How to retrieve the encrypted data on the blockchain

Huige Li, Fangguo Zhang, Peiran Luo, Haibo Tian and Jiejie He
1 School of Data and Computer Science, Sun Yat-sen University
Guangzhou 510006 - China
2 Guangdong Key Laboratory of Information Security
Guangzhou 510006 - China
[e-mail: isszhfg@mail.sysu.edu.cn]
*Corresponding author: Fangguo Zhang

Received January 4, 2019; revised March 20, 2019; accepted May 9, 2019; published November 30, 2019

Abstract

Searchable symmetric encryption (SSE) scheme can perform search on encrypted data directly without revealing the plain data and keywords. At present, many constructive SSE schemes were proposed. However, they cannot really resist the malicious adversary, because it (i.e., the cloud server) may delete some important data. As a result, it is very likely that the returned search results are incorrect. In order to better guarantee the integrity of outsourcing data, and ensure the correction of returned search results at the same time, in this paper, we combine SSE with blockchain (BC), and propose a SSE-on-BC framework model. We then construct two concrete schemes based on the size of the data, which can better provide privacy protection and integrity verification for data. Lastly, we present their security and performance analyses, which show that they are secure and feasible.

Keywords: Searchable encryption, blockchain, cloud-storage, symmetric encryption, privacy

This research was supported by the National Natural Science Foundation of China (No. 61672550), the National Key R&D Program of China (No. 2017YFB0802503), and the Guangxi Key Laboratory of Cryptography and Information Security (No. GCIS201711).

http://doi.org/10.3837/tiis.2019.11.016
1. Introduction

Cloud storage not only allows clients to access their outsourcing data anytime and anywhere, but also charges them a small fee, therefore, more and more people turn to upload data onto it. The cryptography technology provides a technical support on the confidentiality and privacy of these outsourcing data. However, simple cryptography encryption algorithm will hinder the search capability on these encrypted outsourcing data. To address this issue, Song et al. are the first people to propose the concept of searchable encryption (SE) [1]. Because it uses symmetric encryption technology in their paper, it can be seen as a searchable symmetric encryption scheme (SSE).

The data owner and server are the two main participants in SSE. The data owner uses a symmetric encryption algorithm to encrypt data, then he (or she) uploads them on the cloud. When searching the data that contain keyword $w$, he (or she) encrypts this keyword $w$ by using secret key, and generates a search token $t(w)$, which will be sent to the cloud server. The cloud server computes search results by using ciphertexts and $t(w)$, and sends them to the data owner. Lastly, the data owner decrypts these search results locally.

There are many constructive SSE schemes in recent years, such as schemes supporting single keyword [2,3], multiple keyword [4,5,6,7], fuzzy matching [8,9], ranked search [10,11,12], dynamic SSE schemes [13,14,15,16,17,18,19,20,21], parallel SSE scheme [22], and the scheme that supports multi-level access policy [23]. In addition, Bösch et al. did a comprehensive survey of SSE [24].

Some researchers also consider the security level problem in SSE. For example, to against malicious adversary, Kurosawa et al. used the message authentication code technology [25], while Cheng et al. used indistinguishability obfuscation [26]. Dai et al. used the physically unclonable function to prevent memory leakage [27]. While, Li et al. introduced the coercer into SSE [28].

In the above schemes, the cloud server usually is trustworthy, who directly controls the users' data. Although this third party is trustworthy, sometimes it will damage the user's data for its personal benefit. For example, it may tamper with users' data to save its space. Once it happens, the users cannot get true search results. What's worse, if the third party deletes the data that used to verify whether the results are right or not, the users will never judge the correctness of returned results. In order to solve these problems, a simple solution is that the user selects multiple cloud storage platforms to store his (or her) data. He (or she) can perform search on these platforms respectively, and merges the search results together. But this method will waste a lot of network traffic and bandwidth.

However, the blockchain technology can provide a potential solution to the above issue. The blockchain is an emerging technology in recent years, which is stemmed from the Bitcoin system [29] but can be seen as an independent technology. It is composed of blocks one after another. The data is collected and verified by nodes on the blockchain. Only it is accepted by most of nodes, it can be stored in one block. Users can access these data freely, but they cannot tamper with them because the blockchain uses some tools, such as the cryptography hash function and so on.

The data on the blockchain is maintained by everyone. The modified data cannot be accepted as long as the majority of nodes are honest. Therefore, we can use this technology to build a cloud storage system to ensure the data integrity. This it to say, users can store their...
data on a blockchain in the form of transactions. Consequently, except accessing the data flexibly, they do not have to worry about their data being tampered with by illegal users.

Because the size of each block on the blockchain is fixed, the number of data stored in it is limited. When more and more data are generated, the length of the whole blockchain continues to raise. As a result, the problem of how to search data on the blockchain becomes intractable. Taking the Bitcoin system for example, the data on this blockchain are transactions, whose size is small. If Alice wants to find transactions she finished in a certain period of time, she has to find them in the order from back to front. Suppose there are $|T|$ transactions on the blockchain, therefore, the search efficiency is linear in $O(|T|)$.

It is very interesting to consider the privacy security of data and improve the search complexity on the blockchain. Because it not only protects the privacy of data, but also can guarantee the correctness of the search results. Moreover, it can save users’ time. Take the electronic medical systems as an example, at present each hospital keeps the electronic medical records (EMRs) of their patients privately, which can be seen as a private cloud server. These hospitals do not share EMRs to each other. When a patient chooses a new hospital to see a doctor, because he or she cannot obtain all his or her EMRs in time, his or her illness may not be treated quickly. However, this dilemma can be avoided by using blockchain. That is, each hospital uploads the patients' EMRs onto the blockchain in time. The patient then can find his (or her) EMRs at any time and does not need to interact with previous hospital respectively. This scenario was mentioned by Swan in [30]. However, it did not give an effective solution.

**Our contribution.** In this paper, we combine blockchain with SSE, and give a solution to protect the privacy of data and realize search. Our contributions are summarized as below:
- We propose a SSE framework on the blockchain and name it SSE-on-BC, which can better guarantee the integrity of the data and resist the malicious adversary.
- We construct two schemes based on the size of data. Because the smart contract can verify data automatically on the blockchain, the data owner in our schemes can fully believe that the returned search results are correct.
- We complete the security and performance analyses for our schemes, which show that our schemes is adaptively secure and feasible.

**Organization.** The remainder of this paper is organized as follows. In section 2, we review some tools and notations. In section 3, the SSE-on-BC model and its security definition are proposed. There are two concrete constructions in section 4. Next are the analyses of performance and security of our schemes. The conclusion is present in the last section.

## 2. Preliminaries

We will review some tools and notations in this section. It mainly includes negligible function, the model of SSE, Bitcoin system, and so on.

**Definition 1.** A function $f(\cdot)$ is negligible if for every polynomial $p(\cdot)$ there exists an integer $N$ such that for all integer $n>N$ it holds that $f(n)<\frac{1}{p(n)}$.

### 2.1 The model of SSE

In Fig. 1, there have two players: the data owner and the cloud server. In the first stage, the data owner uses his secret key $k$ to encrypt data $D$ into $C$, and builds an invertible index $I$, which are sent to the cloud server. When searching data containing the keyword $w$, the data
owner combines the secret key $k$ with $w$, and gets a search token $t_w$, which is sent to the cloud server. The cloud server returns the search result $C_j$. Lastly, he decrypts $C_{ij}$ locally.

2.2 Bitcoin system

To make readers understand blockchain clearly, in this section, we review some knowledge about the Bitcoin system.

The addresses and transactions are two important elements in the Bitcoin system. To create a transaction, each client must generate a pair of keys (i.e., a private key and a public key) firstly. The private key is used to sign transaction, and gets a signature $\sigma$. The public key is used to generate an address and verify whether the $\sigma$ is valid or not [31]. Compared with traditional electronic cryptocurrency [32, 33], the Bitcoin supports change. To make reader understand clearly, we will use symbol $A = (A.pk, A.sk)$ to denote a key pair of user A. Let $\sigma = \text{sig}_A(m)$ denote a signature about transaction $m$, which is computed by A’s private key $A.sk$, and a verification result about signature $\sigma$ denoted by $\text{ver}_A(m, \sigma)$, which is computed by A’s public key $A.pk$.

A transaction $T$ may have multiple inputs and outputs. The inputs show where these coins come from. The outputs indicate how much money should be given to each recipient, which is represented by an address. Each transaction will have an in-script and an out-script, and both of them are written in Bitcoin scripting language, i.e., the stack based language [34]. Generally, if transaction $T$ wants to redeem transaction $T_x$, its in-script must match with the out-script of $T_x$.

Let BTC represent the bitcoin cryptocurrency symbol. To make the reader understand clearly, we will use Fig. 2 to explain how the transaction works. Suppose Alice wants to pay Bob $v$ BTC $= v_1$ BTC $+ v_2$ BTC (here we do not consider the transaction fee), she needs to create a transaction $T_v$. She finds two unredeemed transactions $T_{v1}$ and $T_{v2}$ from her wallet, such that $v = v_1 + v_2$. In order to show she can spend these money, she puts her signatures $\sigma_1$ and

![Fig. 1. Traditional SSE Model.]

![Fig. 2. The construction process of transaction $T_v$.]
\[\sigma_2\] in the in-script of \(T_x\). Alice adds a function \(\pi_x(\text{body},\sigma)\) in the out-script of transaction \(T_x\) to indicate she will transfer \(v\) BTC to Bob, whose output is a Boolean.

Generally, we can use \(T_x=(y_1,y_2,\pi_x,v,\sigma_1,\sigma_2)\) to denote the transaction \(T_x\), where \(y_1\) is a hash of \(T_y\) and \(y_2\) is a hash of \(T_y\). In addition, the clients can specify a time \(t\) in a transaction, which means that this transaction will be collected by miners after time \(t\). In the Bitcoin system, if a transaction wants to be accepted earlier, it needs to pay some transaction fees. That is, \(v_1+v_2>v\) usually holds, and the difference between them is the transaction fees.

Besides, we enumerate the meanings of some functions and symbols that we will use later, which are shown in Table 1.

| Notations | Meaning |
|-----------|---------|
| \(D\) | the plain document. |
| \(C\) | the encrypted document. |
| \(W=\{w_1,w_2,\cdots,w_m\}\) | The dictionary composed of the keywords \(w_1,w_2,\cdots,w_m\), where \(w_1,\cdots,w_m\) are extracted from the document \(D\). |
| \(k\) | the system parameter. |
| \(a||b\) | a concatenation of string \(a\) and string \(b\). |
| \(F_i (i=1,2,3)\) | a keyed pseudorandom function \(F_i:\{0,1\}^s\times\{0,1\}^n\rightarrow\{0,1\}^s\). |
| \(H\) | a keyed hash function; \(H:\{0,1\}^l\times\{0,1\}^p\rightarrow\{0,1\}^p\), where \(l, p\) are big prime. |
| \(H_1\) | a hash function without key \(H_1:\{0,1\}^p\rightarrow\{0,1\}^p\), where \(l, p\) are big prime. |
| \(\varepsilon=(\varepsilon.\text{Enc}, \varepsilon.\text{Dec})\) | an indistinguishability against chosen-plaintext attacks (IND-CPA) secure symmetric encryption (SE) scheme, where \(\varepsilon.\text{Enc}\) denotes the encryption process and \(\varepsilon.\text{Dec}\) denotes the decryption process. |
| \(\delta=(\delta.\text{Enc}, \delta.\text{Dec})\) | a determinate SE scheme, where \(\delta.\text{Enc}\) denotes the encryption process and \(\delta.\text{Dec}\) denotes the decryption process. |
| \(x\leftarrow\{\text{UTXO}\}\) | \(x\) is sampled from the set \(\{\text{UTXO}\}\) at random, where each UTXO denotes an unredeemed transaction in the blockchain. |
| \(t\) | the upper bound of the size of transaction. |
| \(G=<g>\) | a multiple group of order \(p\), whose generator is \(g\). The \(p\) is a big prime. |
| \(DB(w)\) | a set composed of document transactions related to the keyword \(w\). |
| \(|A|\) | the cardinal number of set \(A\). |
| \([T]\) | The body of the transaction \(T\), which does not contain its in-script value. |

### 3. Our System Model

In this section, the SSE-on-BC model is firstly presented, the following is its security definition.

#### 3.1 The model of SSE-on-BC

It have two participants in Fig. 3 the data owner \(U\) and the server \(S\) (i.e., a receiver of transaction). The data owner \(U\) has \(n\) data \(D_1,\cdots,D_n\). To protect their privacy, he will encrypt them into \(C_1,\cdots,C_n\) by using symmetric encryption algorithm. He then uploads them on the blockchain in the form of transactions \(T_1,\cdots,T_n\) respectively. He then creates a transaction \(Inx\) based on these transactions \(T_1,\cdots,T_n\). To find the data containing the keyword \(w\), he puts the search token \(t(w)\) and the identifier \(TX_{Inx}\) of transaction \(Inx\) into function \(\Phi\), and embeds function \(\Phi\) into transaction \(t\). He then broadcasts it on the blockchain. If the server \(S\) can provide correct search results, it can redeem transaction \(t\).
by using transaction \( s \). Otherwise, the data owner \( U \) will use transaction \( p \) to redeem transaction \( t \).

![Fig. 3. The Model of SSE-on-BC.](image)

Our SSE-on-BC model (i.e., SSE-on-BC= \((\text{Gen, Enc, Trpdr, Search, Dec})\)) contains the following five polynomial-time algorithms:

(a) \( (K, U, S) \leftarrow \text{Gen}(1^k) \): It is a probabilistic algorithm run by the data owner \( U \) and the server \( S \). The inputting parameter is \( k \), and the outputs are a secret key \( K \), a pair of keys \( U=(U.pk, U.sk) \) and a pair of keys \( S=(S.pk, S.sk) \).

(b) \( (T, \text{Inx}, TX_{\text{Inx}}) \leftarrow \text{Enc}(K, U, D, \{T_d\}_{d=0}^{n}) \): is a probabilistic algorithm run by the data owner \( U \). It inputs the secret key \( K \), the pair of keys \( U \), the documents set \( D=(D_1, \ldots, D_n) \) and \( n+1 \) unredeemed transactions \( T_{db}, \ldots, T_{dn} \) and outputs \( n+1 \) transactions \( T=\{T_1, \ldots, T_n\}, \text{Inx} \). Besides, the data owner needs to store the identifier \( TX_{\text{Inx}} \) of transaction \( \text{Inx} \) locally.

(c) \( t \leftarrow \text{Trpdr}(K, w, U, TX_{\text{Inx}}, T_w) \): It is a determinate algorithm, which is run by the data owner \( U \). The inputs are the secret key \( K \), the pair of keys \( U \), keyword \( w \), identifier \( TX_{\text{Inx}} \) and an unredeemed transaction \( T_w \). The output is a transaction \( t \), whose receiver is either \( U \) or the server \( S \).

(d) \( s/p \leftarrow \text{Search}(T, \text{Inx}, TX_{\text{Inx}}, t, S/U) \): It is run either by the server \( S \) or the data owner \( U \). If the server \( S \) can provide correct search results, it needs to take \( T, \text{Inx}, TX_{\text{Inx}}, t, S \) as input, and outputs a transaction \( s \). Otherwise, the data owner inputs \( U \) and \( t \) to output a transaction \( p \), which can be used to redeem transaction \( t \).

(e) \( \{D_{ij}\} \leftarrow \text{Dec}(K, s) \): It is a decryption algorithm run by the data owner \( U \). The inputs are the secret key \( K \) and the transaction \( s \), and it outputs the plaintexts \( \{D_{ij}\} \) locally.

A SSE-on-BC scheme is correct if for all \( k \in \mathbb{N} \), for all \( K, U, S \) output by \( \text{Gen}(1^k) \), for all data \( D \subseteq 2^\Delta \), for all \( (T, \text{Inx}, TX_{\text{Inx}}) \) output by \( \text{Enc}(K, U, D, \{T_d\}_{d=0}^{n}) \), for all keyword \( w \in \Delta \), such that

\[
\text{Search}(T, \text{Inx}, TX_{\text{Inx}}, \text{Trpdr}(K, w, U, TX_{\text{Inx}}, T_w), S) = s \land \text{Dec}(K, s) = \{D_{ij}\}, \text{ for } 1 \leq i \leq n. \quad (1)
\]
3.2 Security Definition

A secure SSE-on-BC scheme should satisfy the following conditions.

- The server \( S \) cannot derive any useful information about the plain data when it accesses to the blockchain for the first time;
- After search, in addition to the search results, the server \( S \) also cannot get any useful information about plaintexts and keywords.
- If the server \( S \) cannot return the right search results to the data owner \( U \), it cannot redeem the transaction \( t \) created by the data owner \( U \).

| \[ \text{Real}_{1}^{1}(k) \] |
|-----------------|
| \((K, U, S) \leftarrow \text{Gen}(1^n)\) |
| \((D, st_A) \leftarrow A_0(1^n)\) |
| \([T_d]_{i=0}^n \leftarrow A_0(st_A, \{\text{UTXO}\})\) |
| \((\{T_i\}_{i=0}^n, \text{Inx}, \text{TX}_{\text{int}}) \leftarrow \text{Enc}(K, U, D, \{T_d\}_{i=0}^n)\) |
| Let \( T = (T_0, \cdots, T_n)\) |
| \((w_i, st_A) \leftarrow A_1(st_A, T, \text{Inx}, \text{TX}_{\text{int}})\) |
| \(T_{w_i} \leftarrow A_1(st_A, \{\text{UTXO}\})\) |
| \(I_{w_i} \leftarrow \text{Trpdr}(K, w_i, U, \text{TX}_{\text{int}}, T_{w_i})\) |
| \(s_i \leftarrow \text{Search}(T, \text{Inx}, \text{TX}_{\text{int}}; I_{w_i})\) |
| Output \( V = (\text{Inx}, T, \text{Tr}, \text{TS}) \) and \( st_A \) |

**Fig. 4.** Game \( \text{Real}_{1}^{1}(k) \).

Adversary either is adaptive or non-adaptive. When the adversary is adaptive, it can select keyword based on the previous keywords and search results. When the adversary is non-adaptive, it should choose all the keywords at once. In this paper, we only consider the former.

**Definition 2.** Let \( II=(\text{Gen}, \text{Enc}, \text{Trpdr}, \text{Search}, \text{Dec}) \) denote a SSE-on-BC scheme, \( L \) be a leakage function that is parameterized by access pattern, search pattern and size pattern defined in [3], \( k \) be the security parameter. Considering the following games \( \text{Real}_{1}^{1}(k) \) and \( \text{Ideal}_{1,5}^{1}(k) \) shown in the **Fig. 4 and 5.**

| \[ \text{Ideal}_{1,5}^{1}(k) \] |
|-----------------|
| \((D, st_A) \leftarrow A_0(1^n)\) |
| \([T_d]_{i=0}^n \leftarrow A_0(st_A, \{\text{UTXO}\})\) |
| \((\{T_i\}_{i=0}^n, \text{Inx}, \text{TX}_{\text{int}}) \leftarrow S_d(L(D, \{T_d\}_{i=0}^n)\) |
| Let \( T = (T_0, \cdots, T_n)\) |
| \((w_i, st_A) \leftarrow A_1(st_A, T, \text{Inx}, \text{TX}_{\text{int}})\) |
| \(T_{w_i} \leftarrow A_1(st_A, \{\text{UTXO}\})\) |
| \(I_{w_i} \leftarrow \text{Trpdr}(K, w_i, U, \text{TX}_{\text{int}}, T_{w_i})\) |
| \(s_i \leftarrow \text{Search}(T, \text{Inx}, \text{TX}_{\text{int}}; I_{w_i})\) |
| Output \( V = (\text{Inx}, T, \text{Tr}, \text{TS}) \) and \( st_A \) |

**Fig. 5.** Game \( \text{Ideal}_{1,5}^{1}(k) \).
We say a SSE-on-BC scheme is adaptively semantically secure if for all polynomial size adversaries $A=(A_0, A_1, \cdots, A_\ell)$ where $q=poly(k)$, there exists a non-uniform polynomial size simulator $S=(S_0, S_1, \cdots, S_\ell)$, such that for all polynomial size $D$, 
\[ |Pr[D(V, st_X) = 1: (V, st_X) \leftarrow \text{Real}_A(k)] - Pr[D(V, st_X) = 1: (V, st_X) \leftarrow \text{Ideal}_{A, S}(k)] | \leq \text{neg}(k), \]
where the probabilities are taken over the coins of $Gen$ and $Enc$.

4. The detailed scheme

Since the size of each block on the blockchain is limited, we should consider the size of the data before uploading. To solve this problem, we present two concrete constructions in this section.

4.1 A SSE-on-BC scheme supports lightweight data

Suppose the size of data array $D=(D_1, \cdots, D_n)$ is small. In order to upload them on the blockchain, the data owner $U$ will do the following steps:

(a) $Gen$: After inputting a security parameter $k$, the data owner $U$ gets a secret key array $K=(K_1, K_2)$, where $K_i \leftarrow \{0,1\}^k (i=1, 2)$. Besides, the data owner $U$ and the server $S$ generate a pair of keys $U=(U.sk, U.pk)=(u_1, g^{u_1})$ and a pair of keys $S=(S.sk, S.pk)=(s_1, g^{s_1})$ respectively, where $u_1, s_1 \in Z_p$ and $g, g^1 \in G$.

(b) $Enc$: For each document $D_j (1 \leq j \leq n)$, the user computes:
\[ C_j = \text{Enc}(K_1, D_j) (i=1, \cdots, n), \]
He then selects an empty set $DB(w_i)$ for each keyword $w_i \in W (i=1, \cdots, m)$. If document $D_j (1 \leq j \leq n)$ contains keyword $w_i$, he puts $C_j$ into $DB(w_i)$. To make readers understand clearly, suppose $\Delta=|DB(w_i)|$, and $DB(w_i) = \{ C_{i_1}, \cdots, C_{i_{\Delta}} \}$. He continues to compute:
\[ t_{w_i} = F_1(K_2, w_i), \]
\[ l_{w_i} = F_2(K_2, w_i), \]
\[ k_{w_i} = F_3(K_2, w_i), \]
\[ h_{w_i} = H(k_{w_i}, C_{i_1} || \cdots || C_{i_{\Delta}}). \]
In order to store the ciphertext $C_i (i=1, \cdots, n)$ on the blockchain, he finds $n$ unredeemed transactions $TX_{\text{D_0}}, \cdots, TX_{\text{D_0}}$ from his own wallet, which contain $d_1, \cdots, d_i$ amount of coins respectively. He then builds transactions $TX_{D_i} (i=1, \cdots, n)$ in the following manner:

1) For transaction $TX_{D_i}$, he embeds $C_i (i=1, \cdots, n)$ into its out-script. Then he uses transaction $TX_{\text{D_0}}$ to compute the body value of transaction $TX_{D_i}$.

2) Sign transaction $TX_{D_i}$ by using his private key $U.sk$, which is broadcasted to the blockchain.

3) If the transactions $TX_{D_1}, \cdots, TX_{D_n}$ appear on the blockchain, the data owner $U$ computes $TXID_{D_i} = H_i(TX_{D_i}) (i=1, \cdots, n)$, which are seen as the identifiers of transactions $TX_{D_1}, \cdots, TX_{D_n}$ respectively.

For each keyword $w_i (i=1, \cdots, m)$, if $C_j \in DB(w_i)$, he replaces $C_j$ with $TXID_{D_i} (1 \leq j \leq n, i =1, \cdots, m)$. 
Let $\Delta = \max \{ \Delta_i \}$. If $\Delta_i < \Delta$, he pads $DB(w_i)$ with $\Delta - \Delta_i$ elements $0^\prime$ such that $|DB(w_i)| = \Delta$, where $i=1,\cdots,m$. Here, we still use symbol $DB(w_i)$ to represent the result after padding.

He chooses an empty array $I$. For each keyword $w_i \in W$, he computes:

$$e_{w_i} = \delta,\text{Enc}(l_{w_i}, DB(w_i))$$

(8)

He stores $(t_{w_i}, e_{w_i}, h_{w_i})$ into array $I$ in a lexicographical manner.

To generate a transaction $Inx$ for documents $D$, the data owner $U$ does:

1) Find an unredeemed transaction $TX_0$ from his wallet, which contains $d_0$ coins.
2) For transaction $Inx$, he embeds $I$ into its out-script.
3) Take transaction $TX_0$ as input, and compute the body of transaction $Inx$.
4) Sign transaction $Inx$ by using $U.sk$, and broadcasts it on the blockchain.

After it appears on the blockchain, he computes its identifier $TX_{Inx} = H_1(Inx)$ and stores it locally, otherwise he needs to recreate transaction $Inx$.

Suppose $\Phi(\cdot, \cdot)$ is a function, which consists of a decryption algorithm and a verification algorithm. It takes two strings $x, y$ as input. It then executes:

1) Use $y$ to find the transaction $q$.
2) Decrypt the information that embedded in transaction $q$ by using $x$. Suppose the decryption results are $(\alpha, \beta)$.
3) Inputs $\alpha, \beta$ and $x$, and it will verify whether $\beta = H(x, \alpha)$ holds or not. If it does, it will outputs $\alpha, 1$, where 1 is a Boolean value. Otherwise it outputs a termination symbol $\bot$.

(c) **Trpdr:** When $U$ wants to find the data containing the keyword $w$, he will create a transaction $ask$ shown in Fig. 6. The concrete construction is as follows:

1) Find an unredeemed transaction $T_q$ from his wallet, which contains $d_t$ coins.
2) Compute $t_w = F_1(K_2, w), l_w = F_2(K_2, w)$ and $k_w = F_3(K_2, w)$, the data owner $U$ then puts $\Phi((t_w, l_w, k_w), TX_{Inx})$ into the out-script of $ask$.
3) Use $T_q$ to compute the body of $ask$.
4) Inputting transaction $ask$, the data owner $U$ and server $S$ compute the body of transaction $Fuse$ respectively. Here, it has a time lock $t$ in the transaction $Fuse$.
5) The server $S$ signs the transaction $Fuse$ by using $S.sk$, and sends it to $U$ to let him add his own signature in it.
6) After signing the transaction $ask$ by using $U.sk$, the data owner $U$ broadcasts it.
7) If transaction $ask$ does not appear on the blockchain until time $t - \max_U$, the data owner $U$ can redeem transaction $T_q$ by using his private key, and quits the protocol immediately. Here, the symbol $\max_U$ means the maximal possible delay time of transaction $T_q$ appears on the blockchain.
(d) **Search**: When the server $S$ wants to redeem the transaction $ask$, it needs to build a transaction $return$ shown in Fig. 6, which contains the information of the search results. The concrete process is as follows:

1) Input transaction $ask$, and compute the body of transaction $return$.

2) Run function $\Phi(t_w, l_w, k_w, TX_{Inx})$:
   i. Use $TX_{Inx}$ to get the information $I$ embedded in the transaction $Inx$.
   ii. Use $t_w$ to find $(e_w, h_w)$, which is stored in $I$.
   iii. Decrypt $e_w$ by using $l_w$: $DB(w) = \delta.\text{Dec}(l_w, e_w)$. For brevity, let us use $DB(w) = \{TXID_{D_{1j}}, TXID_{D_{2j}}, \ldots, TXID_{D_{nj}}\}$ to denote the decryption results, where $TXID_{D_{ij}} = H_i(TX_{D_{ij}})$ ($j=1, \ldots, n$) denotes the identifier of transaction $TX_{D_{ij}}$.
   iv. Read the document ciphertext $C_{ij}$ from transaction $TX_{D_{ij}}$ by using $TXID_{D_{ij}}$ ($j=1, \ldots, n$).

3) Verify whether the equation $H(k_w, C_{ij} || \ldots || C_{ik}) = h_w$ holds or not. If it holds, it puts \{ $C_{ij}$ \} into the in-script of transaction $return$.

4) Sign transaction $return$, and broadcast it onto the blockchain.

(e) **Dec**: After transaction $return$ appearing on the blockchain, the data owner $U$ can read \{ $C_{ij}$ \} from it. He continues to do: $D_{ij} = e.\text{Enc}(K_1, C_{ij})$ ($1 \leq j \leq n$). If the transaction $return$ does not appear on the blockchain after time $t$, he will broadcast transaction $Fuse$ and gets his money back.

### 4.2 A SSE-on-BC scheme supports the Data with big size

If the scale of data is larger, we should deal with it before uploading it. Suppose the data owner $U'$ has $n$ documents $D_1, \ldots, D_n$, whose size is larger. In order to store them on the blockchain, he
will do:

a) **Gen**: It inputs the security parameter $k$, and outputs a secret key array $K = (K_1, K_2)$, where $K_i \leftarrow \{0,1\}^k$ ($i = 1, 2$). Besides, the data owner $U$ and the server $S$ generate a pair of keys $U=(U.sk, U.pk) = (u_1, g_1^{u_1})$ and a pair of keys $S=(S.sk, S.pk) = (s_1, g_1^{s_1})$ respectively, where $u_1, s_1 \in \mathbb{Z}_p$ and $g_1, g_1^{s_1} \in G$.

b) **Enc**: The data owner $U$ encrypts documents $D = (D_1, \cdots, D_n)$ by using the secret key $K_1$:

$$C_i = \varepsilon. \text{Enc}(K_1, D_i) \ (i = 1, \cdots, n).$$  \hfill (9)

1) If $|C_i| > \iota - p$:

He divides $C_i$ into $s$ blocks $C'_1, C'_2, \cdots, C'_s$ such that $|C'_j| + p \leq \iota$, where $s = \left\lceil \frac{|C_i| - \iota}{-p} \right\rceil$.

For each keyword $w_i \in W$ ($i = 1, \cdots, m$), he chooses an empty set $d(w_i)$ and assigns elements to it in this way: If document $D_j$ ($1 \leq j \leq n$) contains keyword $w_i$, he puts $C_j$ into $d(w_i)$.

Suppose $d(w_i) = \{C_{i_1}, \cdots, C_{i_{s_i}}\}$. He computes:

$$t_{w_i} = F_1(K_2, w_i),$$  \hfill (10)

$$l_{w_i} = F_2(K_2, w_i),$$  \hfill (11)

$$k_{w_i} = F_3(K_2, w_i),$$  \hfill (12)

$$h_{w_i} = H(k_{w_i}, C_{i_1} || \cdots || C_{i_{s_i}}).$$  \hfill (13)

He finds $s$ unredeemed transactions $TX_{D^{0}_{i_1}}, \cdots, TX_{D^{0}_{i_{s_i}}}$ from his wallet, which contain $d_{i_1}, \cdots, d_{i_s}$ amount of coins respectively, and builds transactions $TX_{D_k}$ ($k = 1, \cdots, s$) as follows:

For $k = 1$:

i. Embed $C'_1 || 0^p$ into the out-script of transaction $TX_{D_1}$.

ii. Take transaction $TX_{D^{0}_{i_1}}$ as input, and compute the body of transaction $TX_{D_1}$.

iii. Sign transaction $TX_{D_1}$ by using $U'.sk$, and broadcast it onto the blockchain.

iv. After transaction $TX_{D_1}$ appears on the blockchain, he computes its identifier $TXID_{D_1} = H_1(TX_{D_1})$.

For $2 \leq k \leq s$:

i. In the out-script of transaction $TX_{D_k}$, he embeds information $C'_k || TXID_{D_{(k-1)}}$.

ii. Take $TX_{D^{0}_{i_k}}$ as input, and compute the body of transaction $TX_{D_k}$.

iii. Sign it by using $U'.sk$, and broadcast it to the blockchain.

iv. If the transaction $TX_{D_k}$ appears on the ledger, he computes its corresponding transaction identifier $TXID_{D_k} = H_1(TX_{D_k})$.

2) When $|C_i| \leq \iota - p$ ($1 \leq i \leq n$), he finds an unredeemed transaction $TX_{D_0}$ from his wallet, which contains $d_i$ coins. He then builds a transaction $TX_{D_1}$ as follows:

i. In the out-script of transaction $TX_{D_1}$, he embeds information $C_i$.

ii. Inputting transaction $TX_{D_0}$, he computes the body of transaction $TX_{D_1}$. 

iii. Sign it by using $U'.sk$, and broadcast it on the blockchain.

iv. After it appears on the blockchain, he computes its identifier

$$TXID_{D_j} = H_i(TX_{D_j}).$$

For each keyword $w_i$ ($1 \leq i \leq m$), he assigns an empty set $DB(w_i)$. He assigns elements to it in the following way:

i. If $w_i \in D_j$ and $|C_{i_j}| > t - p$, he puts $TXID_{D_j^{w_i}}$ into the set $DB(w_i)$.

ii. If $w_i \in D_j$ and $|C_{i_j}| \leq t - p$, he puts $TXID_{D_j^{w_i}}$ into the set $DB(w_i)$.

Suppose $\Delta_{w_i} = |DB(w_i)|$, and let $\Delta = \max_{1 \leq i \leq m} \{\Delta_i\}$. If $\Delta_i < \Delta$, he pads the set $DB(w_i)$ with $\Delta - \Delta_i$ elements $0^p$ such that $|DB(w_i)| = \Delta$, where $i = 1, \ldots, m$.

He continues to do:

$$e_{w_i} = \delta.\text{Enc}(l_{w_i}, DB(w_i)), \quad (14)$$

For $w_1$, he generates a transaction $TX_{w_{10}}$ as follows:

i. He finds an unredeemed transaction $TX_{w_{10}}$ from his wallet, which contains $d_{w_{10}}$ coins.

ii. Compute $K_{11} = F_2(K_{2,0}^p)$ and $r_1 = \delta.\text{Enc}(K_{11}, t_{w_1} \parallel e_{w_1} \parallel h_{w_1} \parallel 0^p)$.

iii. Embed $r_1$ in the out-script of $TX_{w_{11}}$.

iv. Take transaction $TX_{w_{10}}$ as input, and compute the body of transaction $TX_{w_{11}}$.

v. He signs the transaction $TX_{w_{11}}$, and broadcasts it on the blockchain.

vi. After it appears on the blockchain, he computes its identifier:

$$TI_{w_1} = H(TX_{w_{11}}).$$

vii. If transaction $TX_{w_{11}}$ does not appear on the blockchain, the data owner can redeem transaction $TX_{w_{10}}$ quickly and quits the protocol.

For $w_j \in W$ ($2 \leq j \leq m$), the data owner builds transaction $TX_{w_{j0}}$ as follows:

i. Find an unredeemed transaction $TX_{w_{j0}}$ from his wallet, which contains $d_{w_{j0}}$ coins.

ii. Compute $K_{11} = F_2(K_{2,0}^p)$, and $r_j = \delta.\text{Enc}(K_{11}, t_{w_j} \parallel e_{w_j} \parallel h_{w_j} \parallel TI_{w_{j-1}})$.

iii. Embed $r_j$ in the out-script of $TX_{w_{j1}}$.

iv. Input transaction $TX_{w_{j0}}$, and compute the body of transaction $TX_{w_{j1}}$.

v. Sign transaction $TX_{w_{j1}}$ by using $U'.sk$, and broadcast it on the blockchain.

vi. If the transaction $TX_{w_{j1}}$ appears on the blockchain, he records its identifier

$$TI_{w_j} = H_i(TX_{w_{j1}}).$$

vii. If transaction $TX_{w_{j1}}$ does not appear on the blockchain, the data owner can redeem transaction $TX_{w_{j0}}$ quickly and quits the protocol.

The data owner needs to store $TI_{w_j}$ locally.

Let $\Phi(\cdot, \cdot)$ be the function defined in section 4.1.
c) **Trpdr:** When finding data that contain the keyword \( w \). He needs to create a transaction \( ask \), which is shown in Fig. 7:

i. Find an unredeemed transaction \( T_q \) from his wallet, which contains \( d_i \) coins.

ii. Compute \( t_w = F_1(K_{2, w}), l_w = F_2(K_{2, w}), K_{11} = F_3(K_2, \vartheta) \) and \( k_w = F_3(K_2, w) \).

iii. Embed \( \Phi((t_w, l_w, k_w), K_{11}, TI_{w_a}) \) into the out-script of \( ask \).

iv. To compute the body of transaction \( ask \), he inputs transaction \( T_q \).

v. Taking the transaction \( ask \) as input, for transaction \( Fuse \), the data owner \( U' \) and the server \( S \) compute its body. This transaction \( Fuse \) contains a time \( t \). The server \( S \) signs transaction \( Fuse \) and sends it to \( U' \).

vi. After signing the transaction \( ask \), the data owner \( U' \) broadcasts it.

vii. After time \( t_{maxU} \), if the transaction \( ask \) does not appear on the blockchain, the data owner \( U' \) redeems transaction \( T_q \) by using his private key and quits the protocol immediately, where \( maxU \) is the maximal possible delay of including it in the blockchain.

![Fig. 7. How to return the documents that contain keyword \( w \).](image)

d) **Search:** When the server \( S \) wants to redeem the transaction \( ask \) as shown in Fig. 7, it does:

1) Take transaction \( ask \) as input, and compute the body of transaction \( return \) transaction.

2) Run the function \( \Phi( t_w, l_w, k_w, K_{11}, TI_{w_a} ) \): Firstly, it uses \( TI_{w_a} \) to get the information \( r_m \) from transaction \( TX_{D_m} \). It then computes \( t_w \parallel e_{w_a} \parallel h_{w_a} \parallel TI_{w_a} = \delta . Dec(K_{11}, r_m) \). Next, it will do:

- If \( t_w = t_w \), it continues to do: \( DB_{w_a} = \delta . Dec(l_{w_a}, e_{w_a}) \). For brevity, let we use \( DB_{w_a} = \{ TXID_{D_m}, \ldots, TXID_{D_m} \} \) to denote the decryption results. It then finds ciphertext \( C_i \) by using \( TXID_{D_m} \) (1 \( \leq i \leq \Delta \)):

  i. In the transaction \( TX_{D_m} \), if it contains \( C_m \), it outputs it.
ii. If the information is $i \in i_{m_i s}^{\prime}$ in the transaction $TX_{D_{m_i s}}$, it firstly outputs $C_{m_i s}^{\prime}$, and then uses identifier $TXID_{D_{m_i s}}$ to get the information $D_{m_i j}^{\prime} (j=s-1, \cdots, 1)$ from transaction $TX_{D_{m_i j}}^{\prime} (j=s-1, \cdots, 1)$. Lastly, it sets $C_{m_i s} = C_{m_i s}^{\prime} | \cdots | C_{m_i s}^{\prime}$.

- If $t_{w_s} \neq t_w$, it continues to use transaction identifier $T_{I_{w_s-j}}$ to read information $r_{m_j} (j=1, \cdots, m-1)$ from transaction $TX_{I_{w_s-j}} (j=1, \cdots, m-1)$. If $t_{w_s-j} = t_w$ holds, it stops. That is to say, he does:
  i. Decrypt $t_{w_s-j} || e_{w_s-j} || h_{w_s-j} || T_{I_{w_s-j}} = \delta Dec(K_{11}, r_{m-j})$,

  ii. Verify $t_{w_s-j} = t_w$. If this equation holds, he uses the above method to decrypt $DB_{w_s-j}$ to get $\{C_1, \cdots, C_{m_{j-1}}\}$. If it does not hold, he continues to read the information $r_{m-j}$ embedded in the transaction $TX_{I_{w_s-j-1}}$ until $t_{w_s-j} = t_w$ holds.

  3) Embed the $\{C_1, \cdots, C_{m_{j-1}}, h_{w_s}\}$ into the out-script of transaction return.

  4) After signing the transaction return, he broadcasts it.

e) Dec: After the transaction return appears on the blockchain, the data owner $U$ recovers $\{C_j\}$ from it. He continues to compute $D_j = e Enc(K_1, C_j)$ $(1 \leq j \leq n)$. After time $t$, if the transaction return still does not appear on the blockchain, he will broadcast transaction Fuse to get his money back.

### 5. Security and Performance Analysis

The idea of the scheme presented in section 4.1 is similar to that in section 4.2. The difference between them is that the latter needs to divide documents into blocks before uploading them on the blockchain. When search, the server needs to find all the appropriate blocks and merge them together. Here, we only present performance analysis and security analysis for the first scheme. For the second scheme, readers can derive them by themselves.

#### 5.1 Performance

Our computer configuration is Intel(R) Xeon(R) CPU E3-1230 v5 @ 3.40GHz, 32GB memory. We simulate our scheme on the Fabric with version number 1.4, which is stable. We create an orderer server, three organizations on it, and each organization has two peer nodes. That is to say, we build 6 peer nodes in the blockchain network. The size of the block is set to be 99MB. It takes about 2s to generate a block. We instantiate the pseudorandom functions $F_1, F_2, F_3$ with HMAC-SHA256, the hash function $H$ and $H_1$ with HMAC-SHA256, and SE schemes $\delta, e$ with AES in the CBC mode with a 256 bit key. We sample 9411 RFC files (400MB) from the IETF website (https://www.ietf.org/rfc/) and extract 600 keywords randomly. We then transform them in the form of array (keyword, file). The number of test data ranges from 1000 to $10^3$. 


To show the efficiency of our scheme, we will elaborate from the following points:

**Setup time.** It mainly means the time used to generate an invertible index. The time begins after the documents are uploaded to the blockchain, and ends after the index $I$ appears on the blockchain. The Fig. 8 shows the time to create an index for files with different scales. It is easy to get that as the size of the data grows, the time of creating an index is increasing.

**Search time.** This time includes the search token generation time of the keyword $w$, the time it takes to create a smart contract, and the time to find the files containing keyword $w$. Because the transaction $ask$ contains a function $\Phi$, we can use a smart contract to simulate it. In this smart contract, it contains decryption algorithm, for loop algorithm, and hash verification. Fig. 9 shows the result after it is created on the Fabric.

When searching the data containing the keyword $w$, the server needs to provide the transaction $return$ to complete it. We simulate it by invoke the smart contract that we built above. As shown in Fig. 10, we give its search time respectively under different scales of data.

**Table 1.** Comparison between verifiable SSE schemes. The $n$ denotes the total number of files, $m$, the number of the transactions used to store $n$ files on the blockchain, $r$, the size of indistinguishability obfuscation, and $d(w)$, the number of files containing the keyword $w$. As shown in it, schemes [25] is optimal.

| scheme | computation cost | communication cost | Fully against malicious adversary |
|--------|------------------|--------------------|----------------------------------|
| [25]   | $O(n)$           | $O(d(w))$          | No                               |
| [26]   | $O(n+r)$         | $O(d(w))$          | No                               |
| our scheme | $O(m)$         | $O(d(w))$          | Yes                              |

Fig. 8. The time it takes to create an index for data with different scales. The symbol $w$ represents a keyword, $D$ denotes a file.

Fig. 9. The information about the smart contract

Fig. 10. The time it takes to finish a search on the data with different sizes. The symbol $w$ represents a keyword, $D$ denotes a file.
However, it cannot resist fully malicious adversary, as well as scheme [26]. Though our scheme store files in the form of transaction, the size of each transaction is nearly equal to the size of the ciphertext which is stored in the transaction. That is to say, our schemes is also optimal.

5.2 Security Analysis
In this section, we give its security proof of our first scheme.

Theorem 1. If $F_1, F_2, F_3$ are pseudorandom functions, $H$ and $H_1$ are collision resistant hash function, and $\varepsilon = (\varepsilon.\text{Enc}, \varepsilon.\text{Dec})$ is PCPA-secure symmetric encryption scheme, then the scheme we present in section 4.1 is adaptively IND-CKA2 secure.

Proof. Let we construct a PPT simulator $S = \{S_0, S_1, \ldots, S_q\}$ such that, for an adversary $A = \{A_0, A_1, \ldots, A_q\}$, the output of $\text{Ideal}^{\Pi}_{\text{ind}}(k)$ and $\text{Real}^{\Pi}_{\text{ind}}(k)$ is computationally indistinguishable.

Suppose the simulator $S$ can get access to the trace of a history $L = (|T_1|, \ldots, |T_n|, |\text{Inx}|, \tau(TX_w))$ where $\tau(TX_w)$ denotes the search pattern and the access pattern about keyword $w$. It then generates $(\text{Inx}^*, T_1^*, \ldots, T_n^*, \text{Tr}^*, TS^*)$ and creates transaction $\text{ask}^*$ as follows:

i. Simulating $T_1^*, \ldots, T_n^*$.

If $q=0$, it can set $C_1^* \leftarrow \{0,1\}^{\lfloor C \rfloor}, \ldots, C_n^* \leftarrow \{0,1\}^{\lfloor C \rfloor}$.

Because the encryption algorithm $\varepsilon = (\varepsilon.\text{Enc}, \varepsilon.\text{Dec})$ is PCPA-secure, it means that $C_1^*, \ldots, C_n^*$ are computationally indistinguishable from $C_1, \ldots, C_n$ coming from the $\text{Re}al^{\Pi}_{\text{ind}}(k)$ game. Moreover, the adversary $A$ does not have the private key, therefore, it cannot create valid transactions $T_1^*, \ldots, T_n^*$ which embeds $C_1^*, \ldots, C_n^*$ respectively. If it asks the simulator $S$ to sign these transaction, it will result in the transactions $T_1^*, \ldots, T_n^*$ that generated in the $\text{Real}^{\Pi}_{\text{ind}}(k)$ game.

ii. Simulating $\text{Inx}^*$.

If $q=0$, $S$ sets $t_w^* \leftarrow \{0,1\}^{k}$, $e_w^* \leftarrow \{0,1\}^{k}$, $h_w^* \leftarrow \{0,1\}^{k}$.

Therefore, the $t_w, e_w, h_w$ output by $\text{Enc}$ are computationally indistinguishable from $t_w^*, e_w^*, h_w^*$.

If $q \geq 1$, $S$ selects $l_{w_1}^* \leftarrow \{0,1\}^{k}$, $k_{w_1}^* \leftarrow \{0,1\}^{k}$, and does $e_{w_1}^* = \delta.\text{Enc}(l_{w_1}^*, DB^*(w_1))$, $h_{w_1}^* = H(k_{w_1}^*, C_{w_1}^* \| \ldots \| C_{w_1}^*)$.

Because $F_2, F_3$ are pseudorandom functions, the $(e_{w_1}^*, h_{w_1}^*)$ is computationally indistinguishable from $(e_{w_1}, h_{w_1})$ generated from the step $\text{Enc}$.

Because function $F_1$ is pseudorandom, the $t_{w_1}^*$ output by $\text{Enc}$ is computationally indistinguishable from $t_{w_1}^*$ which is chosen at random from $\{0,1\}^{k}$.

Therefore, $\text{Inx}^*$ is computationally indistinguishable from $\text{Inx}$. 


iii. Simulating \( Tr^* \).

In the transaction \( Tr^* \), it embeds \( t^*_w \) and \( TX_{\text{inc}} \). Because \( TX_{\text{inc}} \) is broadcasted to each other, \( A \) can get it easily. Here we only consider \( t^*_w \) is indistinguishable from \( t_w \). It uses the pseudorandom function \( F_1 \) to generate \( t_w \) for keyword \( w \) in the step \( Tr_{\text{pdr}} \) in the section 4.1, and \( t_w \) is indistinguishable from \( t^*_w \) that \( S \) chooses at random. Therefore, \( Tr^* \) is computationally indistinguishable from \( Tr \).

iv. Claiming the transaction \( ask \) by using transaction \( s \).

When \( q=0 \), if \( A \) wants to get the money from the transaction \( s^* \). \( S \) returns \((\{C_{i1}, \cdots, C_{in}\}, h_w)\) to \( A \), where \( C_{ij} \leftarrow \{0,1\}^k (j=1, \cdots, n) \) and \( h_w \leftarrow \{0,1\}^k \).

When \( q \geq 1 \), \( S \) firstly returns \((\{C_{wq1}, \cdots, C_{wqn}\})\) to \( A \), where \( C_{wqj} (j=1, \cdots, n) \) is the history of access pattern about keyword \( w_q \). \( S \) then sets \( k^*_{wq} \leftarrow \{0,1\}^k \) and computes \( h^*_{wq} = H(k^*_{wq}, C_{wq1} \cdots C_{wqn}) \) which will be sent to \( A \). Because \( F_3 \) is a pseudorandom function, therefore the transaction \( s \) that \( A \) creates cannot claim the money from transaction \( ask \).

6. Conclusion

This paper provided a search method for encrypted data on the blockchain, and constructed two concrete search algorithms based on the size of data. We also give its security and performance analyses. Compared to the existing SSE schemes, our scheme can automatically resist malicious adversary. In addition, the server only needs to find the document transactions which are related to the keyword \( w \), therefore, our search complexity is sub-linear with the total number of documents. Since our scheme can better protect the privacy and integrity of data, it can be applied in many industries, such as medical healthcare, insurance and finance.

At present, the blockchain is still in its infancy, and it only supports static data. Therefore, how to design a scheme supports data update and search on it is very interesting. This is also our next work.

References

[1] D. X. Song, D. Wagner, and A. Perrig, “Practical techniques for searches on encrypted data,” in Proc. of 2000 IEEE Symposium on Security and Privacy, IEEE, pages 44–55, 2000. Article (CrossRef Link)

[2] E. J. Goh, “Secure indexes,” IACR Cryptology ePrint Archive, 2003. http://eprint.iacr.org/2003/216

[3] R. Curtmola, J. A. Garay, S. Kamara, and et al., “Searchable symmetric encryption: improved definitions and efficient constructions,” in Proc. of the 13th ACM conference on Computer and communications security, ACM, pages 79–88, 2006. Article (CrossRef Link)

[4] P. Golle, J. Staddon, and B. Waters, “Secure conjunctive keyword search over encrypted data,” International Conference on Applied Cryptography and Network Security, Springer, pages 31–45, 2004. Article (CrossRef Link)

[5] T. Moataz and A. Shikfa, “Boolean symmetric searchable encryption,” in Proc. of the 8th ACM SIGSAC symposium on Information, computer and communications security, ACM, pages 265–276, 2013. Article (CrossRef Link)
[6] D. Cash, S. Jarecki, C. S. Jutla, and so on, “Highly-scalable searchable symmetric encryption with support for boolean queries,” *Advances in Cryptology–CRYPTO 2013*, Springer, pages 353–373, 2013. [Article (CrossRef Link)]

[7] S. Kamara, T. Moataz, “Boolean searchable symmetric encryption with worst-case sub-linear complexity,” in *Proc. of Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, pages 94–124, 2017. [Article (CrossRef Link)]

[8] J. Li, Q. Wang, C. Wang, and et al., “Fuzzy keyword search over encrypted data in cloud computing,” in *Proc. of INFOCOM*, 2010 Proceedings IEEE, pages 1–5, 2010. [Article (CrossRef Link)]

[9] A. Boldyreva, N. Chenette, “Efficient fuzzy search on encrypted data,” *Fast Software Encryption*, Springer, pages 613–633, 2014. [Article (CrossRef Link)]

[10] W. K. Wong, D. W. Cheung, B. Kao, and et al., “Secure knn computation on encrypted databases,” in *Proc. of the 2009 ACM SIGMOD International Conference on Management of data*, ACM, pages 139–152, 2009. [Article (CrossRef Link)]

[11] N. Cao, C. Wang, M. Li, and et al., “Privacy-preserving multi-keyword ranked search over encrypted cloud data,” *IEEE Trans. Parallel Distrib. Syst.*, 25(1), 222–233, 2014. [Article (CrossRef Link)]

[12] Z. J. Fu, F. X. Huang, K. Ren, and et al., “Privacy-preserving smart semantic search based on conceptual graphs over encrypted outsourced data,” *IEEE Trans. Information Forensics and Security*, 12(8):1874–1884, 2017. [Article (CrossRef Link)]

[13] P. V. Liesdonk, S. Sedghi, J. Doumen, and et al., “Computationally efficient searchable symmetric encryption,” in *Proc. of Workshop on Secure Data Management*, Springer, pages 87–100, 2010. [Article (CrossRef Link)]

[14] S. Kamara, C. Papamanthou, and T. Roeder, “Dynamic searchable symmetric encryption,” in *Proc. of the 2012 ACM conference on Computer and communications security*, ACM, pages 965–976, 2012. [Article (CrossRef Link)]

[15] K. Kurosawa and Y. Ohtaki, “How to update documents verifiably in searchable symmetric encryption,” in *Proc. of Cryptology and Network Security-12th International Conference*, CANS 2013, pages 309–328, Paraty, Brazil, November 20-22, 2013. [Article (CrossRef Link)]

[16] D. Cash, J. Jaeger, S. Jarecki, and et al., “Dynamic searchable encryption via blind storage,” *Security and Privacy (SP), 2014 IEEE Symposium on*, IEEE, pages 639–654, 2014. [Article (CrossRef Link)]

[17] C. Guo, X. Chen, Y. M. Jie, and et al., “Dynamic multi-phrase ranked search over encrypted data with symmetric searchable encryption,” *IEEE Trans. Services Computing*, 2017. [Article (CrossRef Link)]

[18] E. Stefanov, C. Papamanthou, and E. Shi, “Practical dynamic searchable encryption with small leakage,” *NDSS*, volume 14, pages 23–26, 2014. [Article (CrossRef Link)]

[19] R. Bost, P. A. Fouque, and D. Pointcheval, “Verifiable dynamic symmetric searchable encryption: Optimality and forward security,” *IACR Cryptology ePrint Archive*, 2016:62, 2016. [http://eprint.iacr.org/2016/062](http://eprint.iacr.org/2016/062)

[20] R. Bost, “Σφορς: Forward Secure Searchable Encryption,” in *Proc. of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ACM, pages 1143–1154, 2016. [Article (CrossRef Link)]

[21] S. Kamara, C. Papamanthou, “Parallel and dynamic searchable symmetric encryption,” in *Proc. of International Conference on Financial Cryptography and Data Security*, Springer, pages 258–274, 2013. [Article (CrossRef Link)]

[22] J. Alderman, K. Martin, and S. Louise Renwick, “Multi-level access in searchable symmetric encryption,” in *Proc. of International Conference on Financial Cryptography and Data Security*, Springer, pages 35-52, 2017. [Article (CrossRef Link)]
[24] C. Bosch, P. Hartel, W. Jonker, and et al., “A survey of provably secure searchable encryption,” ACM Computing Surveys (CSUR), 47(2), 18, 2015. Article (CrossRef Link)

[25] K. Kurosawa and Y. Ohtaki, “UC-secure searchable symmetric encryption,” Financial Cryptography and Data Security, Springer, pages 285–298, 2012. Article (CrossRef Link)

[26] R. Cheng, J. Yan, C. Guan, and et al., “Verifiable searchable symmetric encryption from indistinguishability obfuscation,” in Proc. of the 10th ACM Symposium on Information, Computer and Communications Security, ASIA CCS ’15, pages 621–626, Singapore, April 14-17, 2015. Article (CrossRef Link)

[27] S. G. Dai, H. G. Li, and F. G. Zhang, “Memory leakage-resilient searchable symmetric encryption,” Future Generation Comp. Syst., 62, 76–84, 2016. Article (CrossRef Link)

[28] H. G. Li, F. G. Zhang, and C. I. Fan, “Deniable searchable symmetric encryption,” Information Sciences, 402:233–243, 2017. Article (CrossRef Link)

[29] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, 2008. Article (CrossRef Link)

[30] M. Swan, “Blockchain: Blueprint for a new economy [M],” O’Reilly Media, Inc., 2015.

[31] M. Andrychowicz, S. Dziembowski, D. Malinowski, and et al., “Fair two-party computations via bitcoin deposits,” in Proc. of International Conference on Financial Cryptography and Data Security, Springer, pages 105–121, 2014. Article (CrossRef Link)

[32] D. Chaum, “Blind signatures for untraceable payments,” Advances in cryptology, pages 199–203, Springer, 1983. Article (CrossRef Link)

[33] D. Chaum, “Blind signature system,” Advances in cryptology, Springer, pages 153–153, 1984. Article (CrossRef Link)

[34] M. Andrychowicz, S. Dziembowski, D. Malinowski, and et al., “Secure multiparty computations on bitcoin,” in Proc. of 2014 IEEE Symposium on Security and Privacy, IEEE, pages 443–458, 2014. Article (CrossRef Link)

Huige Li received her M.S degree from the School of Mathematics and Information Science, Shaanxi Normal University in 2013. She is currently reading for her Ph.D. at the school of Electronics and Information Technology of Sun Yat-sen University, China. Her research focuses on Searchable Encryption.

Fangguo Zhang received his Ph.D. from the School of Communication Engineering, Xidian University in 2001. He is currently a Professor at the School of Data and Computer Science of Sun Yat-sen University, China. He is the co-director of Guangdong Key Laboratory of Information Security Technology. His research mainly focuses on cryptography and its applications.
**Peiran Luo** received his B.E. degree from the School of Computer, South China Normal University in 2018. At present, he is reading for his M.E. at the school of Data and Computer Science of Sun Yat-sen University, China. His research focuses on Blockchain and its applications.

**Haibo Tian** received his Ph.D. from the School of Communication Engineering, Xidian University in 2006. He is currently an associate Professor at the School of Data and Computer Science of Sun Yat-sen University, China. His research mainly focuses on security protocol analysis and its design.

**Jiejie He** received his B.E. degree from the School of Information Technology, Minzu University of China in 2015. At present, he is reading for his M.E. at the school of Data and Computer Science of Sun Yat-sen University, China. His research focuses on Blockchain and its applications.