Block Chain based Searchable Symmetric Encryption

Huige Li[a,b], Haibo Tian[a,b], Fangguo Zhang[a,b,*]

[a]*School of Data and Computer Science, Sun Yat-Sen University, Guangzhou 510006, China*
[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 any thing 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)

Keywords: Searchable Symmetric Encryption, Bitcoin, Block Chain, Fairness, Cloud-Computing

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 [4, 1, 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 blcok 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 ₿ 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 dɃ, 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 $\mathcal{I}$. 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 σ 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 $\sigma = sig_A(m)$ denote the signature of the message m with respect to the $A.sk$ and $ver_A(m, \sigma)$ be the result of the verification by using $A.pk$.

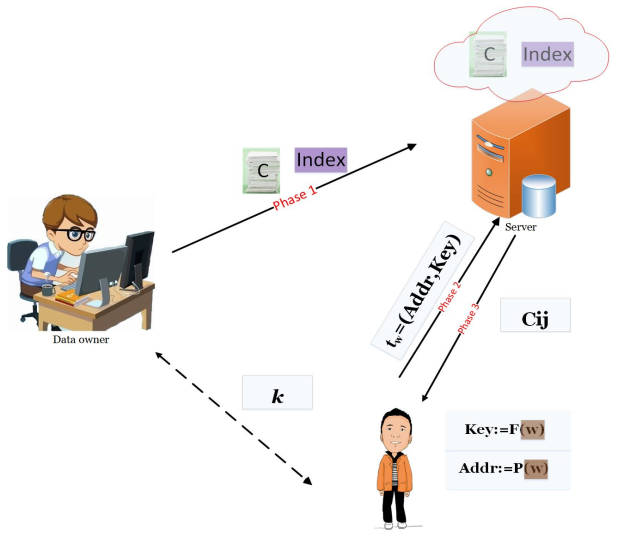


Figure 1: SSE Model

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 adress $A.pk$ to another adress $B.pk$. A transaction can be denote as $T_x = ((y_1, a_1, \sigma_1), ..., (y_l, a_l, \sigma_l), (v_1, \pi_1),..., (v_l, \pi_l), t)$, 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 σ_i is the input-script. This script is written in Bitcoin scripting language, it is a stack based language [25]. The $(v_1, \pi_1), ...(v_l, \pi_l)$ can be seen as the outputs of T_x, where the π_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 $(y_1, a_1), ..., (y_l, a_l), (v_1, \pi_1), ...(v_l, \pi_l), t)$ 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.

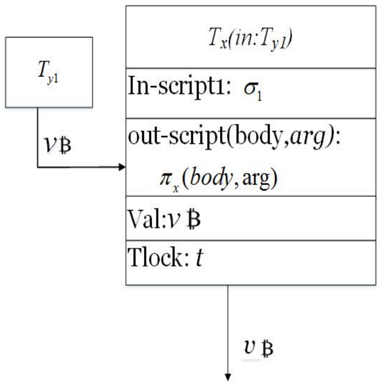


Figure 2: transaction T_x

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 $\mathbf{D} = (D_1, D_2, ..., D_n)$ 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 $T_w = Enc_k(t_w)$. The user sends $T_w = Enc_k(t_w)$ 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 $(d_2 + d)$Ƀ. 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 bby 01. Here, transactions $0, 01$ all embed the signatures of the server and the user.

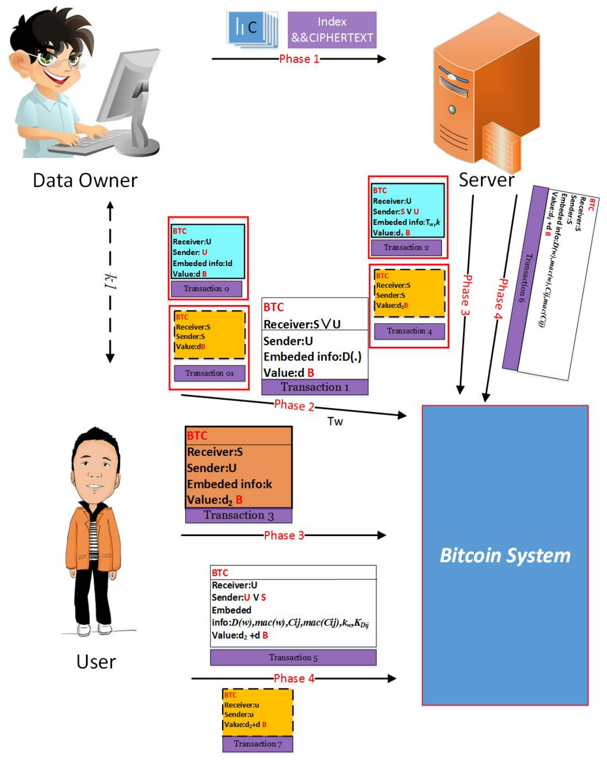


Figure 3: SSE Model

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 chian*-based *SSE* model is as follows:

Definition 1 (*Block chain*-based Searchable Symmetric Encryption). *A block* *chain-based searchable symmetric encryption is a tuple of five polynomial time algorithms* $SSE = (Gen, Enc, Srchtoken, Search, Verify, Red, Dec)$ such that:

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

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

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

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

- $(Get, Prove, Fuse) \leftarrow Verify(\mathcal{C}_w, \mathcal{M}ac_w, K, d + d_1\text{₿})$: The user takes $\boldsymbol{K}$ as input. The user and the server compute the body of transaction $Fuse$ and sign this transaction respectively. The user needs to pledge $d+d_1$₿ and then broadcasts transactions *Get*. If the server wants to claim $d + d_1$₿, he needs to construct transaction*Prove* which embeds $(D(w), MAC(D(w)), C_{ij}, MAC(C_{ij}))$. Then the server broadcasts the transaction *Prove*. Let $D(w)$ denote the set that contains keyword w. $MAC(w)$, $MAC(C_{ij})$ are message authentication codes about $D(w)$ and C_{ij} respectively, C_{ij} is the documents that should return to user. If transaction *Prove* does not appear on the ledger after a fixed time, the user will publish transaction $fuse$.

- *compesa/Redeem*$\leftarrow$ $\text{Red}(id_u, id_s, ask, Pay/withdraw, Get, Prove/Fuse)$: The user needs to provide the addresses used in transactions ask, $Pay/withdraw$, 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.

- $D_{ij} \leftarrow Dec(\mathcal{C}_w, K_1)$: 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* $Real^{\Pi}_{A}(k)$ *and the simulation game* $Ideal^{\Pi}_{A,B,S}(k)$ *as mentioned in the following.*

Let $\Pi = (Gen, Enc, Srchtoken, Search, Verify, Red, Dec)$ 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.

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

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

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

$$|Pr[\mathcal{D}(view_{real}) = 1] - Pr[\mathcal{D}(view_{ideal}) = 1]| \leq negl(\lambda).$$

Remark:In the above experiments, $L(\mathbf{D})$ is the trace related to $\mathbf{D}$. The trace includes the search pattern, access pattern, the length of the encrypted documents and the number of documents in $\mathcal{D}$ 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 ε 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 $\boldsymbol{K} = (K_1, K_2, K_3, K_4, K_5)$, 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 $\boldsymbol{D} = \{D_1, D_2, ..., D_n\}$:

$$C_i = \varepsilon_{K_1}(D_i)(1 \le i \le n),$$
$$\mathcal{MAC}_{C_i} \leftarrow H_2(K_5, C_i)$$

then set

$$\boldsymbol{C} \leftarrow ((C_1, \mathcal{MAC}_{C_1}),\ (C_2, \mathcal{MAC}_{C_2}),\ ...,(C_n, \mathcal{MAC}_{C_n}))$$

– The data owner will generate an Index to optimize the search time complexity for future search. The data owner firstly extracts the keywords collection $\mathcal{W} = \{w_1, w_2, ..., w_m\}$ from the $\boldsymbol{D}$. Let $DB(\cdot)$ denote a n-binary array. For each keyword $w \in \mathcal{W}$, 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:

$$t_w \leftarrow H_1(K_2, w \parallel 0),\ k_{21} \leftarrow H_1(K_2, w \parallel 1)$$
$$e_w \leftarrow Enc(k_{21}, DB(w)),\ K_w = H_1(K_3, w)$$
$$\mathcal{M}ac_w \leftarrow H_1(K_w, DB(w))$$

where Enc is a deterministic symmetric encryption, $\mathcal{M}ac_w$ is a message authentication code for $DB(w)$.

The data owner puts $(t_w, e_w, \mathcal{M}ac_w)$ into $\boldsymbol{I}$ according to the form of the dictionary order. At last, he sends $\boldsymbol{(C, I)}$ 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:

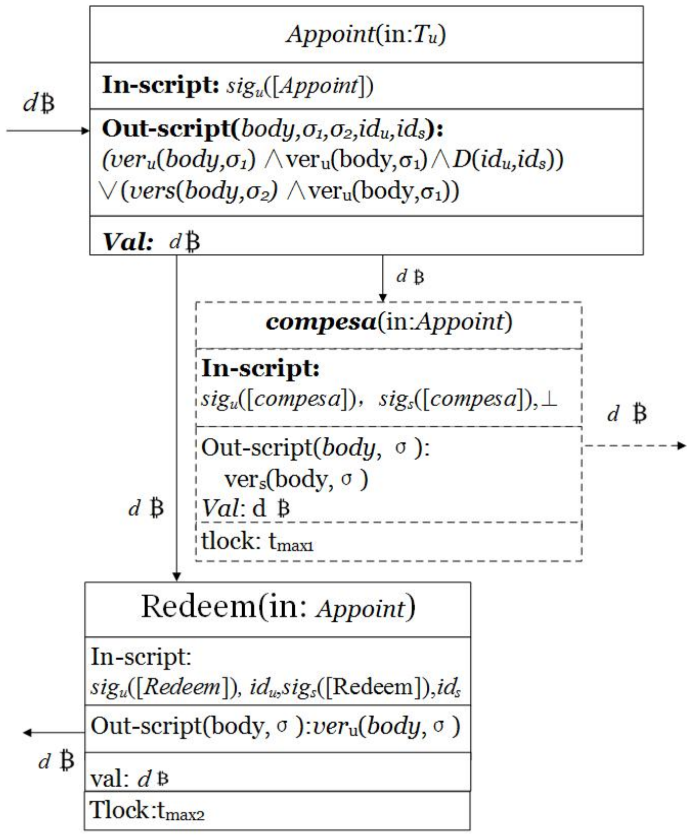
$$t_w \leftarrow H_1(K_2, w \parallel 0),\ k_{21} \leftarrow H_1(K_2, w \parallel 1)$$
$$k_{31} \leftarrow H_1(K_4, w),\ T_w \leftarrow Enc(k_{31}, t_w, k_{21}, H(k_{31}))$$

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

- Before the next step beginning, the user and the the server should negotiate the following transactions. The user computes the body of the transaction *Appoint* using the T_u as input whose value is d₿. The user and the server all compute the body of the transactions *Redeem* and *compesa*. The user signs *Redeem* and *compesa*, then send them to the server. The server also signs them. The user broadcasts the transaction *Appoint*. Next, they will wait until the transaction *Appoint* is included in the block chain. Here, the time t_{max1} and t_{max2} satisfy that $t_{max1} > t_{max2} > t_1 > t$ 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 $Search$ and $Verify$, 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 dɃ by transaction *compesa*.

If the transaction *Appoint* does not appear on the block chain until $t_{max1} - t_{max_0} < t$, then the user will quickly redeem the transaction T_u. Here the t_{max_0} means that the maximal possible delay that broadcasts the *Appoint*. The concrete process is shown in figure 4.



Appoint(in:T_u)
In-script: sig_u([Appoint])
Out-script(body,σ_1,σ_2,id_u,id_s):
$(ver_u(body,\sigma_1) \wedge ver_u(body,\sigma_1) \wedge D(id_u,id_s)) \vee (vers(body,\sigma_2) \wedge ver_u(body,\sigma_1))$
Val: d ฿
d ฿
compesa(in:Appoint)
In-script: sig_u([compesa]), sig_s([compesa]), $\perp$
Out-script(body, σ): ver_s(body, σ)
Val: d ฿
tlock: t_{max1}
Redeem(in: Appoint)
In-script: sig_u([Redeem]), id_u, sig_s([Redeem]), id_s
Out-script(body, σ): ver_u(body, σ)
val: d ฿
Tlock: t_{max2}

Figure 4: Appointment Process

- ***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฿ 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 $\mathcal{V}(\cdot,\cdot) : \{0,1\}^* \times \{0,1\}^* \rightarrow \{1,\perp\}$.

- Suppose that the server has an unredeemed transaction T_{s1} of value d฿. 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 $H(\tilde{k}_{31}) \overset{?}{=} H(k_{31})$. 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.

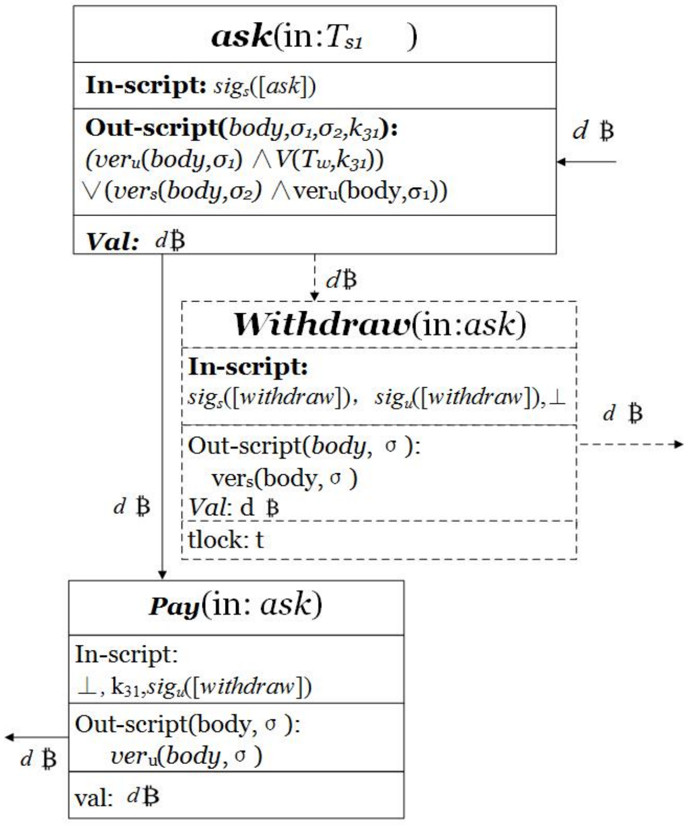

ask(in:T_{s1})
In-script: $sig_s([ask])$
Out-script($body,\sigma_1,\sigma_2,k_{31}$):
$(ver_u(body,\sigma_1) \wedge V(T_w,k_{31}))$
$\vee(ver_s(body,\sigma_2) \wedge ver_u(body,\sigma_1))$
Val: d฿
d ฿
d฿
Withdraw(in:ask)
In-script:
$sig_s([withdraw])$, $sig_u([withdraw]),\perp$
d ฿
Out-script($body$, σ):
$ver_s(body, \sigma)$
Val: d ฿
tlock: t
d ฿
Pay(in: ask)
In-script:
$\perp$, k_{31},$sig_u([withdraw])$
Out-script(body, σ):
$ver_u(body, \sigma)$
d ฿
val: d฿

Figure 5: Search Process

- ***Verify***: In order to reclaim d฿ and get d_1฿ 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+d฿. 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 $\mathcal{V}_1(\cdot)$. 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 $\mathcal{V}_1(\cdot)$ and each one cannot modify it, the $DB(w)$, Mac_w, $C_w, Mac(C_j)$ can be obtained in transaction *Prove*. In the future, the miners will verify the equations:

$$\mathcal{M}ac'_w \leftarrow H_1(K_w, DB(w))$$
$$\mathcal{MAC}'_{C_i} \leftarrow H_2(K_5, C_i)$$

hold or not. If they hold, the miners will accept *Prove*, or they will wait $Fuse$.

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

– If the transaction *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 *Get* into block chain, the user can immediately redeem the T_{u1} and quits the protocol. If the transaction *Prove* is not appeared in the block chain within time $t_1 - 2max_4$, where the max_4 means the maximal possible delay between broadcasting the transaction *Prove* and including it into the block chain. The user can broadcast the transaction $Fuse$ to redeem the $d + d_1$Ƀ. The concrete process is shown in figure 6.

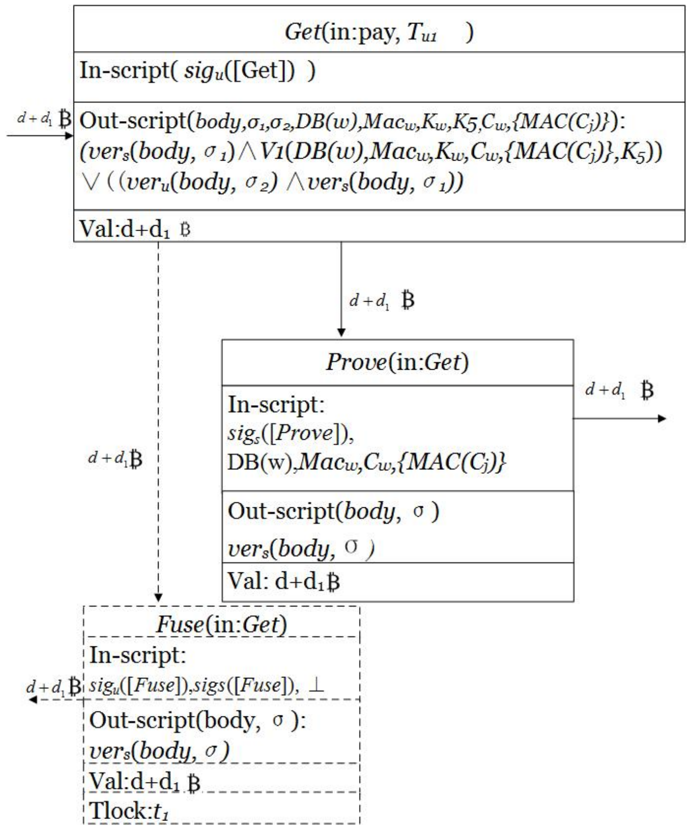
- If the transaction *Redeem* does not appear on the block chain within time $t_{max_1} - 2max_5 < t_{max_2}$, the server can broadcast the transaction *compesa* and gets dɃ. Here, the max_5 means the maximal possible delay between broadcasting the transaction *Redeem* and including it into the block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

- *Dec*: When the user receives $\mathcal{C}_w$, he will uses private key K_1 to decrypt $\mathcal{C}_w$. i.e., $\mathcal{D}_w = \varepsilon.Dec(K_1, \mathcal{C}_w)$.

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.



Get(in:pay, T_{u1})
In-script(sig_u([Get]))
Out-script(body,σ_1,σ_2,DB(w),Mac_w,K_w,K_5,C_w,{MAC(C_j)}):
(ver_s(body, σ_1)$\wedge$V1(DB(w),Mac_w,K_w,C_w,{MAC(C_j)},K_5))
$\vee$ ((ver_u(body, σ_2) $\wedge$$ver_s$(body, σ_1))
Val:d+d_1 ฿
$d+d_1$ ฿
Prove(in:Get)
In-script:
sig_s([Prove]),
DB(w),Mac_w,C_w,{MAC(C_j)}
Out-script(body, σ)
ver_s(body, σ)
Val: d+d_1฿
Fuse(in:Get)
In-script:
sig_u([Fuse]),sigs([Fuse]), ⊥
Out-script(body, σ):
ver_s(body, σ)
Val:d+d_1 ฿
Tlock:t_1

Figure 6: Verify Process

Theorem 5.1 *If H_1, H_2 are pseudorandom function, H is a collision resistant hash function, and $\varepsilon = (Enc, Dec)$ is PCPA-secure symmetric encryption scheme, then the scheme we present above is an adaptive IND-CKA2 block chain-based SSE scheme.*

Proof. We need to construct a PPT simulator $\mathcal{S} = \{\mathcal{S}_0, \mathcal{S}_1, ..., \mathcal{S}_q\}$ for the adversary such that $\mathcal{A} = \{\mathcal{A}_0, \mathcal{A}_1, ..., \mathcal{A}_q\}$ the output of $View_{real}$ and $View_{ideal}$ are computationally indistinguishable. Suppose that the simulator $\mathcal{S}$ is given the trace of a history L, then he can generate ($\mathbf{I}^*$, $\mathbf{C}^*$, t_w, $Appoint^*$, $compesa^*$, $Redeem^*$, ask^*, $Withdraw^*$, Pay^*, Get^*, $Prove^*$, $Fuse^*$) 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 $|\mathcal{I}|$. $\mathcal{S}$ puts the I^* in the state st_S, and set $C_i^* \leftarrow \{0,1\}^{|D_i|}$. Since the state $st_{\mathcal{A}_0}$ does not have the key K_2, K_3, he will uniformly choose $t_w^* \leftarrow \{0,1\}^*$ and $\mathcal{M}ac_w{}^* \leftarrow \{0,1\}^*$ at random for each keyword w.

If $q \geq 1$, the $st_{\mathcal{A}}$ firstly uniformly selects $(st_{\mathcal{A}_0}, t^*_{w^*_0}) \leftarrow \{0,1\}^*$ and $\mathcal{M}ac_{w^*_0}{}^* \leftarrow \{0,1\}^*$ at random for each keyword w^*_0. He then chooses $w^*_i, q \geq i \geq 1$ through $(w^*_{i-1}, st_{\mathcal{A}_{i-1}})$. Then he chooses $(st_{\mathcal{A}_i}, t^*_{w^*_i}) \leftarrow \{0,1\}^*$ and $\mathcal{M}ac_{w^*_i}{}^* \leftarrow \{0,1\}^*$ at random for each keyword w^*_i.

Because the H_1 is undistinguishable from function $f : \{0,1\}^k \to \{0,1\}^k$, we can get I^* is undistinguishable from I generated in $View_{real}$.

Similarly, because the ε is $IND-CPA$, the C^* is indistinguishable from a true ciphertext C. Because the state $st_{\mathcal{A}_0}$ does not have the key K_5, the adversary does not distinguish $\mathcal{MAC}_{C_i}{}^*$ from $\mathcal{MAC}_{C_i}$ generated by step Enc. Here the $\mathcal{MAC}_{C_i}{}^*$ is selected uniformly from $\{0,1\}^{2k}$ at random.

- Simulating the T^*_w. Because the decryption key k_{31} will embed in transaction Pay, each one can see it, so does the Adversary. It only needs to analyze the t^*_w, k_{21w^*} are undistinguishable from true t_w, k_{21} generated in $Srchtoken$ algorithm. If $q = 0$, the $\mathcal{S}$ will select $t^*_{w^*_0}, k_{21w^*_0}$ uniformly at random from $\{0,1\}^k$. If $q \geq 1$, simulator $\mathcal{S}$ will choose $w^*_i, q \geq i \geq 1$ by $st_{\mathcal{A}_{i-1}}, w^*_{i-1}$, $i \geq 1$, then computes $(t^*_{w^*_i}, k_{21w^*_i}) \leftarrow \{0,1\}^*$. Because pseudorandom function H_1 is undistinguishable from function $f : \{0,1\}^k \to \{0,1\}^k$, we can get t^*_w, k_{21w^*} are undistinguishable from true t_w, k_{21}.

- Simulating the transactions $Appoint^*$, $compesa^*$, $Redeem^*$, ask^*, $Withdraw^*$, Pay^*, Get^*, $Prove^*$, $Fuse^*$). Indeed, these transaction embed some information, such as $T_w, \mathcal{MAC}_w$, 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.

■

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 Get. The server can broadcast the transaction $compesa$ and redeems its dɃ pledge.

- If the server is not honest, it means that the server releases false $DB(w)$, C_w. The miners will reject transaction $Prove$, and the user will gets the $d_1 + d$Ƀ.

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 1: The time complexity comparing result

scheme	Search complexity	Communication	Verifiability	Adversary	Fairness
KO[17]	$O(D(w))$	$O(r)$	Yes	malicious server	No
KP[5]	$O(rlog\ (n)/p)$	$O(1)$	Yes	honest-but -curious server	No
Our scheme	$O(D(w))$	$O(6m)$	Yes	maliciou server malicious user	Yes

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 bitcion 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

[1] D. X. Song, D. Wagner, A. Perrig, Practical techniques for searches on encrypted data, in: Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on, IEEE, 2000, pp. 44–55.

[2] O. Goldreich, R. Ostrovsky, Software protection and simulation on oblivious rams, Journal of the ACM (JACM) 43 (3) (1996) 431–473.

[3] E.-J. Goh, et al., Secure indexes., IACR Cryptology ePrint Archive 2003 (2003) 216.

[4] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky, Searchable symmetric encryption: improved definitions and efficient constructions, in: Proceedings of the 13th ACM conference on Computer and communications security, ACM, 2006, pp. 79–88.

[5] S. Kamara, C. Papamanthou, Parallel and dynamic searchable symmetric encryption, in: International Conference on Financial Cryptography and Data Security, Springer, 2013, pp. 258–274.

[6] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, M. Steiner, Highly-scalable searchable symmetric encryption with support for boolean queries, in: Advances in Cryptology–CRYPTO 2013, Springer, 2013, pp. 353–373.

[7] A. Boldyreva, N. Chenette, Efficient fuzzy search on encrypted data, in: Fast Software Encryption, Springer, 2014, pp. 613–633.

[8] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M.-C. Rosu, M. Steiner, Dynamic searchable encryption in very-large databases: Data structures and implementation., IACR Cryptology ePrint Archive 2014 (2014) 853.

[9] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, W. Lou, Fuzzy keyword search over encrypted data in cloud computing, in: INFOCOM, 2010 Proceedings IEEE, IEEE, 2010, pp. 1–5.

[10] B. Wang, S. Yu, W. Lou, Y. T. Hou, Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud, in: INFOCOM, 2014 Proceedings IEEE, IEEE, 2014, pp. 2112–2120.

[11] T. Moataz, A. Shikfa, Boolean symmetric searchable encryption, in: Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, ACM, 2013, pp. 265–276.

[12] E. Stefanov, C. Papamanthou, E. Shi, Practical dynamic searchable encryption with small leakage., in: NDSS, Vol. 14, 2014, pp. 23–26.

[13] R. Bost, P.-A. Fouque, D. Pointcheval, Verifiable dynamic symmetric searchable encryption: Optimality and forward security, Tech. rep., Cryptology ePrint Archive: Report 2016/062 (2016).

[14] R. Bost, $o\varphi o\varsigma$: Forward secure searchable encryption, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2016, pp. 1143–1154.

[15] C. Bösch, P. Hartel, W. Jonker, A. Peter, A survey of provably secure searchable encryption, ACM Computing Surveys (CSUR) 47 (2) (2015) 18.

[16] J. Alderman, K. M. Martin, S. L. Renwick, Multi-level access in searchable symmetric encryption. http://eprint.iacr.org/2017/211.

[17] K. Kurosawa, Y. Ohtaki, Uc-secure searchable symmetric encryption, in: Financial Cryptography and Data Security, Springer, 2012, pp. 285–298.

[18] R. Cheng, J. Yan, C. Guan, F. Zhang, K. Ren, Verifiable searchable symmetric encryption from indistinguishability obfuscation, in: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ACM, 2015, pp. 621–626.

[19] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, http://www.cryptovest.co.uk/resources/Bitcoin%20paper%20Original.pdf.

[20] F. Reid, M. Harrigan, An analysis of anonymity in the bitcoin system, in: Security and privacy in social networks, Springer, 2013, pp. 197–223.

[21] D. Ron, A. Shamir, Quantitative analysis of the full bitcoin transaction graph, in: International Conference on Financial Cryptography and Data Security, Springer, 2013, pp. 6–24.

[22] L. L. Berger, Bitcoin exchange transactions: income tax implications to consider within the south african environment, Ph.D. thesis (2016).

[23] I. Eyal, A. E. Gencer, E. G. Sirer, R. Van Renesse, Bitcoin-ng: A scalable blockchain protocol, in: 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16), 2016, pp. 45–59.

[24] J. Garay, A. Kiayias, N. Leonardos, The bitcoin backbone protocol: Analysis and applications, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2015, pp. 281–310.

[25] M. Andrychowicz, S. Dziembowski, D. Malinowski, L. Mazurek, Secure multiparty computations on bitcoin, in: 2014 IEEE Symposium on Security and Privacy, IEEE, 2014, pp. 443–458.

[26] M. Andrychowicz, S. Dziembowski, D. Malinowski, Ł. Mazurek, Fair two-party computations via bitcoin deposits, in: International Conference on Financial Cryptography and Data Security, Springer, 2014, pp. 105–121.

[27] I. Bentov, R. Kumaresan, How to use bitcoin to design fair protocols, in: International Cryptology Conference, Springer, 2014, pp. 421–439.

[28] V. Buterin, et al., A next-generation smart contract and decentralized application platform, white paper.