Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2021.015210

ARTICLE

Attribute-Based Keyword Search over the Encrypted Blockchain

Zhen Yang[1], Hongao Zhang[1], Haiyang Yu[1,*], Zheng Li[1], Bocheng Zhu[1] and Richard O. Sinnott[2]

[1]Faculty of Information Technology, Beijing University of Technology, Beijing, 100124, China

[2]School of Computing and Information Systems, The University of Melbourne, Victoria, 3010, Australia

[*]Corresponding Author: Haiyang Yu. Email: yuhaiyang@bjut.edu.cn

Received: 02 December 2020 Accepted: 15 March 2021

ABSTRACT

To address privacy concerns, data in the blockchain should be encrypted in advance to avoid data access from all users in the blockchain. However, encrypted data cannot be directly retrieved, which hinders data sharing in the blockchain. Several works have been proposed to deal with this problem. However, the data retrieval in these schemes requires the participation of data owners and lacks finer-grained access control. In this paper, we propose an attribute-based keyword search scheme over the encrypted blockchain, which allows users to search encrypted files over the blockchain based on their attributes. In addition, we build a file chain structure to improve the efficiency of searching files with the same keyword. Security analysis proves the security of the proposed scheme. Theoretical analysis and experimental results in performance evaluation show that our scheme is feasible and efficient.

KEYWORDS

Blockchain; searchable encryption; attribute-based encryption; smart contract

1 Introduction

As an emerging decentralized technology, blockchain has gained attention worldwide over the past decade, which can be applied to many scenarios such as IoT, 5G networks [1], etc. Among the many appealing features of blockchain, highly redundant storage enables each user to possess a full copy of the blockchain, which means users can share their data with anyone in the blockchain network directly for many data driven applications [2]. This feature makes the blockchain a perfect technique for constructing data sharing schemes. However, since data is public to all users, it can be accessed by both data users and external (undesirable) users in the blockchain network, thus giving rise to privacy issues and malicious attacks [3]. To achieve confidentiality, data owners usually opt to encrypt their data before submitting it to the blockchain. Although encryption protects the data from being deciphered and understood by anyone in the network, the encrypted files hinder data users from conducting search operations on the blockchain. A search mechanism over the encrypted blockchain is thus desirable to achieve both data confidentiality and usability.

Several works [4–8] have focused on keyword search over the encrypted blockchain. However, existing schemes mostly adopted symmetric encryption and thereby have several major drawbacks.

Firstly, since data users have to interact with each data owner before searching the data, these schemes cannot support efficient data retrieval over the whole blockchain. Secondly, interactions require data owners and users to share either the secret key or keyword, which can give rise to privacy erosion. Thirdly, since the authorization is entirely dependent on data owners, access control [9] in searching the blockchain is not supported. As an example of these issues, consider the following situation. There are a lot of users in the blockchain network belonging to the same university or company who have similar features or attributes and want to search shared data within the group (and only within the group). Contacting data owners individually to get the search permission makes the data sharing process highly inflexible and inefficient.

Traditional symmetric encryption schemes cannot address these issues well. Therefore, we propose an attribute-based keyword search scheme on blockchain. Data users can get the secret key from a trusted third party and use it to search on the blockchain without any interactions from the data owner, which avoids the possible privacy leakage of users. By improving the structure of the encrypted index, we achieved fine-grained access control and efficient search, which allows users to search encrypted files over the blockchain based on their attributes efficiently and accurately.

In summary, our contributions include:

(1) We proposed an attribute-based keyword search scheme enabling the searching of encrypted data over blockchains. By introducing Attribute-Based Encryption (ABE), the proposed scheme can achieve data retrieval over the whole blockchain with support for fine-grained access control.
(2) Motivated by the blockchain data structure, we design a data structure that is comprised of chained files with the same keyword to address the efficiency of encrypted data retrieval over blockchains.
(3) We analyze the security of our design and further implement a prototype of the proposed scheme on the Hyperledger Fabric platform. Theoretical analysis and experimental evaluation show the efficiency and feasibility of the proposed scheme.

2 Related Work

The concept of searchable encryption was first proposed by Song et al. [10] in 2000. This scheme could search encrypted data stored in remote servers by leveraging symmetric encryption methods. In 2004, Boneh et al. [11] first proposed a public-key encryption scheme with keyword search. However, the keyword privacy was not protected by this scheme. Subsequently, extensive research has been carried out on both symmetric [12–15] and asymmetric searchable encryption [16–19].

Searchable encryption has been well studied in cloud environments, however, limited research has been conducted on searching keywords on blockchains. Li et al. [20] presented a searchable encryption scheme using blockchain, which utilized blockchain techniques to ensure transaction security during the process of outsourced data retrieval. Cai et al. [21] designed an encrypted and distributed search protocol. This scheme considered submitted the retrieval results to the blockchain to achieve verifiability. Zhang et al. [22] proposed a trustworthy keyword search scheme over encrypted data based on blockchain. They adopted blockchains to ensure the security and fairness of payment for data retrieval. However, all these schemes did not search encrypted data over the blockchain, rather they only enhanced the reliability of transaction and payment occurrences during data retrieval over encrypted blockchains. Jiang et al. [7] proposed a stealth

authorization scheme on blockchains to achieve private authorization delivery between data users and data owners, but the data user still had to inform the data owner of the search keywords. Tian et al. [23] designed a storage framework based on blockchain to protect digital data, which ensures the integrity and validity of evidence and better balances privacy and traceability. Yang et al. [8] proposed a blockchain based multi-keyword ranked search system, which realized the top-k ranked ciphertext retrieval on blockchain and used smart contracts to only return the most relevant encrypted files to data users. However, this scheme limited the structure of the index and therefore limited its scalability; it also increased the size of the index. Tahir et al. [24] proposed a privacy preserving searchable encryption framework to achieve data retrieval over encrypted blockchains. By introducing a probabilistic trapdoor, this scheme was able to protect the privacy of keywords. Unfortunately, this scheme only supported data retrieval by the data owner. Hu et al. [4] considered the use of smart contracts to support data retrieval. However, this scheme required the data user to send keywords to the data owner and the data owner would then search the dataset on behalf of the data user due to the adoption of symmetric searchable encryption. When considering searching over blockchains, the data user even needed to send keywords to all data owners, which is neither efficient nor privacy-preserving. Therefore, none of the above schemes meet the needs of data owners who want to share their data and achieve fine-grained access control while users who want to search independently on the blockchain without relying on the data owner.

3 Preliminaries

3.1 Bilinear Pairing

Let G_1 and G_T be two multiplicative cyclic groups of large prime order q. Denote g and g_T as the generators of G_1 and G_T, respectively. A bilinear map is a map e: $G_1 \times G_1 \rightarrow G_T$ which satisfies the following three properties:

1) Bilinearity:

for $u, v \in G_1$ and $a, b \in Z_p^*$, $e\left(u^a, v^b\right) = e(u, v)^{ab}$;

2) Non-degeneracy:

$\exists g \in G_1$ such that $e(g, g) \neq 1$;

3) Computability:

there exists an efficient algorithm to compute the map e.

3.2 System Model

The system model of the proposed scheme consists of three entities, as illustrated in Fig. 1: the attribute authority (AA), the data owner and the data user. An AA is a trusted third party responsible for distributing secret keys to users in the blockchain based on their attributes. Data owners possess files that can be shared within a group of data users with similar attributes. A data owner generates encrypted indexes and submits them to the blockchain. The data user encrypts the searching keyword and sends an encrypted token to the blockchain to retrieve the files it wants. The blockchain system is maintained and shared by all data owners and data users. A smart contract in the blockchain retrieves files and returns the search results to the data user.

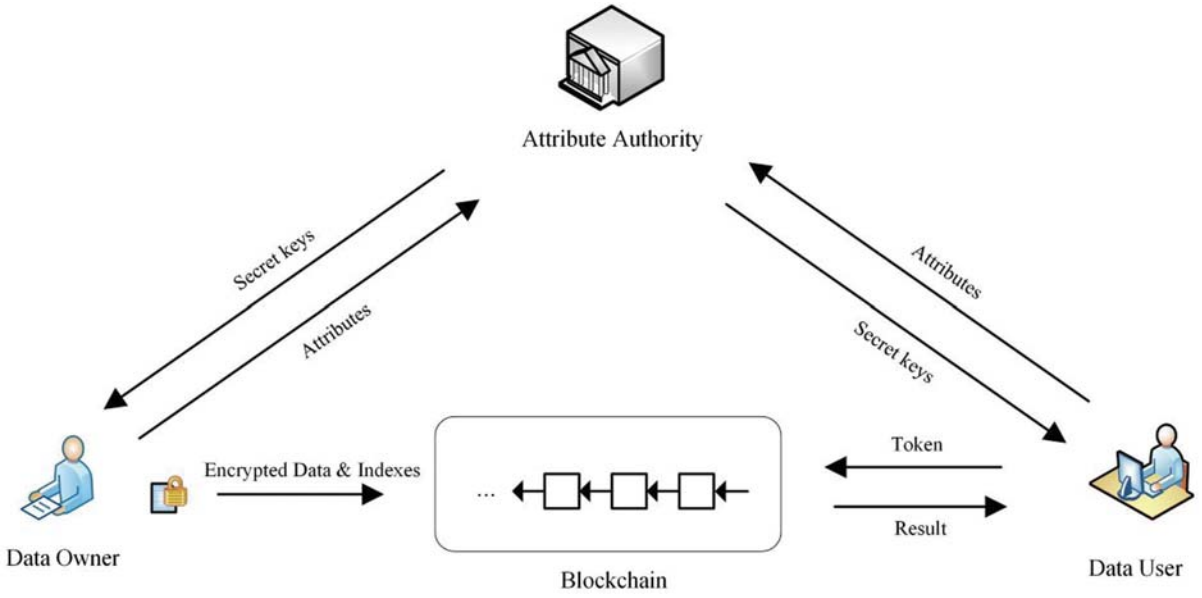

Figure 1: The system model of the proposed scheme including three entities: the attribute authority (AA), the data owner and the data user

3.3 Design Goals

The design goals of the proposed scheme are as follows:

1) Flexibility:

The data user can search files of all data owners in the blockchain without interacting individually with each data owner.

2) Attributed-Based Retrieval:

The encrypted files can only be retrieved and decrypted by data users with specific security attributes.

3) Reliability:

The search should return all matched results to the data user and the results should be the same each time the same query is executed.

4) Efficiency:

The retrieval performance should be higher than traditional traversal search in the blockchain.

3.4 Definition of the Proposed Scheme

Definition 1. *The proposed scheme is comprised of six polynomial time algorithms including Setup, KeyGen, IndexGen, TokenGen, Search and Decrypt.*

$Setup\left(1^k\right) \rightarrow (msk, pm)$. The *Setup* algorithm takes a security parameter k as the input and returns the master key msk and the public parameters pm.

$KeyGen\,(msk, A_s) \rightarrow sk$. The *KeyGen* algorithm takes the master key msk and the attributes A_s as inputs and outputs the secret key sk.

IndexGen $(kw, T, F) \rightarrow \left(cph, \tilde{F}\right)$. The *IndexGen* algorithm requires the keyword kw, the access tree T, and the file F as inputs and returns the encrypted index cph and the encrypted file $\tilde{F}$.

TokenGen $(sk, kw) \rightarrow tk$. The *TokenGen* algorithm takes the secret key sk and the search keyword kw as inputs, and outputs the search token tk.

Search $(tk, cph) \rightarrow rst$. The *Search* algorithm takes the search token tk and encrypted index cph as inputs and returns the search result rst.

Decrypt $(rst) \rightarrow \{F\}$. The *Decrypt* algorithm takes the search result rst as the input and returns the file F.

3.5 Security Model

In this paper, the Selective-Set Security Game [25,26] is introduced to prove the security of the proposed scheme. The detailed game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ is defined as follows:

Init. An adversary $\mathcal{A}$ sends a challenge access policy with a set of attributes S to the challenger $\mathcal{C}$.

Setup. The challenger $\mathcal{C}$ runs the Setup algorithm to generate the public parameters that are forwarded to the adversary $\mathcal{A}$.

Phase 1. By submitting any access trees with attribute set A_s' to the challenger $\mathcal{C}$, the adversary $\mathcal{A}$ is allowed to request the secret key based on the attributes. The restriction is that $A_s \nsubseteq A_s'$.

Challenge. The adversary $\mathcal{A}$ forwards two different keywords *kw1, kw2* to the challenger $\mathcal{C}$. The challenger $\mathcal{C}$ randomly flips a coin $\mu \in \{0, 1\}$ and calculates the encrypted index cph_μ. The challenger $\mathcal{C}$ returns cph_μ to the adversary $\mathcal{A}$.

Phase 2. Same as Phase 1.

Guess. Based on the forementioned steps, the adversary $\mathcal{A}$ outputs the guess μ' of μ.

We say that the proposed scheme is secure if for any probabilistic polynomial time adversary $\mathcal{A}$, the advantage of winning the Selective-Set Security Games is negligible.

$$Adv_{\mathcal{A}} = Pr\left[\mu' = \mu\right] - \frac{1}{2}$$

4 Attribute-Based Keyword Search over Encrypted Blockchain

4.1 File Chain Structure

In a lot of cases, there may be several files with the same keyword when searching files in the blockchain. For instance, a stock analyst in a security company may submit several reports related to the company Apple to his boss. Thus, all these files may have the same keyword 'Apple.' A data owner can build a structure in advance to address this issue and improve the overall retrieval performance. Motivated by the data structure of chained blocks in blockchain, we designed a File Chain Structure (FCS) as shown in Fig. 2. An FCS is a chain structure links all files with the same keyword in the blockchain. Each node in the FCS contains the address of previous file and an encrypted file. All nodes linked end to end comprise the FCS.

The FCS can simplify the retrieval process. When users search on any file with certain keywords in the FCS, the smart contract only needs to search an index table and then traverse

the FCS to retrieve all files with the same keyword instead of searching for other files by different tokens.

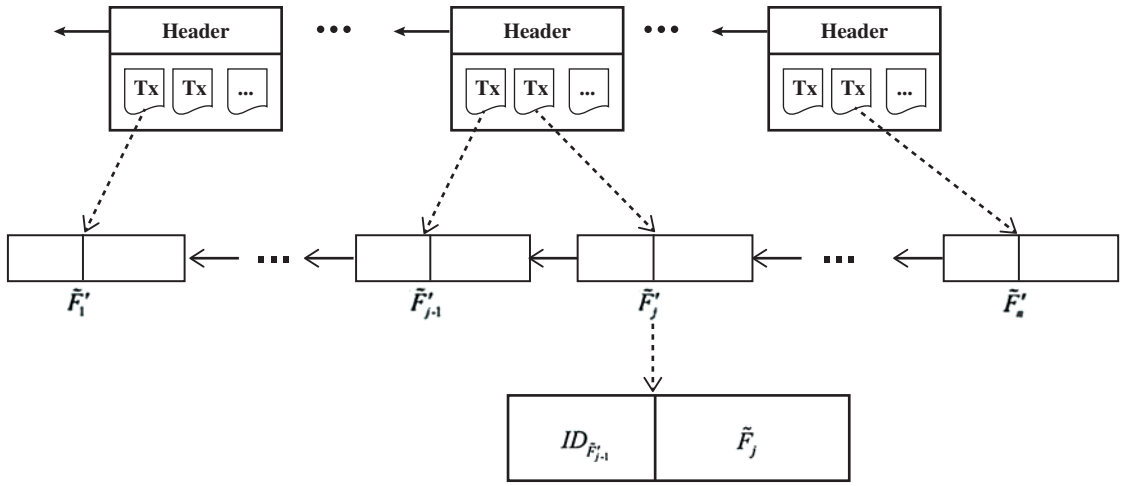

Figure 2: Example of the file chain structure, each node in the FCS contains the address of the previous file and an encrypted file

4.2 The Detailed Construction

As noted, the proposed scheme consists of six algorithms: *Setup*, *KeyGen*, *IndexGen*, *Token-Gen*, *Search*, *Decrypt*.

Setup. k denotes the security parameter. Let G_1 and G_T be two multiplicative cyclic groups of prime order q. Let g be the generator of group G_1. H: $\{0, 1\}^* \rightarrow G_1$ and h: $\{0, 1\}^* \rightarrow Z_p^*$ denotes two hash functions. AA randomly chooses $a, b, c, x \in G_1$, sets the master secret key as $msk = (a, b, c, x)$, and computes $E_x = e(g, g)^x$. Finally, AA publishes the public parameters and keeps the master secret key msk private.

KeyGen. AA generates a random value $r \in Z_p^*$, computes $A = g^{(ac-r)/b}$ and $B = e(g, g)^{x+r}$. Let A_s be the attribute set of a data owner or user. For each attribute $att_i \in A_s$, AA chooses a random value $r_i \in Z_p^*$, computes $M_i = g^r H(att_i)^{r_i}$ and $N_i = g^{r_i}$. AA generates the keyword search key $sk = \{A_s, A, B, \{M_i, N_i \mid att_i \in A_s\}\}$ and returns it to the corresponding data owner or data user.

IndexGen. The data owner first generates a keyword kw_j for each file F_j and then creates the secret key ck for symmetric encryption and encrypts the file F_j to generate $\tilde{F}_j$. Let $\{\tilde{F}_j\}_{j\in[1,n]}$ be a set of files containing the same keyword, where n is the number of files. For $j = 1$, it sets $\widetilde{F}'_1 = null \,||\, \tilde{F}_1$, where $ID_{\widetilde{F}'_n} = h\left(\tilde{F}_n\right)$ is the address of $\widetilde{F}'_n$. For $j = 2, \ldots, n$, the data owner calculates $\widetilde{F}'_j = ID_{\tilde{F}_{j-1}} \,||\, \tilde{F}_j$.

The data owner then chooses random values $r_1, r_2 \in Z_p^*$, calculates $W_0 = g^{cr_1}$, $W_1 = g^{a(r_1+r_2)} g^{bh(kw)r_1}$, $W_2 = g^{br_2}$, $C_0 = ck \cdot E_x^{r_2}$, $C_1 = B^{r_2}$. It constructs the access tree T based on the access policy. The access tree includes a set of attributes T_s where each leaf node represents an attribute. $att(v)$ denotes the attribute of the leaf node v in the access tree T. By using the access

tree T, the data owner shares the secret value r_2 into the leaf nodes. The secret value of each leaf node v is $q_v(0)$, where $q_v(x)$ is the polynomial used in the access tree T. For each leaf node $v \in T$, it computes $W_v = g^{q_v(0)}$ and $D_v = H(att(v))^{q_v(0)}$. Finally, the data owner gets a ciphertext keyword $cph = \{T, W_0, W_1, W_2, C_0, C_1, \{(W_v, D_v) \mid att(v) \in T_s\}\}$ and submits the transaction $T_x = \{cph, \tilde{F}\}$ to the blockchain.

TokenGen. The data user randomly chooses $s \in Z_p^*$ and calculates $tk_1 = \left(g^a g^{bh(kw)}\right)^s$, $tk_2 = g^{cs}$, $tk_3 = A^s$. For each attribute $att_i \in A_s$, the data user computes $M_i' = M_i^s$ and $N_i' = N_i^s$. Finally, the data user generates search token $tk = \left\{tk_1, tk_2, tk_3, A_s, \left\{\left(M_i', N_i'\right) \mid att_i \in A_s\right\}\right\}$.

Search. The data user invokes the search smart contract S_c with the search token tk. The smart contract S_c will search over each ciphertext keyword cph based on the search token tk. The smart contract S_c first checks if there exists a subset of attribute set A_s satisfying the access tree T in ciphertext keyword cph. If not, it will return an empty set. Otherwise, for each attribute $att_i = att(v) \in T_s$, it computes $E_v = e\left(M_i', W_v\right)/e\left(N_i', D_v\right) = e(g, g)^{rsq_v(0)}$. The smart contract S_c then calculates the secret values of the nodes of the access tree T from bottom to top, until recovering the secret value of root node $E_r = e(g, g)^{rsq_r(0)} = e(g, g)^{rsr_2}$. It then checks if

$$e(W_0, tk_1)\, E_r e(tk_3, W_2) = e(W_1, tk_2) \tag{1}$$

If the equation holds, it splits the file $\widetilde{F}_j'$ to get the address $ID_{\tilde{F}_{j-1}}$ and the encrypted file $\tilde{F}_j$. If $null \leftarrow ID_{\tilde{F}_{j-1}'}$, it stops. If $\widetilde{F}_{j-1}' \leftarrow ID_{\tilde{F}_{j-1}'}$, it stores $ID_{\tilde{F}_{j-1}'}$ and continues traversing the files in the FCS. Finally, the smart contract Sc returns the search results $rst = \{\{ID_{\tilde{F}_j'}\}, E_r, C_0, C_1\}$.

Decrypt. The data user invokes the smart contract S_c to read the rst. It then computes $C_0/\left(C_1 \cdot E_r^{(1/s)}\right) = ck$ to get the symmetric encryption key ck. Finally, it decrypts the files included in rst and gets the plaintext files $\{F\}$.

The algorithms in our scheme are described in Fig. 3.

4.3 Correctness

In this section, we illustrate the proof of correctness of Eq. (1) as follows:

$$\begin{aligned} e(W_0, tk_1) &= e\left(g^{cr_1}, \left(g^a g^{bh(kw)}\right)^s\right) \\ &= e(g, g)^{scr_1(a+bh(kw))} \end{aligned} \tag{2}$$

$$E_r = e(g, g)^{rsr_2} \tag{3}$$

$$\begin{aligned} e(tk_3, W_2) &= e\left(A^s, g^{br_2}\right) \\ &= e\left(g^{s(ac-r)/b}, g^{br_2}\right) \\ &= e(g, g)^{r_2 s(ac-r)} \end{aligned} \tag{4}$$

Setup

1. $G_1 \leftarrow \{0,1\}^*$, $Z_p^* \leftarrow \{0,1\}^*$.
2. Random choose $a, b, c, x \in G_1$, set $msk \leftarrow (a, b, c, x)$.
3. $E_x \leftarrow e(g,g)^x$.

KeyGen

1. Generate a random value $r \in Z_p^*$
2. $A \leftarrow g^{(ac-r)/b}$, $B \leftarrow e(g,g)^{x+r}$.
3. For each attribute $att_i \in A_s$:
   a) Choose a random value $r_i \in Z_p^*$.
   b) $M_i \leftarrow g^r H(att_i)^{r_i}$, $N_i \leftarrow g^{r_i}$.
   c) Set $sk \leftarrow \{A_s, A, B, \{M_i, N_i | att_i \in A_s\}\}$.

IndexGen

1. For each file F_j:
   a) Generate a keyword kw_j, create the secret key ck for symmetric encryption.
   b) Encrypt the file F_j to generate $\tilde{F}_j$ by ck. Choose random values $r_1, r_2 \in Z_p^*$.
   c) For $j = 1$:
      i. $\tilde{F}_1' \leftarrow null || \tilde{F}_1$.
   d) For $j = 2, \ldots, n$:
      i. $ID_{\tilde{F}_n'} \leftarrow h(\tilde{F}_n)$.
      ii. $\tilde{F}_j' \leftarrow ID_{\tilde{F}_{j-1}} || \tilde{F}_j$.
   e) $W_0 \leftarrow g^{cr_1}$, $W_1 \leftarrow g^{a(r_1+r_2)} g^{bh(kw)r_1}$, $W_2 \leftarrow g^{br_2}$, $C_0 \leftarrow ck \cdot E_x^{r_2}$, $C_1 \leftarrow B^{r_2}$.
   f) For each leaf node $v \in T$:
      i. $W_v \leftarrow g^{q_v(0)}$, $D_v \leftarrow H(att(v))^{q_v(0)}$.
   g) Set $cph \leftarrow \{T, W_0, W_1, W_2, C_0, C_1, \{(W_v, D_v) | att(v) \in T_s\}\}$
   h) Set $T_x \leftarrow \{cph, \tilde{F}\}$ and sent it to the blockchain.

TokenGen

1. Randomly choose $s \in Z_p^*$.
2. $tk_1 \leftarrow \left(g^a g^{bh(kw)}\right)^s$, $tk_2 \leftarrow g^{cs}$, $tk_3 \leftarrow A^s$.
3. For each attribute $att_i \in A_s$:
   a) $M_i' \leftarrow M_i^s$, $N_i' \leftarrow N_i^s$.
4. Set $tk \leftarrow \{tk_1, tk_2, tk_3, A_s, \{(M_i', N_i') | att_i \in A_s\}\}$.

Search

Data user:

1. Sent the search token tk to the search smart contract S_c.

Smart contract:

1. Search over each ciphertext keyword cph. Assert there exists a subset of attribute set A_s satisfying the access tree T in cph.
2. For each attribute att_i=$att(v) \in T_s$:
   a) $E_v = e(M_i', W_v)/e(N_i', D_v) = e(g,g)^{rsq_v(0)}$.
3. Recover the secret value of root node $E_r = e(g,g)^{rsq_r(0)} = e(g,g)^{rsr_2}$.
4. Assert $e(W_0, tk_1)E_r e(tk_3, W_2) = e(W_1, tk_2)$.
5. Parse $\widetilde{F_j'}$ into $\left(ID_{\tilde{F}_{j-1}}, \tilde{F}_j\right)$, store $ID_{\tilde{F}_{j-1}}$.
6. While $ID_{\tilde{F}_{j-1}} != null$:
   a) Retrieve $\widetilde{F_j'}$ with $ID_{\tilde{F}_{j-1}}$. Parse $\widetilde{F_j'}$ into $\left(ID_{\tilde{F}_{j-1}}, \tilde{F}_j\right)$, store $ID_{\tilde{F}_{j-1}}$.
7. returns the search results $rst \leftarrow \{\{ID_{\tilde{F}'_j}\}, E_r, C_0, C_1\}$.

Decrypt

1. Invoke the smart contract to get rst.
2. $ck \leftarrow C_0 / \left(C_1 \cdot E_r^{(1/s)}\right)$.
3. Decrypt the ciphertext files $\{\tilde{F}\}$ into plaintext files $\{F\}$ by ck.

Figure 3: Construction of our search scheme

thus, we get

$$\begin{aligned} e(W_0, tk_1)\, E_r e(tk_3, W_2) &= e(g,g)^{sacr_2+sacr_1+scr_1 bh(kw)} \\ &= e(g, g)^{sc(a(r_2+r_1)+r_1 bh(kw))} \\ &= e\left(g^{(a(r_2+r_1)+r_1 bh(kw))}, g^{sc}\right) \\ &= e\left(g^{(a(r_1+r_2))} g^{(r_1 bh(kw))}, g^{cs}\right) \\ &= e(W_1, tk_2) \end{aligned} \tag{5}$$

5 Security Analysis

In this paper, the AA and the blockchain infrastructure are assumed to be reliable. Thus, we focus on the secure search related privacy requirements of the proposed scheme, which mainly involves the *keyword semantic security* and *token unlinkability*.

Definition 2. *(The DBDH Assumption). Let $u, v, w, z \in Z_p^*$ be the random values and g be the generator of G_1. The DBDH (Decisional Bilinear Diffie-Hellman) Assumption is that given a polynomial-time adversary $\mathcal{A}$, it cannot distinguish the tuple $(U = g^u, V = g^v, W = g^w, e(g, g)^{uvw})$ from the tuple $(U = g^u, V = g^v, W = g^w, e(g, g)^z)$ with non-negligible probability ε.*

Theorem 1. *Keyword semantic security. If there exists a probabilistic polynomial time adversary that can win the Selective-Set Security Game with non-negligible probability ε, then we can construct a simulator S to solve the DBDH problem with non-negligible probability $\frac{\varepsilon}{2}$.*

Proof. The DBDH challenger $\mathcal{C}$ first selects random values $g, u, v, w, z \in Z_q^*$ and a bilinear map e. Then it flips a fair coin k and if $k = 0$ it sets $U = g^u$, $V = g^v$, $W = g^w$, $Z = e(g, g)^{uvw}$. Otherwise, it sets $U = g^u$, $V = g^v$, $W = g^w$, $Z = e(g, g)^z$. The challenger $\mathcal{C}$ sends g, U, V, W, Z to the simulator $\mathcal{B}$. The simulator $\mathcal{B}$ executes the game by interacting with the adversary $\mathcal{A}$ as follows:

Init. The adversary $\mathcal{A}$ submits an attribute set S to the simulator $\mathcal{B}$.

Setup. The simulator $\mathcal{B}$ sets $x = x' + uv$, where x' is a random value in Z_q^* and then calculates $E_x = e(g,g)^{x'+uv} = e(U, V)\, e(g,g)^{x'}$. Finally, the public parameters are sent to the adversary $\mathcal{A}$.

Phase 1. The adversary $\mathcal{A}$ queries $\mathcal{B}$ for the secret key sk of any access structure T with an attribute set A'_s. The simulator $\mathcal{B}$ computes $B = e(g,g)^{x'+uv+r} = e(U, V)\, e(g,g)^{x'+r}$.

Challenge. The adversary $\mathcal{A}$ provides two keywords kw_1 and kw_2 with equal length to $\mathcal{B}$. The simulator $\mathcal{B}$ flips a fair coin $\mu \in \{0, 1\}$ and encrypts kw_μ with the access policy T and the secret keys. The simulator $\mathcal{B}$ chooses a random number r'_2 and sets $r_2 = r'_2 + w$. It then computes $W_2 = W^b g^{r'_2 b}$. Finally, it sets $C_0 = ck \cdot Z \cdot e(U, V)^{r'_2} \cdot e(g,g)^{x'(w+r'_2)}$ and $C_1 = Z \cdot e(U, V)^{r'_2} \cdot e(g, g)^{(x'+r)(w+r'_2)}$.

Phase 2. The same as Phase 1.

Guess. The adversary $\mathcal{A}$ outputs a guess μ' of μ. At the same time, the simulator $\mathcal{B}$ makes a guess k' of k based on the output of the adversary $\mathcal{A}$. If $\mu' \neq \mu$, the simulator $\mathcal{B}$ guesses $k' = 1$. Otherwise, it guesses $k' = 0$.

If $k = 1$, $Z = e(g, g)^z$, which means the adversary $\mathcal{A}$ cannot get any advantage of the game but a random guess. Therefore, $Pr[\mu' \neq \mu \mid v = 1] = \frac{1}{2}$. When $\mu' \neq \mu$, the simulator $\mathcal{B}$ outputs $k' = 1$ such that $Pr[k' = k \mid k = 1] = \frac{1}{2}$.

If $k = 0$, $Z = e(g, g)^{uvw}$, which means the adversary $\mathcal{A}$ will get a valid *cph*. Therefore, $Pr[\mu' = \mu \mid v = 1] = \frac{1}{2} + \varepsilon$. When $\mu' = \mu$, the simulator $\mathcal{B}$ outputs $k' = 0$ such that $Pr[k' = k \mid k = 0] = \frac{1}{2} + \varepsilon$.

Finally, the overall advantage of the simulator $\mathcal{B}$ for winning this game can be calculated as $Pr[k' = k] - \frac{1}{2} = \frac{1}{2} Pr[k' = k \mid k = 1] + \frac{1}{2} Pr[k' = k \mid k = 0] - \frac{1}{2} = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \left(\frac{1}{2} + \varepsilon\right) - \frac{1}{2} = \frac{\varepsilon}{2}$.

Theorem 2. *Token Unlinkability. Data users are unable to directly distinguish two or more search tokens even those containing the same keyword.*

Proof. The search information submitted to the blockchain should be encrypted before submitting since it can be accessed by all users. The data user encrypts the keyword *kw* into tk_1 with a random value *s*. In this way, other users cannot directly distinguish different search tokens because they are obfuscated by the random value *s*. Therefore, the proposed scheme can achieve token unlinkability.

6 Performance Evaluation

We implement a prototype of the proposed scheme using almost 3760 lines of code. We develop the blockchain system based on the Hyperledger Fabric blockchain platform. Both the data owner and the service peers were deployed on a machine with 8 GB of RAM, 2 Intel cores i5-8300H, running Ubuntu 20.04 LTS. The smart contract was implemented using chaincode in the Hyperledger Fabric platform. We implemented the pseudo-random function using the secure hash algorithm (SHA1).

The database of this experiment was randomly generated, each record consisted of a keyword and a file corresponding to the keyword. Furthermore, we generated two similar-sized databases with different numbers of keywords and of the same size to study the performance of our scheme on different datasets. The database DB1 consists of 3 K keyword-file pairs, 600 keywords where each keyword matched 5 files on average and the database DB2 also consisted of 3 K keyword-file pairs but had 300 keywords where each keyword matched 10 files on average. We ran each execution 20 times and averaged the results. The performance of our scheme for the different phases was recorded.

6.1 Index Generation

During this phase, the data owner encrypts keywords and corresponding files into encrypted indexes and associated encrypted files. Fig. 4 shows the index generation time varying with the number of keyword- file pairs. It can be observed that the time cost increases with the number of keyword-file pairs linearly. The time for generating indexes of 600 keyword-file pairs averages to 21.2 s.

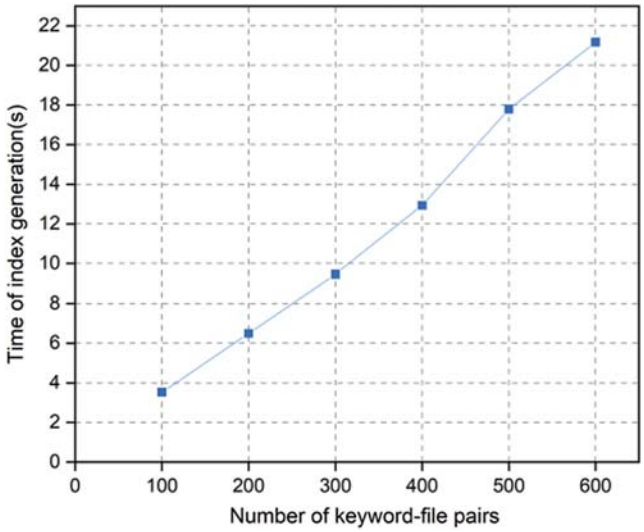

Figure 4: Time of index generation *vs*. the number of keyword-file pairs

6.2 Search

To search the blockchain, the data owner first needs to generate a search token. The time cost of token generation is shown in Fig. 5. It is obvious that the time cost increases with the number of keywords linearly because the data owner generates a search token for each keyword. The time for generating tokens of 600 keywords was 10.3 s on average.

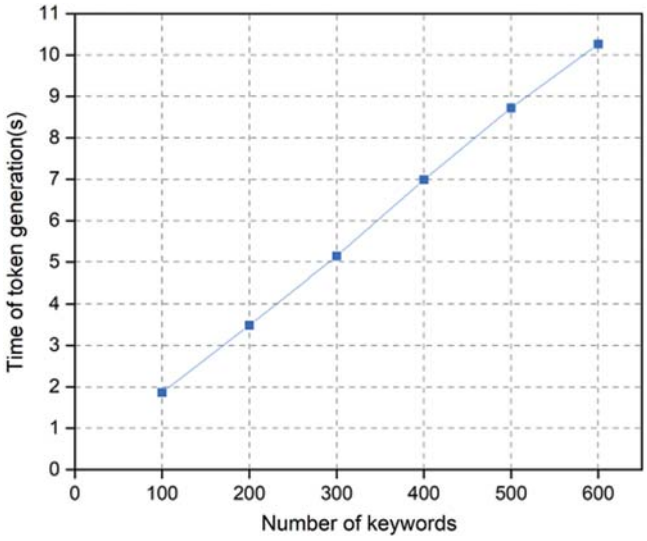

Figure 5: Time of token generation *vs*. the number of keyword-file pairs

On the service side, the smart contract accepts the search request from the data owner/data user and executes the query operation. We execute the query operation with DB1 and DB2 separately. We do not implement FCS here, which helps us focus on the performance of our scheme without additional optimization considerations. The experimental results are shown in

Fig. 6. We can see that the time costs for both DB1 and DB2 increase linearly with the number of matching files. In addition, the search time for DB1 takes about twice as long as that of DB2, since the smart contract needs to traverse all the ciphertext keywords (*cph*) in this phase. The time for each traversal of DB1 is the same as DB2 because DB1 and DB2 are the same size. Each query on DB2 returns 10 results while DB1 returns 5 results, resulting in the time to be half of that for DB2.

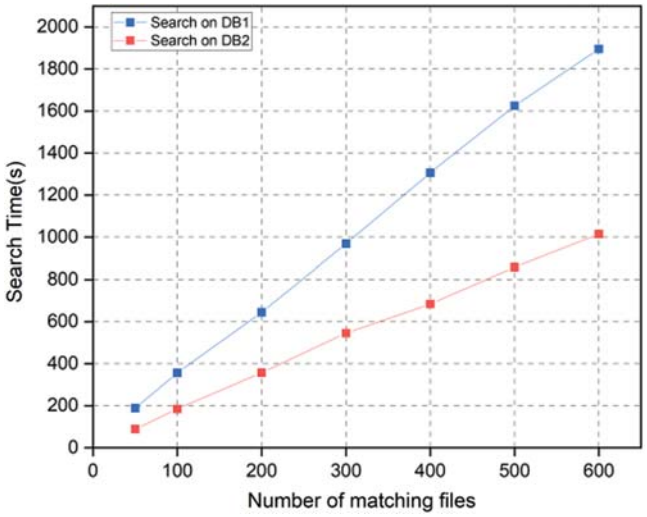

Figure 6: Comparing the search time of our scheme on DB1 and DB2 for various result set sizes

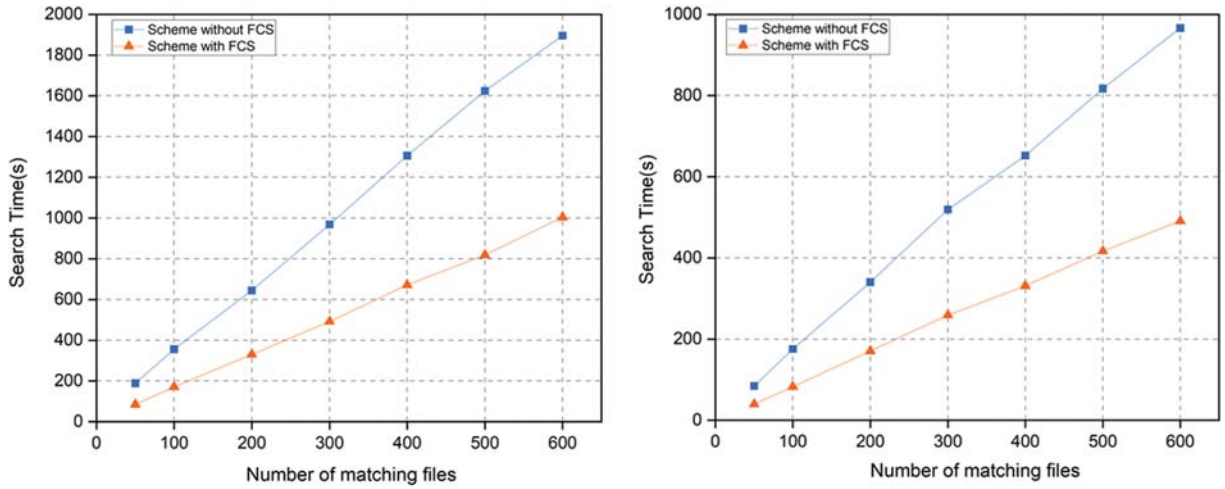

Figure 7: Comparing the search time of our scheme on DB1 and DB2 for various result set sizes, the time cost of scheme with and without FCS are compared on each database (a) Search time cost of our scheme on DB1 (b) Search time cost of our scheme on DB2

To demonstrate the efficiency of FCS, we present the search time cost with and without FCS respectively. Experimental results on DB1 are shown in Fig. 7a and those on DB2 are shown in Fig. 7b. It is obvious that the efficiency of the scheme with FCS is much higher than that without FCS for both DB1 and DB2. This shows the superiority of our FCS scheme. Besides, itis easy to see that both the time costs with and without FCS linearly increase with the number of matching files. This is because each query is independent of the other.

7 Conclusion

In this paper, we consider a scenario of encrypted data retrieval where data users could search over encrypted blockchains based on their attributes. Using attribute-based encryption, we designed an attribute-based encrypted keyword search scheme for blockchains, which allows a data user to search encrypted data across the blockchain. A novel File Chain Structure was proposed to improve the retrieval efficiency when searching files with the same keywords. The security analysis proves that our scheme can provide keyword semantic security and token unlinkability. The performance evaluation shows the feasibility and efficiency of the proposed scheme.

Funding Statement: This work was supported by the National Natural Science Foundation of China (61671030), Industrial Internet Innovation Development Project, China Postdoctoral Science Foundation (2019M660377), and National Key Research and Development Program of China (2020YFB2009501). It was also supported by Engineering Research Center of Intelligent Perception and Autonomous Control, Ministry of Education.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Su, S., Tian, Z., Liang, S., Li, S., Du, S. et al. (2020). A reputation management scheme for efficient malicious vehicle identification over 5G networks. *IEEE Wireless Communications, 27(3),* 46–52. DOI 10.1109/MWC.001.1900456.
2. Tian, Z., Su, S., Shi, W., Du, X., Guizani, M. et al. (2019). A data-driven method for future Internet route decision modeling. *Future Generation Computer Systems, 95(4),* 212–220. DOI 10.1016/j.future.2018.12.054.
3. Li, M., Sun, Y., Lu, H., Maharjan, S., Tian, Z. (2019). Deep reinforcement learning for partially observable data poisoning attack in crowdsensing systems. *IEEE Internet of Things Journal, 7(7),* 6266–6278. DOI 10.1109/JIOT.2019.2962914.
4. Hu, S., Cai, C., Wang, Q., Wang, C., Luo, X. et al. (2018). Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization. *IEEE Conference on Computer Communications*, pp. 792–800. Honolulu, IEEE.
5. Chen, L., Lee, W. K., Chang, C. C., Choo, K. K. R., Zhang, N. (2019). Blockchain based searchable encryption for electronic health record sharing. *Future Generation Computer Systems, 95(3),* 420–429. DOI 10.1016/j.future.2019.01.018.
6. Li, H., Zhang, F., He, J., Tian, H. (2017). A searchable symmetric encryption scheme using blockchain. arXiv preprint. arXiv: 1711.01030.
7. Jiang, S., Liu, J., Wang, L., Yoo, S. M. (2019). Verifiable search meets blockchain: A privacy-preserving framework for outsourced encrypted data. *2019 IEEE International Conference on Communications*, pp. 1–6. Shanghai, IEEE.

8. Yang, Y., Lin, H., Liu, X., Guo, W., Zheng, X. et al. (2019). Blockchain-based verifiable multi-keyword ranked search on encrypted cloud with fair payment. *IEEE Access, 7,* 140818–140832. DOI 10.1109/ACCESS.2019.2943356.
9. Qiu, J., Tian, Z., Du, C., Zuo, Q., Su, S. et al. (2020). A survey on access control in the age of internet of things. *IEEE Internet of Things Journal, 7(6),* 4682–4696. DOI 10.1109/JIOT.2020.2969326.
10. Song, D. X., Wagner, D., Perrig, A. (2000). Practical techniques for searches on encrypted data. *Proceeding 2000 IEEE Symposium on Security and Privacy*, pp. 44–55. Oakland, IEEE.
11. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G. (2004). Public key encryption with keyword search. *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 506–522. Berlin, Heidelberg: Springer.
12. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R. (2011). Searchable symmetric encryption: Improved definitions and efficient constructions. *Journal of Computer Security, 19(5),* 895–934. DOI 10.3233/JCS-2011-0426.
13. Cash, D., Jaeger, J., Jarecki, S., Jutla, C. S., Krawczyk, H. et al. (2014). Dynamic searchable encryption in very-large databases: Data structures and implementation. *NDSS*, vol. 14, pp. 23–26. San Diego, The Internet Society.
14. Xia, Z., Wang, X., Sun, X., Wang, Q. (2015). A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. *IEEE Transactions on Parallel and Distributed Systems, 27(2),* 340–352. DOI 10.1109/TPDS.2015.2401003.
15. Fu, Z., Sun, X., Linge, N., Zhou, L. (2014). Achieving effective cloud search services: Multi-keyword ranked search over encrypted cloud data supporting synonym query. *IEEE Transactions on Consumer Electronics, 60(1),* 164–172. DOI 10.1109/TCE.2014.6780939.
16. Waters, B. R., Balfanz, D., Durfee, G., Smetters, D. K. (2004). Building an encrypted and searchable audit log. *NDSS*, vol. 4, pp. 5–6. San Diego, The Internet Society.
17. Baek, J., Safavi-Naini, R., Susilo, W. (2008). Public key encryption with keyword search revisited. *International Conference on Computational Science and Its Applications*, pp. 1249–1259. Berlin, Heidelberg: Springer.
18. Rhee, H. S., Susilo, W., Kim, H. J. (2009). Secure searchable public key encryption scheme against keyword guessing attacks. *IEICE Electronics Express, 6(5),* 237–243. DOI 10.1587/elex.6.237.
19. Chen, R., Mu, Y., Yang, G., Guo, F., Wang, X. (2015). Dual-server public-key encryption with keyword search for secure cloud storage. *IEEE Transactions on Information Forensics and Security, 11(4),* 789–798. DOI 10.1109/TIFS.2015.2510822.
20. Li, H., Tian, H., Zhang, F., He, J. (2019). Blockchain-based searchable symmetric encryption scheme. *Computers & Electrical Engineering, 73(3),* 32–45. DOI 10.1016/j.compeleceng.2018.10.015.
21. Cai, C., Yuan, X., Wang, C. (2017). Towards trustworthy and private keyword search in encrypted decentralized storage. *2017 IEEE International Conference on Communications*, pp. 1–7. Paris, IEEE.
22. Zhang, Y., Deng, R. H., Shu, J., Yang, K., Zheng, D. (2018). TKSE: Trustworthy keyword search over encrypted data with two-side verifiability via blockchain. *IEEE Access, 6,* 31077–31087. DOI 10.1109/ACCESS.2018.2844400.
23. Tian, Z., Li, M., Qiu, M., Sun, Y., Su, S. (2019). Block-DEF: A secure digital evidence framework using blockchain. *Information Sciences, 491(3),* 151–165. DOI 10.1016/j.ins.2019.04.011.
24. Tahir, S., Rajarajan, M. (2018). Privacy-preserving searchable encryption framework for permissioned blockchain networks. *2018 IEEE International Conference on Internet of Thing and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data*, pp. 1628–1633. Halifax, IEEE.
25. Sun, W., Yu, S., Lou, W., Hou, Y. T., Li, H. (2014). Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. *IEEE Conference on Computer Communications*, pp. 226–234. Toronto, IEEE.
26. Wang, N., Fu, J., Bhargava, B. K., Zeng, J. (2018). Efficient retrieval over documents encrypted by attributes in cloud computing. *IEEE Transactions on Information Forensics and Security, 13(10),* 2653–2667. DOI 10.1109/TIFS.2018.2825952.