TTP-Aided Searchable Encryption of Documents Using Threshold Secret Sharing

Ahmad Akmal Aminuddin Mohd Kamal[1][a] and Keiichi Iwamura[2]

[1]Department of Information and Computer Technology, Tokyo University of Science, Tokyo, Japan

[2]Department of Electrical Engineering, Tokyo University of Science, Tokyo, Japan

Keywords: Searchable Encryption, Threshold Secret Sharing, TTP-Aided, Secure Computation, Secure Storage, Clouds, Cloud Security, Information Security.

Abstract: In recent years, the introduction of services such as storage-as-a-service has enabled users to outsource their data to cloud servers to mitigate the cost of physical storage infrastructure. Moreover, cloud storage allows users to access data anywhere. However, outsourcing sensitive data to cloud servers also introduces concerns regarding data leakage and privacy. Therefore, these data must be encrypted before storage. Searchable encryption (SE) is a method that allows data to be searched in its encrypted state. SE uses symmetric key encryption, public key encryption, or secret sharing. SE using symmetric and public key encryptions can be implemented using one cloud server. However, most SEs utilize the search index for efficiency, which incurs the additional cost of constantly updating the search index. SE using secret sharing is computationally light. Therefore, a direct search over ciphertext is possible without sacrificing the efficiency. However, it requires multiple, independently managed cloud servers. In this study, by effectively using a trusted third party, we demonstrate that realizing an SE with a single cloud server is possible, even if secret sharing is used, thereby reducing the total running cost and communications required. Moreover, we demonstrate that the proposed method is secure against a semi-honest adversary.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

Cloud computing services, such as secure-storage-as-a-service, enable businesses and users to upload their private data into the clouds for easy access from anywhere in the world, without requiring any physical infrastructure. Owing to concerns and risks regarding the privacy and confidentiality of the information stored in the cloud, data must be encrypted before storage (Wang et al., 2017). However, in this approach, searching encrypted data using conventional encryption methods is not possible. That is, the encrypted data must be decrypted momentarily to search for possibilities.

Searchable encryption (SE) allows cloud users to search for and retrieve encrypted data while securely preserving data confidentiality. Conventional SE methods use symmetric encryption, public key encryption, and secret sharing. In recent years, several SE solutions using symmetric and public key encryptions have been proposed. SE solutions using these encryption methods allow users to securely outsource their private data to a single third-party cloud server as long as the encryption keys used to encrypt their data are kept secure. Therefore, only a single cloud server is required for secure storage, minimizing the usage cost.

However, most SEs based on symmetric and public key encryptions proposed thus far only realize the search of encrypted data using a search index. A search index is a body of structured data that a search engine in the cloud server refers to when searching for data that are relevant to a specific search query by the user (Goh, 2003). Index data are related to the keywords in a document, which are encrypted and registered in the search index. However, this method has two main disadvantages: (1) keywords that are not registered in the search index cannot be searched and (2) the search index needs to be updated each time a new document is added or removed.

By contrast, Kamal et al. proposed SEs based on secret sharing (Kamal and Iwamura, 2021). Secret sharing is a method in which a secret input is divided into multiple different values (known as shares)

[a] https://orcid.org/0000-0002-7941-3021

Kamal, A. and Iwamura, K.
TTP-Aided Searchable Encryption of Documents Using Threshold Secret Sharing.
DOI: 10.5220/0011653700003405
In *Proceedings of the 9th International Conference on Information Systems Security and Privacy (ICISSP 2023)*, pages 442-449
ISBN: 978-989-758-624-8; ISSN: 2184-4356
Copyright © 2023 by SCITEPRESS – Science and Technology Publications, Lda. Under CC license (CC BY-NC-ND 4.0)

and stored on multiple cloud servers (Shamir, 1979). SEs using secret sharing enable direct searching over encrypted data, thereby solving the aforementioned drawbacks of using a search index. Moreover, because the secret sharing method is computationally light, the computational cost for encrypting data is extremely low compared to that of SEs using symmetric and public key encryptions.

However, secret sharing requires multiple independently managed cloud servers (generally at least three) to be secure. Therefore, the user must use multiple cloud servers from different providers to implement SE using secret sharing, which is difficult and costly. In addition, a high-speed communication network between all the cloud servers is necessary.

Moreover, the SEs proposed by Kamal et al. were based on a secure computation known as the Tokyo University of Science 2 (TUS 2) method (Kamal and Iwamura, 2017) (details provided in Section 2). In secure computation in the TUS 2 method, the search operation between the query and encrypted data can be computed in its encrypted state using a minimum of two cloud servers. However, the computation cost of the TUS 2 method is extremely large when compared to that of conventional secure computation methods. Therefore, this is not the most efficient and practical SE method.

In contrast, in the latest version of the TUS method (known as the TUS 6 method), by leveraging a trustable third party (TTP), Iwamura et al. proposed a secure computation based on secret sharing with only a single computing server (Iwamura et al., 2022a). This method overcomes the drawbacks of all secret sharing methods.

Our Contributions.

In this study, the *concept* of direct searching in Kamal et al.'s SE (Kamal and Iwamura, 2021) was combined with the *idea* of TTP-assisted secure computation in the TUS 6 method (Iwamura et al., 2022a) to realize an efficient SE based on (k,n) threshold secret sharing with one cloud server.

In particular, we applied the concept of differentiating the parameters used for shares (n) and cloud servers (N) in the TUS 6 method and extended the general secure computation in the TUS 6 method (Iwamura et al., 2022a) to realize an SE method using (k,n) threshold secret sharing, which can realize a direct search over encrypted documents with only a single cloud server.

Thus, our proposed SE can be implemented using a single cloud server, thereby reducing the total running cost required and making the actual implementation more practical. Moreover, because our proposed SE does not require any consecutive computations using the search result, it achieves lower computational and communication costs than those of the TUS 6 method.

We also evaluated the security of our proposed method and showed that it could achieve a higher security level than that of the original TUS 6 method. Finally, we discuss a more efficient SE method by allowing some partial information leakage and also include a discussion on the use of TTP.

2 BUILDING BLOCKS

This section explains the basic methodology required to realize an SE using (k,n) threshold secret sharing with only a single cloud server.

2.1 (k,n) Threshold Secret Sharing

Secret sharing is a method of converting a secret input s into a sequence of n different values (known as *shares*) and distributing them to n different computing servers. A secret sharing that satisfies the following conditions is known as (k,n) threshold secret sharing (Shamir, 1979). However, $n \geq k > 1$.

- Any $k-1$ or fewer shares will not reveal details about the secret s;
- Any k or more shares will allow for the reconstruction of the secret s.

Therefore, if the cloud servers that store the shares are managed by the same organization, k (or more) shares are collected and the secret input is leaked. Examples of (k,n) threshold secret sharing include a method that uses polynomials for the distribution of the secret input proposed by Shamir (Shamir, 1979) (hereinafter called Shamir's (k,n) method), and additive secret sharing. Unless otherwise stated, Shamir's (k,n) method was used, and the prime number was p. In addition, the shares of secret s are denoted by $[s]_i$.

2.2 TUS Methods

Encryption of a secret input using Shamir's (k,n) method is performed using a $(k-1)$-degree polynomial (with the secret input being the constant term) to compute n shares corresponding to n cloud servers. This means that collecting a threshold of k shares allows the user to reconstruct the polynomial and, consequently, the secret. However, the multiplication of two secret inputs results in a $(2k-2)$-degree polynomial, and the number of shares required to reconstruct the result increases to $2k-1$ (Cramer et al., 2000).

Therefore, all secure computations based on Shamir's (k,n) method require at least $n = 2k - 1 = 3$ servers.

TUS methods have been used to represent the secure computation methods proposed by a research group at TUS. TUS methods propose solutions for secure computation using Shamir's (k,n) method without increasing the number of cloud servers required. In all TUS methods, the secret input is first encrypted with a random number before being secret-shared to the cloud servers. Moreover, when performing multiplication, the encrypted secret is momentarily restored and multiplied by the shares of the other encrypted secrets to prevent an increase in the polynomial degree in the resulting shares.

However, all the TUS methods require at least two independently managed cloud servers. Therefore, Iwamura et al. proposed an improved TUS 6 method that uses only a single cloud server with the help of TTP (Iwamura et al., 2022a). This is realized by sending multiple shares of the same secret input to the server. Here, the TTP generates and protects the random numbers used throughout the computation.

This study extends the latest version of the TUS method, known as the TUS 6 method (Iwamura et al., 2022a), to realize SE using a single cloud server. The protocols of the TUS 6 method are not disclosed here owing to page limits.

3 RELATED WORKS OF SE

3.1 SE Using Symmetric Encryption

The first SE using symmetric encryption was proposed by Song et al., in which the search was directly performed over the encrypted keywords of each document (Song et al., 2000). However, this method has low efficiency, and keywords that have not been registered before cannot be searched.

To solve the efficiency problem, SE with a secure index, such as a document-based or keyword-based index, was proposed. Goh et al. proposed an SE using a document-based index, where the construction of the secure index is performed using pseudorandom functions and Bloom filters (Goh, 2003). However, each index only corresponds to one document; therefore, the server must search through multiple indexes when multiple documents are stored. Curtmola et al. proposed an SE using a keyword-based index, where one keyword in the index may correspond to many document identifiers (Curtmola et al., 2006).

The advantage of using a secure index is that instead of scanning the entire ciphertext of keywords in a document, the server only needs to search over the search indexes. However, if the keyword is not pre-registered in the index, the search is not possible. Moreover, the index should be updated whenever a new document is added or removed from the cloud server, thereby increasing overall operation and maintenance costs (Wang et al., 2017).

3.2 SE Using Public Key Encryption

The first SE using public key encryption was proposed by Boneh et al., based on identity-based encryption (Boneh et al., 2004). In SE using public-key encryption, the owner encrypts the keywords using a public key and stores the ciphertext to the cloud server. The searcher with the correct private key produces a search tag and sends it to the cloud server, where the server identifies whether a search tag is in a particular ciphertext.

Many SEs that use public key encryption have been proposed. Abdalla et al. proposed a generic solution for SE by utilizing a keyword search in a mail server (Abdalla et al., 2005). Xu et al. proposed an SE that supports fuzzy keyword searches (Xu et al., 2013). Nonetheless, SE using public key encryption is extremely inefficient and incurs large computational costs. Moreover, the ciphertext size of the registered keyword is often large and requires considerable storage space.

3.3 SE Using Secret Sharing

Kamal et al. proposed a solution for realizing SE using secret sharing (Kamal et al., 2017). In this method, SE is used to search for encrypted images stored in cloud servers. However, because the searching process is performed on each single pixel, and the result is also reconstructed for each pixel, it is not secure against brute-force attacks because the adversary that corrupts the searcher can change the search query for each pixel that does not match.

To solve this problem, Kamal et al. proposed an improved version of SE using secret sharing, which allows the result of multiple characters to be reconstructed simultaneously (Kamal and Iwamura, 2021). However, communication and computation costs are extremely high because this method is based on the TUS 2 method (Kamal and Iwamura, 2017).

In addition, Iwamura et al. proposed an SE that can realize a partial matching search, where a certain difference in the search query is allowed (Iwamura et al., 2022b). However, all SEs based on (k,n) threshold secret sharing require at least two independently managed cloud servers. Therefore, the initial setup and maintenance costs are significantly larger than those of most SEs using key encryption, where the subscription of a single cloud server is sufficient.

4 PROPOSED METHOD

This section explains our SE using (k,n) threshold secret sharing on a single cloud server.

4.1 Overview of the Proposed Method

In the proposed searching method, we implemented the concept of direct searching proposed by Kamal et al. (Kamal and Iwamura, 2021), where, rather than searching for a matching keyword in the precomputed search index, we realize a search process for each character between the search query and registered document in the cloud server.

Moreover, we simultaneously realize the function of total matching search for multiple characters. That is, although search computation is performed per character, the final result is reconstructed only once. However, the method by Kamal et al. (Kamal and Iwamura, 2021) requires at least two cloud servers, considerable communication and computation costs, being extremely inefficient. In this study, a more efficient search method is proposed.

Details of the total matching search computations used in the proposed method are described below. Suppose a registered document a with l characters $a_1,\dots,a_l$, and a search query b with g characters $b_1,\dots,b_g$. The difference for each character between documents a and query b can be computed as $d_1 = a_1 - b_1$, $d_2 = a_2 - b_2$, …, $d_g = a_g - b_g$.

Here, a function f such that $f = 0$ only when all differences are equal to zero can be represented as follows:

$$f(d_1,\dots,d_g) = d_1 \times \cdots \times d_g.$$

Therefore, in the proposed method, we performed the aforementioned computation for each g character using only one cloud server. Moreover, if the search query does not match the first g characters of the documents, the search process can be continued by shifting the search position h to the left by one ($h = h+1$).

4.2 Proposed Protocols

The specific protocols of the proposed method are described below.

The size of each character a_x $(x = 1,\dots,l)$ in the registered document a and the character b_y $(y = 1,\dots,g)$ in the search query are elements within $p-2$. Moreover, all random numbers generated in the protocols are non-zero elements of $GF(p)$. Here, p is a prime number, and all computations are performed within modulus p.

In addition, in the following protocols, let $x = 1,\dots,l$, $y = 1,\dots,g$, $i = 0,\dots,n-1$, $j = 0,\dots,k-1$, and parameter $k \geq 3$. For ease of understanding, we describe the protocols with $n = k$; therefore, the ranges of i and j are the same, that is, $i = j = 0,\dots,k-1$.

Protocol 1.1: Encryption of documents.

- Input: a_x $(x = 1,\dots,l)$
- Output: $\alpha_x(a_x+1)$

1. Each owner generates l random numbers α_x and computes $\alpha_x(a_x+1) = \alpha_x \times (a_x+1)$ for each character of document a. Then, it sends α_x to TTP and $\alpha_x(a_x+1)$ to server S.

Protocol 1.2: Encryption of search query.

- Input: b_y $(y = 1,\dots,g)$
- Output: $\beta_y(b_y+1)$

1. The searcher generates g random numbers β_y and computes $\beta_y(b_y+1) = \beta_y \times (b_y+1)$ for each character of their search query b. Then, it sends β_y to TTP and $\beta_y(b_y+1)$ to server S.

Protocol 1.3: Search process.

- Input: a, b
- Output: a or failed

1. The computing server S and TTP perform the following for g consecutive characters in document a and search query b, starting with the initial position $h = 0$. In the following steps, we let $y = 1,\dots,g$, $v = h+y$, and $h = 0,\dots,l-g$.
   (a) TTP generates a random number γ_h and computes and distributes the following auxiliary random numbers using Shamir's (k,n) method:
   $$\frac{\gamma_h}{\alpha_v}, \frac{\gamma_h}{\beta_y}.$$
   (b) TTP generates random numbers $\tau_{j',h}$ $(j' = 1,\dots,k-1)$, computes, and sends the following to server S, with $\tau_{0,h} = 1$:
   $$\tau_{j,h}\left[\frac{\gamma_h}{\alpha_v}\right]_j = \tau_{j,h} \times \left[\frac{\gamma_h}{\alpha_v}\right]_j,$$
   $$\tau_{j,h}\left[\frac{\gamma_h}{\beta_v}\right]_j = \tau_{j,h} \times \left[\frac{\gamma_h}{\beta_v}\right]_j.$$
   (c) Server S computes the following for each $j = 0,\dots,k-1$:
   $$\sum_{y=1}^{g} \tau_{j,h}\gamma_h[a_v - b_y]_j = \sum_{y=1}^{g} \Big\{ \alpha_v(a_v+1) \times \tau_{j,h}\left[\frac{\gamma_h}{\alpha_v}\right]_j - \beta_y(b_y+1) \times \tau_{j,h}\left[\frac{\gamma_h}{\beta_v}\right]_j \Big\}.$$

(d) TTP distributes constant 0 using Shamir's (k,n) method, and computes $\gamma_h[0]_j = \gamma_h \times [0]_j$.

(e) TTP obtains $\sum_{y=1}^{g} \tau_{m,h}\gamma_h[a_v - b_y]_m$ from server S, and computes $\sum_{y=1}^{g} \gamma_h[a_v - b_y]_m + \gamma_h[0]_m$ as follows. Here, $m = 1,\ldots,k-1$. Then, TTP sends $\sum_{y=1}^{g} \gamma_h[a_v - b_y]_m + \gamma_h[0]_m$ and $\gamma_h[0]_0$ to server S.

$$\sum_{y=1}^{g} \gamma_h[a_v - b_y]_m + \gamma_h[0]_m = \frac{\sum_{y=1}^{g} \tau_{m,h}\gamma_h[a_v - b_y]_m}{\tau_{m,h}} + \gamma_h[0]_m.$$

(f) Server S adds $\gamma_h[0]_0$ to $\sum_{y=1}^{g} \tau_{0,h}\gamma_h[a_v - b_y]_0$, with $\tau_{0,h} = 1$, and reconstructs the computation result $\sum_{y=1}^{g} \gamma_h(a_v - b_y)$.

2. If the reconstructed result is equal to 0, server S and TTP send $\alpha_x(a_x+1)$ and α_x to the searcher.
3. The searcher reconstructs document a from the following:

$$a_x = \frac{\alpha_x(a_x+1)}{\alpha_x} - 1.$$

5 SECURITY OF THE PROPOSED METHOD

In this section, we discuss the security of the proposed SE method. In the proposed SE, we implemented the secure computation of the TUS 6 method (Iwamura et al., 2022a), which was proven computationally secure against a semi-honest adversary, to realize an SE with a single cloud server. However, we further extended the secure computation in the TUS 6 method to realize a more efficient computation and a higher security level. In the proposed method, we assumed a semi-honest adversary. Moreover, the TTP is assumed to be trusted.

5.1 Security of Protocols 1.1 and 1.2

Suppose that the adversary has no information about the registered document a or search query b. However, the adversary has information from the cloud server S. Here, when the adversary can identify the unknown information of a or b, the attack is considered successful.

In Protocols 1.1 and 1,2, cloud server S holds the following information A on document a and search query b. Let $x = 1,\ldots,l$ and $y = 1,\ldots,g$.

$$A = (\alpha_x(a_x+1), \beta_y(b_y+1)).$$

Here, $\alpha_x(a_x+1)$ and $\beta_y(b_y+1)$ are the encrypted information for document a and query b, respectively. However, because the random numbers α_x and β_y used to encrypt document a and search query b, respectively, are not sent to server S, these random numbers cannot be retrieved by the adversary. That is, even if the adversary corrupts server S, the values of a_x and b_y cannot be retrieved. Therefore, protocols 1.1 and 1.2 are information-theoretic secure against a semi-honest adversary, and the following statement is true:

$$H(a_x) = H(a_x|A),\ \ H(b_y) = H(b_y|A).$$

5.2 Security of Protocol 1.3

Herein, we study the security of Protocol 1.3 in detail. In most SEs, security evaluation is performed under the assumption that the adversary only corrupts the cloud server without considering that either the owner or the searcher is the adversary.

However, considering the situation in which the adversary may also corrupt the searcher to learn about the document registered in the server is also important. Therefore, we consider the following two types of semi-honest adversaries.

Adversary 1: The adversary corrupts the cloud server S. Adversary 1 has information from server S, and attempts to learn the content of document a or search query b.

Adversary 2: The adversary corrupts both the cloud server S and the searcher with search query b. Adversary 2 has all the information from Protocol 1.2 in addition to the information from server S, and attempts to learn the content of document a.

Security Against Adversary 1.
Suppose that Adversary 1 has information from server S in Protocol 1.3, in addition to encrypted information of document a and search query b. Adversary 1 has the following information: B. Let $j = 0,\ldots,k-1$, $y = 1,\ldots,g$, $x = 1,\ldots,l$, $h = 0,\ldots,l-g$, $v = h+y$ and $m = 1,\ldots,k-1$.

$$B = \Bigg\{\alpha_x(a_x+1), \beta_y(b_y+1), \tau_{j,h}\left[\frac{\gamma_h}{\alpha_v}\right]_j, \tau_{j,h}\left[\frac{\gamma_h}{\beta_y}\right]_j, \sum_{y=1}^{g}\tau_{j,h}\gamma_h[a_v - b_y]_j, \sum_{y=1}^{g}\gamma_h[a_v - b_y]_m + \gamma_h[0]_m, \gamma_h[0]_0, \sum_{y=1}^{g}\gamma_h(a_v - b_y)\Bigg\}.$$

First, to learn secret inputs a_v of the owner and search query b_y of the searcher from $\alpha_v(a_v+1)$ and $\beta_y(b_y+1)$, Adversary 1 has to learn random numbers

α_v and β_y from $\tau_{j,h}\left[\gamma_h/\alpha_v\right]_j$ and $\tau_{j,h}\left[\gamma_h/\beta_y\right]_j$, respectively.

However, because both random numbers $\tau_{j,h}$ and γ_h are generated by the TTP, Adversary 1 will not be able to learn them from server S. Moreover, Adversary 1 learns k shares of the following from Steps 1 (c) and (e). Adversary 1 will be able to learn $\tau_{0,h} = 1$, but will not be able to learn the remaining $\tau_{m,h}$ from the ratio of the following information without first learning about each individual share, $\gamma_h[0]_m$.

$$\sum_{y=1}^{g} \tau_{j,h}\gamma_h\left[a_v - b_y\right]_j, \sum_{y=1}^{g} \gamma_h\left[a_v - b_y\right]_m + \gamma_h[0]_m.$$

Finally, in the reconstruction of the computational result, Adversary 1 learns $\sum_{y=1}^{g} \gamma_h(a_v - b_y)$. If the result is not equal to zero, a random number is output; otherwise, Adversary 1 learns that $\sum_{y=1}^{g} a_v = \sum_{y=1}^{g} b_y$; however, the information of random number γ_h will not be leaked.

From the arguments above, our proposed method is information-theoretic secure against Adversary 1, and Adversary 1 cannot learn a_v or b_y. Therefore, the following statement is true:

$$H(a_v) = H(a_v|B), \;\; H(b_y) = H(b_y|B).$$

Security Against Adversary 2.
From Protocol 1.2, Adversary 2 has information on the search query b and random numbers β_y inputted by the searcher. Therefore, Adversary 2 contains the following information: C.

$$C = \Bigg\{\beta_y, b_y, \alpha_x(a_x+1), \tau_{j,h}\left[\frac{\gamma_h}{\alpha_v}\right]_j, \tau_{j,h}\left[\frac{\gamma_h}{\beta_y}\right]_j, \sum_{y=1}^{g} \tau_{j,h}\gamma_h[a_v - b_y]_j, \sum_{y=1}^{g} \gamma_h[a_v - b_y]_m + \gamma_h[0]_m, \gamma_h[0]_0, \sum_{y=1}^{g} \gamma_h(a_v - b_y)\Bigg\}.$$

To learn secret a_v of the owner from $\alpha_v(a_v+1)$, Adversary 2 first has to learn a random number α_v from $\tau_{j,h}\left[\gamma_h/\alpha_v\right]_j$.

However, because random numbers $\tau_{j,h}$ and γ_h are generated by the TTP, Adversary 2 will not be able to learn them without corrupting the TTP. Moreover, Adversary 2 holds β_y and $\tau_{j,h}\left[\gamma_h/\beta_y\right]_j$ from the searcher and the cloud server S, respectively.

The TTP in our proposed method does not secret share random number γ_h to cloud server S. Therefore, even if Adversary 2 compute k shares $\tau_{j,h}\left[\gamma_h\right]_j$ from above, the computational attack of guessing random numbers $\tau_{j,h}$ and comparing the reconstructed result of the computed γ_h with the correctly distributed γ_h is not possible, and the random number γ_h is not leaked.

Moreover, Adversary 2 will not be able to learn random numbers $\tau_{m,h}$ from the ratio of k shares of the following without first learning about each individual share of $\gamma_h[0]_m$:

$$\sum_{y=1}^{g} \tau_{m,h}\gamma_h[a_v - b_y]_m, \sum_{y=1}^{g} \gamma_h[a_v - b_y]_m + \gamma_h[0]_m.$$

Finally, in the reconstruction of the computational result, Adversary 1 learns $\sum_{y=1}^{g} \gamma_h(a_v - b_y)$. If the result is equal to zero, Adversary 2 will learn that $\sum_{y=1}^{g} a_v = \sum_{y=1}^{g} b_y$; however, information on the random number γ_h will not be leaked.

From the arguments above, we can state that Adversary 2 cannot learn any a_v. Therefore, the following statement is true:

$$H(a_v) = H(a_v|C).$$

However, Adversary 2 may perform a brute-force search attack by inputting multiple search queries to the cloud server and learning documents a even without the correct search query b, which is true for all SEs without user authentication.

Nonetheless, this can be overcome by introducing a time gap between each search, as implemented in the block-time in Bitcoin blockchains. Alternatively, the attack can also be overcome by the introduction of a password such that the actual SE process using the search query will only be performed if the SE on the password matches. Moreover, because the password can be easily updated by the owner, the aforementioned attacks can be prevented. The detailed computation is omitted here because of space constraints.

6 DISCUSSION

6.1 Tradeoff Between Efficiency and Partial Information Leakage

The proposed SE described in Section 4 was proven to be secure against semi-honest adversaries 1 and 2. However, realizing security against both adversaries requires extra computation and communication costs between the cloud server and TTP.

However, as previously mentioned, some SEs (particularly SEs with a single-client model) only assume an adversarial model where a semi-honest server attempts to compromise the privacy of the data and search queries. This is the same as our definition of Adversary 1, where collusion between the cloud server and the user who provides the search queries is not assumed.

In the case where Adversary 2 is not assumed, the proposed SE can be further optimized to realize a more efficient SE at the cost of *partial* information leakage. In particular, because the adversary has no knowledge of the search query b and random numbers β_y used in Protocol 1.2, the random number γ_h will not be leaked even if γ_h/α_v and γ_h/β_y are sent directly to the server instead of previously encrypting their shares with another random number $\tau_{j,h}$:

Therefore, Steps 1(a)—1(f) in Protocol 1.3 described in Section 4, can be simplified as follows:
Protocol 1.4: Optimized search process.

(a) TTP generates a random number γ_h, computes, and sends γ_h/α_v and γ_h/β_y to server S:

(b) Server S computes $\sum_{y=1}^{g}\gamma_h(a_v - b_y)$ as follows:

$$\sum_{y=1}^{g}\gamma_h(a_v - b_y) =$$
$$\sum_{y=1}^{g}\left\{\alpha_v(a_v+1)\times\frac{\gamma_h}{\alpha_v} - \beta_y(b_y+1)\times\frac{\gamma_h}{\beta_y}\right\}.$$

Security Against Adversary 1.
To learn inputs a_v of the owner and search query b_y of the searcher from $\alpha_v(a_v+1)$ and $\beta_y(b_y+1)$, respectively, Adversary 1 first has to learn random numbers α_v and β_y from γ_h/α_v and γ_h/β_y.

However, because the random number γ_h is generated by the TTP, Adversary 1 will not be able to learn it from server S. Moreover, it will be able to compute $\gamma_h(a_v+1)$ and $\gamma_h(b_y+1)$ from $\alpha_v(a_v+1)$, γ_h/α_v and $\beta_y(b_y+1)$, γ_h/β_y, respectively.

Because $\gamma_h(a_v+1)$ and $\gamma_h(b_y+1)$ are encrypted by the same random number γ_h, if $a_v = b_y$, the same encrypted values of $\gamma_h(a_v+1) = \gamma_h(b_y+1)$ are computed, and the adversary will also learn $\gamma_h a_v = \gamma_h b_y$ for that particular position v and y. This is the same as most deterministic encryption methods, such as RSA (without padding), where the same encryption key and input will result in the same encrypted data each time.

However, because the adversary has no information about b_y, the information of a_v and random number γ_h will not be leaked. Moreover, because the random number γ_h is changed for every new position h, no additional information is leaked, except that both a_v and b_y are equal to an unknown random number. That is, our proposed SE can be more efficient by allowing partial information to be leaked without sacrificing the confidentiality of the documents and search queries.

However, the aforementioned partial leakage can also be solved in the same manner as in most deterministic encryptions, that is, with the implementation of a randomization process. The details are omitted here because of space limitation.

6.2 Trusted Third Party

In the proposed SE, a TTP is required to generate and store random numbers and provides computation assistance during the reconstruction of the search result. However, TTP does not handle any raw secret inputs of document a and search query b.

The use of TTP is common in cryptography. For example, in SEs using public key encryption, TTP is required to realize SE in a multi-user setting. A multi-user setting allows a group of authorized users to submit search queries (Wang et al., 2017). To realize this, Bao et al. (Bao et al., 2008) and Kiayias et al. (Kiayias et al., 2016) proposed the use of TTP to generate secret keys for all parties. In contrast, our proposed method of secret sharing also supports multi-user settings, without requiring any additional processes for sharing secret keys among multiple users.

However, for real-world implementation, a TTP might become a potential limitation. To address this, we can adopt various security techniques and technologies to realize TTP, such as using a trusted execution environment (TEE) (i.e., a secure hardware enclave of Intel Software Guard Extensions (SGX)).

TEE, such as Intel SGX (Costan and Devadas, 2016), offers hardware-based memory encryption that isolates specific application code and data in memory. This allows the TEE to perform the same tasks as the TTP in Section 4 without sacrificing privacy and security. Moreover, because of a TEE in the central processing unit (CPU) of the cloud server, replacing the role of TTP with a TEE means that all communications can be performed locally. This results in faster processing time and extremely low latency.

In our future study, we will consider the details of implementing a TEE to replace TTP.

6.3 Comparing Computation, Communication, and Storage Costs

In this study, instead of symmetric or public-key encryption, we realized SE using secret sharing. Conventional encryption methods, such as public key encryption, are often based on difficult operations to ensure security, resulting in more costly computations. By contrast, secret sharing is computationally light, reducing computational costs.

In addition, to realize SE in a multi-user setting, where the searcher might not be the original data owner, a mechanism of sharing the secret keys is required in both SEs using symmetric and public key encryptions. In contrast, secret sharing does not rely on any encryption keys. Therefore, a multi-user setting is possible without any additional process.

Moreover, the ciphertext size produced by public and symmetric key encryptions is often large compared to that of secret sharing, where the computed shares are almost the same size as the input. Therefore, ciphertext generated by public and symmetric key encryptions incurs more storage costs than shares generated by secret sharing. In other words, our proposed SE requires less storage.

Finally, SEs using symmetric and public key encryptions can be performed using a single cloud server; therefore, all computations are performed locally. In contrast, secret sharing typically requires communication between multiple servers. In the proposed SE, only a single server is required, and computations are performed on the server, except when reconstructing the result, where communication with the TTP is required. In our future study, we will consider using a TEE in the CPU to realize an SE with zero communication. Detailed comparison with conventional SEs is omitted owing to space limitation.

7 CONCLUSION

In this study, we extended the secure computation in the TUS 6 method, combined the concept of direct searching in (Kamal and Iwamura, 2021), and realized an improved SE using (k,n) threshold secret sharing with only a single cloud server. We succeeded in solving the drawbacks of SEs using secret sharing, where multiple cloud servers are required.

In future studies, we will implement the proposed method and perform a detailed comparison with conventional SEs. We also consider the means of increasing the practicality of the proposed method by considering the use of a TEE instead of TTP.

REFERENCES

Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., and Shi, H. (2005). Searchable encryption revisited: consistency properties, relation to anonymous ibe, and extensions. In Shoup, V., editor, *Advances in Cryptology – CRYPTO 2005*, pages 205–222, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bao, F., Deng, R. H., Ding, X., and Yang, Y. (2008). Private query on encrypted data in multi-user settings. In Chen, L., Mu, Y., and Susilo, W., editors, *Information Security Practice and Experience*, pages 71–85, Berlin, Heidelberg. Springer Berlin Heidelberg.

Boneh, D., Di Crescenzo, G., Ostrovsky, R., and Persiano, G. (2004). Public key encryption with keyword search. In Cachin, C. and Camenisch, J. L., editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 506–522, Berlin, Heidelberg. Springer Berlin Heidelberg.

Costan, V. and Devadas, S. (2016). Intel sgx explained. Cryptology ePrint Archive, Paper 2016/086.

Cramer, R., Damgård, I., and Maurer, U. (2000). General secure multi-party computation from any linear secret-sharing scheme. *LNCS*, 1807:316–334.

Curtmola, R., Garay, J., Kamara, S., and Ostrovsky, R. (2006). Searchable symmetric encryption: Improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, page 79–88, New York, NY, USA. ACM.

Goh, E.-J. (2003). Secure indexes. Cryptology ePrint Archive, Paper 2003/216.

Iwamura, K., Kamal, A. A. A. M., and Inamura, M. (2022a). TTP-aided secure computation using secret sharing with only one computing server. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '22, page 1243–1245, New York, NY, USA. ACM.

Iwamura, K., Kamal, A. A. A. M., and Kuroi, D. (2022b). Efficient secret sharing-based partial matching search using error correction code. In Arai, K., editor, *Proceedings of the Future Technologies Conference (FTC) 2021, Volume 3*, pages 65–81, Cham. Springer International Publishing.

Kamal, A. A. A. M. and Iwamura, K. (2017). Conditionally secure multiparty computation using secret sharing scheme for n < 2k-1 (short paper). In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 225–2255.

Kamal, A. A. A. M. and Iwamura, K. (2021). Searchable encryption using secret sharing scheme that realizes direct search of encrypted documents and disjunctive search of multiple keywords. *Journal of Information Security and Applications*, 59:102824.

Kamal, A. A. A. M., Iwamura, K., and Kang, H. (2017). Searchable encryption of image based on secret sharing scheme. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1495–1503.

Kiayias, A., Oksuz, O., Russell, A., Tang, Q., and Wang, B. (2016). Efficient encrypted keyword search for multi-user data sharing. *LNCS*, 9878 LNCS:173–195. SE multi-user with TTP.

Shamir, A. (1979). How to share a secret. *Communications of the ACM*, 22:612–613.

Song, D. X., Wagner, D., and Perrig, A. (2000). Practical techniques for searches on encrypted data. In *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, pages 44–55.

Wang, Y., Wang, J., and Chen, X. (2017). Secure searchable encryption: a survey. *Journal of Communications and Information Networks*, 1(4):52–65.

Xu, P., Jin, H., Wu, Q., and Wang, W. (2013). Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack. *IEEE Transactions on Computers*, 62(11):2266–2277.