Hindawi
Security and Communication Networks
Volume 2022, Article ID 2309834, 15 pages
https://doi.org/10.1155/2022/2309834

Research Article

A Lightweight Blockchain-based Public-Key Authenticated Encryption with Multi-Keyword Search for Cloud Computing

Haorui Du,[1] **Jianhua Chen**,[1] **Fei Lin**,[2] **Cong Peng**,[3] **and Debiao He**[3]

[1]*The School of Mathematics and Statistics, Wuhan University, Wuhan, China*
[2]*Wuhan Maritime Communication Research Institute, Wuhan, China*
[3]*The School of Cyber Science and Engineering, Wuhan University, Wuhan, China*

Correspondence should be addressed to Jianhua Chen; chenjh_ecc@163.com

Received 21 June 2022; Revised 29 August 2022; Accepted 14 September 2022; Published 15 October 2022

Academic Editor: Youwen Zhu

Cloud computing can provide users with sufficient computing resources, storage, and bandwidth to meet their needs. Data security and privacy protection are among the new threats faced by users. Searchable encryption is the combination of search technology and encryption technology. Searchable encryption can upload the user's data to the cloud server after special encryption, and can realize the function of retrieving according to keywords. Comparatively to symmetric searchable encryption (SSE), public key searchable encryption (PEKS) simplifies key management greatly. However, most existing public key authenticated encryption with keyword search (PAEKS) schemes are based bilinear pairing, making them computationally expensive. Apart from this, complex retrieval requirements and the integrity of the results had not been considered. To address these problems, we propose a blockchain-based PAEKS schemes supporting multi-keyword queries and integrity verification. In addition, we provide security proofs for the PAEKS scheme under the decisional oracle Diffie-Hellman (DODH) assumption. This scheme a scheme that requires less storage and computational power than other schemes of the same kind.

1. Introduction

With the rapid development of mobile Internet, cloud computing is also gradually changing people's lifestyles. A service provider aggregates a large amount of computing and storage resources to the cloud. It provides services on-demand to users with limited resources to facilitate a variety of diverse and personalized applications. Based on the high efficiency and low cost that cloud computing can provide, cloud computing has attracted a great deal of attention from academia as well as industry in the past decade. However, third-party service providers are not completely trustworthy, and storing data in plaintext will definitely pose a serious threat to the privacy of data leakage.

The data on an untrusted server must be encrypted before it can be accessed. Data encryption ensures that nobody can access information without a key. Moreover, the data owner also loses the ability to retrieve the data. A conventional approach is to download all the files locally, decrypt them, and then search for them. It is not practical for users to use this method. Another approach is to delegate retrieval to the server, which finds the file and returns it to the user. However, it is a challenge for the server to find what it needs from the encrypted file. To query the encrypted file, the server needs the user's key to decrypt it. Thus, encryption is rendered ineffective. As a result, it is hoped the server side has access to as much search functionality as possible without having to decrypt the data. Researchers proposed a technique called searchable encryption (SE), which could be used for private encryption and search.

There are two types of searchable encryption: symmetric searchable encryption (SSE) and public key encryption with keyword search (PEKS). Song et al. [1] introduced the first practical scheme for searchable encryption, allowing text search without compromising confidentiality. In 2004, Boneh et al. [2] published the first study studying the search problem of data encrypted by the public key system, used as the prototype of all PEKS. It mainly solved how to find the

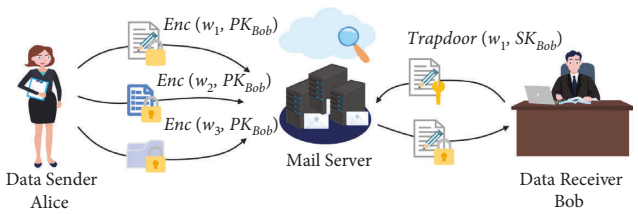

Figure 1: Models of PEKS.

mail required by users from the massive mail system. We must therefore understand the PEKS model clearly. As depicted in Figure 1, a PEKS scheme works as follows:

(i) The user Alice encrypts the file with a symmetric key. Then the user encrypts the keyword w_1, w_2, w_3 with Bob's public key and sends them to the mail server. Finally, the user sends the encrypted file and keyword ciphertext to the mail server.

(ii) The user Bob generates a search trapdoor of w_1 using his secret key, and sends a query to the mail server. Data users who wish to search for data on the cloud server generate search trapdoor information using their secret keys. Bob sends the search trapdoor to the cloud server.

(iii) By matching the ciphertext of the keywords, the mail server delivers the search results to Bob.

SE is invalid, however, when you cannot securely share a secret key over e-mail. The PEKS can effectively solve the key transfer and key management problems in symmetric searchable encryption. Researchers strived to design efficient and effective mechanisms to perform searches on encrypted data. However, PEKS naturally has two major shortcomings.

(i) Having to deal with an ever-growing dataset size and requiring many public key operations limits the usefulness of PEKS.

(ii) Keyword guessing attacks (KGA) and inside keyword guessing attacks (IKGA) can be applied when the space of keywords is smaller than the space of keys.

To resist keyword guessing attacks (KGA) from outside, Rhee et al. [3] constructed a public key encryption scheme for keyword searches of a specified searcher by introducing random numbers in the trapdoor, making it possible to run the test algorithm only on a specified server, thus preventing KGA by external adversaries. Further, the researchers considered the case where the adversary is a malicious server. As a result, the methods are divided into two categories roughly. A potential solution is to expand the range of keywords, which will make keyword guesses more difficult [4]. The other solution is to limit the adversary's ability to create ciphertexts and trapdoors, so that keyword guessing attacks cannot be tested in large numbers by Huang and Li [5]. The methods are resistant to IKGA, but they ignore the fact that trapdoors are algorithmically generated deterministically, which can give away the user's search habits. To meet this goal, the first challenge is to design a searchable encryption scheme that is resistant to IKGA and protects user search patterns and access patterns.

As far as practicality and efficiency are concerned, Xu et al. [6] presented a lightweight PEKS scheme that was quite similar to some practical SSE schemes in terms of search performance. Similarly, those schemes [7–9] also designed the storage structure of ciphertext to improve the retrieval efficiency. Their approach was effective in terms of efficiency. However, we take another perspective is to reduce the consumption of public key operations, thus improving the retrieval efficiency and enhancing the utility. In addition, the current PAEKS schemes do not take into account multi-keyword search, which can affect the user's experience. The second challenging goal of this scheme is to design an effective searchable encryption scheme that can be searched by multiple keywords.

Third-party servers often benefit in practice from providing retrieval services with a risk that ciphertext data may be tampered with by malicious parties or that the server may fail. As a consequence, the integrity of search results is also a concern that cannot be overlooked. Azraoui et al. [10] combined polynomial-based accumulators and merkle trees to implement conjunctive keyword verification. The results of multi-keyword searches used by Wan and Deng [11] were verified using homomorphic Mac. Verifying search results in a flexible and feasible manner was possible by using a blockchain, which proved the reliability and fairness of the process by its non-repudiable property [12–15]. Therefore, designing a PAEKS that supports integrity verification of the retrieved results is the third challenge goal.

1.1. Motivation and Contributions. We analyzed the problems in existing schemes to establish the basis for achieving the above goals. First, the cloud server can perform unlimited trapdoor testing and cipher text matching. The server is too powerful for users to moniton, which can reveal the user's access patterns. Therefore, we prefer that the recipient sends the file identifier to the server itself. In addition, trapdoors for the same keyword are identical, which may reveal the user's search pattern. We introduce random numbers in trapdoor generation to blind trapdoors, and we can also use multiple keywords to reduce the risk of leakage. Secondly, most existing PAEKS algorithm is based on the bilinear map. The generation of ciphertexts and the high overhead of trapdoor computation can affect the experience of computationally constrained users. At the same time, the cloud server has to perform matching test for each ciphertext one by one in the trapdoor retrieval phase, which leads to a large computational burden on the cloud server. This makes the overall utility of the solution low. Besides, most existing PAEKS either do not consider the extension of multi-keyword search or the cost of extending to multiple keywords is costly. Therefore, we use a computationally inexpensive public key cryptographic primitive with support for multi-keyword search. Finally, the integrity of the search results is not considered. Therefore, we should use some new tools to solve this problem efficiently. Following the analysis above, we constructed a new PAEKS scheme

based on our design goals. The contributions of this research are as follows:

(i) Firstly, we design a blockchain-based public key authenticated encryption scheme. Data sender and data receiver can naturally share a common initial key based on Diffie-Hellman key agreement, which can resist IKGA. The search and access pattern are not compromised.

(ii) Secondly, our scheme greatly reduces the computational cost and improves the practicability of the scheme. Specifically, it can reduce the computational cost of ciphertext and trapdoor without using bilinear pairs. In the testing phase, we use the inner product relation of two vectors to determine the set inclusion.

(iii) Finally, our scheme supports the integrity verification of the results. It ensures the reliability of search results by using the tamper proof characteristics of blockchain. We demonstrate that the scheme has adaptive security for CKA, and we perform a series of experiments to evaluate its performance.

1.2. Organization. We introduce a review of the related works in Section 2. Preliminaries containing some cryptographic notations and the system model of our proposed system are introduced in Section 3. We introduce the basic BB-PAKES scheme in Section 4. We give the security proof of scheme in Section 5. We show the comparison of time consumption with other schemes in Section 6. Finally, we conclude this paper in Section 7.

2. Related Works

In the cloud environment, searchable encryption provides a robust solution for retrieving ciphertexts under privacy protection. In this case, the cloud server can search among the ciphertext data uploaded by users and return ciphertext data matching the search criteria when the plaintext information is unavailable. Boneh et al. [2] introduced the public key encryption with keyword search, which is of great significance. The performance of SSE is generally better than the performance of PEKS. Beak et al. [16] pointed out [2] scheme utilized a secure channel between communication parties. The cost of building a secure channel made it not suitable for certain applications. Meanwhile, the adversary could intercept the trapdoor transmitted using a non-secure channel, creating security problems. Reference [16] used the server's public key to encrypt the trapdoor to avoid these drawbacks, and proposed an improved keyword search public key encryption scheme to ensure the security of the trapdoor in the transmission process. There is only one server that can perform the search, and it was the first searchable public key encryption with designated tester (dPEKS) that improved the security of PEKS.

In contrast, Byun et al. [17] observed that the keyword space was much smaller than the ciphertext space, and that we could brute force keywords with low entropy content more easily. Based on these facts, [17] pointed out that [2] could not resist (outside) keyword guessing attacks (KGA). Reference [17] provided specific methods of obtaining keyword information from any captured query message. Yau et al. [18] based on the attack by [17], and presented an attack method against the scheme by [16]. The above literature analysis showed that it was not possible to simply combine keywords and secret keys to generate a trapdoor that was resistant to keyword guessing attacks. This means that any insider/outsider attacker can associate the combination with the public key through pairing, which leads to an offline keyword guessing attack.

To resist KGA, Rhee et al. [3] constructed a designated searcher keyword search public key encryption scheme, which restricted external adversaries from running the test algorithm. Hu and Liu [19] pointed out that [3] could not resist the outside keyword guessing attack of the server, and described the specific attack method and two improvements. They reserved the nature that the designated server can only carry out the keyword search and extended the dPEKS scheme to a bidirectional searchable proxy re-encryption with a designated tester scheme (Re-dPEKS).

To achieve IKGA security against inside attacker, Huang and Liu [5] introduced the concept of public key authenticated encryption with keyword search (PAEKS). In PAEKS, the data sender not only encrypts the keyword, but also authenticates it, so that the verifier will believe that the encrypted keyword can only be generated by the sender. Specifically, Qin et al. [20, 21] concluded that [5] efficient was not adequate to capture a realistic threat, called for outside chosen multi-ciphertext attacks, and provided a new PAEKS model, which captured both (outside) chosen multi-ciphertext attacks and (inside) keyword guessing attacks. However, The trapdoor had been fixed, which would reveal the user's retrieval patterns. On the basis of the above research, we considered several issues.

Additionally, some researchers had suggested some encryption schemes that can support complex retrieval such as conjunctive keyword search, subset search, range query, and semantic keyword search [22–25], which improved retrieval accuracy and provided more complex retrieval expressions for the user. Abdelraheem et al. [26] presented a query evaluation scheme that combined SSE with Bitmap indexes. However, it required two rounds of interaction with the cloud server. Katz et al. [27] constructed a scheme for evaluating inner products based on predicates, which evaluated disjunctions, polynomials, thresholds, and more. It was possible to increase query efficiency by reducing the bilinear pairing during the search process [28–30]. Zhang et al. [31] proposed a public key encryption scheme based on a tree-based index structure. However, there was a keyword arrangement table between the data sender and the data receiver by default. Otherwise, the probability of successful retrieval was very low.

Integrity verification aspects, Cheng et al. [32] proposed a symmetric searchable encryption scheme with verifiable integrity, which applies indistinguishable obfuscation techniques to counter server attacks. It achieved that a

malicious server cannot tamper with the search results arbitrarily, but the indistinguishable obfuscation technique was too ideal. Therefore, it was difficult to implement in practical applications and did not have application value. Wang and Fan [33] proposed a lightweight symmetric searchable encryption scheme that implements support for search result integrity detection and also enables dynamic updating of ciphertexts corresponding to search keywords. They mainly used a tree structure to improve the update efficiency, but only single-user uploading and retrieval of ciphertexts is possible.

3. Preliminaries

As part of this section, we describe notations, cryptography materials, blockchains, the Bloom Filter, and system goals.

3.1. Notations. There is a description of the notations in Table 1.

3.2. Blockchain. Decentralization, public verification, transparency, open audit, and antitampering are characteristics of blockchain, a technology that is gaining attention from academia and industry alike. All nodes have access to the data stored on the blockchain since blockchain runs without a central server. Each node participates and generates calculations, which are stored on the blockchain. Blockchains maintain the same data between all nodes thanks to the consensus mechanism, meaning that no single node can change the recorded data. Given these characteristics, the blockchain can act as a trusted third party for fair verification.

3.3. Bloom Filter. The bloom filter (BF) is a probabilistic data structure. It is very useful for performing collection membership tests quickly and in a space-saving manner, but it has the drawback that false positives do occur [34–36]. False positives are rare enough that many applications outweigh this disadvantage. Particularly, the BF in consists of three polynomial-time algorithms $BF = (BF.Gen, BF.Upd, BF.Check)$:

(i) $BF.\mathrm{Gen}(L, J)$: It requires two integers as inputs $L, J \in N$, and selects a collection of hash functions $\mathscr{H} = \{h_j(-,k)\}_{j\in[J]}$, where $h_j(-,k)$: $\{0,1\}^* \longrightarrow [L]$ is from $\{0,1\}^*$ to a set $[L]$, and k is key. Finally, it outputs $\mathscr{H}$ and an initial L-bit array $BF = 0^L$ with each bit $BF[i]$ for $i \in [L]$ set to 0.

(ii) $BF.Upd(\mathscr{H}, BF, \mathrm{Meg})$: It takes $\mathscr{H} = \{h_j(\mathrm{Meg}, k)\}_{j\in[J]}$, $BF \in \{0,1\}^L$ and an element $\mathrm{Meg} \in \{0,1\}^*$, updates the current array BF by setting $BF[h_j(\mathrm{Meg}, k)] \leftarrow 1$ for all $j \in [J]$, and finally outputs the updated BF.

(iii) $BF.\mathrm{Check}(\mathscr{H}, BF, \mathrm{Meg})$: It takes $\mathscr{H} = \{h_j(\mathrm{Meg}, k)\}_{j\in[J]}$, $BF \in \{0,1\}^L$ and an element $\mathrm{Meg} \in \{0,1\}^*$, and checks if $BF[h_j(\mathrm{Meg}, k)] = 1$ for all $j \in [J]$. If ture, it outputs 1, otherwise returns 0.

Table 1: Summary of notations.

Symbol	Description
G_1	The cyclic group of prime order p
g	The generator of group G_1
H_1, H_2, H_3	Hash functions
PK, SK	Participant's public and private keys
W	The keywords set
Doc	The files set
$[L]$	The number of values in the range $[0, L-1]$
J	The number of the hash functions
BF	Bloom filter
$\overrightarrow{BF}$	L-dimensional vector
$\mathscr{H}$	The hash function family with key
ind_w	A set of file identifiers containing keyword w
EDB	Keyword ciphertext set
Acc	Hash check list of ciphertext
Rlt	Search result set
$\langle \overrightarrow{\ }, - \rangle$	Vector inner product
$\lvert \overrightarrow{\theta} \rvert$	The euclidean norm of an l-vector $\overrightarrow{\theta}$

3.4. Set Containment Search and Vector Scalar Product. One solution to perform aggregate containment search on outsourced data is to transform the aggregate records into a set of binary vectors with fixed dimensions. Based on the scalar product of the set containment problem, one can transform it into a vector search problem [37].

3.5. Pattern Leakage. Let Q be a q-query set whose element is a pair (i, w), where i denotes the timestamp of a query, and w denotes a keyword. The leakage can be represented as follows [38]:

(i) *Access Pattern.* An access pattern describes whether a document contains a query keyword. For each query keyword w, an access pattern is defined as $ap(w) = \{ID(w)\}$, where $ID(w)$ represents the document's ID number.

(ii) *Search Pattern.* It shows the query patterns for every keyword w. The search pattern for each keyword is $sp(w) = \{i | (i, w) \in Q\}$.

3.6. Inverted Index. Database indexes store maps between content (such as words or numbers) and their location in a table, or in a file or group of files.

(i) *Create Inverted Index.* First, all the raw data are numbered to form a list of documents. Then the document data is extracted to obtain a large number of keywords, which are indexed by entry. The numbering information of the documents containing these entries is kept. It can also be referred to as an index matrix. As depicted in Figure 2.

(ii) *Search Process.* The user enters any keyword and brings the keyword to an inverted index list for matching. By looking up these terms, the numbers of all documents containing them can be found. Documents are then found in the document list based on these numbers.

ID	*file*	*Keyword*
1	f_1	w_1, w_2, w_4, w_5
2	f_2	w_1, w_3
3	f_3	w_1, w_2, w_4
4	f_4	w_1, w_2, w_3, w_4, w_5
5	f_5	w_3, w_4

(a)

	f_1	f_2	f_3	f_4	f_5
w_1	1	1	1	1	0
w_2	1	0	1	1	0
w_3	0	1	0	1	1
w_4	1	0	0	1	1
w_5	1	0	0	1	0

(b)

FIGURE 2: Inverted index. (a) Extract keywords from files. (b) Index matrix.

3.7. The Computational Diffie-Hellman Assumption (CDH). Let G be a cyclic group. The order is p. The generator is g. Given two elements of G, g^a and g^b, it is required to compute g^{ab}. CDH is intractable in the underlying group G.

3.8. The Decisional Oracle Diffie-Hellman (DODH). The G is a group. The order is p. The g is a random generator of G. The H: $\{0,1\}^* \longrightarrow \{0,1\}^{h\text{Len}}$ is hash function. The DODH assumption means that the advantage of $\mathscr{A}$ is negligible i $\text{Adv}_{\mathscr{A}}^{\text{DODH}}(\lambda) = |\Pr[\mathscr{A}(g, g^a, g^b, R) = 1] - \Pr[\mathscr{A}(g, g^a, g^b, H(g^{ab})) = 1]|$ where $a, b \in Z_p, R \in \{0,1\}^{h\text{Len}}$ for any PPT adversary $\mathscr{A}$.

3.9. The Definition of BB-PAEKS. A *BB − PAEKS* scheme is consisted of eight PPT algorithms, namely, *Setup*, *KeyGen*, *PAEKS*, *Trapdoor*, *Search*, *Verify*, *Dec*, *Update*. The formal constructions are as follows:

The formal constructions are as follows:

(i) $pp \leftarrow \text{Setup}(\lambda)$: This algorithm generates global parameters. Input a security parameter λ, and outputsa global public parameter pp.

(ii) $(PK, SK) \leftarrow \text{KeyGen}(pp)$: It is responsible for maintaining the public and private keys for participants.

(a) $(PK_{DS}, SK_{DS}) \leftarrow \text{KeyGen}(pp)$: An algorithm that produces public and private keys for a sender. Input pp, and generates (PK_{DS}, SK_{DS}) for the sender.

(b) $(PK_{DR}, SK_{DR}) \leftarrow \text{KeyGen}(pp)$: This algorithm generates the public/private key pair for the receiver using input pp and outputs (PK_{DR}, SK_{DR}).

(iii) $C_w \leftarrow \text{PAKES}(PK_{DR}, SK_{DS}, w)$: The process is carried out by a data sender. Input PK_{DR}, SK_{DS}, w, and returns corresponding keyword's ciphertext C_w.

(iv) $T_{w'} \leftarrow \text{Trapdoor}(PK_{DS}, SK_{DR}, w')$: It is performed by a data user. Input PK_{DS}, SK_{DR}, w', and returns corresponding keyword's trapdoor $T_{w'}$.

(v) $\{0,1\} \leftarrow \text{Search}(C_w, T_w)$: The algorithm is run by a SP.

(vi) $\text{proof} \leftarrow \text{Verify}(Rlt, Acc)$: The algorithm is run by Blockchain.

(vii) $\text{ind}_w \leftarrow \text{Dec}(Rlt, SK_{DR})$: The algorithm is run by DR.

(viii) $(C_w^1, C_w^2) \leftarrow \text{Update}(w, k)$: The algorithm is run by DR.

3.9.1. Correctness. For a PAEKS scheme, Given C_w and $T_{w'}$, it can be converted into two vectors $\overrightarrow{C_w}, \overrightarrow{T_{w'}}$. We formulate consistency as below:

If $w = w'$, then $\langle \overrightarrow{C_w}, \overrightarrow{T_{w'}} \rangle = |\overrightarrow{C_w}|$; else $w \neq w'$, then $\langle \overrightarrow{C_w}, \overrightarrow{T_{w'}} \rangle \neq |\overrightarrow{C_w}|$.

3.10. The Security Models of BB-PAKES. For *BB* − PAEKS, the semantic security model incorporates both the indistinguishability of cipher-keywords and the indistinguishability of trapdoors. Our ciphertext indistinguishable security (CI-security) and Trapdoor indistinguishable security (TI-security) definition is identical to that of [5, 20, 21] in settings.

3.10.1. CI-Security Model. Assume $\mathscr{A}$ is an adversary, while λ is the security parameter.

(i) *Initialization.* A challenger generates the system parameter pp by running the setup algorithm $\text{Setup}(\lambda)$. Then, it runs $\text{KeyGen}(pp)$ to generate the target DS's public/private key pairs (PK_{DS}, SK_{DS}), DR's public/private key pairs (PK_{DR}, SK_{DR}) respectively. The challenger provides pp and PK_{DS} and PK_{DR} to $\mathscr{A}$.

(ii) *Phase 1.* In this adaptive attack, the adversary asks PPT questions to the cipher oracle $(\mathcal{O}_{\mathscr{C}})$ and trapdoor oracle $(\mathcal{O}_{\mathscr{T}})$, and receives the cipher and trapdoor for query keywords.

(iii) *Challenge.* After Phase 1, it outputs two challenge keywords w_0^* and w_1^*. A coin $b \in [0,1]$ is flipped by the challenger. It sends the cipher-keyword $C_{w_b^*} \leftarrow \text{PAKES}(PK_{DR}, SK_{DS}, w_b^*)$ to $\mathscr{A}$.

(iv) *Phase 2.* While the adversary may use the Phase 1 oracle to continue to query the oracles, they may not access $\mathcal{O}_{\mathscr{C}}$ and $\mathcal{O}_{\mathscr{T}}$ with w_0^* and w_1^*.

(v) *Guess.* Finally, $\mathscr{A}$ returns a bit $b' \in [0,1]$ as the guess of b and wins the game if $b' = b$.

The adversary $\mathscr{A}$ wins in the above game if he guesses $b' = b$. The $\mathscr{A}$'s advantage in winning this game is defined as,

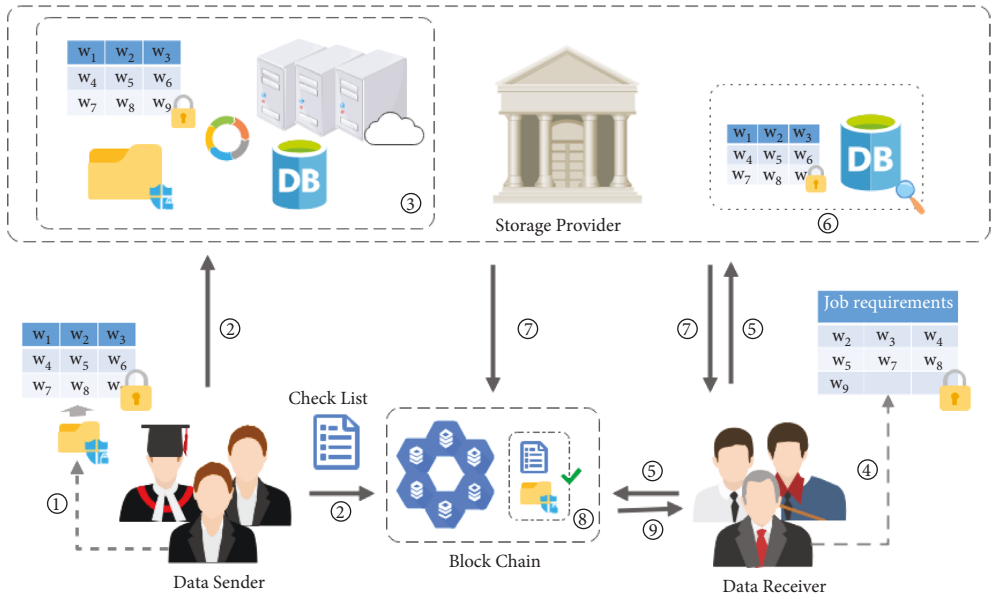

Figure 3: The system model of BB-PAEKS.

$$\mathrm{Adv}_{\mathscr{A}}^{IND-CKA}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|. \quad (1)$$

Definition 1. (CI-security) *ABB − PAEKSscheme satisfies cipher-keyword indistinguishable against chosen keywords attacks if for any PPT adversary$\mathscr{A}$, the advantage-*$\mathrm{Adv}_{\mathscr{A}}^{IND-CKA}(\lambda)$*of succeeding in the above game is negligible.*

3.10.2. TI-Security Model. The adversary $\mathscr{A}$ and the security parameter λ.

(i) *Initialization.* Challenger $\mathscr{C}$ generates the system parameter pp by running the setup algorithm $Setup(\lambda)$. Then, it runs $KeyGen(pp)$ to generate the target DS's public/private key pairs (PK_{DS}, SK_{DS}), DR's public/private key pairs (PK_{DR}, SK_{DR}) respectively. The $\mathscr{C}$ provides pp, PK_{DS} and PK_{DR} to $\mathscr{A}$.

(ii) *Phase 1.* The adversary adaptively performs PPT queries on password-keyword oracles PAEKS and trapdoor oracles, and obtain cipher and trapdoors for the query keywords.

(iii) *Challenge.* After the first phase, it outputs two challenge keywords w_0^* and w_1^* with the constraints w_0^* and w_1^*. In the first stage, w_0^* and w_1^* are never queried by $\mathscr{A}$ for the cryptographic keyword oracle and the trapdoor oracle. The challenger then picks a coin $b \in [0, 1]$ and sends the trapdoor $T_{w_b^*}$. Sends the trapdoor $T_{w_b^*} \leftarrow$ Trapdoor $(PK_{DS}, SK_{DR}, w_b^*)$ to $\mathscr{A}$.

(iv) *Phase 2.* As Phase 1, the adversary can continue to query the oracles, but cannot query $\mathcal{O}_{\mathscr{C}}$ and $\mathcal{O}_{\mathscr{T}}$ with w_0^* and w_1^*.

(v) *Guess.* Finally, $\mathscr{A}$ returns a bit $b' \in [0, 1]$ as the guess of b and wins the game if $b' = b$.

The adversary $\mathscr{A}$ wins in the above game if he guesses $b' = b$. The advantage of $\mathscr{A}$ winning this game is defined as,

$$\mathrm{Adv}_{\mathscr{A}}^{IND-KGA}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|. \quad (2)$$

Definition 2. (TI-security) *ABB − PAEKSscheme satisfies trapdoor indistinguishability against chosen keywords attacks if for any PPT adversary$\mathscr{A}$, the advantage*$\mathrm{Adv}_{\mathscr{A}}^{IND-KGA}(\lambda)$*of succeeding in the above game is negligible.*

3.11. System Model. The system includes the following parties: Storage Provider (SP), Data Sender (DS), Data Receiver (DR) and Blockchain as depicted in Figure 3. The characteristic and function of each party are depicted as follows.

(i) Storage Provider (SP): Service providers have strong computing and storage capabilities. The SP stores encrypted data on behalf of the data sender, and SP retrieves the corresponding ciphertext when DR submits a retrieval request. The final step involves returning the retrieval result to the DR and blockchain.

(ii) Data Sender (DS): The DS collects fle documents with identifiers $ID = \{id_1, id_2, , \ldots, id_{\mathrm{fle}}\}$. Each document id_i contains a set of keywords W_i, which is a subset of the collection of all keywords $W = \{w_1, w_2, \ldots, w_{\mathrm{num}}\}$, i.e., $W_i \subset W$. As a measure of improving query efficiency, DS creates an inverted index database DB for his keywords. For different keywords W_i, there is a set of document identifiers, denoted by ind_i. After that, DS encrypts all files, keyword and ind. The DS outsources the keyword ciphertext T_B and document ciphertext T to the SP. At the same time, the DS performs a hash

operation on the encrypted file identifier set B and sends it to the blockchain.

(iii) Data Receiver (DR): The DR generates search token according to the keywords and then sends the trapdoor to SP and BlockChain, respectively.

(iv) Blockchain: The blockchain is composed of various nodes (such as recruitment companies, candidates, and other identities). Its primary responsibility is to maintain the blockchain network that supports smart contracts, which can be utilized for storing user data and inspecting documents.

The detailed procedures of $BB-PAEKS$ are as follows: ① The DS extracts keyword set W from documents Doc and builds a checklist B from files. At the same time, encrypt keyword index C_w and documents C_{Doc}. ② The DS sends encrypted keyword index C_w to the SP. Meanwhile, the DS sends checklist Acc to the Blockchain. ③ The SP registers C_w and C_{Doc} to the database. ④ The DR computes trapdoor T_w of queried keywords W_q. ⑤ The DR sends trapdoor T_w to the SP and Blockchain. ⑥ The SP searches keyword index C_w in the database by using trapdoor T_w. ⑦ The SP returns the result to the DR and blockchain for verification. ⑧ As soon as the SP returns the results, the blockchain calculates the hash value of every result. From the identity information of DS in the checklist Acc, the blockchain gets the corresponding hash values of files. Finally, the blockchain compares two hash value to generate the $proof$. ⑨ The blockchain sends the $proof$ to DR.

3.12. Threat Model. In this paper, it suppose that DS honestly follow the *PAEKS* algorithm to produce searchable ciphertexts for DO and transmits these ciphertexts to the SP. DR honestly follow the *Trapdoor* algorithm to produce trapdoor and transmits these to the SP. The SP is supposed to be honest and curious, who will honestly perform Test algorithm, and is interested in query results and frequency information of ciphertext. Blockchain is trusted and executes the protocols in the system honestly.

3.13. Design Goals. According to the most basic requirements of public key searchable encryption, our scheme must meet the following characteristics:

(i) Correctness: If the DR provides the correct trapdoor and the scheme is executed in the correct way, the received search results must be correct.

(ii) Confidentiality: The scheme needs to protect the confidentiality of keywords, files, indexes and trapdoors. In other words, the scheme can resist KGA and IKGA.

(iii) Integrity: The scheme supports multi-keyword retrieval and integrity verification of search results.

4. Construction

This section presents a basic construction of PAEKS followed by an extension of the base version. It supports multi-keyword retrieval. Inverted index-based data structures, as illustrated in Figure 2, are better suited to our construction. Our discussion in this paper does not include the topic of how to encrypt files, which is not the focus of our discussion.

4.1. Basic Construction. Our scheme is consisted of eight PPT algorithms, namely, *Setup*, *KeyGen*, *PAEKS*, *Trapdoor*, *Search*, *Verify*, *DEC*, *Update*. The formal constructions are as follows:

(i) $pp \leftarrow Setup(\lambda)$: Choose a G_1 is cyclic groups with prime order p. Select three hash functions H_1: $G_1 \times \{0,1\}^* \longrightarrow Z_p^*$, H_2: $\{0,1\}^* \longrightarrow \{0,1\}^*$, H_3: $Z_p^* \times \{0,1\}^* \longrightarrow Z_p^*$ and a collection of hash functions $\mathscr{H} = \{h_j(k,-)\}$ with key k, where $j \in [J]$. Choose a random generator g of G_1. Output the public parameters $pp = \{G_1, p, g, H_1, H_2, H_3, \mathscr{H}\}$.

(ii) $(PK, SK) \leftarrow KeyGen(pp)$: It is responsible for the public and private key pair of participants.

(a)The DS chooses a random element $y \in Z_p^*$, and sets to $PK_{DS} = g^y \bmod p$, $SK_{DS} = y$.
(b)The DR selects randomly $x \in Z_p^*$, and sets to $PK_{DR} = g^x \bmod p$, $SK_{DR} = x$.

(iii) $(C_w^1, C_w^2) \leftarrow PAEKS(SK_{DS}, PK_{DR}, w)$:

(a)Data Sender.
Version serial number VerInfo $\in Z_p^*$.
DS runs $BF.\mathrm{Gen}(L, J)$.
Compute $u = g^{yx}$, and $k = H_1(u, w)$, which the key is updated with the version information.
Input k and w, run $BF.Upd(\mathscr{H}, BF_{DR}, w)$ algorithm, and output the updated bloom filter BF_{DS}, where a collection of hash functions $\mathscr{H} = \{h_j(k,w)\}$ with key k and keyword w for $j \in [J]$.
$ind_w = \{id_i | w \in id_i\}$.
Choose random $r_1 \in Z_p^*$, $C_1 = g^{r_1}$, $C_2 = g^{xr_1} \cdot \mathrm{ind}_w$, $C_3 = H_2(C_2)$.
Send the $C_w^1 = \{BF_{DS}, C_1, C_2, \mathrm{VerInfo}\}$ to SP, and send the $C_w^2 = \{C_3, \mathrm{VerInfo}\}$ to Blockchain.
(b)Storage Provider.
Store C_w^1 to EDB.
(c)Blockchain
Store C_w^2 to Acc.

(iv) $T_w \leftarrow Trapdoor(SK_{DR}, PK_{DS}, w)$: The algorithm is run by a DR.

(a)Data Recriver.
DR runs $BF.\mathrm{Gen}(L, J)$.
Compute $u = g^{yx}$, and $k = H_1(u, w)$, which the key is updated with the version information.
Input k and w, run $BF.Upd(\mathscr{H}, BF_{DR}, w)$ algorithm, and output the updated bloom filter BF_{DR}, where a collection of hash functions $\mathscr{H} = \{h_j(k,w)\}$ with key k and keyword w for $j \in [J]$.
Randomly select J random numbers $\mathrm{num}_1, \mathrm{num}_2, \ldots, \mathrm{num}_J \in \{0,1\}^*$, and the corresponding

position of the bloom filter BF_{DR} is set to 1, where J means the number of pseudo-random functions in bloom filter.

$T_w = BF_{DR}$.

Send T_w to SP and Blockchain.

(v) $\{0,1\} \leftarrow Search(C_w, T_w)$: The algorithm is run by a SP.

(a)If $\langle \overrightarrow{BF_{DS}}, \overrightarrow{BF_{DR}} \rangle = |\overrightarrow{BF_{DS}}|$, the search algorithm returns 1. Otherwise, it returns 0.
(b)Puts the matching ciphertext C_w^1 into the map Rlt.
(c)Sends the result Rlt to the DR and Bolckchain.

(vi) proof $\leftarrow Verify(Rlt, \text{Acc})$: The algorithm is run by Blockchain.

(a)Gets the search result Rlt, obtains the hash value of each file in the result from B, and gets the Acc.
(b)For all ciphertexts $C_w^1 \in Rlt$, blockchain computes the hash value of C_2, gets $H_2(C_2)$.
(c)Blockchain compares $H_2(C_2)$ and Acc, if they are equal, the proof is true, otherwise false.
(d)The search results Rlt and proof are sent to DR.

(vii) $\text{ind}_w \leftarrow Dec(Rlt, SK_{DR})$: The algorithm is run by DR.

(a)Gets the search result Rlt.
(b)Compute $\text{ind}_w = C_2 / C_1^x$.

(viii) $(C_w^1, C_w^2) \leftarrow Update(w, k)$: The algorithm is run by DR.

(a)Data Receiver. The ciphertext update algorithm here is the same as PAKES, except for the key generation method

Version serial number $\text{VerInfo}^* \in Z_p^*$.

DS runs $BF.\text{Gen}(L, J)$.

Compute $u = g^{yx}$, and $k_1 = H_1(u, w)$, $k_2 = H_3(k_1, w)$.

Input k_2 and w, run $BF.\text{Upd}(\mathcal{H}, BF_{DS}, w)$ algorithm, and output the updated bloom filter BF_{DS}.

$\text{ind}_w = \{id_i | w \in id_i\}$.

Choose random $r_1 \in Z_p^*$, $C_1 = g^{r_1}$, $C_2 = g^{xr_1} \cdot \text{ind}_w$, $C_3 = H_2(C_2)$.

Send the $C_w^1 = \{BF_{DS}, C_1, C_2, \text{VerInfo}^*\}$ to SP, and send the $C_w^2 = \{C_3, \text{VerInfo}^*\}$ to Blockchain.
(b)Storage Provider.

Store C_w^1 to EDB.
(c)Blockchain

Store C_w^2 to Acc.

4.2. Derived Constructions. On the basis of scheme BB – PAEKS, it can be easily extended to public key authenticated encryption with multi-keyword search. The Setup, KeyGen, PAEKS, Search, Verify, Update these algorithms are the same as those of the basic scheme, except that the Trapdoor and Dec algorithms are different. In order to save space, please refer to the basic scheme settings for details of specific schemes.

(i)Trapdoor $(SK_{DR}, PK_{DR}, PK_{SP}, w)$:

(a)Data Receiver.

DR runs $BF.\text{Gen}(L, J)$.

For $W_Q = \{w_1, w_2, .., w_q\}$ Compute $u = g^{yx}$, For each keyword w_i in set $W_Q = \{w_1, w_2, .., w_q\}$, perform the following steps, and finally add them to the bloom filter.

$k_i = H_1(u, w_i)$.

Input k_i and w_i, run $BF.\text{Upd}(\mathcal{H}, BF_{DR}, w_i)$ algorithm, and output the updated bloom filter BF_{DR}.

Choose random $J \in \{0,1\}^*$, and set BF_{DR} to 1, where J means the number of pseudo-random functions in bloom filter.

$T_w = BF_{DR}$.

Send T_w to SP and Blockchain.

(ii)Dec (Rlt, SK_{DR}): The algorithm is run by DR.

(a)Gets the search result Rlt.
(b)Run $\text{Dec}(Rlt, SK_{DR})$, get $\{\text{ind}_{w_1}, \text{ind}_{w_2}, \ldots, \text{ind}_{w_q}\}$.
(c)It can get the results of different keyword combinations locally(conjunction and disjunction).

4.2.1. Remark.

(1) It is not linear that the ciphertext and trapdoor size grow with keywords in our scheme. The trapdoor is the same size as the single keyword trapdoor.

(2) Search Index can be executed with a linear search, but a binary tree search is more efficient because we do not have to test all indexes in a binary tree search.

(3) Several factors need to be considered prior to using the bloom filter, including the length and number of pseudo-random functions. We can determine these parameters by using n and f_p, where n means the maximum number of characters and f_p represents the maximum likelihood of false-positive f_p [35, 39]. We select 2000 keywords, the length of the mapping bit array is 20000, the number of hash functions is 5, and the error rate is 0.00943. According to the parameter standard provided by [40].

5. Security Proof

In this section, we show that our basic scheme is compatible with the design goals, in other words, our construction is reliable. It can resist IKGA and reduce the possibility of access pattern and search pattern disclosure.

5.1. Correctness

5.1.1. Correctness. The sender and receiver compute $u = g^{yx}$, $\alpha = H_2(w)^u$, $k = H_1(\alpha, w)$. They run $BF.\text{Gen}(L, J)$

and $BF.\mathrm{Upd}(\mathscr{H}, BF, w)$ algorithm and get BF_{DS} and BF_{DR} about keyword w. Then, it can be proved to be correct according to bloom filter and set intersection property.

5.1.2. Remark. Our scheme does not consider malicious man in the middle attack and malicious tampering by external enemies. It can be guaranteed by public key encryption system and algorithms.

5.2. Confidentiality. The following theorems illustrate how our scheme satisfies IND-CKA and IND-KGA security.

Theorem 1. *Our scheme BB − PAEKS implements IND-CKA security if the DODH assumption holds.*

Lemma 1. *The advantage $Adv_{\mathscr{A}}^{CKA}$ is negligible for any PPT adversary $\mathscr{A}$.*

Proof. Suppose the adversary guesses the key words in the game, and the correct event is recorded as $(b' = b)$. We define games as follows:

Game$_0$. It is the original IND-CKA game.

(i)$\mathrm{Setup}(\lambda)$: The challenger $\mathscr{C}$ runs $\mathrm{Setup}(\lambda)$ and $\mathrm{KeyGen}(pp)$ to generate the public parameter pp and public/private key pair of participants $(PK_{DS}, SK_{DS}) = (g^y \mod p, (y)$ of DS, $(PK_{DR}, SK_{DR}) = (g^x \mod p, (x)$ of DR. Then, the challenger $\mathscr{C}$ sends $\{pp, PK_{DS}, PK_{DR}\}$ to the adversary $\mathscr{A}$. Hash functions H_1, H_2, H_3 and $\mathscr{H}$ should be collision resistant and secure.

(ii)*Phase* 1: In this adaptive attack, the adversary asks PPT questions to the cipher-keyword oracle $(\mathcal{O}_{\mathscr{C}})$ and the trapdoor oracle $(\mathcal{O}_{\mathscr{T}})$, and $\mathscr{C}$ is simulated as follows:

(a)$\mathcal{O}_{\mathscr{C}}$: Run $BF.\mathrm{Gen}(L, J)$. Compute $u = g^{yx}$ and $k = H_1(u, w)$. Input k and w, run $BF.\mathrm{Upd}(\mathscr{H}, BF_{DS}, w)$ algorithm, and output the updated bloom filter BF_{DS}. Send the BF_{DS} to $\mathscr{A}$.
(b)$\mathcal{O}_{\mathscr{T}}$: Run $BF.\mathrm{Gen}(L, J)$. Compute $u = g^{yx}$, and $k = H_1(u, w)$. Input k and w, run $BF.\mathrm{Upd}(\mathscr{H}, BF_{DR}, w)$ algorithm, and output the updated bloom filter BF_{DR}. Choose random $J \in \{0,1\}^*$, and set BF_{DR} to 1, where J means the number of pseudo-random functions in bloom filter. Send trapdoor BF_{DR} to $\mathscr{A}$.

(iii)*Challenge*: As the adversary $\mathscr{A}$ chooses keywords (w_0^*, w_1^*), and sends them to $\mathscr{C}$. The challenger $\mathscr{C}$ picks a random bit $b \in \{0,1\}$, and encrypts the keyword w_b^*. The ciphertext of the challenge keyword w_b^* as follows:

(a)Choose a random $u = g^{yx}$, and $k = H_1(u, w_b^*)$. $\mathrm{RunBF.Upd}(\mathscr{H}, BF_{DS}^*, w_b^*)$, output BF_{DS}^*.
(b)Finally, the challenge $\mathscr{C}$ sends $C_{w_b^*}^*$ to adversary.

(iv)*Phase* 2: The adversary $\mathscr{A}$ proceeds to query oracles during *Phase*1. It cannot query the ciphertext and trapdoor of (w_0, w_1).

(v)*Guess*: If $b' = b$ then $\mathscr{C}$ wins the game. The adversary $\mathscr{A}$ has the advantage that is

$$\mathrm{Adv}_{\mathscr{A}}^{\mathbf{Game}_0}(\lambda) = \mathrm{Adv}_{\mathscr{A}}^{\mathbf{CKA}}(\lambda). \tag{3}$$

Game$_1$. Let **Game**$_1$ be the same game as **Game**$_0$, except that the challenger chooses $r_1 \in G_1$ instead of computing $u = g^{yx}$. The challenger sends the ciphertext $C_{w_b^*}^*$. We have $|\mathrm{Adv}_{\mathscr{A}}^{\mathbf{Game}_0} - \mathrm{Adv}_{\mathscr{A}}^{\mathbf{Game}_1}| < \mathrm{Adv}_{\mathscr{A}}^{\mathrm{DODH}}(\lambda)$, where $\mathrm{Adv}_{\mathscr{A}}^{\mathrm{DODH}}(\lambda)$ is negligible if the DODH assumption holds.

Game$_2$. Let **Game**$_2$ be the same game as **Game**$_1$, except that the challenger random chooses r_1' instead of $k = H_1(\alpha, w_b^*)$. Due to r_1 and r_1' are random. From adversary's view, the ciphertext $C_{w_b^*}$ of Game_1 and Game_2 are the identical distribution.

We have,

$$\mathrm{Adv}_{\mathscr{A}}^{\mathbf{Game}_2}(\lambda) = \mathrm{Adv}_{\mathscr{A}}^{\mathbf{Game}_1}(\lambda). \tag{4}$$

We conclude that the adversary can only win with probability in Game_2 because $C_{w_b^*}$ is independent of b. Thus, the advantage,

$$\mathrm{Adv}_{\mathscr{A}}^{\mathrm{Game}_2}(\lambda) = \left|\frac{1}{2} - \frac{1}{2}\right| = 0. \tag{5}$$

Finally, according to the Game_0, Game_1, Game_2 we have,

$$\left|\mathrm{Adv}_{\mathscr{A}}^{\mathbf{Game}_2}(\lambda) - \mathrm{Adv}_{\mathscr{A}}^{\mathbf{IN\ D-CKA}}(\lambda)\right| \le \mathrm{Adv}_{A}^{\mathrm{DODH}}(\lambda). \tag{6}$$

where $\mathrm{Adv}_{\mathscr{A}}^{\mathrm{DODH}}(\lambda)$ are negligible. Therefore, the adversary $\mathscr{A}'$ advantage in winning can be ignored in the IND-KGA game. □

Theorem 2. *Our scheme BB − PAEKS implements IND-KGA security if the DODH assumption holds.*

Lemma 2. *The advantage $Adv_{\mathscr{A}}^{KGA}$ is negligible for any PPT adversary $\mathscr{A}$.*

Proof. Suppose the adversary guesses the key words in the game, and the correct event is recorded as $(b' = b)$. We define games as follows:

Game$_0$. IND-KGA is the original version of this game.

(i)$\mathrm{Setup}(\lambda)$: The challenger $\mathscr{C}$ runs $\mathrm{Setup}(\lambda)$ and $\mathrm{KeyGen}(pp)$ to generate the public parameter pp and public/private key pair of participants $(PK_{DS}, SK_{DS}) = (g^y \mathrm{mod} p, y)$ of DS, $(PK_{DR}, SK_{DR}) = (g^x \mod p, x)$ of DR, of $(PK_{SP}, SK_{SP}) = (g^z \mathrm{mod} p, z)$ of SP. Then, the challenger $\mathscr{C}$ sends $\{pp, PK_{DS}, PK_{DR}, PK_{SP}\}$ to the adversary $\mathscr{A}$. Based on what we know so far about hash functions, we assume H_1, H_2, H_3 and $\mathscr{H}$ are secure and collision-resistant.

(ii)*Phase* 1: In this adaptive attack, the adversary asks PPT questions to the cipher-keyword oracle $(\mathcal{O}_{\mathscr{C}})$ and the trapdoor oracle $(\mathcal{O}_{\mathscr{T}})$, and $\mathscr{C}$ is simulated as follows:

(a)$\mathcal{O}_{\mathscr{C}}$: Run $BF.\mathrm{Gen}(L, J)$. Compute $u = g^{yx}$ and $k = H_1(u, w)$. Input k and w, run $BF.\mathrm{Upd}(\mathscr{H},$

BF_{DS}, w) algorithm, and output the updated bloom filter BF_{DS}. Send the BF_{DS} to $\mathcal{A}$.
(b)$\mathcal{O}_{\mathcal{T}}$: Run $BF.\mathrm{Gen}(L, J)$. Compute $u = g^{yx}$, $k = H_1(u, w)$. Input k and w, run $BF.\mathrm{Upd}(\mathcal{H}, BF_{DR}, w)$ algorithm, and output the updated bloom filter BF_{DR}. Choose random $J \in \{0,1\}^*$, and set BF_{DR} to 1, where J means the number of pseudo-random functions in bloom filter. Send trapdoor BF_{DR} to $\mathcal{A}$.

(iii)*Challenge*: The adversary $\mathcal{A}$ selects two keywords (w_0^*, w_1^*) and sends them to $\mathcal{C}$. The challenger $\mathcal{C}$ picks a random bit $b \in \{0,1\}$, and encrypts the keyword w_b^*. The trapdoor of the challenge keyword w_b^* as follows:

(a)Choose a random $u = g^{yx}$ and $k = H_1(u, w_b^*)$. $\mathrm{RunBF.Upd}(\mathcal{H}, BF_{DR}^*, w_b^*)$, output BF_{DR}^*. Choose random $J \in \{0,1\}^*$, and set BF_{DR} to 1, where J means the number of pseudo-random functions in bloom filter. Finally, the challenge $\mathcal{C}$ sends $T_{w_b^*}^*$ to adversary.

(iv)*Phase* 2: *Phase*1 continues to be issued by the adversary $\mathcal{A}$ as queries to Oracles. There is only one restriction is that (w_0, w_1) cannot be chosen for the ciphertext and trapdoor query.

(v)*Guess*: If $b' = b$ then $\mathcal{C}$ wins the game. According to the IND-KGA definition, adversary $\mathcal{A}$ has the advantage that is

$$\mathrm{Adv}_{\mathcal{A}}^{\mathbf{Game}_0}(\lambda) = \mathrm{Adv}_{\mathcal{A}}^{\mathbf{KGA}}(\lambda). \tag{7}$$

Game$_1$. Let **Game**$_1$ be the same game as **Game**$_0$, except that the challenger chooses $r_1 \in G_1$ instead of computing $u = g^{yx}$. The challenger sends the ciphertext $C_{w_b^*}^*$. We have $|\mathrm{Adv}_{\mathcal{A}}^{\mathbf{Game}_0} - \mathrm{Adv}_{\mathcal{A}}^{\mathbf{Game}_1}| < \mathrm{Adv}_A^{\mathrm{DODH}}(\lambda)$, where $\mathrm{Adv}_A^{\mathrm{DODH}}(\lambda)$ is negligible if the DODH assumption holds.

Game$_2$. Let **Game**$_2$ be the same game as **Game**$_1$, except that the challenger random chooses r_1' instead of $k = H_1(u, w_b^*)$. Due to r_1 and r_1' are random, which the ciphertext $C_{w_b^*}$ of Game$_1$ and Game$_2$ are the same distribution from adversary's view.

We have,

$$\mathrm{Adv}_{\mathcal{A}}^{\mathbf{Game}_2}(\lambda) = \mathrm{Adv}_{\mathcal{A}}^{\mathbf{Game}_1}(\lambda). \tag{8}$$

As we know, the adversary can only win Game$_2$ with probability since $C_{w_b^*}$ is independent of b. Thus, the advantage,

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Game}_2}(\lambda) = \left|\frac{1}{2} - \frac{1}{2}\right| = 0. \tag{9}$$

Finally, according to the Game$_0$, Game$_1$, Game$_2$ we have,

$$\left|\mathrm{Adv}_{\mathcal{A}}^{\mathbf{Game}_2}(\lambda) - \mathrm{Adv}_{\mathcal{A}}^{\mathbf{KGA}}(\lambda)\right| \le \mathrm{Adv}_A^{\mathrm{DODH}}(\lambda), \tag{10}$$

where $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{DODH}}(\lambda)$ are negligible. Therefore, the advantage of adversary $\mathcal{A}$ wins in the IND-KGA game is negligible. □

5.3. Security. This study, adversary a is mainly aimed at the curious server. That's because resisting external adversary can be easily solved by introducing the server public key. In other words, IKGA is mainly considered. In addition, *PAKES* and *Trapdoor* algorithms are most likely to disclose keyword information, So we focused on the analysis *PAKES* and Trapdoor algorithms.

(i)Resistance against IKGA. In PAEKS, only the DS can generate the legal ciphertext of keyword. The adversary executes the *PAEKS* algorithm, it cannot generate k. Similarly, the adversary cannot generate the trapdoor of keyword. Finally, the adversary cannot obtain any information by running the Test algorithm. Thus, our scheme can resist the IKGA.

(ii)Access pattern. We use DR's public key to encrypt the file identifier ind_w that meets the keyword w. The SP returns the result to DR, who decrypts the file identifier information with his private key. Finally, the DR submits file identifier to SP according to his actual needs. By adding a round of communication, we ensure the privacy of the matching relationship between keywords and file identifiers. Thus, our scheme protects the access pattern of DS.

(iii)Search pattern. The trapdoors of our scheme are generated randomly, that is, the trapdoors of the same keyword are different. Thus, our scheme protects the search pattern.

6. Comparison and Analysis

In this section, we compare our scheme with the authentication based searchable schemes (HL17 [5], QC + 20 [20], QC + 21 [21] and PL21 [41]), which are mainly focused on security comparison. Then, we count the number of different operations of other schemes and conduct an empirical performance evaluation using the Relic and GMP library.

6.1. Comparative Analysis of Security. A comparison of the security guarantees provided by these PAEKS schemes is shown in Table 2. Reference [21] introduced the definition of fully CI-security, fullyTI-security. The mDLIN, DBDH and BDHI stand for modified Decision Linear (mDLIN) assumption, Decisional Bilinear Diffie-Hellman (DBDH) assumption and Bilinear Diffie-Hellman Inversion (BDHI) assumption respectively. The BDH and CODH stand for Bilinear DiffieHellman and Computational Oracle Diffie-Hellman. HL17 [5], QC + 20 [20], QC + 21 [21] and PL21 [41] have a common feature that they use the DR's public key in ciphertext generation and the DR's private key in trapdoor generation, which can naturally resist IKGA. A comparison of Table 2 shows that only our scheme has fully CI-security and fully TI-security in PAEKS. Specifically, in our scheme, the ciphertext with the same keyword is encrypted with different keys, and the trapdoor is blinded with random numbers. In their CI-security model, HL17 and PL21 still prohibited adversaries from querying cipher-keyword oracles with challenge keywords. QC + 21 and QC + 20 are fully CI-security. Trapdoors are generated by deterministic algorithms, so they are not fully TI-security. In other words, the trapdoor generation algorithm of their scheme will leak the retrieval habit. We reduce the possibility of access

Table 2: Comparative security.

Schemes	Fully CI-security	Fully TI-security	Integrity	Assumption
HL 17 [5]	X	X	X	mDLLN and DBDH
QC + 20 [20]	√	X	X	BDH
QC + 21 [21]	√	X	X	BDH and CODH
PL21 [41]	X	X	X	BDHI
Ours	√	√	√	DODH

√: Schemes supporting corresponding features are supported. X: The scheme cannot support the corresponding feature.

Table 3: Operations comparison of PAEKS schemes.

Schemes	PAEKS	Trapdoor	Test
HL 17 [5]	$3E + H$	$3E + P + H$	$2P$
QC + 20 [20]	$3E + P + 2H$	$2E + H$	$H + P$
QC + 21 [21]	$3E + P + H$	$2E + H$	P
PL21 [41]	$3E + H$	$3E + P + H$	$2P$
Ours	$3E + 2H$	$E + H$	$2D$

H: denoting a hash-to-point operation. P: denoting a bilinear pairing operation. E: denoting a modular exponentiation. D: denoting a dot product.

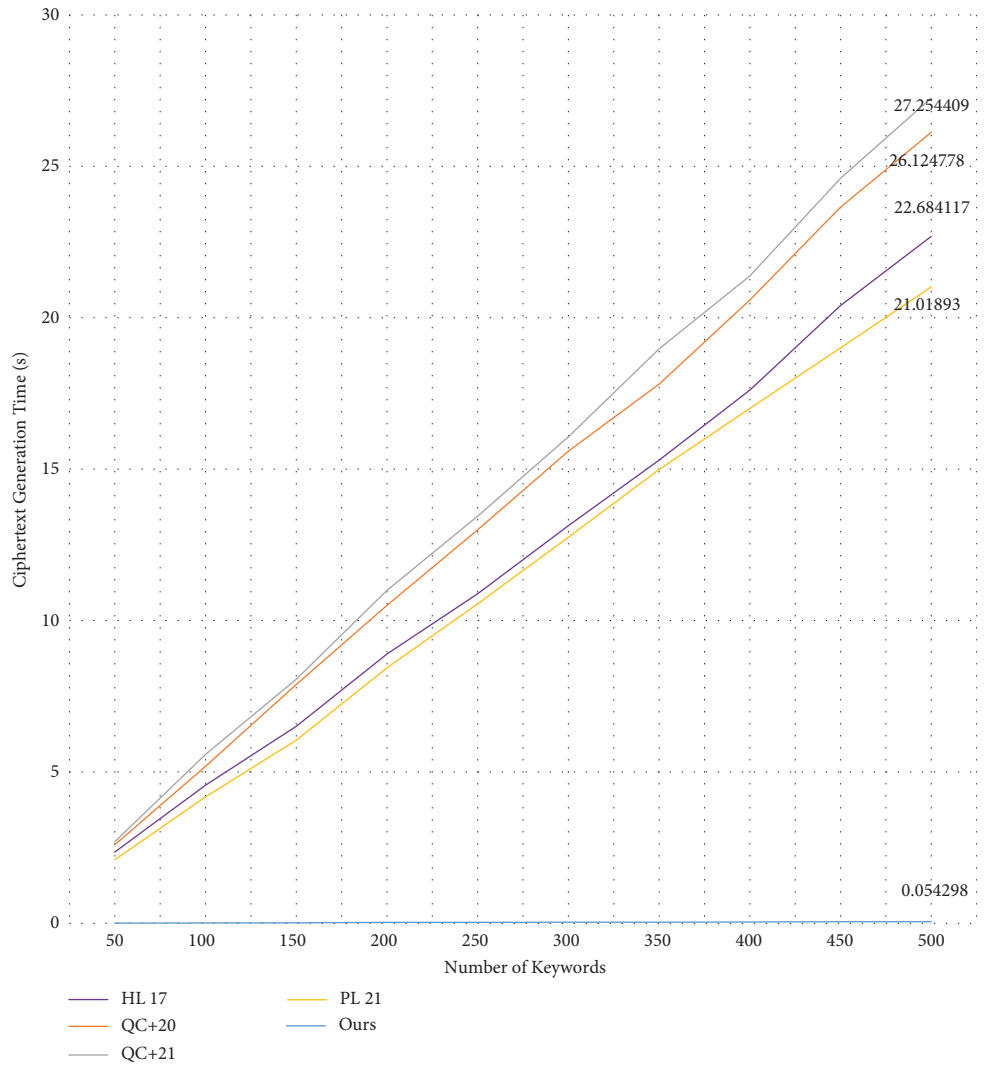

Figure 4: The cost of ciphertext.

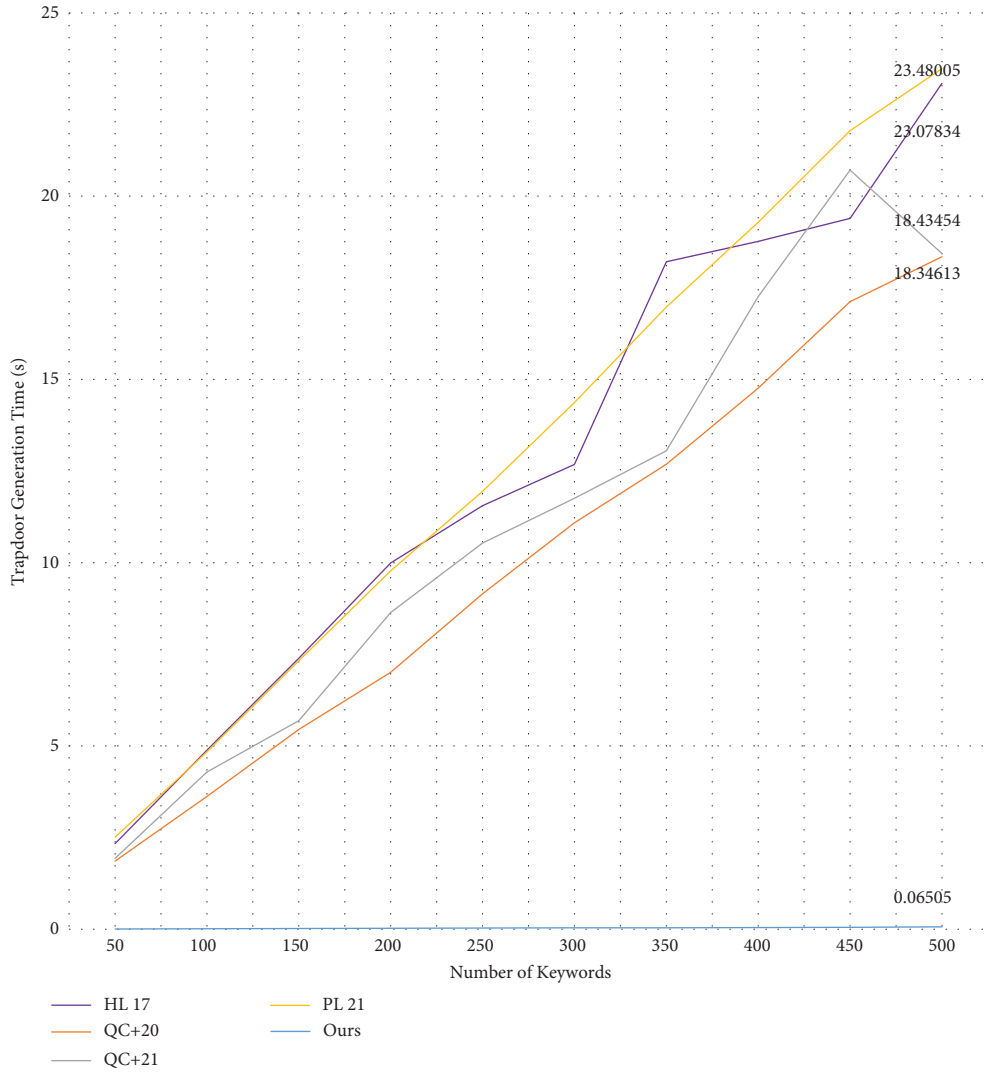

Figure 5: The cost of trapdoor.

pattern leakage by adding a round of communication. Because the server does not know the specific index entries that meet the keyword. In addition, only our scheme also meets the result integrity verification.

6.2. Time Complexity. Table 3 shows the number of operations of each algorithm. E, P, H, F denotes the operation of exponentiation, the pairing operation, hash function, pseudo-random function respectively. In the table, we disregard operations with low costs, such as normal hashing. Considering that Setup and KeyGen of different schemes have no significant difference in computational costs and the common algorithm, we only consider $Encrypt/PAEKS$, $Trapdoor$, and $Test/Search$ algorithms. The schemes of HL17 and PL21 be slightly faster than that of QC + 20 and QC + 21, as they only calculated the hash to point operation in the keyword encryption algorithm. However, our solution is much faster than theirs. Our scheme is the fastest compared with the schemes of HL17, QC + 20, QC + 21 and PL21 in the $Trapdoor$ algorithm, because it does not need pairing. For $Test$ generation algorithm, our scheme requires one time inner product operation. This is almost optimal among these $Test$ schemes.

6.3. Evaluation. By incorporating Relic and GMP, we are able to evaluate the effectiveness of the various schemes (HL17, QC + 20, QC + 21 and PL21). Platforms used in this experiment include Ubuntu 18.04.5 LTS with Intel (R) Xeon (R) CPU E5-2620 v4 @ 2.10 GHz and 16.00 GB of RAM. The pseudo random permutation was computed using the AES algorithm (CBC model, 128 bit key). The hash functions were computed using the SHA-256 algorithm. We choosed the real Encron e-mail Dataset (Version 20150307, about 423 MB) to demonstrate the practical performance of our proposed scheme, which contains the data from about 150 users. We choosed about 2000 keywords whose lengths are not less than 5 characters and the total number of occurrences is higher than 20. We compared our proposed scheme to HL17, QC + 20, QC + 21, PL21, with respect to ciphertext generation, trapdoor generation, and test algorithm.

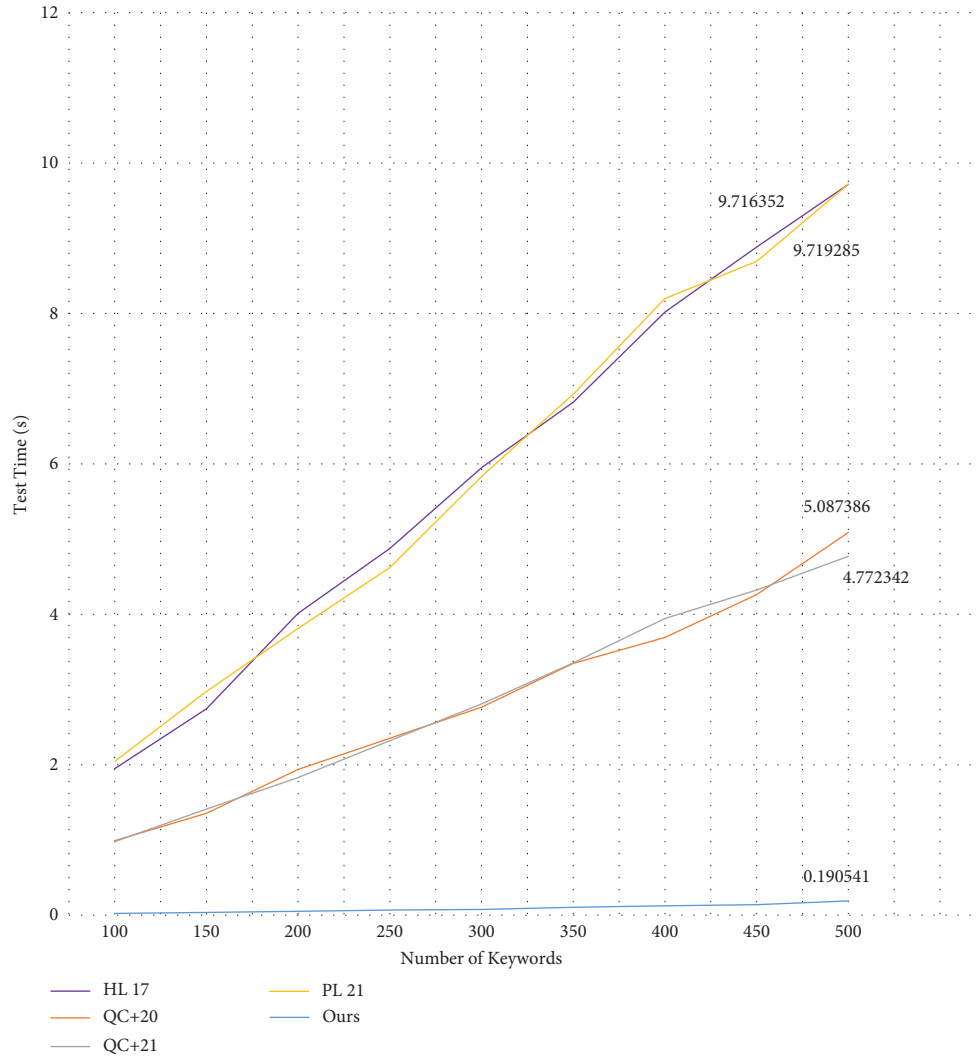

Figure 6: The cost of the test.

Furthermore, this experiment also used random keywords. We have the smallest computation costs of the five schemes based on Figures 4–6. The main reason is that our scheme does not need bilinear pairing operation, which can save a lot of computing overhead. In the ciphertexts generation algorithm, Pan and Li [41] computational overhead is about 420 times that of ours. Our trapdoor generation algorithm has a computation overhead 282 times higher than Qin et al.'s [20]. In the test algorithm, Qin et al.'s [21] computational overhead is about 25 times that of ours.

7. Conclusion

This paper proposes a blockchain-based public key authenticated encryption with keyword search for cloud computing scheme. The scheme can not only resist the IKGA and minimize the leakage of retrieval information, but also realize the functions of multi-keyword retrieval, result integrity verification and so on. At the same time, the security analysis and theoretical analysis of BB-PAEKS are carried out. Then, we evaluate its performance by simulation experiments. The disadvantage is that our scheme uses bloom filter, which will lead to certain false positive events. However, the error rate can be reduced as much as possible by selecting appropriate parameters. Our future work will focus on the design of lightweight multi-user public key searchable encryption schemes.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

The work was supported by the National Key Research and Development Program of China (No. 2021YFA1000600) and

the National Natural Science Foundation of China (Nos. U21A20466, 62172307, 61972294, 61932016).

References

[1] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of the 2000 IEEE Symposium on Security and Privacy. S&P 2000*, pp. 44–55, IEEE, Berkeley, CA, USA, May 2000.

[2] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 506–522, Springer, Heidelberg, Germany, June 2004.

[3] H. S. Rhee, W. Susilo, and H.-J. Kim, "Secure searchable public key encryption scheme against keyword guessing attacks," *IEICE Electronics Express*, vol. 6, no. 5, pp. 237–243, 2009.

[4] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: Public-Key Encryption with Fuzzy Keyword Search: A Provably Secure Scheme under Keyword Guessing Attack provably secure scheme under keyword guessing attack," *IEEE Transactions on Computers*, vol. 62, no. 11, pp. 2266–2277, 2013.

[5] 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.

[6] P. Xu, S. He, W. Wang, W. Susilo, and H. Jin, "Lightweight searchable public-key encryption for cloud-assisted wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3712–3723, 2018.

[7] S. Lai, S. Patranabis, A. Sakzad et al., "Result pattern hiding searchable encryption for conjunctive queries," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 745–762, New York, NY, USA, October 2018.

[8] B. Chen, L. Wu, H. Wang, L. Zhou, and D. He, "A blockchain-based searchable public-key encryption with forward and backward privacy for cloud-assisted vehicular social networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 5813–5825, 2020.

[9] P. Xu, S. Tang, P. Xu, Q. Wu, H. Hu, and W. Susilo, "Practical multi-keyword and boolean search over encrypted e-mail in cloud server," *IEEE Transactions on Services Computing*, vol. 14, no. 6, pp. 1877–1889, 2021.

[10] M. Azraoui, K. Elkhiyaoui, M. Önen, and R. Molva, "Publicly verifiable conjunctive keyword search in outsourced databases," in *Proceedings of the 2015 IEEE Conference on Communications and Network Security (CNS)*, pp. 619–627, IEEE, Florence, Italy, September 2015.

[11] Z. Wan and R. H. Deng, "Vpsearch: achieving verifiability for privacy-preserving multi-keyword search over encrypted cloud data," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 1083–1095, 2018.

[12] 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, HI, USA, April 2018.

[13] H. Li, H. Tian, F. Zhang, and J. He, "Blockchain-based searchable symmetric encryption scheme," *Computers & Electrical Engineering*, vol. 73, pp. 32–45, 2019.

[14] Y. Guo, C. Zhang, and X. Jia, "Verifiable and forward-secure encrypted search using blockchain techniques," in *Proceedings of the ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, Dublin, Ireland, June 2020.

[15] W. Xu, J. Zhang, Y. Yuan, X. Wang, Y. Liu, and M. I. Khalid, "Towards efficient verifiable multi-keyword search over encrypted data based on blockchain," *PeerJ Computer Science*, vol. 8, p. e930, 2022.

[16] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Proceedings of the International Conference on Computational Science and its Applications*, pp. 1249–1259, Springer, Heidelberg, Germany, June 2008.

[17] J. W. Byun, H. S. Rhee, H.-A. Park, and D. H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *Proceedings of the Workshop on Secure Data Management*, pp. 75–83, Springer, Heidelberg, Germany, October 2006.

[18] W. C. Yau, S. H. Heng, and B. M. Goi, "Off-line keyword guessing attacks on recent public key encryption with keyword search schemes," in *Proceedings of the International Conference on Autonomic and Trusted Computing*, pp. 100–105, Springer, Heidelberg, Germany, July 2008.

[19] C. Hu and P. Liu, "An enhanced searchable public key encryption scheme with a designated tester and its extensions," *Journal of Computers*, vol. 7, no. 3, pp. 716–723, 2012.

[20] B. Qin, Y. Chen, Q. Huang, X. Liu, and D. Zheng, "Public-key authenticated encryption with keyword search revisited: Security model and constructions," *Information Sciences*, vol. 516, pp. 515–528, 2020.

[21] B. Qin, H. Cui, X. Zheng, and D. Zheng, "Improved security model for public-key authenticated encryption with keyword search," in *Proceedings of the International Conference on Provable Security*, pp. 19–38, Springer, New York, NY, USA, May 2021.

[22] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Proceedings of the International Conference on Applied Cryptography and Network Security*, pp. 31–45, Springer, Heidelberg, Germany, July 2004.

[23] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proceedings of the Theory of Cryptography Conference*, pp. 535–554, Springer, Heidelberg, Germany, February 2007.

[24] Y. H. Hwang and P. J. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Proceedings of the International Conference on Pairing-Based Cryptography*, pp. 2–22, Springer, Heidelberg, Germany, June 2007.

[25] M. Wen, R. Lu, X. Liang, J. Lei, and X. S. Shen, "Range query over encrypted metering data for financial audit," in *Proceedings of the Querying over Encrypted Data in Smart Grids*, pp. 51–75, Springer, New York, NY, USA, April 2014.

[26] M. A. Abdelraheem, C. Gehrmann, M. Lindström, and C. Nordahl, "Executing boolean queries on an encrypted bitmap index," in *Proceedings of the 2016 ACM on Cloud Computing Security Workshop*, pp. 11–22, New York, NY, USA, October 2016.

[27] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," *Journal of Cryptology*, vol. 26, no. 2, pp. 191–224, 2013.

[28] B. Zhang and F. Zhang, "An efficient public key encryption with conjunctive-subset keywords search," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 262–267, 2011.
[29] C. Song, X. Liu, and Y. Yan, "Efficient public key encryption with field-free conjunctive keywords search," in *Proceedings of the International Conference on Trusted Systems*, pp. 394–406, Springer, Heidelberg, Germany, December 2014.
[30] Y. Zhang, Y. Li, and Y. Wang, "Efficient conjunctive keywords search over encrypted e-mail data in public key setting," *Applied Sciences*, vol. 9, no. 18, p. 3655, 2019.
[31] Y. Zhang, L. You, and Y. Li, "Tree-based public key encryption with conjunctive keyword search," *Security and Communication Networks*, vol. 2021, pp. 1–16, 2021.
[32] R. Cheng, J. Yan, C. Guan, F. Zhang, and K. Ren, "Verifiable searchable symmetric encryption from indistinguishability obfuscation," in *Proceedings of the 10th ACM Symposium on Information, computer and communications security*, pp. 621–626, New York, NY, USA, April 2015.
[33] B. Wang and X. Fan, "Lightweight verification for searchable encryption," in *Proceedings of the 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)*, pp. 932–937, IEEE, New York, NY, USA, August 2018.
[34] S.-F. Sun, R. Steinfeld, S. Lai et al., "Practical non-interactive searchable encryption with forward and backward privacy," in *Proceedings of the 2021 Network and Distributed System Security Symposium. Internet Society*, June 2021.
[35] T. Suga, T. Nishide, and K. Sakurai, "Secure keyword search using bloom filter with specified character positions," in *Proceedings of the International Conference on Provable Security*, pp. 235–252, Springer, Heidelberg, Germany, July 2012.
[36] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.
[37] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Achieving Efficient and Privacy-Preserving Set Containment Search over Encrypted Data," *IEEE Transactions on Services Computing*, vol. 2021, Article ID 3065240, 9 pages, 2021.
[38] Q. Song, Z. Liu, J. Cao, K. Sun, Q. Li, and C. Wang, "Sap-sse: SAP-SSE: Protecting Search Patterns and Access Patterns in Searchable Symmetric Encryptionrotecting search patterns and access patterns in searchable symmetric encryption," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1795–1809, 2021.
[39] E. J. Goh, *Secure Indexes*, Cryptology ePrint Archive, 2003.
[40] https://pages.cs.wisc.edu/%20cao/papers/summary-cache/node8.html.
[41] X. Pan and F. Li, "Public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability," *Journal of Systems Architecture*, vol. 115, Article ID 102075, 2021.