, this kind of information leaking can have significant consequences, such as insider trading, confidentiality violations, or even manipulation of stock prices or public opinion [32].

V. CONCLUSION

The rising dependency on cloud storage solutions is the driving force behind the investigation of SSE, and more specifically the Curtmola model. This study emphasises the urgent need for encryption techniques that strike a compromise between maximum security and effective data recovery. The model developed by Curtmola et al. [1] provides a workable answer to the urgent problems of data privacy and operational effectiveness in the cloud environment, rather than just being a theoretical construct. The research examines the core methodology of the Curtmola model, breaking down its processes, gauging its effectiveness, and determining how resilient it is to changing operational needs and security threats. Further investigation reveals that the system is robustly protected against several attack vectors by the implementation of AES-256 encryption in CBC mode. The ability of the system to carry out sub-linear search times and efficiently handle dynamic updates is evidence that this encryption technique maintains great operational efficiency while guaranteeing data security.

To assess the search efficiency of the Curtmola SSE system, the study employed a comprehensive testing technique. In order to replicate the complexity and unpredictability of real-world settings, this required using large datasets and a variety of search queries. The system's exceptional capacity to handle searches with less results is demonstrated by the performance results, which also highlight the system's potential for quick access to encrypted data. The system's vulnerability to count attacks, a sophisticated kind of leaking abuse attack, was a notable discovery. This discovery is important because it points to a possible way that malicious groups may obtain sensitive data via metadata analysis rather than direct content access. This study's finding highlights a crucial area where SSE systems still need to be strengthened and improved.

Leakage-Abuse Attacks (LAAs) in Searchable Symmetric Encryption (SSE) require a sophisticated approach that combines strict access restrictions with cutting-edge cryptographic algorithms [20]. By hiding genuine access patterns, strategies like dummy queries and query randomization make it more difficult for adversaries to do analytical research. By enabling safe calculations on ciphertexts, the strategic use of both deterministic and random encryption for indexing, in conjunction with the implementation of complex index structures such as encrypted Bloom filters, improves the security fabric [33]. Oblivious RAM (ORAM) and Private Information Retrieval (PIR), which conceal query-to-data links, provide further fortification. Leakage is greatly decreased by using Trusted Execution Environments (TEEs), like Intel SGX, which provide a safe execution area [30].

The operational effectiveness and security robustness of the Curtmola SSE technology are well understood in this thesis, which significantly advances the subject of SSE. It sets the basis for further advancements in this field by highlighting the crucial relationship between security and efficiency in encrypted search engines. The need for sophisticated SSE systems that can adjust to the ever-changing landscape of data storage and security will grow along with the increasing dependence on cloud storage. Researchers and industry practitioners may use the study's observations, conclusions, and suggestions as a complete reference to help them navigate the complexity and difficulties involved in creating and implementing safe, effective SSE systems.

Finally due to restricting our work mainly in the symmetric key domain, our search efficiency is theorized to perform faster than the prototype by Lai and Heng, 2022 [34] which utilizes asymmetric key cryptography, but it

would be interesting to see a side-by-side comparison. Another avenue of research is to see if the same leakage attacks that we study will work on their prototype and if not, how effective are the defenses.

ACKNOWLEDGEMENT

The research for this paper was not funded.

Github repository link: https://github.com/prithvipc/SSEImplementation_2023

REFERENCES

[1] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *Proceedings of the 13th ACM conference on Computer and communications security*, Alexandria Virginia USA: ACM, Oct. 2006, pp. 79–88. doi: 10.1145/1180405.1180417.

[2] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proceedings of the 2012 ACM conference on Computer and communications security*, Raleigh North Carolina USA: ACM, Oct. 2012, pp. 965–976. doi: 10.1145/2382196.2382298.

[3] L. Xu, H. Duan, A. Zhou, X. Yuan, and C. Wang, "Interpreting and Mitigating Leakage-Abuse Attacks in Searchable Symmetric Encryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 5310–5325, 2021, doi: 10.1109/TIFS.2021.3128823.

[4] L. Chen, Y. Xue, Y. Mu, L. Zeng, F. Rezaeibagha, and R. H. Deng, "CASE-SSE: Context-Aware Semantically Extensible Searchable Symmetric Encryption for Encrypted Cloud Data," *IEEE Trans. Serv. Comput.*, vol. 16, no. 2, pp. 1011–1022, Mar. 2023, doi: 10.1109/TSC.2022.3162266.

[5] Y.-C. Liu and M.-F. Liu, "Implementation of Grover's Algorithm & Bernstein-Vazirani Algorithm with IBM Qiskit," *J. Inform. Web Eng.*, vol. 3, no. 1, pp. 76–95, 2024.

[6] Babitha M.P. and K. R. R. Babu, "Secure cloud storage using AES encryption," in *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*, Pune, India: IEEE, Sep. 2016, pp. 859–864. doi: 10.1109/ICACDOT.2016.7877709.

[7] M. I. Salam *et al.*, "Implementation of searchable symmetric encryption for privacy-preserving keyword search on cloud storage," *Hum.-Centric Comput. Inf. Sci.*, vol. 5, no. 1, p. 19, Dec. 2015, doi: 10.1186/s13673-015-0039-9.

[8] Q. Song, Z. Liu, J. Cao, K. Sun, Q. Li, and C. Wang, "SAP-SSE: Protecting Search Patterns and Access Patterns in Searchable Symmetric Encryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 1795–1809, 2021, doi: 10.1109/TIFS.2020.3042058.

[9] K. A. Dongre, R. S. Thakur, and A. Abraham, "Secure cloud storage of data," in *2014 International Conference on Computer Communication and Informatics*, Coimbatore, India: IEEE, Jan. 2014, pp. 1–5. doi: 10.1109/ICCCI.2014.6921741.

[10] C. Hu, L. Han, and S. M. Yiu, "Efficient and secure multi-functional searchable symmetric encryption schemes," *Secur. Commun. Netw.*, vol. 9, no. 1, pp. 34–42, Jan. 2016, doi: 10.1002/sec.1376.

[11] K. S. Kim, M. Kim, D. Lee, J. H. Park, and W.-H. Kim, "Forward Secure Dynamic Searchable Symmetric Encryption with Efficient Updates," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, Dallas Texas USA: ACM, Oct. 2017, pp. 1449–1463. doi: 10.1145/3133956.3133970.

[12] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, 2000, pp. 44–55. doi: 10.1109/SECPRI.2000.848445.

[13] C. Bosch *et al.*, "Distributed Searchable Symmetric Encryption," in *2014 Twelfth Annual International Conference on Privacy, Security and Trust*, Toronto, ON, Canada: IEEE, Jul. 2014, pp. 330–337. doi: 10.1109/PST.2014.6890956.

[14] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-Abuse Attacks Against Searchable Encryption," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, Denver Colorado USA: ACM, Oct. 2015, pp. 668–679. doi: 10.1145/2810103.2813700.

[15] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation".

[16] F. Chen, T. Xiang, Y. Yang, and S. S. M. Chow, "Secure cloud storage meets with secure network coding," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, Toronto, ON, Canada: IEEE, Apr. 2014, pp. 673–681. doi: 10.1109/INFOCOM.2014.6847993.

[17] G. S. Poh, J.-J. Chin, W.-C. Yau, K.-K. R. Choo, and M. S. Mohamad, "Searchable Symmetric Encryption: Designs and Challenges," *ACM Comput. Surv.*, vol. 50, no. 3, pp. 1–37, May 2018, doi: 10.1145/3064005.

[18] X. Song, C. Dong, D. Yuan, Q. Xu, and M. Zhao, "Forward Private Searchable Symmetric Encryption with Optimized I/O Efficiency," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 5, pp. 912–927, Sep. 2020, doi: 10.1109/TDSC.2018.2822294.

[19] G. Chen, T.-H. Lai, M. K. Reiter, and Y. Zhang, "Differentially Private Access Patterns for Searchable Symmetric Encryption," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, Honolulu, HI: IEEE, Apr. 2018, pp. 810–818. doi: 10.1109/INFOCOM.2018.8486381.

[20] K. He, J. Chen, Q. Zhou, R. Du, and Y. Xiang, "Secure Dynamic Searchable Symmetric Encryption With Constant Client Storage Cost," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 1538–1549, 2021, doi: 10.1109/TIFS.2020.3033412.

[21] J. Li *et al.*, "Searchable Symmetric Encryption with Forward Search Privacy," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 1, pp. 460–474, Jan. 2021, doi: 10.1109/TDSC.2019.2894411.

[22] Q. Gan, X. Wang, D. Huang, J. Li, D. Zhou, and C. Wang, "Towards Multi-Client Forward Private Searchable Symmetric Encryption in Cloud Computing," *IEEE Trans. Serv. Comput.*, vol. 15, no. 6, pp. 3566–3576, Nov. 2022, doi: 10.1109/TSC.2021.3087155.

[23] G. Asharov, G. Segev, and I. Shahaf, "Tight Tradeoffs in Searchable Symmetric Encryption," *J. Cryptol.*, vol. 34, no. 2, p. 9, Apr. 2021, doi: 10.1007/s00145-020-09370-z.

[24] J. Wang and S. S. M. Chow, "Forward and Backward-Secure Range-Searchable Symmetric Encryption," *Proc. Priv. Enhancing Technol.*, vol. 2022, no. 1, pp. 28–48, Jan. 2022, doi: 10.2478/popets-2022-0003.

[25] K. Salmani, "An Efficient, Verifiable, and Dynamic Searchable Symmetric Encryption with Forward Privacy," in *2022 19th Annual International Conference on Privacy, Security & Trust (PST)*, Fredericton, NB, Canada: IEEE, Aug. 2022, pp. 1–10. doi: 10.1109/PST55820.2022.9851964.

[26] L. Chen, J. Li, and J. Li, "Toward Forward and Backward Private Dynamic Searchable Symmetric Encryption Supporting Data Deduplication and Conjunctive Queries," *IEEE Internet Things J.*, vol. 10, no. 19, pp. 17408–17423, Oct. 2023, doi: 10.1109/JIOT.2023.3274390.

[27] S.-Y. Tan, J.-J. Chin, G.-S. Poh, Y. H. S. Kam, and W.-C. Yau, "A Client-Server Prototype of a Symmetric Key Searchable Encryption Scheme Using Open-Source Applications," in *2015 5th International Conference on IT Convergence and Security (ICITCS)*, Kuala Lumpur, Malaysia: IEEE, Aug. 2015, pp. 1–5. doi: 10.1109/ICITCS.2015.7292892.

[28] "How Do You Write Test Cases for a Text Box?," *Testsigma Blog*. Jan. 2024. Accessed: Feb. 09, 2024. [Online]. Available: https://testsigma.com/blog/test-cases-for-text-box/

[29] B. Minaud and M. Reichle, "Hermes: I/O-Efficient Forward-Secure Searchable Symmetric Encryption," in *Advances in Cryptology – ASIACRYPT 2023*, vol. 14443, J. Guo and R. Steinfeld, Eds., Singapore: Springer Nature Singapore, 2023, pp. 263–294. doi: 10.1007/978-981-99-8736-8_9.

[30] B. Zhao, Z. Chen, and H. Lin, "Cycle ORAM: A Practical Protection for Access Pattern in Untrusted Storage," *IEEE Access*, vol. 7, pp. 26684–26695, 2019, doi: 10.1109/ACCESS.2019.2900304.

[31] E. Stefanov *et al.*, "Path ORAM: an extremely simple oblivious RAM protocol," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS '13*, Berlin, Germany: ACM Press, 2013, pp. 299–310. doi: 10.1145/2508859.2516660.

[32] C. V. Wright and D. Pouliot, "Early Detection and Analysis of Leakage Abuse Vulnerabilities." 2017. Accessed: Feb. 10, 2024. [Online]. Available: https://eprint.iacr.org/2017/1052

[33] Z. Gui, K. G. Paterson, and S. Patranabis, "Rethinking Searchable Symmetric Encryption," in *2023 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA: IEEE, May 2023, pp. 1401–1418. doi: 10.1109/SP46215.2023.10179460.

[34] J.-F. Lai and S.-H. Heng, "Secure File Storage On Cloud Using Hybrid Cryptography," *J. Inform. Web Eng.*, vol. 1, no. 2, pp. 1–18, Sep. 2022, doi: 10.33093/jiwe.2022.1.2.1.