ENHANCING THE CLOUD SERVICES AND PRIVACY WITH SEARCHABLE ENCRYPTION SEARCH PATTERN

#1METHUKU SOUMYA, M.Tech Student, Dept of CSE,
#2Dr.GULAB SINGH, Associate Professor, Dept of CSE,
#3Dr.CHANDRAMOULI NARSINGOJU, Associate Professor & HOD, Dept of CSE
VAAGESWARI COLLEGE OF ENGINEERING, KARIMNAGAR, TS.

ABSTRACT:

Dynamic symmetrical searchable encryption (SSE) has emerged in recent years, allowing data users to search for and dynamically update codes that are stored on a cloud server while maintaining the highest level of security possible. Searching and updating operations, on the other hand, would result in additional privacy leaks in numerous previously reported SSE systems, such as search pattern privacy, privacy, and reverse privacy, which have all been published previously. To our knowledge, neither the dynamic SSE system nor the static SSE system maintains the confidentiality of the search pattern. Many privately held SSE systems, on the other hand, do leak certain critical information, such as identifiers that are now associated with a certain term in the database, which might be detrimental. To address the aforementioned concerns, we present in this article a practical SSE technique that not only protects the privacy of the search pattern but also protects the privacy of the search pattern's reverse. For the time being, we will use k-anonymity and encryption in particular to construct an obscure approach. To handle single term searches while boosting confidentiality and backward privacy, the next step is to build a simple, dynamic SSE method based on blurring technique, pseudorandom feature, and pseudorandom generator that is simple and dynamic in nature. Besides that, we revise our proposed technique in order to accommodate more efficient boolean queries. Based on the results of the security analysis, our proposed scheme is capable of achieving the essential data protection characteristics, and rigorous performance studies have demonstrated that our proposed technique is efficient in terms of overhead communication and computation costs.

Index Terms—Dynamic SSE, search pattern privacy, enhanced backward privacy, boolean query.

---

1. INTRODUCTION

There is an increasing trend of hosting programmes on a third-party cloud system, such as databases, e-mail, and file systems, among other things, to save money. The keyword search function in these programmes is one of the most often used features. Despite the fact that data encryption preserves privacy, keyword searching on encrypted data becomes a practical impossibility when using encrypted data. Symmetric searchable encryption (SSE) has been developed as a fundamental encryption method that allows a client to search for encrypted data on a non-trusted server. The Dynamic SSE (DSSE) variation lets the client to make changes to the data after the server has been outsourced.

The majority of DSSE approaches build an encrypted search-efficiency invert index from outsourced data, which is then used to do searches. It is necessary to modify the accompanying index as well in order to maintain consistency in the addition and deletion of files; in the meantime, a privacy notation is achieved. It is common to utilise a pair as an inverted index (key, value), where the key is one keyword and the value is a list of all of the keyword file identifiers in the database. When a keyword is searched for on the server, the server runs a search operation on the index and returns the identification of the corresponding files to the client at the start. Following the search results, the customer can access and download the files.

Leakages and Attacks of DSSE

Practical DSSEs exchange security in order to reach a suitable level of efficiency through the leakage of specific information. A leak's access pattern, search pattern, and magnitude are all factors that influence how it is modelled in general. Identifying the access pattern enables you to see where, when, and how frequently keywords and modifications are discovered in the encrypted indexes. The search pattern indicates whether or not the consumer searches for the same keyword more than once in a given period of time. During inquiries and entity updates, the size pattern can be drawn based on a variety of factors, such as the file number in the search results, the number of indices in files that are added or deleted, and so on.

According to some authors, these basic SSE leaks can result in significant attacks on a network. Leakage abuse, file injection attacks, and count attacks, in particular, make use of access pattern leakage, pattern search leakage, and size pattern leaks to get access to sensitive information. With enough information about the dataset, the attackers can effectively extract the query and even encrypted data from the database. More later attacks were recommended, either to improve the rate of keyword retrieval or to attack specific leak patterns that had been identified. While numerous SSE strategies have been developed to prevent information leakage through privacy accomplishment, the relationship between search operations and updates can only be disrupted by breaking the interaction between search operations and updates (i.e., search-update link ability).

In the instance of a new file being added, no keyword information is leaked into the new file, however in the case of a deleted file being removed, no keyword information can be released into search searches for keywords after the file has been deleted.

While moving privacy forward or backward can reduce the effectiveness of attacks that rely on update leaks, search/access/size patterns are still susceptible to attack. Adaptive variants of the file-injection assault, for example, will only be able to withstand the adaptive variation of the private SSE system. However, the non-adaptive file injection attack that makes use of the access pattern in private SE continues to be effective in the meanwhile (see for more details). A counter-attack against huge pattern attacks was proposed by Cash et al. [7] in response to such attacks. When playing Counter Attack, the opponent should be aware of certain information about the underlying data set, for example. The goal is to determine which keyword is asked for the unique size when the number of documents matched is unique, or to determine which keyword is requested in a single size when the number of documents matched is unique. The IKK Attack is used to rebuild the query by access pattern by overlapping the search results across many inquiries. Contrary to the IKK Attack, the counterattack is more efficient and accurate; practically all Enron dataset queries may be retrieved with this method. Liu and colleagues were the first to examine search behaviour leveraging attacks. Recently, some attacks utilising range search patterns and k-NN queries have been devised and tested. These attacks try to increase the rate at which plaintext is recovered without the use of a pre-determined query distribution. Recent assaults have had an impact on the size pattern leak as well. Overall, future planning and retrospective preparations for private SSE facilities continue to confront numerous leakage risks.

Ignored Size Pattern Leakage

The purpose of certain SSE systems is to achieve a higher level of safety by preventing the leaking of search/access patterns that define the data block that is being protected (key, value). Current SSE-inspired ORAM algorithms, on the other hand, do not maintain pattern leaks, resulting in the search results being always leaked.

To further illustrate the effects of size pattern leakage, we establish two types of connections in Figure 1 that are independent of search-update connectivity: (1) search link capability to identify the searches for the same keyword that were performed by different people; and (2) update connectivity capability to demonstrate the identification of changes to the same file that were performed by different people.

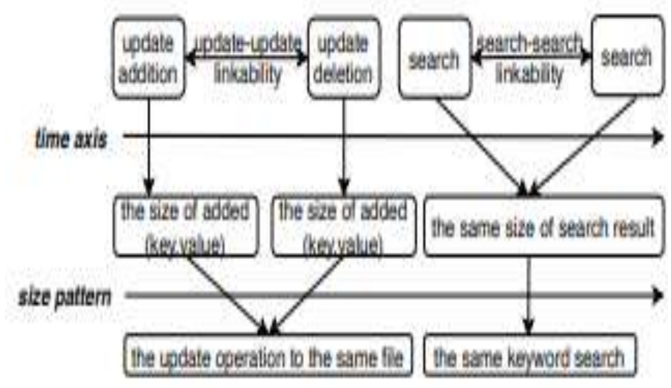
update addition
update-update linkability
update deletion
search
search-search linkability
search
time axis
the size of added (key,value)
the size of added (key,value)
the same size of search result
size pattern
the update operation to the same file
the same keyword search

Fig. 1: The ignored size pattern in general SSE scheme.

Searching for connectivity is a never-ending process. The search connectivity provided by SSE enabled the attacker to discover search queries for the same terms across a large number of large inquiries. Different methods of leaking connectivity may be used to accomplish this. For example, if search tokens are related in some way, the server may simply match requests, which is now the most common cause of search pattern leakages in most SSE systems. Another possible leak option is to access the memory pattern, which means that the server can link queries together to see whether they have an impact on the same memory during search queries, which is another possibility. However, the situation is made worse by the fact that the findings of the matching document size pattern can also disclose the existence of connectedness. Because the size pattern of a given keyword is unique, the server can easily connect queries by simply observing the size pattern of that keyword; this is not an exaggeration – it has been demonstrated that the unique size pattern can be used to significantly increase attack efficiency and recover almost all queries from the Enron dataset.

Improve the linkability of your website. Improve the linkability of your website. If the keyword locations of a file cannot be preserved, the connectivity to a previous addition operation can simply be bypassed if the file is removed after it has been created. However, hiding the update pattern alone is not sufficient because the size pattern can be leaked as a result of the high bandwidth cost associated with the update request. The failure to identify a file with which the queries are associated, for example, after adding or removing the results of several searches, is referred to as the connection error.

Motivation. Long-term viability of the leaky size pattern is estimated to be very high. The IKK attack recovery rate can be abused indefinitely as long as the Counter Attack does not display the size pattern of the IKK attack. When this was written, it was not known whether an assault that just mimics a model's size existed. The situation has changed recently as a result of a number of new attacks against leakage and misuse. Blackstone and others have proposed attacks on significantly weaker assumptions that may be utilised simply by exploiting document-size information, not only for SSE with standard leakage, but also for ORAM, and they have demonstrated their effectiveness. In a similar vein, many encrypted database attacks have been developed that only expose size patterns. All of these attacks demonstrate that a size pattern that appears to be risky may be leveraged to deliver devastating blows.

2. LITERATURE REVIEW

Song and colleagues [4] developed the world's first searchable encryption method, which they claim will make the search for outsourced encrypted information significantly more efficient. Researchers Goldwasser et al. [5] and Garg et al. [6] have developed completely homomorphic encryption algorithms that are searchable in their

ORAM systems. Despite the fact that both methodologies are capable of achieving extraordinarily secure searchable encryption, the high cost of ORAM computation and the completely homomorphic encryption approach prevent these systems from being cost-effective.

Then, symmetric searchable encryption (SSE), which enhanced search performance at the expense of minor leaks such as access patterns and seard patterns, was proposed as a way to strike a balance between the safety of searchable encryption systems and the research efficiency of the field. Access patterns provide information about which documents were located in the query as well as which searches returned results that contained the same phrase as in the query as the access pattern. Following the first proposals to SSE[4], Curtmola et al. [7] have defined the SSE safety paradigm and implemented the first SSE inverted index-based system as a result of their research. The inverted index technology maps document IDs from each term and serves as a critical foundation for a wide range of additional activities, including our work in this study, which is described in detail below.

While the methods described above are helpful for searching for encrypted data, they are primarily static SSE and do not have the ability to dynamically upgrade the outsourced encrypted material. Kamara et al. [8] provided support for this dynamic updating by developing the first dynamic SSE with sub-linear search time and a number of further dynamic SSE programmes to complement it. As a result of the updating mechanisms used, dynamic SSE systems are more prone than static SSE systems to leak information into the network.

Performing this additional step may provide information on whether or not the current updated word, for example, forward privacy in earlier queries, has been looked for. Additionally, the current query may display documents that match the current search keyword, i.e. retrograded privacy, but have already been removed from the system. private

Additional privacy was first implemented, but it became increasingly important after Zhang et al. conducted a file assault, which was documented in the media. A dynamic SSE attack of this nature is particularly effective when the dynamic SSE methods used are not patented. Because clients outsource all of their papers in "closed systems," as previously stated, an adversary's ability to introduce documents into the system is severely limited; as a result, the attack is difficult in "closed systems." In terms of backward privacy, it was Bost et al. who provided the first accurate definition[13].

Searchable encryption (SE) is becoming increasingly common among cloud storage services since it prevents unencrypted information from being leaked into the impacted server while yet retaining search functionality. It is possible to use two SE regions at the same time: public search encoding (PEKS) and searchable symmetrical Encoding (SSE) (SSE). In our work, we make extensive reference to the SSE.

In 2012, Kamara et al.[7] developed an inverted index-based DSSE system that delivered sublinear, secure CKA2 search complexity while maintaining a high level of security. After that, they developed a novel dynamic, searchable red-black tree index encryption mechanism that allowed them to simultaneously search for keywords, add and delete parallel files while maintaining security. Naveed et al. [9], Xia et al. inverted index [10], and Guo et al. inverted index [11] are examples of other techniques. Aside from that, DSSE leaks information about the search pattern (search query pattern), the size pattern[18] (the number of search results returned), and the access pattern [19]. (how the encrypted data or indexes are accessed). These drew the attention of the general people.

In 2016, Zhang et al. [17] proposed a file-injection attack, in which the attacker can detect keywords in a token by introducing files containing different sentences, as described in the paper. The effectiveness of the attack necessitates the strengthening of DSSE security. The process of downloading and decrypting fully encrypted files in order to extract the relevant files is not a straightforward method of encrypting the search. Using secure two-part computations, fully uniform encoding, and disregarded RAM, you may reduce security waste while increasing store capacity and processing and communication complexity. Both options are prohibitively expensive and impractical.

After introducing the concept of "future privacy" in 2014[18], Stefanov and colleagues have set a new and safe goal for themselves: the realistic implementation of dynamic searchable encryption systems. Different methods have been proposed to accomplish this since 2014, including Stefanov et al.[18] based on a hierarchical logarithmic level structure; Bost[19] based on trapdoor permutations; and they have proposed a forward and backward schema based on primitive functions, including restricted pseudorandom functions and punctual encoding[35]; Wang et al.[36] based on a forward and backward schema based on primitive functions, including restricted

DSSE for multiple users has been the subject of some research. The bilinear accumulator was employed by Nair and Rajasree [13] in order to develop a fine grain multi-user solution for search and access regulation using the multireader technology. Popa and Zeldovich[36] presented a method for encrypting data with different keys that was later used. Regarding the single writer/multilectualist, Curtmola et al. [31] proposed the first system structure based on broadcast encryption in 2018, which was implemented in the following year. According to Wang et al. [20], multiuser forward, safe, and symmetrical searchable symmetric encryption are all viable options. However, the amount of research being done on dynamically searchable multi-user symmetrical (FBM-DSSE) encryption is insufficient.

3. RELATED WORK & ALGORITHM

Searchable Encryption

There are descriptions of both the general system and unfavourable SE models in this section.

System Model

Searchable Encryption allows a firm to search for data on a distant server that has been encrypted using a searchable key. There is also a location on the server of encrypted data objects that include a given keyword, which is included in this section. An easily searchable encryption system is often comprised of three entities:

the name given to a faraway server that stores encrypted information (such as a pay-as-you-go cloud server). We assume that the information stored on the server is sensitive, which means that any third party, including the server, does not want to risk exposing it to unauthorised individuals. It is important to note that in some circumstances, such as medical records, it may be illegal to disclose information about the content. On the server, we presume that the data items are always encrypted, and this is the case.

Data Owner: This entity is in charge of creating the schema and is in possession of the master secrecy key, which is used to build a search query in a single user scenario or a search query in a multi user situation with secret user keys (in the multi-user setting, the data owner is enrolled as the user and generates a user secret key in order to produce search queries). The data owner has the ability to send search queries (also known as read data) to the server at any time and encrypt the server data in the process (referred to as writing data). The data owner can check to see if any other users (if any) have access to or write data in the system by logging into the system.

User(s): The SE schemes have a variable number of users, each of whom can either write, read, or execute these two functions in accordance with the data owner's specifications.

A metadata record is created by the data owner for each data item to be encrypted. The metadata record is often composed of keywords or keywords that characterise the contents of the data item. Every data object in the collection is encrypted separately using a standard encryption approach. As a result, the searchable encryption is concerned with the method by which the encrypted index is constructed. Neither the server retrieval of encrypted data chunks nor the transmission of encrypted data chunks are considered in this work. A data owner or user prepares a server search query for data items to be found in the encrypted index to search for a keyword, and the server searches the encrypted index for data items to be found in the encrypted index for the keyword.

The basic architecture of the SE system is depicted in Figure 1 as a series of five steps:

1. Encryption of data. It is the data owner who encrypts and transfers to the server the information that is supplied to the server (or a user, depending on the circumstance). Most of the time, encrypted data is divided into two parts:

(a) Data items that have been encrypted: Key encryption is used to symmetrically encrypt data objects that have been outsourced to the server by their respective owners. It is possible to identify each encrypted data item by its unique identification (ID).

In addition, a secure index enables the server to utilise a search token to locate IDs on the server that correspond to the encoded data items that have been identified.

2. The transfer of the user's secret key. The secret key for a user is generated and transferred by the data proprietor. The user can generate search queries by using the secret key provided.

3. I'd want to conduct a search. A user creates and sends a keyword search query to the server, which then determines which encrypted data items match the user's search criteria.

4. The results of the search. The search results that are displayed to the user can be divided into two categories:

On encrypted data items, (a) IDs that match a search token query are returned; (b) encrypted data items that match a search token query are returned

If access to the actual data item is not required by the user (for example, while analysing statistically encrypted data), the first response to this question is the best option because it entails the least amount of communication. Finally, the server searches for encrypted data items that match the query using the ID set, which increases the amount of time it takes to complete the search. Systems that generate search results consisting of IDs of encrypted data items that only meet the search token request are considered in this study; however, recovering actual data items is beyond the scope of this work.

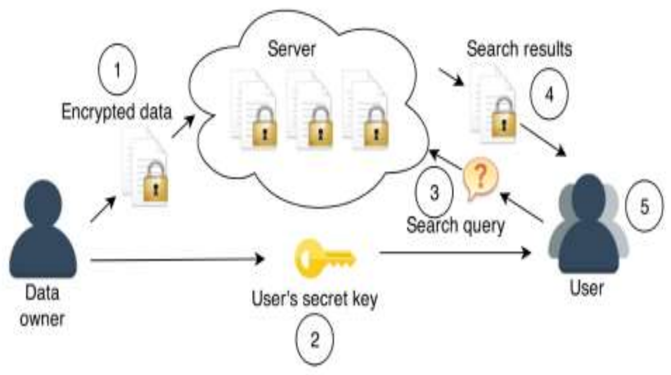

Figure 2.Searchable encryption model.

ALGORITHM

Algorithm 1 is a formal description of our M-SSE scheme, which is shown in Algorithm 2. The system allows for the addition and deletion of operations, and the client transmits these data blocks to cloud c1 or cloud c2 for use in a variety of applications. H is a hash key function with an output of -bits, while IND-CPA symmetrically encrypts Ek (m) and Dk (d) with a -bit output (c).

Algorithm 1 M-SSE Basic Construction of Multi-Cloud SSE

```
Setup()
  1: $K_\Sigma \xleftarrow{\$} \{0,1\}^\lambda$
  2: $(SK, PK) \xleftarrow{\$} KeyGen(1^\lambda)$
  3: $W, T \leftarrow$ empty map
Search($w$, $\sigma$; EDB)
Client:
  1: $K_w \leftarrow F(K_\Sigma, w)$
  2: $(ST_c, c) \leftarrow W[w]$
  3: If $(ST_c, c) = \perp$ and $Tag \neq \perp$
  4:     documents matching w are stored in $c_3$
  5: Else
  6:     Send $(K_w, ST_c.token_l, c_l)$ to $c_1$, $(K_w, ST_c.token_r, c_r)$ to $c_2$
     Cloud $c_1$ and $c_2$:
  7: Run $\Sigma o \varphi o \varsigma - B.Search(w, \sigma; EDB)$ and get $E_{K_S}(ind, op)$,
      store the result in S
  8: Send S to the client.
     Client:
  9: Decrypt S and get $\{ind : \exists i, (ind_i, op_i) \cap \forall j > i, (ind_j, op_j) \neq (ind, del)\}$
Update($w$,$c_1$, $c_2$; EDB)
Client or cloud $c_3$:
  1: $K_S \xleftarrow{\$} \{0,1\}^\lambda$
  2: Randomly select documents sending to $c_1$ and $c_2$ respectively.
  3: Run $\Sigma o \varphi o \varsigma - B.Update(w, E_{K_S}(ind, op), \sigma; EDB)$ on cloud $c_1$ and $c_2$
```

UPDATE OPERATION

When a document that matches the Keyword w is changed, the client will create a new data block and transmit it to cloud c1 or cloud c2 at random, depending on the document. The action of updating a document comprises the addition and deletion of documents. The document can be a completely new document or a copy of an existing c3 document that was searched for and downloaded for the addition procedure. In order for the document to be deleted, it must already be in the delete procedure database. The following is a step-by-step procedure:

Step 1 (Choose a random cloud as the target cloud): Make a choice about which cloud the new block should be transmitted to. An encrypted and operated version of the new document identifier is contained within the new block. As a result, a new block can be placed in two different locations, increasing privacy.

Step 2 (New Token Generation): Tokens of a new generation will be issued. A new token for the new block is generated by the consumer in step 1 of the process. The token has something to do with the phrase w that matches it in the new block. As a result, the individual who purchases the token will be able to receive the matching document w.

Step 3 (Update to Map W): If the new block is sent to cloud c1, W[w] will be updated accordingly. The new value updates STw.tokenl and W[w].cl if it is not already updated, else the new value updates W[w]. W[w].cr and STw.tokenr are two examples of W[w].cr.

SEARCH OPERATION

This is accomplished by the customer creating a search token that corresponds to the keyword w. The search token t will allow the server to receive document IDs that correspond to the term w when the search token is used. The following is a more detailed description of the approach:

Step 1 (RELEASE W MAP) consists of the following steps: With the keyword w, we will be able to obtain the current state of w. If W[w].STc.tokenl is not â€," it is moved to cloud c1. Otherwise, it is transferred to cloud c2. W[w].STc.tokenl. Instead, the W[w].STc.tokenr will be delivered to cloud c2 through the internet. If the W[w].STc.tokenl and W[w].STc.tokenr versions are the same, then w in cloud c3 is equal to 1.

Step 2 (Merge Identifier) is as follows: Cloud data blocks are encrypted with KS, which is then used to decrypt the blocks. In the case if we have both (indi, add), this is not taken into consideration (indi, del)

Step 3 (papers that have been received): According to the position map, we can determine the location of the document for each ind = DKw (EKw (indi, op)). When a match is found, the matching ind can be broadcast from step 2 to numerous clouds.

Step 4 (Upload Cloud c3 search results): After the w keyword has been searched for and the documents have been obtained, the encrypted documents and IDs are uploaded to c3 and the Tag is reset to zero. After a period of time, these documents will be promoted to c1 and c2 status and become new documents.

4. EXPERIMENT RESULTS

The purpose of this experiment is to compare the efficiency of M-SSE systems to that of other conventional SSE systems. The M-SSE system divides the search into two parts: one provides a token on the client side, and the other searches on the server side, which is done in parallel. The update can also be divided into two sections: first, a new block is produced on the client side, and second, the new block is uploaded to the server side, which completes the cycle. Our monitoring includes bandwidth, server and client computing, as well as client and server computing. These activities are divided into two components:

A. IMPLEMENTATION DETAILS

M-SSE is implemented using the C/core C++ function and benchmark, and it is tested. Primitive cryptography is implemented in M-SSE using source code 6oo[2]. We implement RSA with the help of the OpenSSL BigNum module, which uses the HMAC key-ash function. Furthermore, we employ RSA to achieve trapdoor permutation.

1) EXPERIENCE WITH ACCOUNTS. The server map is stored in RocksDB, which is a relational database. In the Desktop PC, we are experimenting with an Intel Core i7-7700 3.60GHz processor, 2GB of RAM, and Ubuntu 14.0.4 operating system.

2) MAKE A RESERVATION. The safe parameter has been set to 128 bits in this case. In practise, the maximum number of keywords and documents can be anywhere between 140 and 140 00000, depending on the situation. When it comes to length, the cryptographic keys are 128 bits long, whereas RSA keys are 2048 bits in length.

B. EVALUATION

We tested the performance of M-SSE on 140,000 keyword pairs to see how well it performed. The experiment is comprised of three operations: the creation of a token, the searching for information, and the updating of information.

1) TOKEN GENERATION

As illustrated in Figure 3, we evaluate the efficiency with which tokens are generated during search. As far as we are aware, Fides provides the highest level of creative performance conceivable. The 6o Scheme serves as the foundation for all M-SSE and Agenzia Fides operations. M-SSE and Fides conduct nearly identical tasks during the token creation process, resulting in token performance that is comparable. According to the results of the testing, M-speed SSEs are no worse than Fides and are nearly the same as Fides in terms of performance. M-SSE, on the other hand, provides more privacy than Fides. TWORAM is the most inefficient algorithm, even if only the bare minimum of information is provided, as seen in Figure 3.

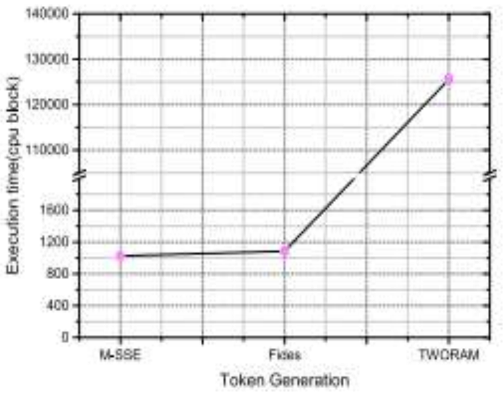

FIGURE.3.Comparison of token generation.

2) SSE OPERATION

As illustrated in Figures 4 and 5, we examine the performance of search and update algorithms that are used in different situations. M-SSE and Fides disc access will not be fully integrated with RSA operations at the start of the search procedure because of a technical limitation. Latency, on the other hand, is caused by mutexes and storage access. M-SSE saves the keyword index for w on two different clouds, allowing us to access the index at the same time. As a result, the pace of the search approach is approximately 2 pounds faster than that of Fides. Because of the complexity of ORAM, the search and update efficiency of TWORAM is much lower than that of ORAM.

These approaches are examined using the same keywords, keywords, and benchmarks as the previous approaches. Thus, the initialization and benchmarks are the same as they were previously. We may conclude that M-SSE is significantly greater than TWORAM and Fides.

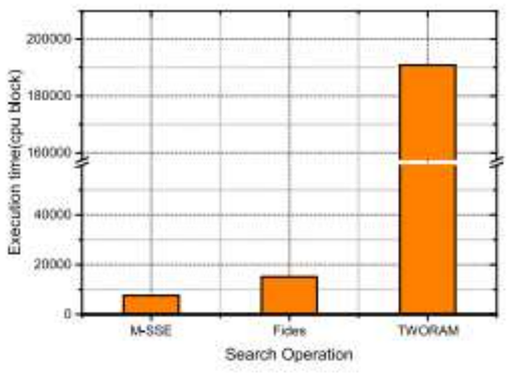

FIGURE 4. Comparison of Search operations

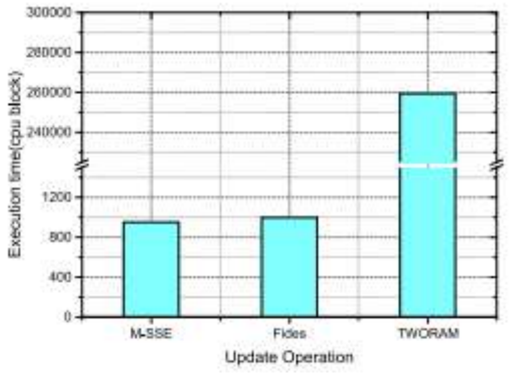

FIGURE 5. Comparison of Update operations. Keyword-document pairs is set to 140000.

3) SECURITY LEVEL

M-SSE has the ability to disperse leaks across many clouds due to its multi-cloud features. If the clouds do not pollute, the amount of knowledge that each cloud knows about the environment can be reduced. As a result, M-SSE is capable of protecting the size pattern. M-SSE can help safeguard users' privacy from file injection assaults, which are becoming increasingly common. In part due to its backward-looking private level, M-SSE is able to protect the pattern of updates going forward.

M-SSE is nearly as good as Fides, which is the best performer in SSE but has a higher level of reliability. Despite the fact that TWORAM has a minor level of confidentiality, its performance is significantly worse than that of M-SSE. The goal of M-main SSE is to prevent leakage caused by searchable safe encryption. As a result, we may conclude that M-SSE achieves a satisfactory balance between efficiency and security.

5. CONCLUSION

As part of this research, we developed a functional SSE system that provides privacy for search patterns as well as privacy and privacy. K-anonymity and encryption technology were used to create a hidden approach, which was particularly effective. Next, utilising the dynamic SSE technique and pseudorandom and pseudorandom to accommodate the efficient boolean query, we created a single keyword query that was optimised for speed and efficiency. Parallel to this, we examined the security of our system and demonstrated that the suggested system met all of the necessary security standards, including search privacy, privacy, and superior backward privacy, among others. We have also performed extensive experiments to evaluate the performance of our suggested system, with the findings demonstrating that the scheme is cost-effective in terms of overall communication and processing expenses. Our long-term goal is to better optimise search and update activities while maintaining the confidentiality of the SSE system as much as possible.

REFERENCES

❖ D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, ‘‘Public key encryption with keyword search,’’ in Proc. Eurocrypt, 2004, pp. 506–522.

❖ R. Bost, ‘‘6oϕoς: Forward secure searchable encryption,’’ in Proc. ACM CCS, 2016, pp. 1143–1154.

❖ R. Bost, P.-A. Fouque, and D. Pointcheval, ‘‘Verifiable dynamic symmetric searchable encryption: optimality and forward security,’’ Int. Assoc. Cryptol. Res., Las Vegas, NV, USA, Tech. Rep. 2016/062, 2016, vol. 62. [Online]. Available: https://eprint.iacr.org/2016/062 PWC. 2015

Information Security Breaches Survey; Technical Report; PWC: London, UK, 2015.

- Bösch, C.; Hartel, P.H.; Jonker, W.; Peter, A. A Survey of Provably Secure Searchable Encryption. ACM Comput. Surv. 2014, 47, 18.
- Bao, F.; Deng, R.H.; Ding, X.; Yang, Y. Private Query on Encrypted Data in Multi-user Settings. In Proceedings of the 4th International Conference on Information Security Practice and Experience, Sydney, Australia, 21–23 April 2008; Volume 4991, pp. 71–85.
- D. X. Song, D. A. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," *IEEE Symposium on Security and Privacy*, vol. 3, no. 3, pp. 44-45, 2000.
- D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," *Advances in Cryptology-EUROCRYPT 2004*, vol. 26, pp. 506–522, 2004.
- E. Goh, "Secure indexes," *IACR Cryptology ePrint Archive*, vol. 216, 2003.
- J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Proceedings of the Computational Science and Its Applications – ICCSA 2020*, pp. 1249–1259, Cagliari, Italy, July 2008.
- "Data-sharing and cloud: A big data match made in heaven," https: //www.computerweekly.com/blog/Ahead-in-the-Clouds.
- Y. Zheng, R. Lu, B. Li, J. Shao, H. Yang, and K. R. Choo, "Efficient privacy-preserving data merging and skyline computation over multisource encrypted data," Inf. Sci., vol. 498, pp. 91–105, 2019.
- Y. Zheng, R. Lu, and J. Shao, "Achieving efficient and privacypreserving k-nn query for outsourced ehealthcare data," J. Medical Systems, vol. 43, no. 5, pp. 123:1–123:13, 2019.
- D. X. Song, D. A. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in IEEE Symposium on Security and Privacy, 2000, pp. 44–55.
- S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F. Liu, A. Sahai, E. Shi, and H. Zhou, "Multi-input functional encryption," in Advances in Cryptology - EUROCRYPT, 2014, pp. 578–602.