Block Chain based Searchable Symmetric Encryption

Huige Li\textsuperscript{a,b}, Haibo Tian\textsuperscript{a,b}, Fangguo Zhang\textsuperscript{a,b,*}

\textsuperscript{a}School of Data and Computer Science, Sun Yat-Sen University, Guangzhou 510006, China
\textsuperscript{b}Guangdong Key Laboratory of Information Security, Guangzhou 510006, China

Abstract

The mechanism for traditional Searchable Symmetric Encryption is pay-then-use. That is to say, if a user wants to search some documents that contain special keywords, he needs to pay to the server firstly, then he can enjoy search service. Under this situation, these kinds of things will happen: After the user paying the service fees, the server may either disappear because of the poor management or returning nothing. As a result, the money that the user paid cannot be brought back quickly. Another case is that the server may return incorrect document sets to the user in order to save his own cost. Once such events happen, it needs the arbitration institution to mediate which will cost a long time. Besides, to settle the disputes the user has to pay to the arbitration institution. Ideally, we deeply hope that when the user realizes the server has a tendency to cheat in the task of searching, he can immediately and automatically withdraw his money to safeguard his right. However, the existing SSE protocols cannot satisfy this demand.

To solve this dilemma, we find a compromised method by introducing the block chain into SSE. Our scheme achieves three goals stated below. Firstly, when the server does not return anything to user after he gets the search token, the user can get some compensation from the server, because the server can infer some important information from the Index and this token. Besides, the user also doesn’t pay the service charge. Secondly, if the documents that the server returns are false, the server cannot receive service fees, meanwhile, he will be punished. Lastly, when the user receives some bitcoin from server at the beginning, he may terminate the protocol. Under this situation, the server is a victim. In order to prevent such thing from happening, the server will broadcast a transaction to redeem his pledge after an appointed time.

\*Corresponding author

Email addresses: tianhb@mail.sysu.edu.cn (Haibo Tian), isszhfg@mail.sysu.edu.cn (Fangguo Zhang)

Preprint submitted to Elsevier

April 26, 2017
1. Introduction

Searchable Symmetric Encryption was firstly proposed by Song et al. [1]. In cloud computing environment, each user realizes that their own local storage spaces are limited. If the data are stored in local devices, it will also require a lot of manpower and financial resources to manage them. Instead, the cloud server can provide convenient storage service for users and it only charges a small service fee. Moreover, users see that when uploading their data to the cloud server, they can conveniently get access to their data at any device. So more and more users prefer to store their data on the server.

The data usually contain some sensitive messages, which often cause more attention of Hacker. In recent years, there are many events of leakage about users’ privacy documents, which appeals us to encrypt users’ data before uploading. To the ciphertexts, if there do not have auxiliary messages, the search efficiency will be low. That is to say, when search, the user has to download all the ciphertexts and decrypt them on the local devices which is an awkward idea. Alternatively, the data owner can authorize the secret key to the server and let the server only return the data that the user needs. However, if the server is not honest, it will also leak the data’s privacy. Therefore, how to implement efficient and secure search on the ciphertexts is a challenging work.

Song et al. [1] firstly proposed a searchable encryption (SE, abbreviation) protocol. In the SE model, it involves three main parties in the framework of searchable symmetric encryption: The data owner, the server and the user. The data owner has \( n \) documents \( D_1, D_2, \ldots, D_n \) that need to upload onto the server. Firstly, the data owner encrypts these documents into ciphertexts \( C_1, C_2, \ldots, C_n \). In their scheme, it mainly uses two-layered encryption. Because it is a sequential scan, the search efficiency is \( O(n) \) where \( n \) denotes the stream cipher and block cipher operations.

Undoubtedly, a stronger privacy guarantee for searchable symmetric encryption can be achieved using oblivious RAMs [2], which needs multiple interactions between the server and the user. Therefore, the search efficiency is low which is not practical. Song et al. [1] pointed that to improve the search efficiency, it is feasible to appropriately weaken the privacy security level. They thought the security on the searchable encryption should satisfy two demands:

1. The server cannot learn anything about the plain documents when it only gets the ciphertexts.
2. When the server executes search algorithm, it also cannot learn
anything about the plain documents and the search plain keyword except the search results.

Goh et al.\[3\] constructed an auxiliary information (namely, Index) to optimize the search complexity. In their scheme, they used Bloom Filter which brought wrong result, so their search result may contain wrong result. They also firstly gave the security definition for the SSE, which is called IND-CKA. But this definition is not right completely, because it can construct an insecure SSE scheme.

Curtmola et al.\[4\] used data structure to construct document Index, their scheme reduced the search time into $O(D(w))$, where $O(D(w))$ denotes the number of documents that contain keyword $w$. Their scheme supports exact search. They also redefined the security of SSE: IND-CKA1 and IND-CKA2. The following work all followed these principles. From their scheme, we can obtain the general model for the SSE. That is to say, in order to facilitate effective search operation on the ciphertexts in the future, the data owner usually constructs an Index according to the documents. Then the data owner uploads the ciphertexts and Index to the server. When a user wants to search documents that contain keywords $w_1, w_2, ..., w_k$, firstly, he/she needs to communicate with the data owner to generate corresponding search tokens. Then he/she sends these tokens to the server. When the server receives them, he uses the tokens to find those corresponding document identities. The server returns the documents to the user that identities point to.

Kamara et al. proposed a parallel search scheme \[5\]. When searching for a keyword $w$, it executes in $o(r)$ parallel time, where $r$ is the number of documents containing keyword $w$. Cash et al. proposed the first sub-linear scheme \[6\] which achieves optimal search time. The search function form expands the exact single keyword \[3, 4, 1\] into complex keywords form \[7, 8, 9, 10, 11\].

Stefanov et al. firstly considered the forward privacy for dynamic SSE scheme \[12\], which used hierarchical structure. Bost et al. found the scheme \[12\] not secure, and then they gave an improved schemes \[13, 14\]. Bösch et al. made a whole survey for SSE protocols \[15\].

Alderman et al. proposed a SSE scheme supporting multi-level access policy \[16\]. Like the \[4\], it also used the list, but it is different from that in \[4\]. That is to say, the items in the list for the same keyword $w$ are ordered according to the access level. The document with the highest classification will be placed at the front, and those with the lowest classification is at the end. When search, if the user has low permission, he only get the search token which has low classification. This search token points to the element in the back of the list. However, their construction cannot guarantee the privacy of the users. That is to say, after the two users search for the same keyword, the server will deduce who has higher permission.
Most jobs are against honest but curious adversary [1, 11, 6]. It also exists other adversary, such as the malicious adversary. The malicious adversary either does not execute the search task honestly or arbitrary changes the search result. In order to resist the malicious adversary, Kurosawa et al. firstly gave an efficient solution under the universal composability framework [17], which uses message authentication code in the Index. Cheng et al. designed a protocol that also can against the malicious adversary by using the indistinguishability obfuscation [18], and their scheme also can resist the malicious user. Other works that resist malicious adversary are [13, 12, 14].

**Bitcoin** is an emerging virtual digital currency generated in peer-to-peer (P2P) network. It was firstly proposed by Satoshi Nakamoto [19] in 2008. He generated the first bucket of Bitcoin in 2009 and released the open source software of Bitcoin. The publish of Bitcoin does not depend on the trusted entity. That is to say, anybody in Bitcoin system may issue a certain amount of Bitcoin as long as he mines a right nonce. It is a purely decentralized system. It uses the POW to confirm a transaction. According to the original assumption, there are only 21 million Bitcoin which will entirely come into the market in 2040. In Bitcoin system, it requires that the majority of nodes in peer-to-peer network are honest. For the transaction in P2P network system, it will be accepted if it is verified by at least 6 nodes. Once the transaction is admitted, it would not be reversible.

Because the Bitcoin is decentralized and irreversible, there has a boom of research of Bitcoin mechanism [20, 21, 22, 23, 24]. Andrychowicz et al. and Bentov et al. respectively introduced the Bitcoin into multiparty computations [25, 26, 27] to resolve the fairness problem.

The block chain is a basic low-layer technology in Bitcoin, and it can be used to construct other forms of cryptography currency. Each people also can define some functionality on it, which is called as smart contract. The smart contract is firstly appeared in [28]. The smart contract can be used to maintain intellectual property, solve the fairness and so on. In fact, the protocol proposed by Andrychowicz can be seen as a smart contract, because it introduces a commitment algorithm $h(x)$ in out-script of a transaction, which is not permitted in Bitcoin system. In the block chain, everyone can spent the money. In order to understand easily, we will use $B$ to represent the form of currency in such block chain that can support smart contract.

In fact, the fairness problem in searchable symmetric encryption still exists. When the adversary is malicious, he can arbitrarily change the result or does not execute the protocol at all. However, the existing schemes [17, 18] only consider the former case. If the server still does not return anything after a fixed time, in order to redeem the service fee, the user has to ask for the court to make an arbitration. This solution process will cost a long time and the user has to pay to the arbitration institution to
settle this dispute. In addition, once the server got the search token, he can deduce some information from the search token, Index and ciphertext which breaches the data’s privacy. Ideally, in order to safeguard the data’s privacy, the user deeply hopes that search token that server has got will become insignificance. This means that data stored on the server should update regularly, which is very intractable at present. As an alternative way, we can force the server to pay a fine if he does not return anything to the user. At the same time, the user can withdraw her service fee as soon as possible. However, the existing schemes can not achieve such goal.

By introducing the block chain, we can easily solve the fairness of SSE. Namely, we can use the following mechanism: Once the server does not follow the protocol honestly, it should suffer financial penalties to decrease its credibility, while, the victim (means the user) can redeem the money automatically that the user pledges. Concretely speaking, the user firstly sends a value $T_w$ to the server, where the $T_w$ denotes an encryption of the search token corresponding to the keyword $w$. The user then broadcasts a transaction $T_0$ of value $dB$, which embeds a smart contract about judgment algorithm. The server establishes transaction $T_1$ to get the decryption key for the $T_w$. After the $T_1$ appears on the block chain, the server can decrypt the search results from the Index $I$. To redeem the ransom that the user pledges in $T_0$, the user has to construct transaction $T_2$ to get these results. If the transaction $T_3$ is accepted which is connected with $T_0$, the user will redeem his pledge, otherwise, the server will get this money. If they honestly execute the protocol, each one can get their things respectively. That is to say, the user gets the search result, while the server can charge the service fees. If anyone does not honestly executes the protocol, he will be punished.

**Our contribution:** Through above analysis, we want to realize three goals which are listed as follows:

- After the server gets the search token, if he does not return any thing to user, the user will get some compensation from the server to safeguard his right. Because the server can infer some important information from the Index and this token.

- If the documents returned from the server are wrong, we wants to achieve two goals. Firstly, the server does not get service fee. Secondly, the server will also lose some money which can be seen as a punishment for his credibility.

- Bitcoin is a virtual currency, in the face of it the user will also be not honest. It is necessary to consider the honesty of user. In our scheme, if the user firstly terminates the search protocol after he gets Bitcoin from the server, the server also can redeem his money naturally.
Organization. The remainder of this paper is organized as follows. In Section 2, we review some preliminaries that will be used in our construction. Then we propose model of block chain-based SSE and its security definition. In Section 4 we present our concrete block chain-SSE scheme. Next we give the analysis of performance and security for our scheme. The last Section is conclusion.

2. Preliminaries

In this section, we review the Bitcoin currency system and searchable symmetric encryption model that we need to use.

2.1. Searchable Symmetric Encryption

There are three participants in traditional Searchable Symmetric Encryption: The data owner, the server and the user. As shown in figure 1, suppose that the data owner has \( n \) documents which denotes \( D_1, D_2, ..., D_n \). Before uploading them to the server, he makes a preprocess: He needs to encrypt these documents into ciphertexts \( C = (C_1, C_2, ..., C_n) \), and generates a corresponding Index \( I \). Then he sends \( C, I \) to the server. Suppose that the data owner and the user share the private key \( k \). The user wants to search documents that contain keyword \( w \). He computes search token \( t_w \) for keyword \( w \) and sends it to the server. The server finds the results \( C_{ij} \) by combining \( t_w, C, \) and \( I \). Lastly, the user decrypts \( C_{ij} \) locally.

A searchable symmetric encryption is secure if the following properties hold:

- The server cannot learn anything about the plain documents when it only gets the ciphertexts.

- When the server executes search algorithm, it also can not learn anything about the plain documents and the search plain keyword except the search results.

2.2. Bitcoin currency system

Because the block chain is a lower-level technology in Bitcoin, we will review the bitcoin. A Bitcoin system consists of addresses and transactions between them. The address usually means that a hash value of the user’s public key. Each user in each transaction can have a pair of key: private key and the corresponding public key [26]. The public key is used to verify whether the signature \( \sigma \) for the transaction is valid, while the private key is used to sign the transaction. For brevity, we will use the capital letter (e.g. \( A \)) to denote the pair key \((A.pk, A.sk)\). Let \( \sigma = \text{sig}_A(m) \) denote the signature of the message \( m \) with respect to the \( A.sk \) and \( \text{ver}_A(m, \sigma) \) be the result of the verification by using \( A.pk \).
In the real Bitcoin system, each transaction can have multiple inputs, but only has two outputs at most. It describes a circulation of an amount $v$, that is to say, it is transferred from an address $A.pk$ to another address $B.pk$. A transaction can be denote as $T_x = (((y_1, a_1, \sigma_1), \ldots, (y_l, a_l, \sigma_l), (v_1, \pi_1), \ldots, (v_l, \pi_l), t)$, where $y_i$ is a hash value of previous transaction $T_{y_i}$, $a_i$ is an index of the output of $T_{y_i}$ and $\sigma_i$ is the input-script. This script is written in Bitcoin scripting language, it is a stack based language [2]. The $(v_1, \pi_1), \ldots, (v_l, \pi_l)$ can be seen as the outputs of $T_x$, where the $\pi_i$ is output-script which is also written in Bitcoin scripting language. Each transaction can have a time lock $t$, it means that this transaction is valid only after $t$ time. The $(y_1, a_1), \ldots, (y_l, a_l), (v_1, \pi_1), \ldots, (v_l, \pi_l), t)$ is called the body of $T_x$ which we denote $[T_x]$. The script usually is used to define how the transaction can be redeemed. A transaction is valid if it satisfies that: (1) The official time is reached. (2) Each evaluation of $\pi_i([T_x], \sigma_i)(1 \leq i \leq l)$ is true. (3) The involved transactions are not redeemed. (4) It needs at least 6 nodes in Bitcoin system to verify. When a transaction is finished, it will be collected by one node in the Block Chain. For example, as shown in figure 2, it is a transaction $T_x = (y_1, \phi_x, v, t, \sigma_1)$. The input script in this transaction is a signature, and the output script is a verification algorithm. We can call it a standard transaction, and the address against which the verification is done will be called the recipient of a transaction.
3. Our System Model

In this section, we give our definition of block chain-based SSE and its security model.

3.1. Definition for block chain-based SSE

As shown in figure 3, in the block chain-based searchable symmetric encryption system, there mainly have four participants: The data owner, the server, the user and the miners. The data owner has \( n \) documents. Let \( D = (D_1, D_2, ..., D_n) \) denote these documents. The server’s duty is managing these documents and executing search task for user. Suppose that the data owner and the user share the private key \( k_1 \). The miners will verify whether the transaction is accepted or not.

In phase 2, the user computes search token \( t_w \), then encrypts it into \( T_w = \text{Enc}_k(t_w) \). The user sends \( T_w = \text{Enc}_k(t_w) \) to the server, at the same time, the user will construct transaction \( T_1 \), and send transactions 0, 01 with his signature to the server, the server then puts his own signature in these transaction 0, 01. Here each transaction of 0 and 01 embeds a stipulated time, after the appointed time, the user or the server can get the money that in \( T_1 \). The 0, 01 will broadcast after the phase 3.

In phase 3, the server constructs transaction 2 and transaction 4. He firstly broadcasts the transaction 2. The user will use the transaction 3 to get the money originated from transaction 2. Here the information \( k \) is embedded in transaction 3. If the transaction 3 is accepted by miners, the user can get the \( d_2 \), or the server will redeem the money by transaction 4. Here the transaction embeds the signatures of the server and the user.
Next, to redeem the money stemmed from the transaction $T_1$, the user will broadcast the transaction 5 of value $(d_2 + d)B$. The server can get these money by transaction 6, as long as he can return right search result, which is embedded in 6. If transaction 6 does not appear in the block chain, the user can broadcast transaction 7 to redeem these money until time $t$, in which the signatures of the server and the user are embedded.

If the above steps finish, the user can broadcast transaction 0 to redeem the money originated from transaction 1. If within time $t_0$, transaction 0 does not appear on the block chain, the server can get the money by 01. Here, transactions 0, 01 all embed the signatures of the server and the user.

The search keyword can be single or multiple, here we only consider the single keyword case, the multiple keywords case is similar to the single one. The definition of our block chain-based SSE model is as follows:

**Definition 1** *(Block chain-based Searchable Symmetric Encryption)*. *A block chain-based SSE model* is a protocol that allows the extraction of information from encrypted data without the need to decrypt the data. The model is based on the block chain protocol, which provides a decentralized, secure, and transparent way to store and access data.
chain-based searchable symmetric encryption is a tuple of five polynomial time algorithms \( \text{SSE} = (\text{Gen}, \text{Enc}, \text{Srchtoken}, \text{Search}, \text{Verify}, \text{Red}, \text{Dec}) \) such that:

- \( K \leftarrow \text{Gen}(1^k) \) : is a probabilistic algorithm that takes a security parameter \( k \) and outputs an array key \( K \).

- \( (I, C) \leftarrow \text{Enc}(K, D, W) \) : takes as input the the secret key \( K \), the data file collection \( D \) and the keyword collection \( W \), the data owner outputs the ciphertexts \( C \) and the invertible Index \( I \).

- \( (\text{Appoint}, \text{compesa}, \text{Redeem}, T_w) \leftarrow \text{Srchtoken}(w, K, dB) \) : is a deterministic algorithm. The user takes as input the search keyword \( w \), the private key \( K \) and an unredeemed transaction of value \( dB \), outputs \( T_w \), the transactions \( \text{Appoint}, \text{compesa} \) and \( \text{Redeem} \). The transactions \( \text{Appoint} \) embeds a smart contract about judgement algorithm \( D() \). While, \( \text{Redeem} \) needs the user to provide the addresses that use in the following transactions happening between the user and the server. The \( \text{Appoint} \) will firstly appear in the block chain, the \( \text{compesa} \) and \( \text{Redeem} \) will appear in the block chain after a long time.

- \( (\text{ask}, \text{Pay}, \text{withdraw}) \leftarrow \text{Search}(T_w, dB, K, w) \): The server and the user put their respective signatures into the transaction \( \text{withdraw} \). The server takes \( T_w \), his signature and an unredeemed transaction of value \( dB \) as input and outputs transaction \( \text{ask} \). The user and the server wait until transaction \( \text{ask} \) appeared in the block chain. If the user wants to claim \( dB \), he needs to embed the secret key \( K \) into transaction \( \text{pay} \), then he broadcasts it. If transaction \( \text{pay} \) does not appear on the ledger after a fixed time, the server will publish transaction \( \text{withdraw} \).

- \( (\text{Get}, \text{Prove}, \text{Fuse}) \leftarrow \text{Verify}(C_w, \text{Mac}_w, K, d + d_1B) \) : The user takes \( K \) as input. The user and the server compute the body of transaction \( \text{Fuse} \) and sign this transaction respectively. The user needs to pledge \( d + d_1B \) and then broadcasts transactions \( \text{Get} \). If the server wants to claim \( d + d_1B \), he needs to construct transaction \( \text{Prove} \) which embeds \( (D(w), \text{MAC}(D(w))), C_{ij}, \text{MAC}(C_{ij}) \). Then the server broadcasts the transaction \( \text{Prove} \). Let \( D(w) \) denote the set that contains keyword \( w \). \( \text{MAC}(w), \text{MAC}(C_{ij}) \) are message authentication codes about \( D(w) \) and \( C_{ij} \) respectively, \( C_{ij} \) is the documents that should return to user. If transaction \( \text{Prove} \) does not appear on the ledger after a fixed time, the user will publish transaction \( \text{Fuse} \).

- \( \text{compesa}/\text{Redeem} \leftarrow \text{Red}(id_u, id_s, \text{ask}, \text{Pay}/\text{withdraw}, \text{Get}, \text{Prove}/\text{Fuse}) \): The user needs to provide the addresses used in transactions \( \text{ask}, \text{Pay}/\text{withdraw}, \text{Get}, \)
Prove/Fuse) in the Redeem, the miners will verify whether it it right, if it appears on the block chain, the user can redeem the money. Otherwise, the server will get the money.

- $D_{ij} \leftarrow \text{Dec}(C_w, K_1)$: The users takes the private key $K_1$, the received document set $C_{ij}$ as input, and outputs the plain document set $D_{ij}$.

### 3.2. Security Definition

Our security definition mainly adopts the real/ideal simulation paradigm stated in [4]. Many of the SSE jobs follow this security definition.

**Definition 2:** If a block chain-based SSE protocol is secure, the adversary should not distinguish the real game $\text{Real}^{\Pi}_{\mathcal{A}}(k)$ and the simulation game $\text{Ideal}^{\Pi}_{\mathcal{A}, \mathcal{B}, \mathcal{S}}(k)$ as mentioned in the following.

Let $\Pi = (\text{Gen}, \text{Enc}, \text{Srchtoken}, \text{Search}, \text{Verify}, \text{Red}, \text{Dec})$ be a block chain based SSE scheme, $k$ be the security parameter. Let $\mathcal{A}$ denote the adversary, $\mathcal{S}$ denote a simulator, $\mathcal{B}$ denote the environment that he can simulate Bitcoin system and $\mathcal{C}$ be a Challenger.

$\text{Real}^{\Pi}_{\mathcal{A}}(k)$: Challenger $\mathcal{C}$ runs $K \leftarrow \text{Gen}(1^k)$ to get key array $K$ and gives signature verification public key to the Adversary. The adversary $\mathcal{A}(1^k)$ randomly chooses data files collection $D = \{D_1, D_2, ..., D_n\}$ and then give them to the challenger $\mathcal{C}$. Then challenger $\mathcal{C}$ runs $(I; C) \leftarrow \text{Enc}(K; D)$ and then sends $(I; C)$ to $\mathcal{A}$. The adversary $\mathcal{A}$ makes polynomial number of queries by choosing different keywords $w_i$, the challenger $\mathcal{C}$ returns $(\text{Appoint}, \text{compesa}, \text{Redeem}, T_w)$ to the adversary. The adversary asks the $\mathcal{C}$ to query $\text{ask}, \text{Pay/withdraw}, \text{Get}, \text{Prove/Fuse}, \text{compesa}/\text{Redeem}$ and $D_{ij}$ Finally, $\mathcal{A}$ returns a bit $b$ that is output by the experiment.

$\text{Ideal}^{\Pi}_{\mathcal{A}}(k)$: Adversary $\mathcal{A}$ randomly chooses $D \leftarrow \{0, 1\}^*$ such that $|D_{\text{Ideal}}| = |D_{\text{Real}}|$. The simulator $\mathcal{S}$ outputs $(I; C) \leftarrow S(L(D))$ and then sends $(I; C)$ to $\mathcal{A}$. The adversary $\mathcal{A}$ makes polynomial number of queries by choosing different keywords $w_i$, the challenger $\mathcal{C}$ responds $(\text{Appoint}, \text{compesa}, \text{Redeem}, T_w)$ $\text{ask}, \text{Pay/withdraw}, \text{Get}, \text{Prove/Fuse}, \text{compesa}/\text{Redeem}$ and $D_{ij} \leftarrow S(L(D))$ in $\mathcal{B}$ environment. Finally, $\mathcal{A}$ returns a bit $b$ that is output by the experiment.

We say that $\Pi$ is semantically secure if for all the PPT $\mathcal{A}$, there exists a PPT $\mathcal{S}$ such that for all polynomial size distinguisher $\mathcal{D}$,

$$|\Pr[\mathcal{D}(\text{view}_{\text{real}}) = 1] - \Pr[\mathcal{D}(\text{view}_{\text{ideal}}) = 1]| \leq \text{negl}(\lambda).$$
Remark: In the above experiments, \( L(D) \) is the trace related to \( D \). The trace includes the search pattern, access pattern, the length of the encrypted documents and the number of documents in \( D \) which are defined in [4].

Because we introduce Block chain technology, it is necessary to define fairness for our scheme.

**Definition 3**: The Block chain-based SSE scheme satisfies fairness if the following properties hold:

- If the server dose not honestly execute the search protocol, he cannot receive the service fee coming from the user. He will lose some money simultaneously which can be seen as a punishment for his credibility.

- If the user firstly terminates the search protocol after he gets Bitcoin from the server, the server can use transaction to redeem his money.

4. The detailed scheme

In the block chain-based SSE scheme, it mainly have four participants: the data owner, the server, user and the miners. The data owner has \( n \) documents \( D_1, D_2, ..., D_n \) which need to upload onto the server. Before uploading, the data owner chooses some secure function. Let \( \varepsilon \) be an IND-CPA secure symmetric encryption scheme, \( H_1 : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^k \), \( H_2 : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^{2k} \) be secure pseudorandom functions. The \( H : \{0,1\}^k \rightarrow \{0,1\}^k \) is a collision resistant hash function. The \( k \) denote the security parameter. The algorithm for each step is as follows:

- **Gen**: The data owner takes a security parameter \( k \) as input. He outputs a secret key array \( K = (K_1, K_2, K_3, K_4, K_5) \), where \( (K_1, K_2, K_3, K_4, K_5) \leftarrow \{0,1\}^k \). \( K_1 \) is used to encrypt the data documents, \( K_2 \) is used to generate the search token \( t_w \) for the keyword \( w \). \( K_3 \) is used to generate the Index, \( K_4 \) is used to encrypt \( t_w \). \( K_5 \) is used to make a MAC for \( C_i \). Each participant needs to generate a signature key pair \( U_i(pk, sk) \) which will be used in the transaction.

- **Enc**: In this phase, there are two steps.
  - Firstly, the data owner uses the private key \( K_1 \) to encrypt these documents collection \( D = \{D_1, D_2, ..., D_n\} \):
    
    \[
    C_i = \varepsilon_{K_1}(D_i)(1 \leq i \leq n),
    \]
    \[
    MAC_{C_i} \leftarrow H_2(K_5, C_i)
    \]
then set
\[ C \leftarrow ((C_1, MAC_{C_1}), (C_2, MAC_{C_2}), \ldots, (C_n, MAC_{C_n})) \]

- The data owner will generate an Index to optimize the search time complexity for future search. The data owner firstly extracts the keywords collection \( W = \{w_1, w_2, \ldots, w_m\} \) from the \( D \). Let \( DB(\cdot) \) denote a \( n \)-binary array. For each keyword \( w \in W \), he computes \( DB(w) \): If \( i \)-th document contains keyword \( w \), then sets \( DB(w)[i] = 1 \), otherwise, it will be 0. He also computes:

\[
\begin{align*}
& t_w \leftarrow H_1(K_2, w \parallel 0), k_{21} \leftarrow H_1(K_2, w \parallel 1) \\
& e_w \leftarrow Enc(k_{21}, DB(w)), K_w = H_1(K_3, w) \\
& Mac_w \leftarrow H_1(K_w, DB(w))
\end{align*}
\]

where \( Enc \) is a deterministic symmetric encryption, \( Mac_w \) is a message authentication code for \( DB(w) \).

The data owner puts \((t_w, e_w, Mac_w)\) into \( I \) according to the form of the dictionary order. At last, he sends \((C, I)\) to the server.

- **Srchtoken** : Suppose that the user is authorised from the data owner, then he can get the secret key \((K_1, K_2, K_3, K_4, K_5)\). To search documents that contain the keyword \( w \), he has to compute that:

\[
\begin{align*}
& t_w \leftarrow H_1(K_2, w \parallel 0), k_{21} \leftarrow H_1(K_2, w \parallel 1) \\
& k_{31} \leftarrow H_1(K_4, w), T_w \leftarrow Enc(k_{31}, t_w, k_{21}, H(k_{31}))
\end{align*}
\]

He sends \( T_w \) to the server, and waits for the server to reply.

- Before the next step beginning, the user and the server should negotiate the following transactions. The user computes the body of the transaction \textit{Appoint} using the \( T_u \) as input whose value is \( dB \). The user and the server all compute the body of the transactions \textit{Redeem} and \textit{compesa}. The user signs \textit{Redeem} and \textit{compesa}, then send them to the server. The server also signs them. The user broadcasts the transaction \textit{Appoint}. Next, they will wait until the transaction \textit{Appoint} is included in the block chain. Here, the time \( t_{\text{max}1} \) and \( t_{\text{max}2} \) satisfy that \( t_{\text{max}1} > t_{\text{max}2} > t_1 > t \) where the \( t_1, t \) are defined in the below.

The algorithm \( D(\cdot) \) is a judgment statement. The user inputs his address and the server’s address which are used in the phase of \textit{Search} and \textit{Verify}, and
this algorithm can automatically count the number of the transactions that initiated by the user. And the receiver is the server in these transactions. If the number is greater than 1, the user will redeem his money by the transaction Redeem. Otherwise, the server will obtain $d_B$ by transaction compesa.

If the transaction Appoint does not appear on the block chain until $t_{max1} - t_{max0} < t$, then the user will quickly redeem the transaction $T_u$. Here the $t_{max0}$ means that the maximal possible delay that broadcasts the Appoint. The concrete process is shown in figure 4.

![Figure 4: Appointment Process](image)

- **Search**: When the server has got the $T_w$, if he wants to help user to search, he has to performs the steps stated below to get the decryption key $t_w, k_{21}$:

  Suppose that the user has negotiated with the server such stipulation: In each transaction, the money they deposit is linear to the information that the search
token contains. That is to say, if the search token can deduce large information, the guarantee deposit will be big. We will use $d_B$ to denote this deposit.

Before the transaction beginning, the server needs to make an agreement to the content of smart contract with the user, which we denote a public verification algorithm $\mathcal{V}(\cdot, \cdot) : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{1, \bot\}$.

- Suppose that the server has an unredeemed transaction $T_{s1}$ of value $d_B$. The server constructs transactions $ask, withdraw$. For the $ask$ it satisfies that:

1.) It contains his signature of the body of transaction $ask$.

2.) The out-script contains a public verify algorithm $\mathcal{V}(T_w, k_{31})$.

$\mathcal{V}(T_w, k_{31})$ means that the miners firstly decrypt the $T_w$ into $(\tilde{t}_w, \tilde{k}_{12}, H(\tilde{k}_{31}))$ by using the $k_{31}$, and then verify whether $H(\tilde{k}_{31}) = H(k_{31})$. If this equation does not hold, the miners will accept transaction $Pay$, otherwise they will accept transaction $withdraw$.

- The server signs the transaction $withdraw$ using the body of transaction $withdraw$, and sends it to the user. The user also signs it and returns it to the server. The server retains it.

- The server signs the transaction $ask$ and broadcasts it. Then they wait until the transaction $ask$ is included into the block chain.

- If the transaction $ask$ does not appear on the block chain until $t - max_1$, where the $max_1$ means the maximal possible delay of including $ask$ into block chain, the server can immediately redeem the $T_{s1}$ and quits the protocol.

- The user broadcasts the transaction $Pay$ to the user which embeds his signature. The user needs to put $k_{31}$ in its in-script, then the user publishes it on the nodes.

- When the miners see transaction $Pay$, he firstly uses $k_{31}$ to decrypt $T_w$ into $t_w, k_{21}, H(k_{31})$. If $H(k_{31})$ is right, the miner will accept transaction $Pay$, otherwise they will wait $withdraw$.

- If the transaction $Pay$ does not appear on the block chain until $t - 2max_2$ time, the server will broadcast the transaction $withdraw$ and gets his money. Here the $max_2$ means the maximal possible delay between broadcasting the transaction $pay$ and including it into the block chain. The concrete process is shown in figure 5.
**Verify**: In order to reclaim $dB$ and get $d_1B$ service fee from the user, the server has to return the corresponding documents to the user. So he does:

- The server uses the $k_{21}$ to decrypt the $DB(w)$, i.e., $DB(w) \leftarrow Dec(k_{21}, e_w)$. If $DB(w)[j] = 1$, he puts document $C_j, MAC(C_j)$ into array $C_w$. The server waits the user to issue a transaction (Here, we assume that the length of each $C_j$ does not beyond the appointed range, if it does, we can make it into some blocks and the length of each block is lower than the specified length which is permitted by a transaction).

- The user constructs transaction $Fuse$. He signs the transaction $Fuse$ using the body of transaction $Fuse$, and ask the server to embed his signature into it.

- The user signs transaction $Get$ where he deposits $d_1 + dB$. It should satisfy
that:

1.) It contains user’s signature of the body of transaction Get. 2.) The out-script contains a public verifying algorithm \( V_1(\cdot) \). This verifying algorithm should inputs \( DB(w), Mac_w, K_w, K_5, C_w, Mac(C_j) \). Concretely, the user needs to provide \( K_w, K_5 \), the server provides \( DB(w), Mac_w, C_w, Mac(C_j) \). The \( K_w, K_5 \) have be embedded in algorithm \( V_1(\cdot) \) and each one cannot modify it, the \( DB(w), Mac_w, C_w, Mac(C_j) \) can be obtained in transaction \( \text{Prove} \). In the future, the miners will verify the equations:

\[
Mac'_w \leftarrow H_1(K_w, DB(w)) \\
MAC'_C_i \leftarrow H_2(K_5, C_i)
\]

hold or not. If they hold, the miners will accept \( \text{Prove} \), or they will wait \( \text{Fuse} \).

- The server publish Transaction \( \text{Prove} \). It should satisfy that: a) It contains server’s signature of the body of transaction \( \text{Prove} \). b) The server has put \( DB(w), Mac_w, C_w, Mac(C_j) \) into its in-script.

- If the transaction \( \text{Get} \) is not appeared in the block chain within time \( t_1 - \max_3 < t_\max_1 \), where the \( \max_3 \) means the maximal possible delay of including \( \text{Get} \) into block chain, the user can immediately redeem the \( T_u1 \) and quits the protocol. If the transaction \( \text{Prove} \) is not appeared in the block chain within time \( t_1 - 2\max_4 \), where the \( \max_4 \) means the maximal possible delay between broadcasting the transaction \( \text{Prove} \) and including it into the block chain. The user can broadcast the transaction \( \text{Fuse} \) to redeem the \( d + d_1 \). The concrete process is shown in figure 6.

• If the transaction \( \text{Redeem} \) does not appear on the block chain within time \( t_\max_1 - 2\max_5 < t_\max_2 \), the server can broadcast the transaction \( \text{compesa} \) and gets \( d \). Here, the \( \max_5 \) means the maximal possible delay between broadcasting the transaction \( \text{Redeem} \) and including it into the block chain.

• \( \text{Dec} \): When the user receives \( C_w \), he will uses private key \( K_1 \) to decrypt \( C_w \). i.e., \( D_w = \varepsilon. \text{Dec}(K_1, C_w) \).

5. Security and Performance Analysis

5.1. Security Analysis

In this section, we will give two theorems to prove that our scheme is secure and fair.
Theorem 5.1 If \( H_1, H_2 \) are pseudorandom function, \( H \) is a collision resistant hash function, and \( \varepsilon = (\text{Enc}, \text{Dec}) \) is PCPA-secure symmetric encryption scheme, then the scheme we present above is an adaptive IND-CCA2 block chain-based SSE scheme.

Proof. We need to construct a PPT simulator \( S = \{S_0, S_1, ..., S_q\} \) for the adversary such that \( A = \{A_0, A_1, ..., A_q\} \) the output of \( \text{View}_{\text{real}} \) and \( \text{View}_{\text{ideal}} \) are computationally indistinguishable. Suppose that the simulator \( S \) is given the trace of a history \( L \), then he can generate \( (I^*, C^*, t_w^*, \text{Appoint}^*, \text{compesa}^*, \text{Redeem}^*, \text{ask}^*, \text{Withdraw}^*, \text{Pay}^*, \text{Get}^*, \text{Prove}^*, \text{Fuse}^*) \) as follows:

- Simulating \( I^* \). If \( q = 0 \), the simulator can get the size of all documents from the trace of \( L \). Then he can use these information to set \( I^* \) be a random strings whose size is equal to \( |I| \). \( S \) puts the \( I^* \) in the state \( st_{\text{S}} \), and set \( C^*_1 \leftarrow \{0, 1\}^{|D_i|} \). Since the state \( st_{\text{A}} \) does not have the key \( K_2, K_3 \), he will uniformly choose \( t_w^* \leftarrow \{0, 1\}^* \) and \( \text{Mac}_{w}^* \leftarrow \{0, 1\}^* \) at random for each keyword \( w \).
If \( q \geq 1 \), the \( st_A \) firstly uniformly selects \((st_{A_0}, t^*_{w_0}) \leftarrow \{0, 1\}^* \) and \( \text{Mac}_{w_0}^* \leftarrow \{0, 1\}^* \) at random for each keyword \( w_0 \). He then chooses \( w_i^*, q \geq i \geq 1 \) through \((w^*_{i-1}, st_{A_{i-1}}) \). Then he chooses \((st_{A_i}, t^*_{w_i}) \leftarrow \{0, 1\}^* \) and \( \text{Mac}_{w_i}^* \leftarrow \{0, 1\}^* \) at random for each keyword \( w_i^* \).

Because the \( H_1 \) is undistinguishable from function \( f : \{0, 1\}^k \rightarrow \{0, 1\}^k \), we can get \( I^* \) is undistinguishable from \( I \) generated in \( \text{View}_{\text{real}} \).

Similarly, because the \( \varepsilon \) is \( \text{IND-CPA} \), the \( C^* \) is indistinguishable from a true ciphertext \( C \). Because the state \( st_{A_0} \) does not have the key \( K_5 \), the adversary does not distinguish \( \text{MAC}_{C_i^*} \) from \( \text{MAC}_{C_i} \) generated by step \( \text{Enc} \). Here the \( \text{MAC}_{C_i^*} \) is selected uniformly from \( \{0, 1\}^{2k} \) at random.

- Simulating the \( T_w^* \). Because the decryption key \( k_{31} \) will embed in transaction \( \text{Pay} \), each one can see it, so does the Adversary. It only needs to analyze the \( t^*_{w_i}, k_{21w_i} \) are undistinguishable from true \( t_w, k_{21} \) generated in \( \text{Srchtoken} \) algorithm. If \( q = 0 \), the \( S \) will select \( t^*_{w_0}, k_{21w_0} \) uniformly at random from \( \{0, 1\}^k \). If \( q \geq 1 \), simulator \( S \) will choose \( w^*_i, q \geq i \geq 1 \) by \( st_{A_{i-1}}, w^*_{i-1}, i \geq 1 \), then computes \((t^*_w, k_{21w}^*) \leftarrow \{0, 1\}^* \). Because pseudorandom function \( H_1 \) is undistinguishable from function \( f : \{0, 1\}^k \rightarrow \{0, 1\}^k \), we can get \( t^*_w, k_{21w^*} \) are undistinguishable from true \( t_w, k_{21} \).

- Simulating the transactions \( \text{Appoint}^*, \text{compesa}^*, \text{Redeem}^*, \text{ask}^*, \text{Withdraw}^*, \text{Pay}^*, \text{Get}^*, \text{Prove}^*, \text{Fuse}^* \). Indeed, these transaction embed some information, such as \( T_w, \text{MAC}_w \), if the adversary wants to get the money point to these transactions, he has to construct a new different block chain which violates the irreversibility.

\[ \text{Theorem 5.2: If the Block chain is irreversible, our scheme can satisfy Fairness for all participants.} \]

- If the user is not honest, it means that he does not construct the transaction \( \text{Get} \). The server can broadcast the transaction \( \text{compesa} \) and redeems its \( d\mathcal{B} \) pledge.

- If the server is not honest, it means that the server releases false \( DB(w), C_w \). The miners will reject transaction \( \text{Prove} \), and the user will gets the \( d_1 + d\mathcal{B} \).
5.2. Performance Analysis

In this section, we give a performance analysis of our scheme and compare it with other existing works.

Before uploading those ciphertexts onto the server, the data owner needs a pre-process in the \( Enc \) phase, which is similar to other prior works. Therefore, we will mainly focus on the computation and communication overheads in the Search phase, which can affect user experience more due to the high frequency. Table 1 shows a comparison between our scheme and other related works. Let \( n \) denote the number of total documents, \( m \) be size of the transaction and \( r \) stands for the number of the retrieving files when search a certain keyword. As shown in table 1, Kamara et al.[5] provides the fastest search time, because it adopts a parallel search method. However, it does not satisfy Fairness. That is to say, if the server returns false result and disappears after he gets the search token, the user will lose money. It is unfair to user. Our scheme can resist malicious server and malicious user. But, in order to reach the Fairness, it needs at least 6 transactions and 3 rounds communication which may delay the time that the user gets the result.

<table>
<thead>
<tr>
<th>scheme</th>
<th>Search complexity</th>
<th>Communication</th>
<th>Verifiability</th>
<th>Adversary</th>
<th>Fairness</th>
</tr>
</thead>
<tbody>
<tr>
<td>KO[17]</td>
<td>( O(D(w)) )</td>
<td>( O(r) )</td>
<td>Yes</td>
<td>malicious server</td>
<td>No</td>
</tr>
<tr>
<td>KP[5]</td>
<td>( O(r \log (n)/p) )</td>
<td>( O(1) )</td>
<td>Yes</td>
<td>honest-but -curious server</td>
<td>No</td>
</tr>
<tr>
<td>Our scheme</td>
<td>( O(D(w)) )</td>
<td>( O(6m) )</td>
<td>Yes</td>
<td>malicious server malicious user</td>
<td>Yes</td>
</tr>
</tbody>
</table>

6. Conclusion

How to implement the searchable symmetric encryption under the environment of bitcoin is very interesting, because it considers the “true” fairness. The existed protocols about \( SSE \) can resist the malicious server when adopting the \( MAC \) algorithm, but it is valid only under the hypothesis of the server is still running. Suppose that the user pays the money for the server and hopes the server can execute search service. However the server now is facing bankruptcy, when the server receives the service charge, he begins to disappear. The server does not worry about he will be accused by user because he uses a fake name. How does the user protect his right...
in the condition of shutdown of the server. Obviously, traditional methods can not resolve this dilemma.

From the point of security, the server can deduce some important information from the search token sent from user. Ideally, the information, (namely means the ciphertext and search Index) stored on the server should be up-to-date. But how to design such protocol is still intractable. Alternative way is that the server should be punished when he does nothing after he gets the search token. The bitcoin system can provide the punishment mechanism, so we firstly introduce the bitcoin to the SSE aims to solve the fairness.

In our protocol, we can see that the server will be punished by losing his pledge once he does not follow with the protocol. He can redeem his ransom when he executes it honestly. Moreover, we also consider the reliability of the user because not every one do not love the money. When the server honestly returns the set that the user needs, the user may still tells a lie which will influence the credibility of the server. In the Bitcoin, the verification is done by the miners. If the user does not lose money, he must honest execute the protocol, or he must control 51% nodes in the Bitcoin system which is impossible to achieve.

The idea we proposed only considers non-dynamic condition, it is easy to apply it to the dynamic searchable symmetric encryption, the reader can deduce similarly.

Here we only consider one server, that is to say, data owner will only choose one server to store his documents. Once the server system has crashed, the documents will lose and never come back again, we do not want to see it happens. So our next work is that design such protocol based on bitcoin: The user will choose different servers. He divides the documents into different parts and stores these parts onto different servers. When a user searches he can get all documents that he needs.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (No. 61379154 and NO. 61672550) and Natural Science Foundation of Guangdong province (No. 2015A030313133).

References


