ISSN: 2582-8118 ***Volume 2, Issue 1; Jan 2022***

INTERNATIONAL JOURNAL OF RESEARCH AND ANALYSIS IN SCIENCE AND ENGINEERING

Web: https://www.iarj.in/index.php/ijrase/index

6. Blockchain-Based Public Key Encryption with Multi-Keyword Search Secure against Inside Keyword Guessing Attacks

Bunyad Hussain, Chungen Xu and Shahid Hussain
School of Mathematics and Statistics,
Nanjing University of Science and Technology,
Nanjing, JS, China,

ABSTRACT

In the last two decades, with the prevalence of Internet of Things (IoT) and cloud computing, many individuals and organizations have outsourced large amount of data to cloud server for storage. Outsourced data onto the cloud server contains sensitive information such as medical records, organization's financial records etc. Thus, the privacy of data retrieved by the user must be protected. In order to ensure that user information is secure when retrieving data, searchable encryption technology for the cloud environment is used. However, most schemes only support single-keyword search and do not support updating files, which limits the flexibility of the scheme. To solve these problems, in this paper we developed a Blockchain-based public key encryption scheme with multi-keyword search (BC-PKEMS), which supports file update operations. Besides, we utilized a smart contract to ensure the fairness of transactions between the data owner and user without introducing a third party. Our scheme achieves verifiability during the data storage phase by numbering the files and ensuring that the ciphertext received by the user is complete. Security and performance analysis shows that our scheme is secure against inside keyword guessing attacks (KGAs) and has better computation and storage overhead than other existing schemes.

KEYWORDS

Blockchain, Public key Encryption with Multi-Keyword search, Keyword Guessing Attacks, File Updates, Confidentiality.

1. Introductions:

As a promising computing paragon, cloud computing has brought great convenience to people's life. It provides huge benefits to data owners such as inclusiveness, flexibility, scalability, and rapid retrieval of data. Thus, the security problem of data is increasingly important. The main goal is to meet the visits required by customers. While eliminating the

user's local storage hardware and management cost, the data are out of the user's physical control, so data security is greatly threatened. When users upload data to cloud storage media, they need to solve the security problem of the data, and people often upload it after encryption. The secure search usually refers to the effective search of encrypted data; to solve the problem of how to use the server to complete the secure keyword search when the encrypted data are stored in the cloud under the premise of incomplete trust, researchers proposed searchable encryption (SE) as the core technology of secure search.

SE is a new technology that enables user to perform secure keyword search in a cloud storage environment using untrusted servers so that users can securely search data in ciphertext form, especially, search the keywords according to the keywords of interest. In terms of the usability of the SE scheme, multi-keyword search is more in line with the user's search experience. Compared with a single-keyword search, it can locate the search more accurately. In cloud computing, the server may be honest but curious and will want to obtain some sensitive information. Therefore, to enhance the security of data in cloud storage and resist keyword guessing attacks, a practical public key searchable encryption scheme needs to be designed and proposed.

2. Related Work and Challenges:

In the present era, cloud computing technology has been rapidly developed and a series of research has been conducted on security issues. In order to enhance the security, Dawn et al. [1] proposed a symmetric SE scheme, but it was in one-to-one model, that triggered people's research on SE because the one-to-one model cannot meet people's needs. For the many-to-one model, Boneh et al. [2] first proposed the public-key SE scheme and gave the concept of SE security based on public-key encryption in 2004. But in certain environments, the many-to-one mode is not feasible. In 2011, Curtmola et al. [7] constructed a one-to-many SE model based on Naor broadcast encryption technology [8], but the user's key replacement in this model requires a great deal of overhead. In a large-scale network environment, data transmission is complicated. Wang et al. [9] constructed a many-to-many model encryption scheme based on Shamir's secret sharing technology [10] and the identity-based encryption technology in [2] to realize the interaction retrieval of multiple users in the server. To effectively solve the problem of interactive retrieval when there are multiple recipients, Yuan et al. [11] proposed a one-to-many public key ciphertext time-release searchable encryption cryptographic model. In the one-to-many model, only authenticated users can enjoy the search service, and the queried keywords are specified, and they can decrypt it when it knows that it will be released in the future. Zhong et al. [12] proposed a many-to-one homomorphic encryption scheme, which overcomes the limitations of the traditional one-to-one model. In terms of the security of SE, about the scheme [2] proposed by Boneh, only the semantic security of index ciphertext can be achieved, but it cannot resist KGAs. In 2009, Tang and Chen [13] put forward a public key SE scheme. The keywords should be registered before use, which can resist KGAs, but the keywords must be registered in advance, which makes the performance of the scheme not high. In 2013, Fang et al. [14] presented the scheme belonging to public-key cryptography, which can resist KGAs; the scheme defines a public key SE model and two important security

concepts: one is for inside attacks and the other is for external attacks. However, a large number of bilinear pairing calculations result in the low efficiency of Fang's scheme [14]. In recent years, scholars have conducted a lot of research on inside attacks. In 2012, Xu et al. [15] proposed a scheme with two trapdoors (fuzzy trapdoor and precision trapdoor) and claimed that the scheme can resist inside KGAs. In this scheme, the adversary intelligently obtains the fuzzy trapdoor, but some keyword information about the trapdoor is not known, and it is restricted in terms of security and efficiency. In 2015, Chen et al. [16] introduced a new framework to prevent inside KGAs. They used two servers to realize the scheme, but the limitation is that the two servers cannot be associated. However, anyone can generate legal trapdoors for keywords, which will make data privacy issues easy to discover. Shao et al. proposed a method [17] that can resist KGAs. In the SE scheme of a designated tester, the security of the scheme is redefined as IND-KGA-SERVER. In the presence of a digital signature, it can resist the server's KGAs. In 2016, Chen et al. [18] proposed a scheme using two servers to resist inside KGAs, and the scheme has high efficiency. However, due to the two assumptions that cloud servers cannot be connected, this is difficult to achieve in practice. In 2017, Huang and Li [19] proposed a public key authentication encryption scheme based on keyword search. The ciphertext generation process of this scheme requires the key of the data owner. Although the scheme can resist the inside KGAs, it cannot achieve the chosen keyword ciphertext indistinguishability. Kang and Liu [20] proposed a completely secure public-key encryption scheme composed of bilinear pairing and TF/IDF algorithm. This scheme achieves security under static assumptions. By comparing with previous SE schemes, their scheme's performance is superior to other schemes. In terms of security, this scheme can avoid revealing privacy due to the curiosity of the adversary. In 2018, Wu et al. [21] proposed a secure and efficient public key SE scheme with privacy protection. This scheme uses a DH shared key and is proven to resist KGAs.

In the Internet of Things (IoT) environment, Wu et al. [22] proposed a certificateless searchable public-key authentication encryption scheme, which can resist KGAs at the same time and also has higher efficiency. Ma et al. [23] designed a new multi-keyword certificateless public-key encryption scheme for IoT deployment. Lu and Li [24] proposed a new PEKS scheme, which not only can resist the existing three types of KGAs but also improves the shortcomings of the designated server. With the development of blockchain [25, 26], the combination of searchable encryption technology and blockchain technology solves the problem of trusted third party in traditional schemes and greatly improves the practicability of searchable encryption. Li et al. [27] proposed a searchable encryption system model of blockchain and designed a practical scheme for the system model. In 2019, Li et al. put forward a scheme [28] based on [29], which improved enablement. In order to be suitable for the electronic medical scene, Chen et al. [29] proposed a SE scheme suitable for this scene under the blockchain technology. This scheme also adopts symmetric encryption method and uses smart contract as the authoritative entity to ensure the credibility of the server in the scheme. Zheng et al. [4] proposed an SE scheme that can verify the correctness of the results, but it cannot support data update operation. The SE scheme proposed by Sun et al. [5] and Xia et al. [6] can not only support dynamic updates but also verify the results, and it also has low computational efficiency. Therefore, we are committed to solving these problems. In 2021, Mei et al [35] proposed an efficient forward

and backward private SSE scheme for multiple data sources (FBSSE-MDS), which supports both forward privacy and backward privacy.

2.1 Our Contributions:

In this paper, we propose the BC-PKEMS scheme in the blockchain scenario; the major contribution of our system are manifold:

- ***Multi-Keyword Search.*** The BC-PKEMS scheme has certain good features, such as multi-keyword search and file updates. In addition, the data owner and data user can generate a shared key when encrypting files. They can get the shared key without any interaction by utilizing the Diffie–Hellman (DH) key exchange protocol.
- ***Fairness.*** In this scheme, the blockchain mechanism is used to ensure the fairness of the transaction between the data owner and user without the involvement of a third party.
- ***Verifiability.*** On the blockchain platform, to ensure the accuracy of search results, we utilize smart contract to store index and trapdoor information and perform search services. In addition, we number the files, and the user can verify the ciphertext of the file after getting the result, which can avoid some malicious behavior of the cloud server.
- ***Inside Keyword Guessing Attacks:*** We developed a BC-PKEMS scheme to resist Inside Keyword guessing attacks without a single-point-of-failure concern, where multiple dedicated key servers are employed to assist users in encrypting keywords without knowing any information about the keywords. BC-PKEMS supports key renewal on key servers to fight against the key compromise. Compared with related schemes, BC-PKEMS gives a stronger security assurance.

3. Preliminaries:

3.1.1. Notations:

In Table 1, we introduced some important notations for understanding formulation and statement in this paper.

Table 1. Notations and Description

Notations	Description
pk_o , sk_o	Public/secret key pair of data owner
pk_u , sk_u	Public/secret key pair of data user
pk_s , sk_s	Public/secret key pair of cloud server
K	Shared key for data owner and data user

Notations	Description
N_i	Encrypted file index
N_S	Encrypted file index set
$F = \{f_i\}_{i\in[1,t]}$	File index set
F'	Returned file set
$C = \{C_i\}_{i\in[1,t]}$	Ciphertext set of F
$C' = \{C_i'\}_{i\in[1,t]}$	Packed ciphertext
$W = \{w_i\}_{i\in[1,m]}$	Keyword dictionary
$I = \{I_0, I_1, I_2, I_j\}_{j\in[1,m]}$	Index set for F
σ_2'	The intermediate value
σ_2	The final value
$W' = \{w_i'\}_{i\in[1,l]}$	Queried keyword set
$T_{w'} = \{T_{W,1}', T_{W,2}'\}$	Trapdoor for W'
$L = \{\rho(\tau)\}_{\tau\in[1,l]}$	Location set of W' in W, $\rho: w_\pi' \to w_{\rho(\pi)}$

3.1.2. Bilinear Pairing.

Bilinear pairing used in our scheme has found extensive application in cryptography. Due to its non-generality, we introduce its definition in this section.

Let G_1, G_2 be two multiplicative cycle groups. A map $e: G_1 \times G_1 \to G_2$ is called a symmetric bilinear pairing if it has the following properties:

(1) Bilinear. $e(u^a, v^b) = e(u,v)^{ab}$, $\forall u, v \in G_1$, and $\forall a, b \in Z_p$.

(2) Nondegenerate. $e(g,g) \neq 1$. Let $1 \in G_2$ be the identity element of G_2 group.

(3) Computable. $\forall u, v \in G_1$, $e(u,v)$; there is a polynomial-time algorithm that can easily calculate e.

3.1.3. Decisional Diffie–Hellman (DDH) Problem.

Given a generator g of G_1, then $\{g, g^a, g^b, g^c\} \in G_1$, where $a, b, c \in Z_P$. The DDH problem is to determine whether g^c is equal to g^{ab}. Assuming that the DDH problem is

difficult, it means that no adversary can solve the problem with a probability that cannot be ignored.

3.1.4. Blockchain.

Blockchain has a wide range of applications, such as the Internet of Things and edge computing, and blockchain can be used in 5G handover authentication [30]. Blockchains are best known for their crucial role in cryptocurrency systems, such as Bitcoin, for maintaining a secure and decentralized record of transactions. The innovation with a blockchain is that it guarantees the fidelity and security of a record of data and generates trust without the need for a trusted third party.

Smart Contract. The smart contract (SC) is considered as the core technology of the second-generation blockchain, which was proposed by Szabo [31]. The carrier of the SC is the blockchain, and its essence is an automatically executed computer code. The code describes the terms of the agreement between the buyer and the seller and is directly written into the code of the blockchain. Satisfying the predetermined terms is the trigger condition for the code to be executed. Since the execution of the code does not require human intervention, it is called automatic execution. As a computer program, a SC is a part of application software and a digitally represented program [32]. Although it is a code representation of contract terms, it is not a contract in the legal sense. In addition, the construction of SC comes from the blockchain framework, which is a public billing system, which can carry out secure value transfer without a trusted third party, and the correctness of the contract code execution is guaranteed by the consensus mechanism. Therefore, SC can be understood as a computer protocol, which can be executed automatically without human intervention.

Gas System. In Ethereum, once the SC is set, it is forbidden to modify it. In order to prevent malicious users from setting up an infinite loop running contract, Ethereum requires users to pay for each step of the deployment contract. The basic unit of cost is gas. Gas is equivalent to the fuel needed to deploy and execute SC. Without fuel, SC cannot be used. This mechanism maintains the operation of the economic system of Ethereum. In a gas system, there are some important parameters. Gas price means that users need to pay for each unit of gas. Each block has a gas limit, that is, the maximum amount of gas allowed in a single block, which can be used to determine how many transactions can be packaged in a single block. Both gas price and gas limit are set by the transaction sender itself. If the total amount of gas consumed by the operation exceeds gas limit, the operation will be voided, the transaction is not packaged in the block and the transaction amount is refunded, and the gas fee that has been performed will still be charged [33]. Only if the user's current amount is greater than the gas limit times gas price, the transaction will be executed successfully. For the gas price, if the value is too high, the transaction may be executed first, and if it is too low, the transaction speed will be slow.

3.2. Proposed Framework

There are five major entities involved in our proposed framework which are Data Owners (DO), Data Users (DU), Cloud Server (CS) smart contract (SC) and Trusted Authority (TA) as illustrated in

- ***Data Owner (DO).*** The main work of the data owner is to calculate the keyword index and the file ciphertext and then send the file ciphertext to the cloud server and the keyword index to the smart contract.
- ***Data User (DU).*** The main work is to calculate the trapdoor and upload it to the smart contract. Then, the data user gets the corresponding file ciphertext from the cloud server and verifies it. Finally, the file ciphertext is decrypted by the data user.
- ***Cloud Server (CS).*** The main work of the cloud server is to store the data uploaded by the data owner and receive the file index from the smart contract. Then, the cloud server forwards the corresponding file ciphertext to the data user.
- ***Smart Contract (SC).*** The Smart contract's main work is to receive indexes and trapdoors to match and then send the search result to the cloud server through a transaction.
- **Trusted Authority (TA).** The trusted authority is responsible for generating public/private key pairs for data owner, data user and the cloud server.

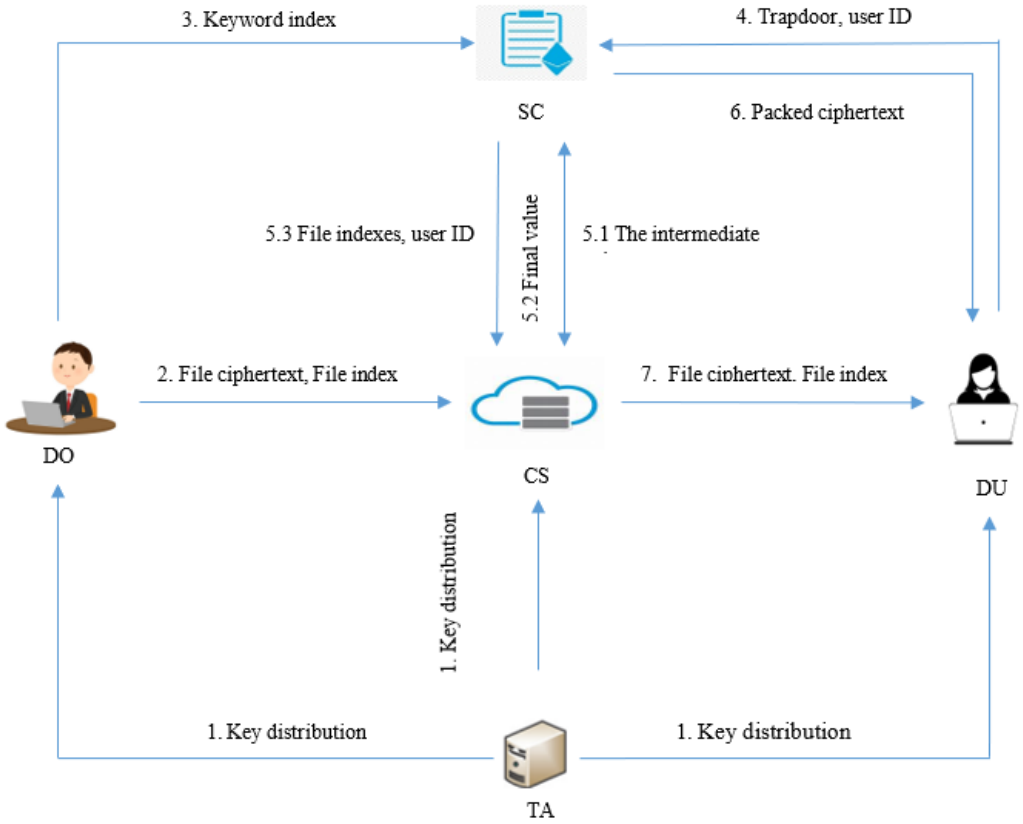

Fig 1: The System Model

3.3 Adversary Model:

In our scheme, the trusted authority (TA) is completely trusted, the data user (DU) is malicious, and the cloud server (CS) is semi-trusted. For example, the semi-trusted CS may want to know the original file information or return partial search results. DU may also

maliciously accuse the CS of not returning correct search results. In the payment phase, the CS may want to obtain the search fee from the DU without providing the search result. In addition, a malicious DU may want to obtain the correct search results from the CS without paying the search fee.

3.4 Design of BC-PKEMS Scheme:

In this section, we describe the six algorithms used in our scheme: Setup, Key generation, Encryption, Trapdoor, Search, and Verification and Decryption.

Setup (1^{λ})**.** The algorithm inputs a public parameter λ and outputs a global public parameter SP

Key Generation (SP)**.** This algorithm takes SP as inputs, and it outputs the DO's public key pk_o and private key sk_o. The public and private keys of DU and CS are generated like DO.

Encryption (SP, pk_u, sk_o, F)**.** This algorithm inputs SP, pk_u, and sk_o. Then, it outputs the keyword indexes I, file ciphertext C, packed ciphertext C', and encrypted file index set N_S.

Trapdoor (W', pk_s, pk_o, sk_u)**.** This algorithm takes queried keyword set W', CS's public key pk_s, DO's public key pk_o, and DU's private key sk_u as input and it outputs the corresponding trapdoor $T_{W'}$ and location information L.

Search $(I, T_{w'}, L, sk_s)$**.** This algorithm inputs I, $T_{w'}$, L, the CS's private key sk_s. Then, it outputs the encrypted file index set Ns. Note that the search process is run in the blockchain, using the privacy key of CS. Therefore, in the execution of smart contract, there will be an interaction with CS first.

Verification and Decryption (SP, C, C', N_s). The algorithm takes SP, N_S, file ciphertext set C, and packed ciphertext C' as input and it outputs the verification results and file set F'.

4.3 The Detailed Construction of BC-PKEMS Scheme:

Setup. Input a security parameter λ, and then TA runs the Setup algorithm to generate the system parameters $SP = (g, h, H_1)$.We set g as a generator of G_1, and h and H_1 are two collision-resistant hash functions, where $h: \{0,1\}^* \to Z_p, H_1: \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$. Then, TA publishes the public parameters SP.

Key Generation. The scheme runs the KeyGen algorithm to generate the public/private key pair for DO, DU, and CS. The detailed generation process is as follows:

(1) $KenGen_o$**:** randomly choose an element $a \in Z_P$ as the private key sk_o and then compute the public key $pk_o = g^a$.

(2) $KenGen_u$: pick an element $b \in Z_p$ as the DU's private key sk_u and compute the public key $e(g,g)^{(1/b)}, g^b$ .The DU's public key has two parts, which we define as $pk_u = \{pku_1, pku_2\}$, where $pku_1 = e(g,g)^{(1/b)}$, $pku_2 = g^b$.

(3) $KeyGen_s$**:** choose an element $c = Z_p$ as the private key sk_s and then compute the CS's public key $pk_s = g^c$.

Ciphertext Generation. Before generating a keyword index, DO first defines the reward offer to be paid per search to himself and sends this setting to the SC. Upon receiving the file set $F = \{f_1, ..., f_t\}$, we define the keyword dictionary as $W = \{w_1, ..., w_n\}$. DO extracts the keywords in each file. The DO uses the Enc algorithm to output the indexes I , file ciphertexts C, and packed ciphertext C'.

(1) First, DO needs to generate keyword index, $I_i = \{I_{i,0}, I_{i,1}, I_{i,2}, I_{i,j}\}$, where $i \in [1,t]$. DO randomly chooses an element $r \in Z_p$. Next, he calculates the $I_{i,0} = e(g,g)^r$, $I_{i,1} = (pku_2)^r = g^{br}$, $I_{i,2} = e(g,g)^{(ar/b)}$, $I_{i,j} = g^{-rh(w_j)}$, where $j \in [1,n]$.

(2) Second, DO encrypts each file $f_i \in F$. Here, we use a symmetric encryption algorithm when encrypting files. The difference is that we use the idea of DH key exchange to share the key K for DO and DU, and DO uses its own private key sk_o and DU's public key pku_2 to calculate it, where $k = pku_2^{sk_o} = g^{ba}$. Then, for each file f_i, $C_i = Enc_K(f_i)$.

(3) Third, DO numbers the file f_i, encrypts the file index i with the key K, obtains the encrypted file index $N_i = Enc_K(i)$, stores the N_i and the ciphertext C_i together, and then performs a hash operation to obtain the result $M_i = H_1(N_i, C_i)$.

The file indexes N_i, M_i are packed as ciphertext C_i'. Then, upload the encrypted file index set N_S and ciphertext set C to the CS. Then, send the packed ciphertext set C' and index set I to the SC for querying operation.

Ciphertext Update. In this section, we describe how to update files, for example, modify, insert, and delete operations. For modification and insertion of files, blockchain and encryption protect the index and encrypted files from leaking sensitive information. The detailed file update operations are shown in

(1) **Modification.** Suppose a file m_k needs to be changed to m_k', and DO needs to recalculate its ciphertext, that is, $c_k' = Enc_K(m_k')$.

(2) **Insertion.** When adding a new file at the $k-th$ position, add the ciphertext at the corresponding position with c_k'.

(3) **Deletion.** When a file needs to be deleted, only the file and index value need to be deleted from the CS.

Trapdoor. In this section, the Trap algorithm was run by DU. When a DU wants to query keyword $W' = \{w_1', ..., w_l'\}$, he needs to generate trapdoor $T_{w'}$ for these keywords. The trapdoor consists of two parts, one is $T_{w',1}$ and the other is $T_{w',2}$.

File index	1	2	3	4	5	6	7	8	9
	m_1	m_2	m_3	m_4	m_5'	m_6	m_7	m_8	m_9

↓ Modify m_5 as m_5'

File index	1	2	3	4	10	6	7	8	9
	m_1	m_2	m_3	m_4	m_5'	m_6	m_7	m_8	m_9

↓ Insert m'

File index	1	2	3	4	10	6	11	7	8	9
	m_1	m_2	m_3	m_4	m_5'	m_6	m'	m_7	m_8	m_9

Delete m_3' ↓

File index	1	2	4	10	6	11	7	8	9
	m_1	m_2	m_4	m_5'	m_6	m'	m_7	m_8	m_9

Fig. 2. The update operations of files

(1) DU randomly selects an element $\varphi \in Z_P$, let $T_{w',1} = \varphi$.

(2) DU computes $T_{w',2} = (pk_s^{-\varphi} pk_o^{(1/b)})^{(1/(b-\sum_{\pi=1}^{l} h(w_\pi')))}$.

We need to record the keyword location L, which expresses the location from W′ to W. We define a mapping function $\rho: w_\pi' \to w_{\rho(\pi)}$. After the user generates the trapdoor $T_{w'}$,

the user sets a time limit node T_1, uploads the trapdoor with the location $L=\{\rho(1),...,\rho(l)\}$ to the SC, and performs the deposit operation from his account. Then, the user sends the trapdoor $T_{w'}$ and the time limit node T_1 to the SC. Next, he uploads his identity ID to request the SC to perform the search service.

Search. The search algorithm is run by SC. SC and blockchain are combined for search services. Here, we define some symbols. owner and user represent the respective accounts of DO and DU. userdeposit expresses the current deposit in the blockchain. DU deposits his account balance into the blockchain system user. The price per unit of gasoline is denoted by gasprice. The total cost of each complete search operation is expressed as cost. Gaslsrch and Gassrch, respectively, express the gas limit and the cost of calling the search algorithm. After receiving the DU's ID and requesting the search service, perform the following algorithm.

1. First, check whether the current time T_2 is less than T_1. If yes, perform the following steps. If not, the process is stopped.
2. Check whether userdeposit is greater than Gaslsrch $\times$ gasprice; if yes, the user's current deposit userdeposit can complete the next search service, and the SC starts to run. If not, stop it.
3. The **SC** computes the intermediate value $\sigma_1' = I_{i,0}^{T_{w',1}}$. Then σ_2' is sent to **CS**. **CS** calculates the final value $\sigma_2 = \sigma_2'^{c}$ and returns it to **SC**.
4. *Compute* $\sigma_1 = e(I_{i,1}.\prod_{\tau=1}^{l} I_{i,\rho(\tau)}, T_{w',2})$ and $\sigma_3 = I_{i,2}$.
5. *Calculate* whether the equation $\sigma_1.\sigma_2 = \sigma_3$ is true. If so, output 1 to indicate that the search was successful. Then, the SC sends the search results to the CS. Otherwise, output 0, indicates failure, and the search service will be stopped. Finally, the SC will record the encrypted file index N_i and then start the next query until all files are retrieved. Finally, SC sends the file index set N_S to CS. We describe the transaction during the search in ALGORITHM 1.

Verification and Decryption Phase. In this section, DU performs the verification and decryption algorithms. The SC sends the file index set N_S and DU's I D that satisfies the search request to CS. Then, CS transmits the file ciphertext set C and encrypted file index set N_S to the DU according to N_i. In ALGORITHM 1, we describe the search process for each round.

During the interaction between SC and DU, the packed ciphertext C_i' is obtained by the DU after the SC is successfully retrieved. Then, the user verifies $N_{SC} = N_{CS}$, where N_{SC}

represents the file index sent by the SC and N_{CS} represents the file index sent by CS. If above indexes are the same, it proves that the CS did not send wrong files, and then verify $M' = H_1(N_i, C_i), M \stackrel{?}{=} M'$.

If the file index N_i and ciphertext C_i are hashed and the result is equal, it proves that the CS has not tampered with the ciphertext data. Finally, DU uses its own private key sk_u and DO's public key pk_o to generate the shared key $K = pk_o^{sk_u} = g^{ab}$ of the encrypted file and decrypts the file ciphertext C_i, where $f_i = Dec_K(C_i)$. Finally, DU gets the decrypted file set F'.

(1) if $T_2 < T_1$ and \$userdeposit $>$ Gaslsrch $\times$ \$gasprice $+$ \$offer then;
(2) Compute $\sigma_1, \sigma_2, \sigma_3$;
(3) if $\sigma_1 \cdot \sigma_2$ is the same as σ_3 then;
(4) Return the file indexes N_i to CS;
(5) else;
(6) Return 0;
(7) Set cost = offer + Gassrch $\times$ gasprice;
(8) Send offer to owner. Then, send Gassrch $\times$ gasprice to executor of a deal;
(9) Finally, set userdeposit = userdeposit-cost;
(10) else;
(11) Send userdeposit to user;
(12) end;

ALGORITHM 1. Ciphertexts retrieval

Correctness. Formula (1) given below indicates that the index and trapdoor match successfully

$$
\begin{aligned}
& e\left(I_{i,1} \cdot \prod_{\tau=1}^{l} I_{i,\rho(\tau),} T_{w',2}\right) I_{i,0}^{cT_{w',1}} = e\left(g^{br} \cdot \prod_{\tau=1}^{l} g^{-vh\left(\omega\rho_{(\tau)}\right)}, \left(g^{(a/b} g^{c(-\varphi)}\right)^{1/\left(b-\sum_{\tau=1}^{1} h(\omega'_\tau)\right)}\right) e(g,g)^{rc\varphi} \\
& = e\left(g^r, g^{(a/b - c\varphi)}\right) e(g,g)^{rc\varphi} \\
& = e(g,g)^{(ar/b - rc\varphi)} e(g,g)^{rc\varphi} \\
& = e(g,g)^{(ar/b)} \\
& = I_{i,2.}
\end{aligned}
\tag{1}
$$

5. Security Analysis:

In order to show that our scheme is secure and practical, we describe the security in detail.

Fairness. Because the blockchain interacts with each entity on a transaction basis and each transaction is transparent, it can be guaranteed that the results of each query are correct and there will be no malicious tampering. Fairness is achieved through the use of SC. In Ethereum, all operations or transactions are associated with gas, and each operation will consume some of the gas on SC, and the person who provides the data (such as DO) will be rewarded accordingly. At the same time, users also need to pay for the files they retrieve. Without the involvement of a third party, the blockchain can ensure that users get correct and complete search results, and malicious operations will be detected. In addition, the user has determined a limited time to ensure the fairness of the transaction because the transaction needs to be completed within the specified time node. If the time limit is exceeded, the user will stop the search service.

Credibility. The search results given by the blockchain must be honest and credible. The operations on SC are transparent and cannot be tampered, so we can be confident that the results returned by the SC are credible. At the same time, it effectively prevents the malicious server from attacking this scheme. In addition, the transparency of the blockchain can ensure the correctness of the results, and the verification on the user side can also achieve the same effect. Nothing can be used as a malicious tamper with the search results. Entities connected to the blockchain can verify the actions of other entities at any time.

Confidentiality. This scheme can resist KGAs in theory. The security of this scheme should realize the indistinguishability of the index and trapdoor. Note that in Game 1, adversary **A** can query both the private key and trapdoor. Importantly, trapdoor queries need to exclude previously defined challenge keywords. Corresponding to the definition of Game 2, we can get that **A** can query the index ciphertext and CS's private key, the limitation is that **A** cannot query the challenge keywords w_0 and w_1 .

The detailed process is as follows. In Game 1, if the DDH assumption holds, the scheme achieves index indistinguishability. In Game 2, the scheme can ensure that it can resist chosen keyword attacks under the random oracle model.

(1) In Game 1, we analyze the private key used to encrypt files between DO and DU, which is generated through negotiation between the two entities. The CS must obtain the private key of one before it can generate a shared key or intercept it during the transmission of the public channel. However, our scheme does not require transmission. Therefore, the CS must obtain the private key of one of them to decrypt the ciphertext of the file f_i . Therefore, in our scheme, the shared key K is secure.

The security of our scheme can be analyzed in two parts. The first is the generation of the index. Assuming that a DU wants to query a keyword set W' , CS must generate a valid index $\vec{I} = \{I_0, I_1, I_2, I_j\}$. CS first needs to obtain the private key $sk_o = a = Z_p$ of DO. The private key of DO is kept secret; CS can only assume that it has obtained a private key a of the DU. But the size of Z_p is p , which is a large prime number. Therefore, the probability of selecting the right one is $(1/p)$, which is negligible. On the other hand, CS

assumes that the keyword set $W' = \{w'_1, ..., w'_l\} = \tilde{W}' = \{\tilde{w}_1, ..., \tilde{w}_l\}$ selected by CS is equal to the keyword set that DU wants to query, which is equivalent to randomly selecting l equal sets from n keywords, with a probability of $(1/C_n^l)$. Assuming that the range of the key set n is large enough, the above probability is also small enough. In summary, CS cannot perform inside KGAs.

(2) In Game 2, given a valid index $I = \{I_0, I_1, I_2, I_j\}$, CS cannot generate a valid trapdoor for matching. The generation of the trapdoor requires the use of the private key sk_u of the DU. We assume that the private key of the DU is $sk_u = b$. CS randomly selects an element $\tilde{b} \in Z_p$ as the private key of the DU. The equal probability is $(1/p)$, so the probability can be ignored. Through the above analysis, our scheme can resist inside KGAs.

Here, we introduce the location privacy of keywords. In the paper, we use the location mapping function $\rho(.)$. The location privacy of queried keywords can be protected using random mask technology, for example, pseudorandom functions. The pseudorandom function confuses the position of the real keyword so as not to riot the position of the real keyword. Try not to let users know more information. For the cloud server, the index location is exposed, but the keywords are encrypted, so the security of the scheme will not be affected

6. Performance Analysis:

In order to show that our scheme is more effective, in this part, we compare three schemes in terms of function. In addition, we discuss the computation and communication cost of our scheme with two other schemes [3, 34]. First, we compare the functions of the three schemes, as shown in Table 2, We can see that by comparing the functions of the five aspects, we can see the functional differences between those schemes. The checkmark means that this condition is satisfied, and the wrong sign means that the condition is not satisfied. It is compared by whether it supports multi-keyword retrieval, whether it supports the dynamic update of files, whether it supports blockchain, and whether it supports fair payment between users, whether it resists keywords guessing attacks. We can see that our scheme supports the five functions, scheme [34] only supports multi-keyword search and resists keyword guessing attacks and scheme [3] only does not support dynamic update of files. The dynamic update of files can ensure the flexibility of the scheme. By using the blockchain, you can take advantage of transparency, immutability, and traceability. Especially the SC running on the blockchain can ensure fair payment between users.

Table 2. Functional comparison of related work

	BC-PKEMS	**[3]**	**[34]**
Blockchain	✓	✓	×

	BC-PKEMS	**[3]**	**[34]**
Multi Keyword	✓	✓	✓
Update	✓	×	×
Fair payment	✓	✓	×
Against KGAs	✓	✓	✓

6.1. Theoretical Analysis:

In table Table 3, we compare the computation overhead of our scheme with the other two schemes [3, 34]. In terms of computation overhead, we mainly consider some time-consuming operations; T_M represents a multiplication operation, T_H represents a hash operation, T_E represents an exponential operation and T_P represents a pair operation. In Table 4, we compare our scheme with other schemes [3, 34] in terms of communication overhead. We define the element length of G_1, G_2 , Z_P as $|G_1|, |G_2|, |Z_P|$. In addition, we define m to represent the number of keywords contained in each file and l to represent the number of queried keywords.

Regarding the computation overhead, we compare the characteristics of each scheme in Table 3. In the key generation stage, we can see that our scheme is in the middle of the three at this stage, and the efficiency is higher than that of the scheme [3] and lower than that of scheme [34]. In the keyword encryption and trapdoor generation phases, the calculation amount of the three schemes increases linearly with the number of encrypted keywords and queried keywords, but our scheme is the most efficient among the three, which are $(3 + m)T_E + mT_E + T_P$ and $3T_E + lT_H + T_M$, respectively. In the search stage, we set the number of keywords to be queried to 1. It can be seen from the table that the calculation amount of the three schemes is constant, but our scheme has the highest efficiency. Therefore, based on the above theoretical analysis, our scheme has the highest efficiency.

Regarding communication overhead, we compare the public key size, encryption size, and trapdoor size with the other two schemes. We can see from Table 4, that the size of the public key generated by the three schemes remains unchanged. In the encryption phase, the size of the storage of our scheme is almost the same as the scheme [3] but is smaller than the storage size of scheme [34]. In the trapdoor generation stage, in scheme [34] the size of the trapdoor increases linearly with the number of queried keywords, and therefore, it will consume a lot of storage resources. Our scheme and scheme [3] are constant and therefore have good storage characteristics.

Table 3. Computation overhead

	BC-PKEMS	**[3]**	**[34]**
Key Gen	$4T_E + T_P$	$2T_H + 4T_E$	$3T_E$

	BC-PKEMS	[3]	[34]
Enc	$(3 + m)T_E + mT_H + T_P$	$mT_M + mT_H + (2m + 3)T_E$	$mT_M + 3mT_H + (2m + 2)T_E$
Trapdoor	$3T_E + lT_H + T_M$	$(l + 1)T_M + lT_H + (2l + 3)T_E$	$3lT_H + (2l + 1)T_E$
Search	$T_M + T_E + T_P$	$T_M + 3T_P$	$T_M + T_E + 3T_P$

Table 4. Communication overhead

	BC-PKEMS	[3]	[34]
Pk size	$\|G_1\|$	$\|G_1\|$	$\|G_1\|$
Enc size	$(m+1)\|G_1\|+2\|G_2\|$	$(m+3)\|G_1\|$	$(m+2)\|G_1\|+m\|Z_p\|$
Trapdoor size	$\|G_1\|+\|Z_P\|$	$3\|G_1\|$	$3\|G_1\|+l\|Z_p\|$

6.2. Experimental Analysis:

In this section, we compare our scheme with two other schemes: Yang's scheme [3] and Xu's scheme [34]. We used the Java Pairing-Based Cryptography (JPBC) Library. The implementation equipment of the scheme is a desktop computer with a 3.00 GHz Intel Core i5-8500 processor and 8.00 GB RAM. In the experiment, we used the Type A elliptic curve. We analyzed three schemes by comparing algorithms Enc, Trap, and Search algorithms. In the Enc algorithm, we set the number of keywords in steps of 10, increasing from 1 to 50 in turn. In Trap and Search, the number of keywords we set, is also increasing from 10 to 50 in steps of 10. In each of the above experiments, after 50 cycles, the average value of the calculation cost is calculated to ensure that the results are relatively valid. It can be seen from Fig.5. that our scheme is the most effective. Below we briefly explain the content of the iconn.

In Fig.3, we can see that our scheme has the smallest slope, which has great advantages compared with the other two schemes. Due to the frequent hashing operations and exponential operations, the coefficients of our scheme $(m$ and $3+m3+m)$ are larger, so the structure of the scheme is simpler. With the increase of keywords, the advantages will become more and more obvious.

In Fig.4, we can find that the time consumed is constant with the number of keywords that users query. In the process of generating trapdoors of our scheme, exponential operations and multiplication operations are constants, and hash operations increase linearly with the increase of keywords. However, you can see that in the other two schemes, the slope of

growth is much larger than that of our scheme, and it takes time to hash to Z_p which is much shorter than hashing to group G.

The efficiency gap between our scheme and the other two schemes is not obvious, as shown in Fig.5. Because of the pair operation, the number of operations of exponential operation is almost constant. For the operation after hashing the keyword, whether it is the aggregation of addition or the aggregation of multiplication, the time consumed by a single operation is very small. Therefore, as the number of keywords increases, the trend of time changes is not obvious. But judging from the changing trend in Fig.5, our scheme still has some benefits.

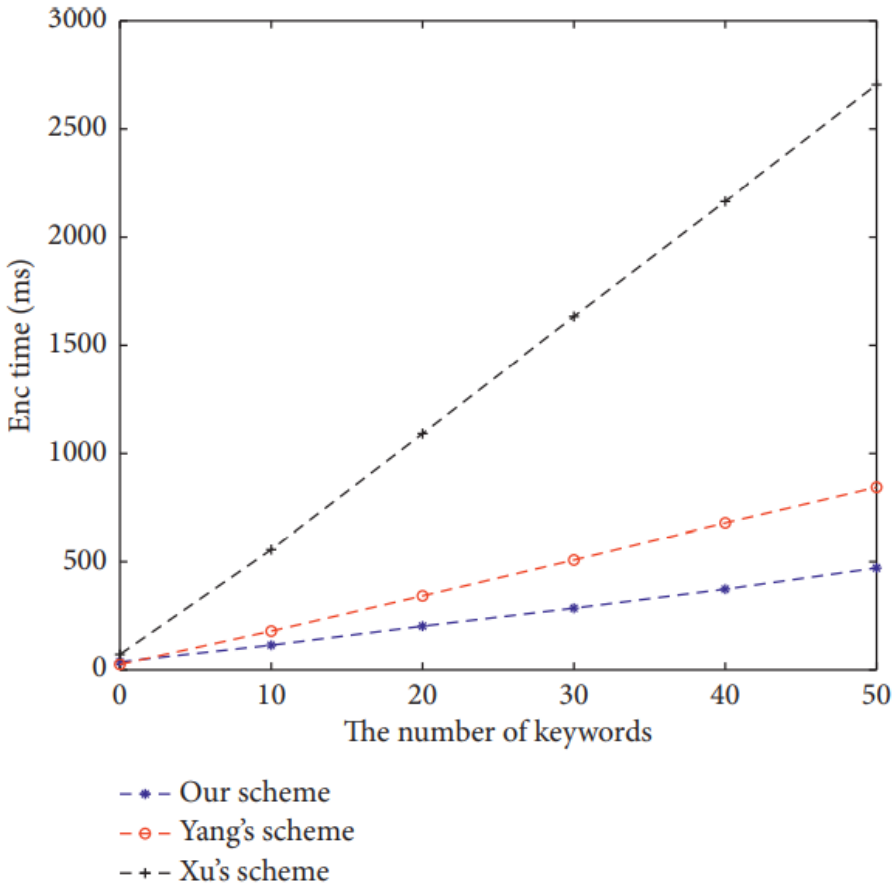

Fig. 3. The computation overhead of Encryption Algorithm

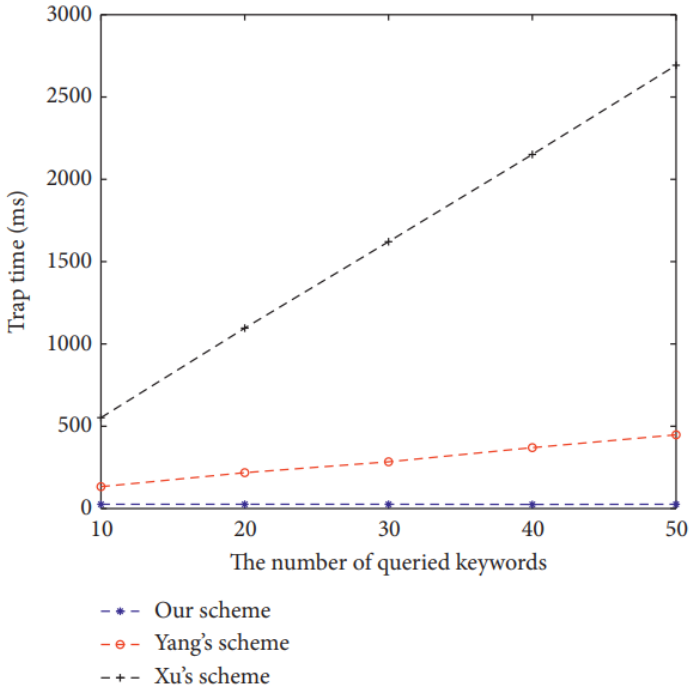

Fig. 4. The computation overhead of Trapdoor Algorithm

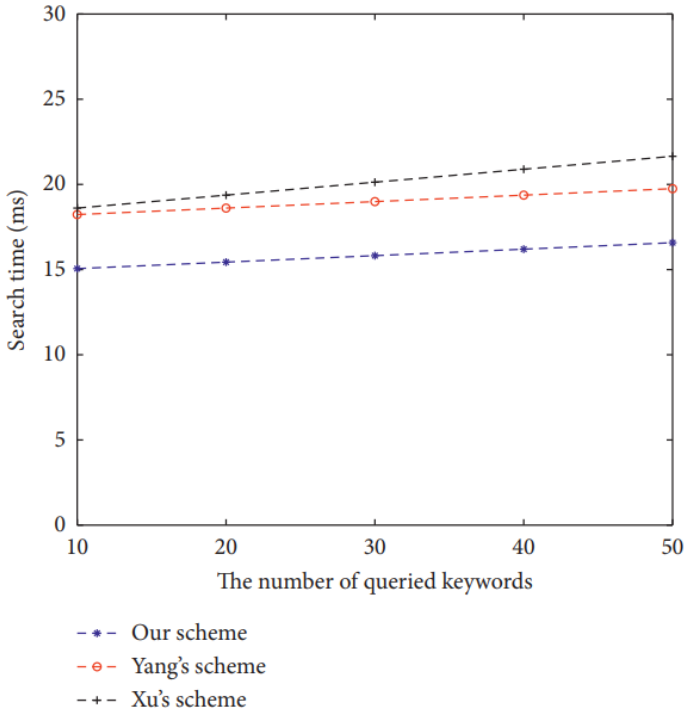

Fig. 5. The computation overhead of search Algorithm

7. Conclusion:

With the widespread applications of cloud computing, data privacy has become one of the critical security issues for data users. In order to achieve data privacy for cloud storage, in this paper, we introduced a secure and efficient BC-PKEMS scheme in blockchain scenario and conduct deep security and experimental analysis. Our proposed scheme supports secure multi-keywords search, dynamic update of files and verification of ciphertext. Our scheme can also resist inside KGAs. In terms of efficiency, we implemented this scheme through simulation and compared it with other related schemes [3, 34] and the comparison shows that our scheme is more secure and practical. For the future work, we will study the potentials for enhancing the security, efficiency, and functionality of data outsourcing systems.

References:

1. D. X. Song, D. W. Wagner, and A. Perrig, "Practical Techniques for Searches on Encrypted Data," in Proceedings of the 2000 IEEE Symposium on Security and Privacy, pp. 44–55, Berkeley, CA, USA, February 2000.
2. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in Advances in Cryptology-EUROCRYPT 2004, C. Cachin and J. L. Camenisch, Eds., pp. 506–522, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.
3. X. Yang, G. Chen, M. Wang, T. Li, and C. Wang, "Multi-keyword certificateless searchable public-key authenticated encryption scheme based on blockchain," IEEE Access, vol. 8, pp. 158765–158777, 2020.
4. Q. Zheng, S. Xu, and G. Ateniese, "Vabks: verifiable attribute-based keyword search over outsourced encrypted data," in Proceedings of the IEEE INFOCOM 2014-IEEE Conference on Computer Communications, pp. 522–530, IEEE, Toronto, ON, Canada, May 2014.
5. W. Sun, X. Liu, W. Lou, Y. T. Hou, and H. Li, "Catch you if you lie to me: efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data," in Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM), pp. 2110–2118, IEEE, Kowloon, HK, USA, May 2015.
6. Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 2, pp. 340–352, 2015.
7. R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," Journal of Computer Security, vol. 19, 2011, https://www.microsoft.com/en-us/research/publication/searchable-symmetric-encryption-improved-definitions-andefficient-constructions-2/.
8. A. Fiat and M. Naor, "Broadcast encryption" in Annual International Cryptology Conference, pp. 480–491, Springer, Berlin, Germany, 1993.
9. P. Wang, H. Wang, and J. Pieprzyk, "Privacy-preserving keyword searches," in Proceedings of the International Conference on Current Trends in Theory and Practice of Computer Science, pp. 646–658, Springer, Novy Smokovec, SL, Europe, January 2008.

10. A. Shamir, “How to share a secret,” Communications of the ACM, vol. 22, no. 11, pp. 612-613, 1979.
11. K. Yuan, Z. Liu, C. Jia, J. Yang, and S. Lv, “Public key timed-release searchable encryption in one-to-many scenarios,” Acta Electronica Sinica, vol. 43, no. 4, pp. 760–768, 2015.
12. H. Zhong, J. Cui, R. Shi, and C. Xia, “Many-to-one homomorphic encryption scheme,” Security and Communication Networks, vol. 9, no. 10, pp. 1007–1015, 2016.
13. Q. Tang and L. Chen, “Public-key encryption with registered keyword search,” in Proceedings of the European Public Key Infrastructure Workshop, pp. 163–178, Springer, Pisa, Italy, September 2009.
14. L. Fang, W. Susilo, C. Ge, and J. Wang, “Public key encryption with keyword search secure against keyword guessing attacks without a random oracle,” Information Sciences, vol. 238, pp. 221–241, 2013.
15. P. Xu, H. Jin, Q. Wu, and W. Wang, “Public-key encryption with a fuzzy keyword search: a provably secure scheme under keyword guessing attack,” IEEE Transactions on Computers, vol. 62, no. 11, pp. 2266–2277, 2012.
16. R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, “A new general framework for secure public key encryption with keyword search,” in Proceedings of the Australasian Conference on Information Security and Privacy, pp. 59–76, Springer, Brisbane, QLD, Australia, July 2015.
17. Z.-Y. Shao and B. Yang, “On security against the server in designated tester public key encryption with keyword search,” Information Processing Letters, vol. 115, no. 12, pp. 957–961, 2015.
18. R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, “Dual-server public-key encryption with keyword search for secure cloud storage,” IEEE Transactions on Information Forensics and Security, vol. 11, no. 4, pp. 789–798, 2015.
19. Q. Huang and H. Li, “An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks,” Information Sciences, vol. 403, pp. 1–14, 2017.
20. Y. Kang and Z. Liu, “A fully secure verifiable and outsourced decryption ranked searchable encryption scheme supporting synonym query” in Proceedings of the 2017 IEEE Second International Conference on Data Science in Cyberspace (DSC), pp. 223–231, IEEE, Shenzhen, China, June 2017.
21. L. Wu, B. Chen, S. Zeadally, and D. He, “An efficient and secure searchable public key encryption scheme with privacy protection for cloud storage,” Soft Computing, vol. 22, no. 23, pp. 7685–7696, 2018.
22. L. Wu, Y. Zhang, M. Ma, N. Kumar, and D. He, “Certificateless searchable public-key authenticated encryption with designated tester for cloud-assisted medical internet of things,” Annals of Telecommunications, vol. 74, no. 7-8, pp. 423–434, 2019.
23. M. Ma, D. He, N. Kumar, K.-K. R. Choo, and J. Chen, “Certificateless searchable public key encryption scheme for the industrial internet of things,” IEEE Transactions on Industrial Informatics, vol. 14, no. 2, pp. 759–767, 2017.
24. Y. Lu and J. Li, “Efficient searchable public key encryption against keyword guessing attacks for cloud-based emr systems,” Cluster Computing, vol. 22, no. 1, pp. 285–299, 2019.

25. Y. Zhang, R. Deng, X. Liu, and D. Zheng, "Outsourcing service fair payment based on blockchain and its applications in cloud computing," IEEE Transactions on Services Computing, vol. 123, pp. 89–100, 2018.
26. Y. Zhang, R. H. Deng, X. Liu, and D. Zheng, "Blockchain based efficient and robust fair payment for outsourcing services in cloud computing," Information Sciences, vol. 462, pp. 262–277, 2018.
27. H. Li, F. Zhang, J. He, and H. Tian, "A searchable symmetric encryption scheme using blockchain," 2017, http://arxiv.org/ abs/1711.01030.
28. H. Li, H. Tian, F. Zhang, and J. He, "Blockchain-based searchable symmetric encryption scheme," Computers & Electrical Engineering, vol. 73, pp. 32–45, 2019.
29. L. Chen, W.-K. Lee, C.-C. Chang, K.-K. R. Choo, and N. Zhang, "Blockchain based searchable encryption for electronic health record sharing," Future Generation Computer Systems, vol. 95, pp. 420–429, 2019.
30. Y. Zhang, R. Deng, E. Bertino and D. Zheng, "Robust and universal seamless handover authentication in 5g Hetnets," IEEE Transactions on Dependable and Secure Computing, vol. 99, 2019.
31. N. Szabo, "Formalizing and securing relationships on public networks," First Monday, vol. 2, no. 9, 1997.
32. A. Miller, Z. Cai, and S. Jha, "Smart contracts and opportunities for formal methods," in Proceedings of the International Symposium on Leveraging Applications of Formal Methods, pp. 280–299, Springer, Limassol, Cyprus, November 2018.
33. S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren, "Searching an encrypted cloud meets blockchain: a decentralized, reliable and fair realization," in Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications, pp. 792–800, IEEE, Honolulu, Hawaii, USA, April 2018.
34. L. Xu, J. Li, X. Chen, W. Li, S. Tang, and H.-T. Wu, "Tc-PEDCKS: towards time controlled public key encryption with delegatable conjunctive keyword search for internet of things," Journal of Network and Computer Applications, vol. 128, pp.11–20, 2019.
35. L. Mei, C. Xu, L. Li, "Efficient Forward and Backward Private Searchable Symmetric Encryption for Multiple Data Sources". Advances in Artificial Intelligence and Security. ICAIS 2021. Communications in Computer and Information Science, vol 1424. Springer, Cham. https://doi.org/10.1007/978-3-030-78621-2_10.