Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 2336685, 15 pages
https://doi.org/10.1155/2022/2336685

Research Article

A Lightweight Authenticated Searchable Encryption without Bilinear Pairing for Cloud Computing

Haorui Du,[1,2] **Jianhua Chen**,[1] **Ming Chen**,[3] **Cong Peng**,[4] **and Debiao He**[5,6]

[1]*School of Mathematics and Statistics, Wuhan University, Wuhan 430072, China*
[2]*Shandong Provincial Key Laboratory of Computer Networks, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China*
[3]*Wuhan Maritime Communication Research Institute, Wuhan, China*
[4]*School of Cyber Science and Engineering, Wuhan 430072, China*
[5]*Shanghai Key Laboratory of Privacy-Preserving Computation, MatrixElements Technologies, Shanghai 201204, China*
[6]*School of Cyber Science and Engineering, Wuhan, China*

Correspondence should be addressed to Jianhua Chen; chenjh_ecc@163.com and Debiao He; hedebiao@163.com

Received 13 July 2022; Revised 27 September 2022; Accepted 6 October 2022; Published 19 October 2022

Academic Editor: Yushu Zhang

Outsourcing data to cloud services is a good solution for users with limited computing resources. Privacy and confidentiality of data is jeopardized when data is transferred and shared in the cloud. The development of searchable cryptography offers the possibility to solve these problems. Symmetric searchable encryption (SSE) is popular among researchers because it is efficient and secure. SSE often requires the data sender and data receiver to use the same key to generate key ciphertext and trapdoor, which will obviously cause the problem of key management. Searchable encryption based on public key can simplify the key management problem. A public key encryption scheme with keyword search (PEKS) allows multiple senders to encrypt keywords under the receiver's public key. It is vulnerable to keyword guessing attacks (KGA) due to the small size of the keywords. The proposal of public key authenticated encryption with keyword search (PAEKS) is mainly to resist inside keyword guessing attacks. The previous security models do not involve the indistinguishability of the same keywords ($w_0 \times \times = w_1$), which brings the user's search pattern easy to leak. The essential reason is that the trapdoor generation algorithm is deterministic. At the same time, most of the existing schemes use bilinear pair design, which greatly reduces the efficiency of the scheme. To address these problems, the paper introduces an improved PAEKS model. We design a lightweight public key authentication encryption scheme based on the Diffie-Hellman protocol. Then, we prove the ciphertext indistinguishability security and trapdoor indistinguishability security of the scheme in the improved security model. Finally, the paper demonstrates its comparable security and computational efficiency by comparing it with previous PAEKS schemes. Meanwhile, we conduct an experimental evaluation based on the cryptographic library. Experimental results show that the computational overhead of our scheme compared with the ciphertext generation algorithm, trapdoor generation algorithm and test algorithm of other schemes Our scheme reduces 274, 158 and 60 times, respectively.

1. Introduction

It has become increasingly important and popular to use cloud storage services due to the rapid development of cloud computing. Based on the characteristics of high efficiency and low cost, cloud computing has attracted great attention from academia and industry. Users are able to upload their own data such as text, audio, and video. It can not only save their own local storage costs but also facilitate data access at any time. In addition, cloud storage services provide the ability to share data. Since cloud servers are not fully trusted, they are likely to be bribed by third parties through other means, which makes the transfer and storage of plaintext data a privacy breach. The data owner is not in direct

management and control of the data. How to ensure the security and privacy of data during data sharing is the first challenge encountered in cloud server storage.

Encrypting sensitive data before sending it to a cloud server is the most natural method. In the case where the ciphertext is saved directly to the cloud server, finding a specific file from among hundreds or thousands of encrypted files becomes challenging. Cloud storage is the most convenient way for users to download and decrypt all their files, and then locate the needed files or data from the plaintext obtained. Considering the high communication overhead between users and cloud storage, and the processing costs to decrypt data from user-owned devices, this solution is clearly unrealistic. It would be ideal if the cloud server would search for ciphertext data based on requirements of each user, then return the matching ciphertext data, and then the user would decrypt it. To solve the above problems, searchable encryptions (SE) are proposed.

Symmetric searchable encryption (SSE) and public key searchable encryption (PEKS) are two types of searchable encryption. In 2004, Boneh et al. [1] were the first to study the search problem of data encrypted by the public key system. Any data sender (DS) can use the public key of the data receiver to encrypt keywords. The data receiver (DR) can generate a trapdoor and send it to the server through a secure channel to search the encrypted data. They also discussed the relationship between PEKS and identity based encryption (IBE) and pointed out that PEKS scheme with semantic security under adaptive keyword selection attack contains IBE scheme with IND-ID-CCA security [2].

It was Byun et al. [3] who observed the keyword space was very small compared to the key space, which discovered that offline keyword attacks are possible and successfully thwarted Boneh's scheme. The adversary could intercept the trapdoor and could perform unlimited trapdoor tests, which allowed keyword guessing attacks to succeed. This enables external attacks to be prevented from being launched by the adversary by protecting the trapdoor from being leaked, e.g. a secure method is achieved by establishing a secure communication channel between the recipient and the server, where only the server can access the trapdoor. Thus, its security can be ensured. Another solution idea is by limiting the testing capability of the adversary. That is, by specifying the PEKS of the tester. However, both methods do not work against malicious servers. How to resist malicious servers is another problem solved by PEKS.

Huang and Li [4] introduced authentication techniques in PEKS, namely Public Key Authentication Encryption with Keyword Search (PAEKS) to resist malicious server attacks. In PAEKS, DS encrypted and authenticated keywords, and trapdoors can only be generated by legitimate recipients, which prevented adversaries from performing brute force tests. It was not until Qin et al. [5] that the PAEKS model was brought into question as [4] was unable to capture a real threat. In this scenario, the outsider choses multiciphertext attacks, i.e. determining if two encrypted files share some keywords. The PAEKS scheme Pan and Li [6] proposed follows scheme [5], achieving multiciphertext indistinguishability (MCI-security) and multitrapdoor indistinguishability (MTI-security) with high computational complexity. A security flaw was discovered soon after Qin et al. [7] spotted the scheme [6]. Their ciphertext indistinguishability security (CI-security) models all fail to be resistant to the fully selected keyword attack.

1.1. Motivation and Contributions. This study is motivated by the algorithm proposed by Qin et al. [7] and their analysis of the security system of PAEKS. In their analysis of keyword guessing attacks, [7] pointed out that most schemes often take into account the indistinguishability of ciphertext, i.e., the semantic security of the selected keywords. In other words, if the adversary does not get the trapdoor that challenges the keywords w_0 and w_1, it only requires the adversary to distinguish between the keyword w_0 encryption or w_1 encryption. In light of the following basic fact, Qin considered that a document usually contained multiple keywords, and the same keyword may appear several times.

Therefore, [7] reconsidered the ciphertext indistinguishability security model of PAEKS, which is multiciphertext indistinguishability. It is worth noting that adversaries can ask for ciphertext for any keyword, including challenging keywords. This means if the model can resist an attack that involves fully selected keywords; it can protect the user's access pattern. However, [7] only considers the security of full ciphertext, not the security of full trapdoor. Trapdoor generation algorithm is a deterministic encryption algorithm in [7]. A malicious adversary can accurately record how often a user searches for different trapdoors, thus providing insight into his search pattern. In that case, the search pattern of users must be protected. In addition, most of the existing PAEKS schemes are constructed based on bilinear mapping. The result is high computing overhead for the client side and poor practicability. We need to design a lightweight PAEKS scheme for lightweight computing devices in order to better promote the application.

(i) Firstly, we discuss different types of adversary capabilities, analyze the security of the PAEKS scheme, and describe the security model of the hidden matching relationship

(ii) Secondly, we design a PAEKS scheme without bilinear pairing. Our scheme satisfies both trapdoor indistinguishability and ciphertext indistinguishability before the cloud server runs the test algorithm. In other words, our base scheme protects the user's search pattern and access pattern against adversaries *Type-II*

(iii) Finally, we provide an extended keyword approach that hides the matching relationship between keyword ciphertexts and trapdoors after the cloud service executes the test algorithm against adversaries Type-III. We analyze the security and efficiency comparison

1.2. Organization. In section 2, contains a review of related works. Preliminary cryptographic notations and definitions are presented in section 3. We analyze previous PAEKS schemes in section 4. We present the PAEKS scheme and a

searchable encryption scheme derived from it in section 5. In section 6, we present the security proof of the scheme. In section 7, we present our time consumption comparison with others. Lastly, we present our conclusions in section 8.

2. Related Works

The KGA is a serious concern for public key searchable encryption. It is shown that one of the main reasons the keyword will be attacked by an adversary under KGA if the keyword space is in a polynomial size. In the literature, there have been some attempts to implement IKGA security. Xu et al. [8] proposed Public Key Encryption for Fuzzy Keyword Search (PEFKS), where each keyword corresponds to an exact keyword search trapdoor and a fuzzy keyword search trapdoor. The keyword space of PEFKS was very small, but malicious parties were also unable to learn accurate keywords. Later, Chen et al. [9] pointed out that in [8] the server cannot accurately guess the keyword. As long as it knows which small set the basic keyword belongs to, it cannot protect the keyword privacy very well. [9] analyzed another main reason for keyword guessing attacks.

The vulnerability is caused by the fact that anyone with knowledge of an external receiver's public key can generate the PEKS ciphertext of any keyword. A malicious server may select a guess keyword when presented with a trapdoor, and then use it to generate ciphertext using PEKS. Then the server can guess whether the keyword hidden under the trapdoor is the right keyword until the correct keyword is found. As long as the two servers do not collide, [9] proposed a method for preventing attacks using dual-server public key encryption with keyword search. Sun et al. [10] took into account the fact that the server was malicious, and they explored whether a PEKS scheme against inside KGA can be built based on different public key cryptosystems, such as PKI based, identity-based, or certificateless cryptosystem. In order to improve the scheme's efficiency, the researchers proposed a construction of PAEKS based on word-independent smooth projective hash functions (SPHFs) and PEKS [11–19].

Combining attribute-based encryption (ABE) and searchable encryption (SE), a fine-grained search scheme is obtained. This combination leads to the development of SE schemes suitable for multiuser scenarios. The ciphertext is linked to the access policy in ciphertext-policy of ABE (CP-ABE), and the user's key is associated with a set of attributes. Therefore, searchable scheme based on attribute encryption can resist IKGA, which mainly depends on the security of attribute encryption. Zheng et al. [20] developed the first fine-grained searchable encryption scheme by using both the variants of CP-ABE. In order to reduce the amount of calculation, efforts had been made in some schemes [21–23]. Schemes [24–28] tried to reduce the storage and calculation cost of keyword ciphertext, key and search token. In this paper, we mainly improve the computational efficiency and retrieval efficiency of key ciphertext and trapdoor in the smallest basic unit of searchable encryption structure, that is, only two parties share data. If we study the most basic units clearly, it will promote multiuser scenario.

3. Preliminaries

In this section, we introduce some basic concepts of cryptography, the definition of schemes, security models, threat models, and design goals.

3.1. Notations. The notations are described in Table 1.

3.2. Pattern Leakage. The set Q is a q-query set consisting of pairs (i, w), with i representing the timestamp of a query and w representing a keyword. The access pattern [29] is as follows:

(i) Access pattern. Documents containing a given keyword appear in the search results. The access pattern is defined as $\mathrm{ap}(\mathrm{w}) = \{\mathrm{ID}(\mathrm{w})\}$ for quering keyword w, where $\mathrm{ID}(w)$ denotes the IDs of documents containing w

(ii) Search pattern. It shows the links between particular keywords and search queries. Its search pattern is defined as $sp(w) = \{i|(i, w) \in Q\}$ for quering keyword w,.

3.3. 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 1

(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

3.4. The Adversary Attack Capability. Here we divide the adversary capabilities into 3 Types.

(i) *Type-I*: The adversary collects some the ciphertext and trapdoor of keywords, and they do not know the correspondence between ciphertext and plaintext

(ii) *Type-II*: The adversary collects all the ciphertext and part of the trapdoor of the keyword submitted by the data receiver, and only knows part of the matching relationship between the submitted trapdoor and the ciphertext. It is worth noting that the adversary does not know the matching relationship between the trapdoor being challenged and the ciphertext

(iii) *Type-III*: The adversary stores all the ciphertexts of keywords and all the trapdoors submitted by the data receiver, and knows the matching relationship

Table 1: BRIEF SYNTHESIS OF NOTATIONS.

Symbol	Description
G_1	The cyclic group
g	The generator of G_1
H_1, H_2	Hash function
PK, SK	Public and private keys of each participants
W	The keywords space
$ID(w)$	The IDs of documents containing w
$ap(w)$	Set of file IDs containing the keyword w
Q	Set of q query
(i, w)	i denotes the timestamp of a-query

between all the trapdoors and ciphertexts, including the trapdoors that challenge keywords

3.5. The Computational Diffie-Hellman Assumption (CDH). Computational Diffie-Hellman assumption (CDH): let G be a cyclic group, which has a prime order q, and Pi be a generator of G1. Given the tuple (*Pi*, aPi, bPi) *E* Gi, R where *a*, *b*-*Z*, there is no probabilistic polynomial time (PPT) algorithm to get abP, *E* Gj. We define the advantage.

Let G is a cyclic group, which has a prime order p, and g be generator of G. We get two elements g^a and g^b from G. The g^{ab} needs to be calculated. CDH is hard to solve in the G

3.6. The Hash Diffie-Hellman Assumption (HDH). G is a cyclic group. Its order is p. The generator is g. Given $H : \{0,1\}^* \longrightarrow \{0,1\}^{hLen}$ be a security cryptographic hash function. The advantage of $\mathscr{A}$ is negligible for a polynomial adversary $\mathscr{A}$. The advantage is $Adv_{\mathscr{A}}^{HDH}(\lambda) = |\Pr[\mathscr{A}(g, g^a, g^b, R) = 1] - \Pr[\mathscr{A}[(g, g^a, g^b, H(g^{ab})) = 1]|$, where $a, b \in Z_p$, $R \in [0,1]^{hLen}$

3.7. The Definition of PAEKS. A PAEKS scheme includes five polynomial algorithms, which are Setup, KeyGen, PAEKS, Trapdoor, and Test. The specific construction is described as follows:

(i) $pp \longleftarrow \text{Setup}(\lambda)$: it is an algorithm for generating global parameters. Input a security parameter λ, and outputs a global public parameter pp

(ii) $(PK, SK) \longleftarrow \text{KeyGen}(pp)$: providing participants with a public/private key pair is its responsibility

(a) $(PK_S, SK_S) \longleftarrow \text{KeyGen}(pp)$: it is sender's key generation algorithm. Input pp, and generates public and private key (PK_S, SK_S)

(b) $(PK_R, SK_R) \longleftarrow \text{KeyGen}(pp)$: the receiver generates keys with this algorithm. Input pp, and generates public and private key (PK_R, SK_R)

(c) $(PK_{CS}, SK_{CS}) \longleftarrow \text{KeyGen}(pp)$: this is the algorithm used by the cloud server to generate keys. Input pp, and generates public and private key (PK_{CS}, SK_{CS}).

(iii) $C_w \longleftarrow PAEKS(SK_S, PK_R, PK_S, w)$: the DS performs the operation. Input SK_S, PK_R, PK_S, w, and returns corresponding ciphertext C_w

(iv) $T_{w'} \longleftarrow \text{Trapdoor}(SK_R, PK_R, PK_S, PK_{CS}, w')$: The procedure is performed by DR. Input $SK_R, PK_R, PK_S, PK_{CS}, w'$, and returns corresponding trapdoor $T_{w'}$

(v) $0/1 \longleftarrow \text{Test}(PK_R, PK_S, SK_{CS}, C_w, T_w)$: the cloud server manages the process. Input $PK_R, PK_S, SK_{CS}, C_w, T_{w'}$. If $w' \in \mathscr{W}$, outputs 1; else, outputs 0

Correctness: given C_w and $T_{w'}$, and we formulate consistency as follows:

If $w = w'$, $\Pr[\text{Test}(SK_{CS}, PK_R, PK_S, C_w, T_{w'}) = 1] = 1$

3.8. System Model. The system includes the following parties: Cloud Server (CS), Data Sender (DS), and Data Receivers (DR) as depicted in Figure 2.

(i) Data Sender (DS): The DS produces his own public key and private key upon inputting the security parameter. Moreover, DS extracts keywords from files, and generates index matrix, and computes the searchable keywords ciphertexts. Finally, the DS stores ciphertexts on the CS

(ii) Data Receiver (DR): Request Users utilize targeted keywords to generate search trapdoors and send them to the CS

(iii) Cloud sever (CS): The Cloud Server has almost unlimited storage and computing power in the PAEKS system. The CS is in charge of storing searchable ciphertexts received from DS. Then, the CS addresses search queries and returns corresponding searching results ciphertexts to DR

3.9. Threat Model. In this paper, it supposes that DS honestly follow the PAEKS algorithm to produce searchable ciphertexts for DO. DR honestly follows the Trapdoor algorithm to produce trapdoor. The CS is supposed to be honest and curious, who will honestly perform Test algorithm, and is interested in query results and frequency information of ciphertext.

3.10. The Security Models of PAEKS. As part of PAEKS semantic security model, there is fully cipher-keyword indistinguishability (Fully CI-security) and fully trapdoor indistinguishability (Fully TI-security). Our fully CI-security and fully TI-security model is the same as [4, 5, 7] in settings. In addition, we propose that a PAEKS scheme for adversary Type-II, which should also meet the security of hiding its matching relationship. ie the matching relationship security model of Hidden Ciphertext and Trapdoor (HMR-security).

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) Extract keywords from files

	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) Index matrix

Figure 1: Inverted index.

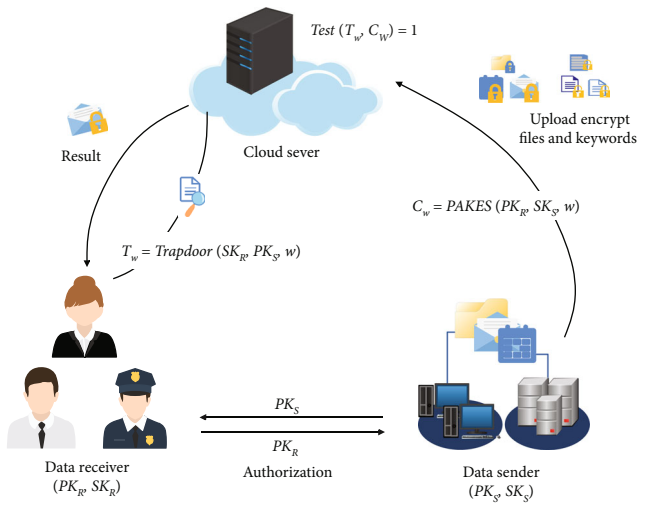

Figure 2: The system model of PAEKS.

3.11. Fully CI-Security Model

(i) Given adversary $\mathscr{A}$ and the security parameter λ

(ii) *Initialization.* The challenger $\mathscr{C}$ firstly runs the algorithm $Setup(\lambda)$ to generate the system parameter pp. $\mathscr{C}$ runs KeyGen(pp) to generate DS's key pairs (PK_S, SK_S), DR's key pairs (PK_R, SK_R) and CS's key pairs (PK_{CS}, PK_{CS}), respectively. $\mathscr{C}$ gives pp, PK_S, PK_R and PK_{CS} to $\mathscr{A}$

(iii) *Phase 1.* $\mathscr{A}$ asks for polynomial oracle $\mathcal{O}_{\mathscr{C}}$ and $\mathcal{O}_{\mathscr{T}}$

(iv) *Challenge.* After Phase 1, outputs w_0^* and w_1^*. $\mathscr{C}$ picks a coin $b \in [0, 1]$. Sends $C_{w_b^*}$ to $\mathscr{A}$

(v) *Phase 2.* $\mathscr{A}$ can remain to query the oracles as *Phase*1

(vi) *Guess.* Finally, $\mathscr{A}$ returns a bit $b' \in [0, 1]$ as the guess of b

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

(i)

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

Definition 1 (Fully CI-Security). A PAEKS scheme satisfies cipher-keyword indistinguishable against chosen keywords attacks if the advantage $Adv_{\mathscr{A}}^{IND-CI}(\lambda)$ of succeeding in the above game is negligible for any polynomial $\mathscr{A}$.

3.12. Fully TI-Security Model

(i) Given adversary $\mathscr{A}$ and security parameter λ

(ii) *Initialization.* $\mathscr{C}$ firstly performs Setup(λ) to generate parameter pp. Then, $\mathscr{C}$ runs KeyGen(pp) to generate DS's key pairs (PK_S, SK_S), data receiver's key pairs (PK_R, SK_R) and cloud sever's key pairs (PK_{CS}, PK_{CS}), respectively. $\mathscr{C}$ gives pp and PK_S, PK_R and PK_{CS} to $\mathscr{A}$

(iii) *Phase 1.* $\mathscr{A}$ adaptively asks $\mathscr{C}$ queries on $\mathcal{O}_{\mathscr{C}}$ and $\mathcal{O}_{\mathscr{T}}$, and gets ciphertext and trapdoor of query keywords

(iv) *Challenge.* After Phase 1, it outputs w_0^* and w_1^* with the restriction that w_0^* and w_1^* never be queried on oracle $\mathcal{O}_{\mathscr{C}}$ and $\mathcal{O}_{\mathscr{T}}$ by $\mathscr{A}$ in Phase 1. Then, $\mathscr{C}$ picks a coin $b \in [0, 1]$ and sends the trapdoor keyword $T_{w_b^*}$ to $\mathscr{A}$

(v) *Phase 2.* As Phase 1, $\mathscr{A}$ can continue to query the oracles

(vi) *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

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

Definition 2 (Fully TI-Security). A PAEKS scheme satisfies trapdoor indistinguishability against chosen keywords attacks if for any PPT adversary $\mathscr{A}$, the advantage $Adv_{\mathscr{A}}^{IND-TI}(\lambda)$ of succeeding in the above game is negligible.

3.13. HMR-Security Model. Let $\mathscr{A}$ be an adversary and λ be the security parameter.

(i) *Initialization.* $\mathscr{C}$ first runs the algorithm $Setup(\lambda)$ to generate the system parameter pp. Then, it runs *Ke*

$yGen(pp)$ to generate DS's key pairs (PK_S, SK_S), DR's key pairs (PK_R, SK_R) and CS's key pairs (PK_{CS}, PK_{CS}), respectively. $\mathscr{C}$ gives parameters pp, PK_S, PK_R and PK_{CS} to $\mathscr{A}$

(ii) *Phase 1.* In this approach, $\mathscr{A}$ asks PPT queries on the cipher-keyword, trapdoor, and test oracles, and then receives cipher and trapdoor responses to the query keywords and test results, as follows:

(a) $\mathscr{C}$ runs $PAKES(PK_R, SK_S, w)$ for any keyword w, and sends C_w to $\mathscr{A}$

(b) $\mathscr{C}$ runs $\text{Trapdoor}(PK_S, SK_R, PK_{CS}, w')$ for any keyword w', and sends $T_{w'}$ to $\mathscr{A}$

(c) For any the cipher C_w and trapdoor $T_{w'}$ of keyword w and w', $\mathscr{C}$ runs $Test(SK_{CS}, PK_R, C_w, T_{w'})$, and if $w' = w$, outputs 1; else, outputs 0

(iii) *Challenge.* After Phase 1, it outputs two challenge keywords w_0^* and w_1^*. Then, $\mathscr{C}$ picks a coin $b \in [0, 1]$ and $d \in [0, 1]$. Sends $C_{w_b^*}$ and $T_{w_d^*}$ to $\mathscr{A}$

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

(v) *Guess.* Finally, $\mathscr{A}$ returns two bit $b', d' \in [0, 1]$ as the guess of b, d

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

$$Adv_{\mathscr{A}}^{HMR}(\lambda) = \left| \Pr\left[\left(b' = b\right) \wedge \left(d' = d\right) \right] - \frac{1}{2} \right|. \quad (3)$$

Definition 3 (HMR-Security). A PAEKS scheme satisfies matching relationship security model of Hidden Ciphertext and Trapdoor (HMR-security) if the advantage $Adv_{\mathscr{A}}^{HMR}(\lambda)$ succeeding in the above game is negligible for any polynomial $\mathscr{A}$.

3.14. Design Goals. Our goal is to design an efficient PAEKS scheme, which can resist external keyword guessing attack and internal keyword guessing attack, and prevent the leakage of search and access pattern for adversary Type-II.

4. Previous PAEKS Schemes

In this section, we firstly analyze the relationship between keyword ciphertext indistinguishability and access pattern and the relationship between trapdoor indistinguishability and search pattern. Secondly, it analyzes whether the previous PAEKS really meet the indistinguishability of ciphertext and trapdoors for external enemies and malicious servers. Finally, we give suggestions and methods for adversary with different abilities.

4.1. Why Should we Define the Indistinguishability of Cipher and Trapdoor of Keywords?

(i) *TI-Security and Search Pattern.* It is well known that the disclosure of search pattern will reflect users' search habits. If the PAEKS scheme does not meet the indistinguishability of trapdoors, trapdoors for the same keyword are the same. This allows the adversary to classify the keywords through the trapdoor, thus revealing the user's search pattern. It can be seen that if the trapdoor does not meet the indistinguishable properties, the user's search pattern will be divulged. The adversary can accurately guess the keywords by combining some a priori knowledge

(ii) *CI-Security and Access Pattern.* Access pattern disclosure will reflect the data file information matching keywords, such as the frequency of different keywords and documents. If the PAEKS scheme does not meet the indistinguishability of keyword ciphertext, the ciphertext of the same keyword is the same. Therefore, the adversary can guess the keywords through the matched file information and prior knowledge

Based on the above analysis, we believe that PAEKS should meet at least the above two points to be considered secure if we want to achieve the purpose of search and access pattern without being compromised.

4.2. Security Analysis of Previous PAEKS Schemes. We discuss the close relationship between indistinguishable ciphertext and indistinguishable trapdoor from the following three aspects for different types of adversaries.

(i) *CI-Security and Not TI-Security.* If the ciphertext of the keyword in PAEKS meets the indistinguishability, the trapdoor is discontent the indistinguishability, as shown in Figure 3(a). Three different ciphertexts of keyword w_1 are represented by ellipsoids of three different colors. We can see that the ciphertext C_1, C_2, and C_3 are indistinguishable. The trapdoor does not satisfy indistinguishability. If the adversary obtains trapdoor T_1, after testing the algorithm, he can understand that C_1, C_2, and C_3 are the ciphertext of the same keyword, which the indistinguishability of the ciphertext loses its function. The access pattern will be compromised. Therefore, this Test algorithm plays a key role in above process. It is worth noting that this also provides an idea for us to solve the problem of access pattern disclosure

(ii) *Not CI-Security and TI-Security.* As shown in Figure 3(b), three boxes with different colors indicate that the three trapdoors T_1, T_2, T_3, for the

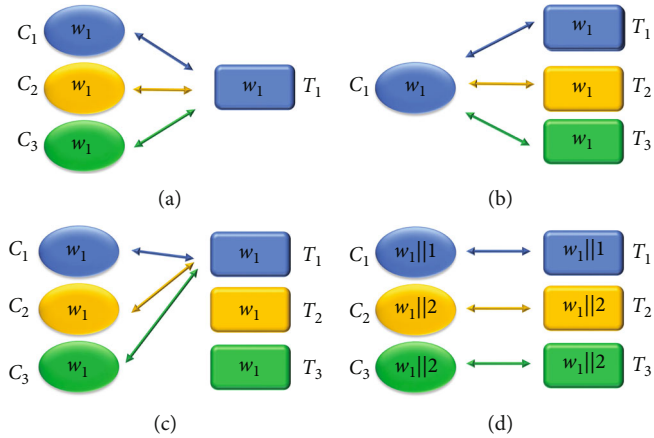

Figure 3: Matching relationship.

keyword w_1 are different. So, it meets the indistinguishability of trapdoor. C_1 does not satisfy the indistinguishability of ciphertext. If the adversary obtains ciphertext C_1, after testing the algorithm, he can understand that T_1, T_2, T_3 is the trapdoor of the same keyword, which makes the user's search pattern leak

(iii) *CI-Security and TI-Security.* As shown in Figure 3(c), combined with Figures 3(a) and 3(b) analysis, the adversary can also get relevant information. To be specific, if the adversary gets T_1, the matching ciphertext is $\{C_1, C_2, C_3\}$. Similarly, for T_2, T_3, the matching ciphertext is $\{C_1, C_2, C_3\}$. The adversary can know that T_1, T_2, and T_3 are trapdoors with the same keyword. The solution is the same as above

Based on the study above, for adversary Type-I and Type-II, we can solve the problem of access pattern and search pattern disclosure by specifying an honest server. However, if it is the adversary Type-III, the existing schemes do not protect their privacy from the perspective of access pattern and search pattern [4–7]. In other words, none of them can resist leakage abuse attacks [30]. One possible solution is to store data distributed on servers so that statistics are not completely compromised if there is an honest server. Next, for the adversary Type-III, we will solve this problem from another angle.

5. Construction

In this section, we first introduce a PAEKS scheme without bilinear pairs. Secondly, based on this scheme, we padding the keywords to achieve one cipher at a time from the ciphertext perspective. In addition, appropriate processing of the documents containing the key makes it possible to protect the access patterns.

5.1. Basic Construction. Our algorithm is consisted of five PPT algorithms, namely, *Setup*, KeyGen PAEKS, Trapdoor, and Test. The formal constructions are as follows:

(i) Setup(λ): G_1 is cyclic groups. The order of G is prime p. Select two hash functions $H_1 : \{0,1\}^* \longrightarrow Z_p^*$ and $H_2 : G_1 \longrightarrow \{0,1\}^*$. Choose a random generator g of G_1. Output the public parameters $pp = \{G_1, p, g, H_1, H_2\}$.

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

(iii) The DS chooses a random element $y \in Z_p^*$, and sets $PK_S = g^y \bmod p$, $SK_S = y$

(iv) The DR selects randomly $x \in Z_p^*$, and sets $PK_R = g^x \bmod p$, $SK_R = x$

(v) The CS selects randomly $z \in Z_p^*$, and sets be $PK_{CS} = g^z \bmod p$, $SK_{CS} = z$

(vi) PAEKS(SK_S, PK_R, PK_S, w): Choose a random $r \in Z_p^*$, compute $C_1 = g^{xr} \bmod p$, $C_2 = H_2(g^{xr} \cdot g^t \bmod p)$, where $t = g^{H_1(w||l||PK_S||PK_R)}$, $l = PK_R^y = g^{xy}$. The ciphertext is $C_w = (C_1, C_2)$.

(vii) Trapdoor($SK_R, PK_R, PK_S, PK_{CS}, w$): Compute $l = PK_S^x = g^{yx}$, for keyword w, $t = g^{H_1(w||l||PK_S||PK_R)}$. Then, choose a random $k(1 \le k \le p-1)$, gcd $(k, p-1) = 1$, compute $u = g^k \bmod p$, $T_1 = g^{kz}$, $T_2 = (t - x \cdot u)k^{-1} \bmod (p-1)$. The trapdoor is $T_w = (T_1, T_2)$.

(viii) Test($PK_R, PK_S, SK_{CS}, C_w, T_w$): Compute $u' = T_1^{1/z} \bmod p$, $H_2(PK_R^{u'} \cdot u'^{T_2} \cdot C_1 \bmod p) \stackrel{?}{=} C_2$. If so, it outputs 1; otherwise, it outputs 0

5.2. Derived Construction. In basic construction, we can understand that if the PAEKS scheme itself can not hide the matching relationship between ciphertext and trapdoor for the adversary Type-III, which it always gets the relationship between ciphertext and trapdoor through Test algorithm. We consider that if we only achieve the hidden matching relationship in the simple PAEKS scheme, the solution is to fill in the keyword or attach some state

information, so that the keyword can achieve the security of one time pad.

To be specific, in order to prevent the malicious server from classifying ciphertext and trapdoor through the tested algorithm this leads to the disclosure of access and search pattern. We have to expand the external form of keywords to achieve that the same keyword has different ciphertext and trapdoor forms Figure 3(d). Meanwhile, if we want to essentially solve the problem of search pattern and access pattern leakage, we have to operate on the original index matrix. The purpose of this operation is to achieve the uniqueness of keywords from the external form.

According to the keywords of the diagram and the frequency information of the document and you can also make a choice according to your actual situation on how to construct different expressions of a keyword, here we give an example Figure 1. Provide the uniqueness of keywords as follows: Solve the uniqueness of keywords as follows:

(i) *Maximum Number of Searches I.* Before uploading the ciphertext, the DS stipulates that the same keyword can be searched for several times at most without disclosing the search pattern, and broadcast on the public channel. If users search more than this number of times, there is a risk of revealing the search pattern

(ii) *Divided into Several Parts J.* To solve the problem that different keywords return different numbers of documents, we suggest dividing the returned documents so that the number of documents returned each time is not much different. Here, the data DS needs to determine the maximum number of parts of a document and broadcast it to the Data Receiver

(iii) *Keyword Form.* A keyword w is set to $w\|i\|j$, where i stands for the i-th search and j represents the j-th part of the search. For example Figure 2, we make the following settings. Note the following facts that is $ap(w_1) = \{f_1, f_2, f_3, f_4\}$, $ap(w_2) = \{f_1, f_3, f_4\}$, $ap(w_3) = \{f_2, f_4, f_5\}$, $ap(w_4) = \{f_1, f_4, f_5\}$, $ap(w_5) = \{f_1, f_4\}$, let $I = 3, J = 2$. Thus, the keyword w_1, w_2, w_3, w_4, w_5 extension to Table 2

Next, we apply the scheme proposed in Section VII to encrypt the filled keywords to achieve our purpose, which is to solve the problem that malicious servers can resist keyword guessing, search, and access pattern disclosure.

Remark. Once the data receiver searches for a keyword more than the predetermined number of times, it is possible to disclose the search behavior and access pattern. Another flaw is that the storage cost has doubled. Fortunately, the peace point of security and efficiency is left to the user to decide.

Table 2: Keyword Extension.

Keyword	Tag(time I)	Tag(part J)
	$w_1\|\|1\|\|1$	$w_1\|\|1\|\|2$
w_1	$w_1\|\|2\|\|1$	$w_1\|\|2\|\|2$
	$w_1\|\|3\|\|1$	$w_1\|\|3\|\|2$
	$w_2\|\|1\|\|1$	$w_2\|\|1\|\|2$
w_2	$w_2\|\|2\|\|1$	$w_2\|\|2\|\|2$
	$w_2\|\|3\|\|1$	$w_2\|\|3\|\|2$
w_3	$w_3\|\|1\|\|1$	$w_3\|\|1\|\|2$
	$w_3\|\|2\|\|1$	$w_3\|\|2\|\|2$
	$w_3\|\|3\|\|1$	$w_3\|\|3\|\|2$
	$w_4\|\|1\|\|1$	$w_4\|\|1\|\|2$
w_4	$w_4\|\|2\|\|1$	$w_4\|\|2\|\|2$
	$w_4\|\|3\|\|1$	$w_4\|\|3\|\|2$
	$w_5\|\|1\|\|1$	
w_5	$w_5\|\|2\|\|1$	
	$w_5\|\|3\|\|1$	

6. Security Proof

In this section, we demonstrate that our basic scheme matches the design goals in the sense that it is capable of providing soundness and confidentiality. It can resist IKGA and reduce the possibility of access pattern and search pattern disclosure for adversary Type-II

6.1. Correctness. The ciphertext and trapdoor of keyword w and w' are $C_w = (C_1, C_2)$ and $T_{w'}$, respectively. That is $C_1 = g^{xr} \bmod p$, $C_2 = H_2(g^{xr} \cdot g^t \bmod p)$. let $T_{w'}$ be the trapdoor of keyword w' generated by the receiver. $u = g^k \bmod p$, $T_1 = g^{kz}$, $T_2 = (t' - x \cdot u)k^{-1} \bmod (p-1)$, where $l = PK_S^x = g^{yx}$, $t' = g^{H_1(w'\|l\|PK_S\|PK_R)}$. It follows that

(i) $u = T_1^{1/z} = (g^{kz})^{1/z} = g^k$

(ii) $H_2(PK_R^u \cdot u^{T_2} \cdot C_1 \bmod p) = H_2(g^{xu} \cdot (g^k)^{(t'-x\cdot u)k^{-1} \bmod (p-1)} \cdot g^{xr} \bmod p) = H_2(g^{xu} \cdot g^{(t'-xu) \bmod (p-1)} \cdot g^{xr} \bmod p) = H_2(g^{t'} \cdot g^{xr} \bmod p)$

If $w = w'$, $t = t'$, then $H_2(g^{t'} \cdot g^{xr} \bmod p) = C_2$ with probability 1; otherwise, $t \neq t', H_2(g^{t'} \cdot g^{xr} \bmod p \neq C_2$ with overwhelmig probability.

6.2. Confidentiality. The following theorems illustrate that our scheme is IND-CKA and IND-KGA security.

(i) *Setup*(λ): $\mathscr{C}$ runs Setup(λ) and KeyGen(pp) to generate the public parameter pp and public/private key pair of participants $(PK_S, SK_S) = (g^y \bmod p, y)$ of Data Sender, $(PK_R, SK_R) = (g^x \bmod p, x)$ of DR, of $(PK_{CS}, SK_{CS}) = (g^z \bmod p, z$ of CS. Then, $\mathscr{C}$ sends $(pp, PK_S, PK_R, PK_{CS})$ to $\mathscr{A}$. Based on our

assumptions, we assume that hash function H_1 and H_2 are secure and resistant to collisions

(ii) *Phase* 1: $\mathscr{A}$ sends queries to $\mathcal{O}_{\mathscr{C}}$ and $\mathcal{O}_{\mathscr{T}}$, and $\mathscr{C}$ is simulated as follows:

(a) $\mathcal{O}_{\mathscr{C}}$: choose a random $r \in Z_p^*$, compute $C_1 = g^{xr} \bmod p$, $C_2 = H_2(g^{xr} \cdot g^t \bmod p)$, where $t = g^{H_1(w\|l\|PK_S\|PK_R)}$, $l = PK_R^y = g^{xy}$. The ciphertext is $C_w = (C_1, C_2)$.

(b) $\mathcal{O}_{\mathscr{T}}$: compute $l = PK_S^x = g^{yx}$, for keyword w, $t = g^{H_1(w\|l\|PK_S\|PK_R)}$. Then, choose a random $k(1 \le k \le p-1)$, $\gcd(k, p-1) = 1$, compute $u = g^k \bmod p$, $T_1 = g^{kz}$, $T_2 = (t - x \cdot u)k^{-1} \bmod (p-1)$. The trapdoor is $T_w = (T_1, T_2)$

(iii) *Challenge*: $\mathscr{A}$ selects (w_0^*, w_1^*). Sends them to $\mathscr{C}$. $\mathscr{C}$ picks a random bit $b \in \{0, 1\}$. Encrypts w_b^* as follows:

(a) Choose a random $r^* \in Z_p^*$, compute $C_1^* = g^{xr^*} \bmod p$, $C_2 = H_2(g^{xr^*} \cdot g^{t^*} \bmod p)$, where $t^* = g^{H_1(w_b^*\|l\|PK_S\|PK_R)}$, $l = PK_R^y = g^{xy}$. The ciphertext is $C_{w_b^*}^* = (C_1^*, C_2^*)$

(iv) *Phase* 2: Oracles continue to be consulted as *Phase*1

(v) *Guess*: $\mathscr{C}$ outputs a guess $b' \in \{0, 1\}$. If $b' = b$ then, $\mathscr{C}$ wins the game. $\mathscr{A}$ has the advantage that is

$$Adv_{\mathscr{A}}^{\mathbf{Game_0}}(\lambda) = Adv_{\mathscr{A}}^{\mathbf{KGA}}(\lambda). \tag{4}$$

Theorem 5. *Our scheme PAEKS implements IND-CKA security if the HDH assumption holds.*

Lemma 6. *The advantage Adv_A^{CKA} is negligible for any polynomial 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. □

Game_1. Let Game_1 be the same game as Game_0. Assuming that Game_1 is the same game as Game_0, except that $\mathscr{C}$ chooses $R \in G$ instead of computing $t^* = g^{H_1(w_b^*\|l\|PK_S\|PK_R)}$. $\mathscr{C}$ sends $C_{w_b^*}^* = (C_1^*, C_2^*) = (g^{xr^*} \bmod p, H_2(g^{xr^*} \cdot g^R \bmod p)$. We have $|Adv_{\mathscr{A}}^{\text{Game}_0} - Adv_{\mathscr{A}}^{\text{Game}_1}| < Adv_A^{HDH}(\lambda)$, where $Adv_A^{HDH}(\lambda)$ is negligible if the HDH assumption holds.

Game_2. Let Game_2 be the same game as Game_1, except that $\mathscr{C}$ random selects $C_1, C_2 \in G$ instead of $C_{w_b^*}^* = (C_1^*, C_2^*) = (g^{xr^*} \bmod p, H_2(g^{xr^*} \cdot g^R \bmod p)$. Given that r and R are random values, the $C_{w_b^*}$ of $Game_1$ and $Game_2$ are of the same distribution as shown by $\mathscr{C}'$ perspective

We have

$$Adv_{\mathscr{A}}^{\mathbf{Game_2}}(\lambda) = Adv_{\mathscr{A}}^{\mathbf{Game_1}}(\lambda). \tag{5}$$

$\mathscr{A}$ can only win $Game_2$ with probability since $C_{w_b^*}$ is independent of b. Thus, the advantage is

$$Adv_{\mathscr{A}}^{Game_2}(\lambda) = \left|\frac{1}{2} - \frac{1}{2}\right| = 0. \tag{6}$$

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

$$\left|Adv_{\mathscr{A}}^{\mathbf{Game_2}}(\lambda) - Adv_{\mathscr{A}}^{\mathbf{KGA}}(\lambda)\right| \le Adv_A^{HDH}(\lambda), \tag{7}$$

where $Adv_{\mathscr{A}}^{HDH}(\lambda)$ are negligible. Therefore, the advantage of $\mathscr{A}$ wins in the IND-CKA game is negligible.

(i) *Setup*(λ): $\mathscr{C}$ runs Setup(λ) and KeyGen(pp) to generate the public parameter pp and public/private key pair of participants $(PK_S, SK_S) = (g^y \bmod p, y)$ of DS, $(PK_R, SK_R) = (g^x \bmod p, x)$ of DR, of $(PK_{CS}, SK_{CS}) = (g^z \bmod p, z)$ of CS. Then, $\mathscr{C}$ sends $(pp, PK_S, PK_R, PK_{CS})$ to $\mathscr{A}$. Based on what we know so far about hash functions, we assume H_1 and H_2 are secure and collision-resistant

(ii) *Phase* 1: $\mathscr{A}$ adaptively issues queries to $\mathcal{O}_{\mathscr{C}}$ and $\mathcal{O}_{\mathscr{T}}$, and $\mathscr{C}$ is simulated as follows:

(a) $\mathcal{O}_{\mathscr{C}}$: choose a random $r \in Z_p^*$, compute $C_1 = g^{xr} \bmod p$, $C_2 = H_2(g^{xr} \cdot g^t \bmod p)$, where $t = g^{H_1(w\|l\|PK_S\|PK_R)}$, $l = PK_R^y = g^{xy}$. The ciphertext is $C_w = (C_1, C_2)$

(b) $\mathcal{O}_{\mathscr{T}}$: compute $l = PK_S^x = g^{yx}$, for keyword w, $t = g^{H_1(w\|l\|PK_S\|PK_R)}$. Then, choose a random $k(1 \le k \le p-1)$, $\gcd(k, p-1) = 1$, compute $u = g^k \bmod p$, $T_1 = g^{kz}$, $T_2 = (t - x \cdot u)k^{-1} \bmod (p-1)$. The trapdoor is $T_w = (T_1, T_2)$

(iii) *Challenge*: $\mathscr{A}$ chooses two keywords (w_0^*, w_1^*) and sends them to $\mathscr{C}$. $\mathscr{C}$ chooses a random bit $b \in \{0, 1\}$, and then the trapdoor of the challenge keyword w_b^* as follows:

(a) Compute $l = PK_S^x = g^{yx}$, for keyword w_b^*, $t = g^{H_1(w_b^*\|l\|PK_S\|PK_R)}$. Then, choose a random $k(1 \le k \le p-1)$, $\gcd(k, p-1) = 1$, compute $u = g^k$

mod p, $T_1 = g^{kz}$, $T_2 = (t - x \cdot u)k^{-1}$ mod $(p - 1)$. The trapdoor is $T_{w_b^*} = (T_1, T_2)$.

(iv) *Phase* 2: Oracles continue to be consulted as *Phase*1

(v) *Guess*: $\mathscr{C}$ outputs a guess $b' \in \{0, 1\}$. If $b' = b$ then $\mathscr{C}$ wins the game. $\mathscr{A}$ has the advantage that is

$$Adv_{\mathscr{A}}^{\mathbf{Game_0}}(\lambda) = Adv_{\mathscr{A}}^{\mathbf{KGA}}(\lambda) \tag{8}$$

Theorem 7. *Our PAEKS scheme implements IND-KGA security if the HDH assumption holds.*

Lemma 8. *The advantage of $Adv_{\mathscr{A}}^{KGA}$ is negligible for any polynomial 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$_0$. *Game*$_0$ is the original IND-KGA. □

Game$_1$. Let Game$_1$ be the same game as Game$_0$, except that $\mathscr{C}$ chooses $R \in G$ instead of computing $t^* = g^{H_1(w_b^* \| l \| PK_S \| PK_R)}$. $\mathscr{C}$ sends the trapdoor is $T_{w_b^*} = (T_1, T_2)$, where $T_1 = g^{kz}$, $T_2 = (R - x \cdot u)k^{-1}$ mod $(p - 1)$, $k(1 \le k \le p - 1)$, gcd $(k, p - 1) = 1$, $u = g^k$ mod p

We have

$$\left| Adv_{\mathscr{A}}^{\mathbf{Game_0}}(\lambda) - Adv_{\mathscr{A}}^{\mathbf{Game_1}}(\lambda) \right| < Adv_{\mathscr{A}}^{HDH}(\lambda), \tag{9}$$

where $Adv_A^{HDH}(\lambda)$ is negligible if the HDH assumption holds.

Game$_2$. Let Game$_2$ be the same game as Game$_1$, except that the $\mathscr{C}$ random chooses $T_1, T_2 \in G$ instead of $T_{w_b^*} = (T_1, T_2)$, where $T_1 = g^{kz}$, $T_2 = (R - x \cdot u)k^{-1}$ mod $(p - 1)$, $k(1 \le k \le p - 1)$, gcd $(k, p - 1) = 1$, $u = g^k$ mod p. Due to k and R are random, which $T_{w_b^*}$ of *Game*$_1$ and *Game*$_2$ are the same distribution from adversary's view.

We have

$$Adv_{\mathscr{A}}^{\mathbf{Game_1}}(\lambda) = Adv_{\mathscr{A}}^{\mathbf{Game_2}}(\lambda). \tag{10}$$

We know that $\mathscr{A}$ can only win with probabilityin *Game*$_2$ because $T_{w_b^*}$ is independent of b. Thus, the advantage is

$$Adv_{\mathscr{A}}^{Game_2}(\lambda) = \left| \frac{1}{2} - \frac{1}{2} \right| = 0. \tag{11}$$

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

$$\left| Adv_{\mathscr{A}}^{\mathbf{Game_2}} - Adv_{\mathscr{A}}^{\mathbf{KGA}} \right| \le Adv_A^{HDH}(\lambda). \tag{12}$$

where $Adv_{\mathscr{A}}^{HDH}(\lambda)$ are negligible. Therefore, the advantage of $\mathscr{A}$ wins in the IND-KGA game is negligible.

(i) *Setup*(λ): The challenger $\mathscr{C}$ runs *Setup*(λ) and *Key Gen*(pp) to generate pp and $(PK_S, SK_S) = (g^y \text{ mod } p, y)$, $(PK_R, SK_R) = (g^x \text{ mod } p, x)$, and $(PK_{CS}, SK_{CS}) = (g^z \text{ mod } p, z)$. Then, $\mathscr{C}$ sends $(pp, PK_S, PK_R, PK_{CS})$ to $\mathscr{A}$. Based on our assumptions, we assume that hash function H_1 and H_2 are secure and resistant to collisions

(ii) *Phase*1: $\mathscr{A}$ adaptively sends queries to $\mathscr{O}_{\mathscr{C}}$, $\mathscr{O}_{\mathscr{T}}$ and $\mathscr{O}_{Test}$, and $\mathscr{C}$ is simulated as follows:

(a) $\mathscr{O}_{\mathscr{C}}$: Choose a random $r \in Z_p^*$, compute $C_1 = g^{xr}$ mod p, $C_2 = H_2(g^{xr} \cdot g^t \text{ mod } p)$, where $t = g^{H_1(w \| l \| PK_S \| PK_R)}$, $l = PK_R^y = g^{xy}$. The ciphertext is $C_w = (C_1, C_2)$.

(b) $\mathscr{O}_{\mathscr{T}}$: Compute $l = PK_S^x = g^{yx}$, for keyword w, $t = g^{H_1(w \| l \| PK_S \| PK_R)}$. Then, choose a random $k(1 \le k \le p - 1)$, gcd $(k, p - 1) = 1$, compute $u = g^k$ mod p, $T_1 = g^{kz}$, $T_2 = (t - x \cdot u)k^{-1}$ mod $(p - 1)$. The trapdoor is $T_w = (T_1, T_2)$.

(c) $\mathscr{O}_{Test}$: For any the cipher C_w and trapdoor $T_{w'}$ of keyword w and w', $\mathscr{C}$ runs $Test(SK_{CS}, PK_R, C_w, T_{w'})$, and outputs 1 if $w' = w$; otherwise, it outputs 0

(iii) *Challenge*: $\mathscr{A}$ selects (w_0^*, w_1^*). Sends them to $\mathscr{C}$. $\mathscr{C}$ selects a random bit $b \in \{0, 1\}$, and encrypts w_b^*. $\mathscr{C}$ picks a random bit $d \in \{0, 1\}$, and the trapdoor of the challenge keyword w_d^* as follows:

(a) Choose a random $r^* \in Z_p^*$, compute $C_1^* = g^{xr^*}$ mod p, $C_2 = H_2(g^{xr^*} \cdot g^{t^*} \text{ mod } p)$, where $t^* = g^{H_1(w_b^* \| l \| PK_S \| PK_R)}$, $l = PK_R^y = g^{xy}$. The ciphertext is $C_{w_b^*}^* = (C_1^*, C_2^*)$.

(b) Compute $l = PK_S^x = g^{yx}$, for keyword w_d^*, $t = g^{H_1(w_d^* \| l \| PK_S \| PK_R)}$. Then, choose a random $k(1 \le k \le p - 1)$, gcd $(k, p - 1) = 1$, compute $u = g^k$ mod p, $T_1 = g^{kz}$, $T_2 = (t - x \cdot u)k^{-1}$ mod $(p - 1)$. The trapdoor is $T_{w_d^*} = (T_1, T_2)$.

Theorem 9. *Our scheme PAEKS implements HMR-Security if the DL assumption holds.*

Lemma 10. *The advantage $Adv_{\mathscr{A}}^{HMR}$ is negligible for any polynomial adversary $\mathscr{A}$.*

Proof. We define games as follows:

Game$_0$. The *Game*$_0$ is the original version of this game. □

Finally, $\mathscr{C}$ sends $C_{w_b^*}^*$ and $T_{w_d^*}$ to adversary.

Table 3: Security Comparison.

Schemes	Fully CI-security	Fully TI-security	Search pattern	Assumption
HL17 [4]	×	×	×	mDLLN and DBDH
QC+20[5]		×	×	BDH
QC+21 [7]		×	×	BDH and CODH
PL21 [6]	×	×	×	BDHI
Ours				HDH and DL

√: The schemes supporting corresponding features are supported. ×: The scheme cannot support the corresponding feature.

Table 4: Operations comparison of PAEKS schemes.

Schemes	KeyGen	PAEKS	Trapdoor	Test
HL17 [4]	$2E_G$	$3E_G + H_1$	$3E_G + P + H_1$	$2P$
QC+20[5]	$2E_G$	$3E_G + P + H_1$	$2E_G + H_1$	P
QC+21 [7]	$2E_G$	$3E_G + P + H_1$	$2E_G + H_1$	P
PL21 [6]	$2E_G$	$3E_G + H_1$	$3E_G + P + H_1$	P
Ours	$3E_G$	$4E_G + H_1$	$4E_G + H_1$	$3E_G$

H_1: denoting a hash-to-point operation. P: denoting a bilinear pairing operation. E_G: denoting a modular exponentiation.

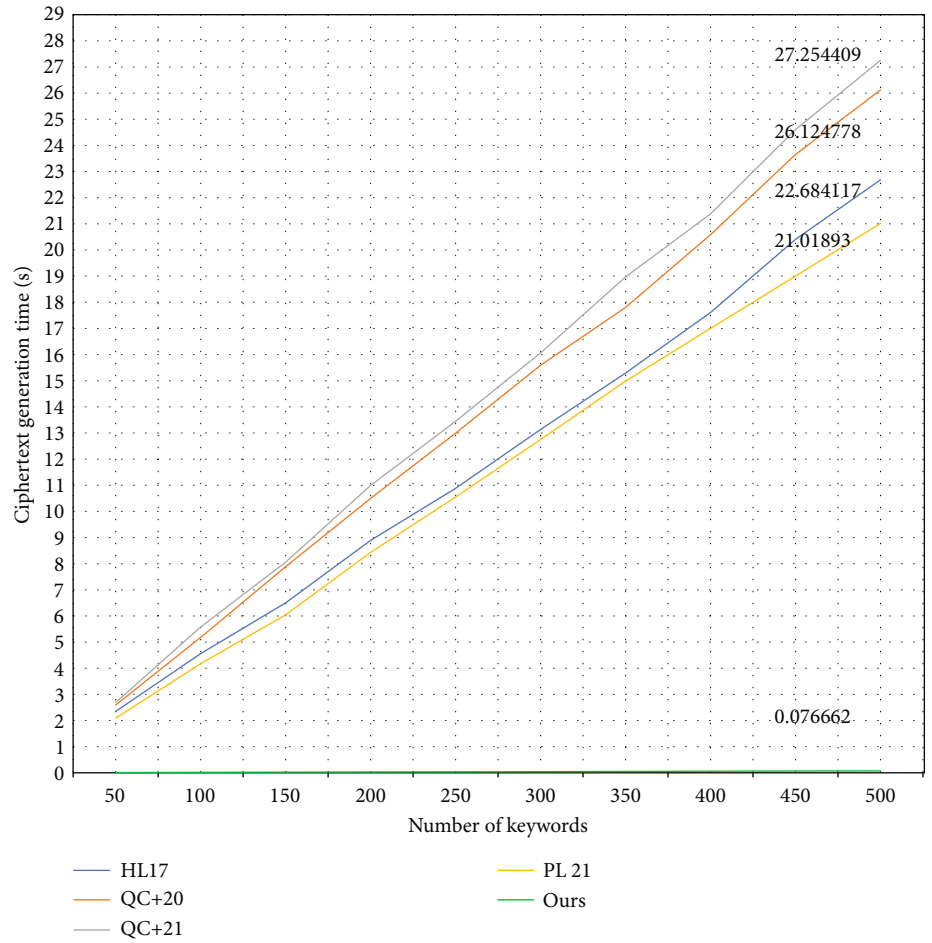

Figure 4: Computation cost of ciphertext generation.

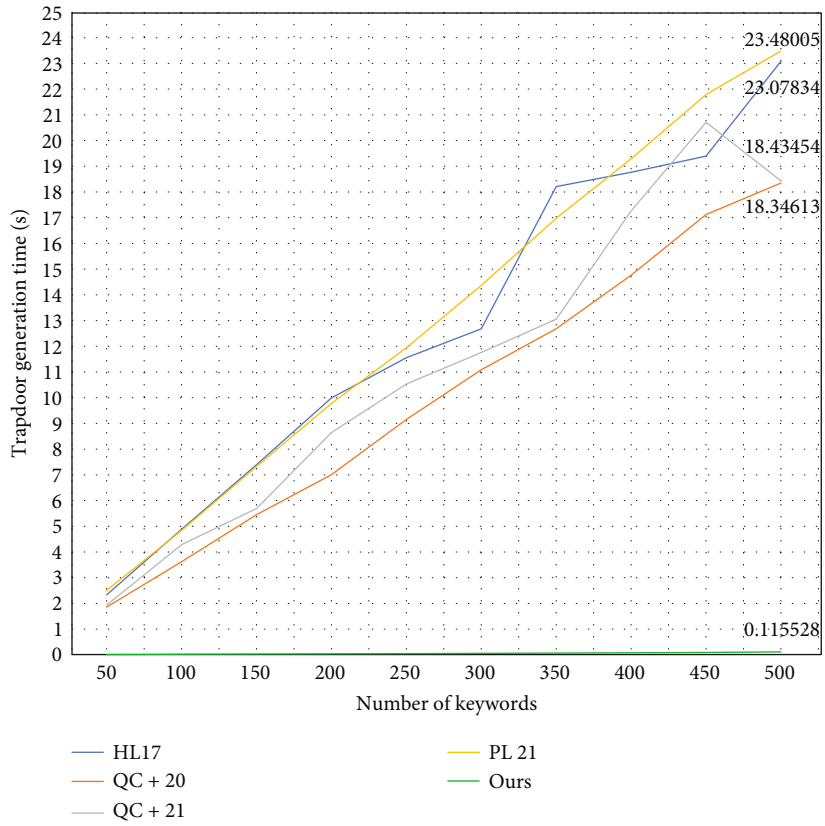

Figure 5: Computation cost of trapdoor generation.

(iv) *Phase* 2: *Phase* 1 continues to be issued by $\mathscr{A}$ as queries to Oracles. There is only one restriction is that neither $C^*_{w_b^*}$ or $T_{w_d^*}$ to $\mathcal{O}_{Test}$

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

According to the definition of the HMR game, the advantage of adversary $\mathscr{A}$ is

$$Adv_{\mathscr{A}}^{\mathbf{Game_0}}(\lambda) = Adv_{\mathscr{A}}^{\mathbf{HMR}}(\lambda). \tag{13}$$

According to Theorem 5 and Theorem 7, we know that the distribution of ciphertext and trapdoor of keywords is the same as that of random tuples from adversary view. So, $\mathscr{A}$ has to test algorithm to verify whether the ciphertext matches the trapdoor. Given g^z and g^{kz}, which calculating the value of k is DL problem.

Finally, we have

$$\left| Adv_{\mathscr{A}}^{\mathbf{Game_0}} - Adv_{\mathscr{A}}^{\mathbf{HMR}} \right| \leq Adv_A^{DL}(\lambda), \tag{14}$$

where $Adv_{\mathscr{A}}^{DL}(\lambda)$ are negligible. Therefore, the advantage of $\mathscr{A}$ wins in the HMR game is negligible.

6.3. Security Analysis

(i) *Resisting attack IKGA.* As far as PAEKS is concerned, only the DS has the ability to generate a legal ciphertext. The adversary executes the *PAEKS* algorithm, it cannot generate k. Similarly, the adversary can not generate the trapdoor of keyword. *Test* algorithm cannot provide any information to the adversary. Thus, our scheme can resist IKGA

(ii) *Access pattern and Search pattern.* For adversary Type-I, we can know that the ciphertext and trapdoor of keywords meet the indistinguishability according to Theorem 5 and Theorem 7. Therefore, the access pattern and search pattern are not compromised. However, the adversary Type-III can run the test algorithm, which will reveal some search information of the user. Then, the adversary can carry out leakage abuse attacks by combining some prior knowledge. In addition, we cannot hide the matching relationship between the ciphertexts and trapdoors from this type of powerful adversary

7. Comparison and Analysis

We compare our scheme with the authentication based searchable schemes (HL17 [4], QC+20 [5], QC+21 [7], and PL21 [6]), 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 libraries.

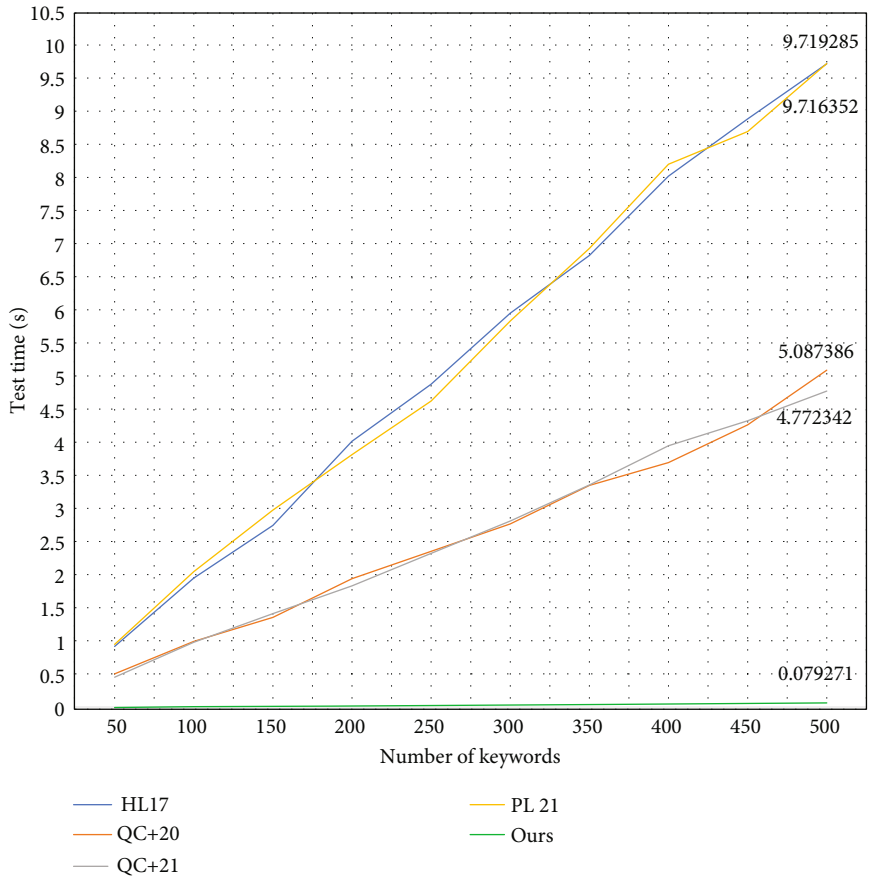

Figure 6: Computation cost of test algorithm.

7.1. Comparative Analysis of Security. The Table 3 indicates the comparison results among the proposed PAEKS. Again, we emphasize that since the server can perform test operations, the inseparability of ciphertext and the indistinguishability of trapdoor for the same keyword have no practical significance unless the keyword is filled in like the extended version of the scheme. Therefore, we only compare the indistinguishability of external adversaries or curious servers before a retrieval token is given. We adopted the security description of [7]. In the table, we are only comparing the security of adversary Type-II. Qin et al. [7] introduced the definition of fully CI-security and fully TI-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 [4], QC+20 [5], QC+21 [7], and PL21 [6] 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. Table 3 shows that only our scheme achieves the fully CI-security and fully TI-security. Meanwhile, it can protect the user's search pattern.

7.2. Time Complexity. The Table 4 shows the number of operations of each algorithm. E_G is a symbol for exponentiation in group G. P is a symbol for the pairing operation. H_1 is a symbol for a group element that maps any string to G. In Table 4, other schemes employ bilinear pair operation, which greatly reduces the efficiency. The computational costs of Setup and KeyGen of various schemes are both similar and have the same algorithm. Therefore, we just consider Encrypt, Trapdoor, and Test algorithms. Qin et al.'s [5, 7] schemes need to compute 3 exponentiation operations, one hash-to-point operation and one pair operation in the ciphertext phase. Their scheme needs to compute 2 power operations and one hash-to-point operation in the trapdoor phase. Huang et al.'s [4] and Pan et al.'s [6] operate the same number of times in the cipher phase and the trapdoor phase. Notice that their schemes all use bilinear bilinear operations, which leads to a large computational overhead. We can see that our scheme is the fastest in the ciphertext, trapdoor, and test phases. This is because our algorithm uses power operations which are computationally less expensive. It can be seen that our scheme is more suitable for lightweight devices.

7.3. Evaluation. We can evaluate the effectiveness of the various schemes (HL17, QC+20, QC+21, and PL21) by

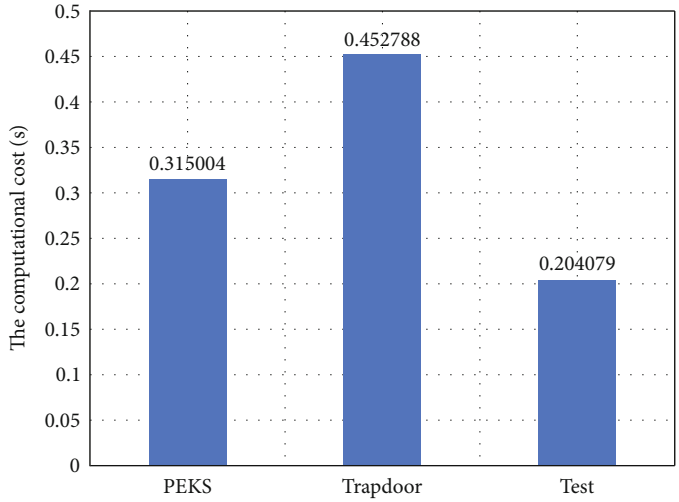

Figure 7: Computation cost of PAEKS in real dataset.

incorporating Relic and GMP. Platforms used in this experiment include Ubuntu 18.04.5 LTS with Intel(R) Xeon(R) CPU E5-2620 v4@2.10GHz and 16.00GB of RAM. The pseudo random permutation was computed using the AES algorithm (CBC model, 128bit key). The hash functions were computed using the SHA256 algorithm. We choose the real Encron Email Dataset (Version 20150307, about 423 MB) to demonstrate the practical performance of our proposed scheme, which contains the data from about 150 users [31]. We choose about 2000 keywords whose lengths are not less than 5 characters and the total number of occurrences is higher than 20.

This paper compares the proposed scheme and the schemes in HL17, QC+20, QC+21, and PL21 in terms of PAEKS, Trapdoor, and Test. The keywords were also chosen randomly for this experiment. As shown in Figures 4–6, we have the lowest computation cost among the five schemes for generating ciphertexts, trapdoors, and test. A key point of the high computational efficiency of our scheme is that it does not require some bilinear pairing operations, which can save a lot of computing overhead. In the ciphertexts generation algorithm, Pan and Li [6] computational overhead is about 274 times that of ours. Compared to our trapdoor generation algorithm, Qin et al.'s [5] computation overhead is about 158 times higher. It is estimated that Qin et al.'s [7] computational overhead is about 60 times ours in the test algorithm.

To evaluate the efficiency of the PAEKS algorithm, we give the time consumption of each algorithm for testing 2000 keywords, as shown in Figure 7. Communication and network effects were removed in the experimental results. Note that to generate 2000 keyword ciphertexts in our scheme, the trapdoor and test take 315.004 ms, 452.788 ms, and 204.079 ms, respectively. The time cost of generating a cipher text is 0.1575 ms. The time cost to generate a trapdoor is 0.2264 ms. The search algorithm is efficient, with an average time cost of about 0.102 ms for matching. Therefore, our scheme is more suitable for some lightweight or computationally constrained devices.

8. Conclusion

In this paper, we have defined different types of adversary capabilities and firstly, analyze the relationship between ciphertext indistinguishability, trapdoor indistinguishability, access pattern, and search pattern. Then, we discuss the security of existing PAEKS schemes. Based on [7], we add the full TI-security model so that we can resist the leakage of search pattern. For adversary Type-II, we design a PAEKS scheme without bilinear pairs, which greatly improves efficiency compared with the previous project. Because the scheme does not use bilinear pairs as a whole, it is more suitable for edge servers and clients with limited computing power. In addition, we also put forward a solution on how to protect access pattern disclosure against adversary Type-III. Unfortunately, the memory expansion is too large. The future work is possible to construct a PAEKS scheme that can protect the search pattern and access pattern and has less memory overhead.

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 Shandong Provincial Key Research and Development Program (No. 2020CXGC010107, 2021CXGC010107), the Special Project on Science and Technology Program of Hubei Provience (No. 2020AEA013), the Natural Science Foundation of Hubei Province (No. 2020CFA052) and the Wuhan Municipal Science and Technology Project (No. 2020010601012187).

References

[1] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *International conference on the theory and applications of cryptographic techniques*, pp. 506–522, Springer, 2004.

[2] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Annual international cryptology conference*, pp. 213–229, Springer, 2001.

[3] 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 *Workshop on Secure Data Management*, pp. 75–83, Springer, 2006.

[4] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Information Sciences*, vol. 403-404, pp. 1–14, 2017.

[5] 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.

[6] 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 102075, 2021.

[7] B. Qin, H. Cui, X. Zheng, and D. Zheng, "Improved security model for public-key authenticated encryption with keyword search," in *International Conference on Provable Security*, pp. 19–38, Springer, 2021.

[8] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: a provably secure scheme under keyword guessing attack," *IEEE Transactions on Computers*, vol. 62, no. 11, pp. 2266–2277, 2012.

[9] 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. 1–798, 2015.

[10] L. Sun, C. Xu, M. Zhang, K. Chen, and H. Li, "Secure searchable public key encryption against insider keyword guessing attacks from indistinguishability obfuscation," *SCIENCE CHINA Information Sciences*, vol. 61, no. 3, pp. 1–3, 2018.

[11] M.-S. Hwang, C.-C. Lee, and S.-T. Hsu, "An elgamal-like secure channel free public key encryption with keyword search scheme," *International Journal of Foundations of Computer Science*, vol. 30, no. 2, pp. 255–273, 2019.

[12] Y.-C. Chen, X. Xie, P. S. Wang, and R. Tso, "Witness-based searchable encryption with optimal overhead for cloud-edge computing," *Future Generation Computer Systems*, vol. 100, pp. 715–723, 2019.

[13] X. Xie, Y.-C. Chen, J.-R. Wang, and Y. Wu, "Witness-Based Searchable Encryption with Aggregative Trapdoor," in *International Conference on Security with Intelligent Computing and Big-data Services*, pp. 561–573, Springer, 2020.

[14] W. Zhang, B. Qin, X. Dong, and A. Tian, "Public-key encryption with bidirectional keyword search and its application to encrypted emails," *Computer Standards & Interfaces*, vol. 78, article 103542, 2021.

[15] S. Niu, W. Liu, S. Han, and L. Fang, "A data-sharing scheme that supports multi-keyword search for electronic medical records," *PLoS One*, vol. 16, no. 1, article e0244979, 2021.

[16] T. Chi, B. Qin, and D. Zheng, "An efficient searchable public-key authenticated encryption for cloud-assisted medical Internet of Things," *Wireless Communications and Mobile Computing*, vol. 2020, Article ID 8816172, 11 pages, 2020.

[17] M. Noroozi and Z. Eslami, "Public-key encryption with keyword search: a generic construction secure against online and offline keyword guessing attacks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 2, pp. 879–890, 2020.

[18] K. Emura, *Generic construction of public-key authenticated encryption with keyword search revisited: stronger security and efficient construction*, Cryptology ePrint Archive, 2022.

[19] Z.-Y. Liu, Y.-F. Tseng, R. Tso, M. Mambo, and Y.-C. Chen, *Public-Key authenticated encryption with keyword search: A Generic Construction and Its quantum-resistant instantiation*, Cryptology ePrint Archive, 2021.

[20] Q. Zheng, S. Xu, and G. Ateniese, "Vabks: verifiable attribute-based keyword search over outsourced encrypted data," in *IEEE INFOCOM 2014-IEEE conference on computer communications*, pp. 522–530, IEEE, 2014.

[21] H. Li, Y. Yang, Y. Dai, S. Yu, and Y. Xiang, "Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data," *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 484–494, 2020.

[22] Q. Xu, C. Tan, W. Zhu, Y. Xiao, Z. Fan, and F. Cheng, "Decentralized attribute-based conjunctive keyword search scheme with online/offline encryption and outsource decryption for cloud computing," *Future Generation Computer Systems*, vol. 97, pp. 306–326, 2019.

[23] J. Cui, H. Zhou, Y. Xu, and H. Zhong, "OOABKS: online/offline attribute-based encryption for keyword search in mobile cloud," *Information Sciences*, vol. 489, pp. 63–77, 2019.

[24] J. Ye and Y. Ding, "Controllable keyword search scheme supporting multiple users," *Future Generation Computer Systems*, vol. 81, pp. 433–442, 2018.

[25] B. Gupta, "An efficient KP design framework of attribute-based searchable encryption for user level revocation in cloud," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 18, article e5291, 2020.

[26] B. Gupta, "An attribute-based keyword search for m-health networks," *Journal of Computer Virology and Hacking Techniques*, vol. 17, no. 1, pp. 21–36, 2021.

[27] Y. Zhang, T. Zhu, R. Guo, S. Xu, H. Cui, and J. Cao, *Multi-Keyword Searchable and Verifiable Attribute-Based Encryption Over Cloud Data*, IEEE Transactions on Cloud Computing, 2021.

[28] B. B. Gupta and M. D. Lytras, *Fog-enabled secure and efficient finegrained searchable data sharing and management scheme for IOT-based healthcare systems*, IEEE Transactions on Engineering Management, 2022.

[29] Q. Song, Z. Liu, J. Cao, K. Sun, Q. Li, and C. Wang, "Sap-sse: protecting search patterns and access patterns in searchable symmetric encryption," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1795–1809, 2020.

[30] L. Blackstone, S. Kamara, and T. Moataz, *Revisiting leakage abuse attacks*, Cryptology ePrint Archive, 2019.

[31] 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.