*future internet*

MDPI

*Article*

# How Can We Achieve Query Keyword Frequency Analysis in Privacy-Preserving Situations?

**Yiming Zhu, Dehua Zhou *, Yuan Li, Beibei Song and Chuansheng Wang**

College of Information Science and Technology/College of Cyber Security, Jinan University, Guangzhou 510632, China; layer_zym@163.com (Y.Z.); yuanli.931012@gmail.com (Y.L.); ssong0802b@gmail.com (B.S.); tcswang@jnu.edu.cn (C.W.)
* Correspondence: tzhoudh@jnu.edu.cn

**Abstract:** Recently, significant progress has been made in the field of public key encryption with keyword search (PEKS), with a focus on optimizing search methods and improving the security and efficiency of schemes. Keyword frequency analysis is a powerful tool for enhancing retrieval services in explicit databases. However, designing a PEKS scheme that integrates keyword frequency analysis while preserving privacy and security has remained challenging, as it may conflict with some of the security principles of PEKS. In this paper, we propose an innovative scheme that introduces a security deadline to query trapdoors through the use of timestamps. This means that the keywords in the query trapdoor can only be recovered after the security deadline has passed. This approach allows for keyword frequency analysis of query keywords without compromising data privacy and user privacy, while also providing protection against keyword-guessing attacks through the dual-server architecture of our scheme. Moreover, our scheme supports multi-keyword queries in multi-user scenarios and is highly scalable. Finally, we evaluate the computational and communication efficiency of our scheme, demonstrating its feasibility in practical applications.

**Keywords:** searchable encryption; keyword frequency analysis; multi-keyword search; keyword guessing attacks; multi-user access



**Citation:** Zhu, Y.; Zhou, D.; Li, Y.; Song, B.; Wang, C. How Can We Achieve Query Keyword Frequency Analysis in Privacy-Preserving Situations? *Future Internet* **2023**, *15*, 197. https://doi.org/10.3390/fi15060197

Academic Editor: Jingsha He

Received: 16 April 2023
Revised: 26 May 2023
Accepted: 26 May 2023
Published: 29 May 2023



## 1. Introduction

With the exponential growth in massive data generated by the internet of things [1,2], traditional storage systems are struggling to meet the storage demands. As a result, cloud storage technology [3] based on cloud computing has gained popularity among businesses and individuals for its scalability and flexibility.

Despite the rapid and recent development of cloud storage technology, most businesses and individuals prefer to encrypt their important data before uploading it to a cloud server, due to concerns about data security. This does effectively guarantee the security of data; however, it also makes it difficult to retrieve and analyze the encrypted data.

Searchable encryption (SE) is a robust solution that addresses the challenge of searching encrypted data while preserving data confidentiality. The concept of public key encryption with keyword search (PEKS) was introduced by Boneh [4] et al. in their seminal work. A PEKS scheme typically involves three entities: a data owner, a data user, and a cloud server. To share data with data users, the data owner extracts keywords from files and encrypts them with the public key of the designated data user, thereby generating searchable ciphertexts. The data owner then sends the searchable ciphertext, together with the corresponding files, to the cloud server. Subsequently, the designated data user is the only entity capable of generating a query trapdoor, by using their private key and specifying a keyword of interest. The generated query trapdoor is then sent to the cloud server to retrieve the associated file. Finally, the cloud server will match the searchable ciphertext with the query trapdoor one-by-one, and send the matching result to the data

user. Since SE was proposed, it has been applied in many fields, such as online medical care [5,6], cloud manufacturing [7], etc.

In many instances of cloud storage applications, counting the frequency of keyword queries is important to improve the search experience of users. For example, in a cloud manufacturing system [7], an upstream supplying plant (the data owner) will encrypt some production data from its workshop and store the ciphertext on a cloud server. Subsequently, the service platform providing the ciphertext retrieval service can respond to the search request of the purchasing company (the data users), and provide the supply factory that meets their needs. If another service platform that provides keyword frequency analysis services can analyze the frequency of the query keywords while ensuring the privacy of the purchasing company, the user's search experience can be significantly optimized (e.g., by placing the data of the most frequent keywords in the best position in the database). Meanwhile, the cloud server shares analysis results with upstream supply factories, which can also help factories to make targeted production adjustments, thus greatly improving production efficiency. In our solution, the query token used by the purchasing company (the data users) to query is embedded with a timestamp specified by the user themselves, which is similar to the user setting a validity period for their query records, during which the query operation will not compromise the user's privacy. After this validity period, a third party can recover the keyword information of the query token, but cannot associate this keyword information with the user's identity. In this way, we propose a PEKS scheme that perfectly combines security with keyword frequency analysis. Meanwhile, the scheme has many advantages: it is multi-user, has the capacity for multi-keyword queries, and can resist KGA (keyword-guessing attack).

### *1.1. Our Contribution*

In this work, we propose a PEKS scheme with keyword frequency analysis (KFA-PEKS), which takes into account privacy protection and frequency analysis of keywords. Furthermore, KFA-PEKS supports multi-user participation and multi-keyword search; therefore, it is suitable for distributed IoT scenarios such as cloud manufacturing [7]. Overall, our contributions in this paper are as follows:

- We propose a novel KFA-PEKS scheme that combines searchable encryption with keyword frequency analysis, while maintaining high levels of security and scalability. Our solution allows the server to periodically analyze the frequency of queried keywords without linking this information to the user's identity, thereby preserving user privacy.
- We conduct a comprehensive functional comparison of our KFA-PEKS scheme with existing SE schemes. The comparison results demonstrate that our scheme is unique in achieving multi-keyword search, resistance to KGA, and frequency analysis of query keywords within a multi-writer/multi-reader model. Furthermore, the complexity analysis of our scheme indicates that it is highly practical and applicable.
- We implement our KFA-PEKS scheme using Python, and perform extensive security evaluations, including computational and communication overhead analysis, to demonstrate its feasibility and effectiveness. Our experimental results highlight the advantages of our KFA-PEKS scheme compared to other PEKS schemes, showcasing its superior performance and suitability for real-world applications.

### *1.2. Paper Organization*

In Section 2, we will review the relevant literature and related work. In Section 3, some preliminaries required to understand our proposed KFA-PEKS will be given. In Section 4, we will formally describe the KFA-PEKS scheme. Next, in Section 5, we will present a concrete construction of KFA-PEKS and provide a proof of its security. The efficiency of our construction will be analyzed theoretically in Section 6, and we will showcase its practicality through experimental results. Finally, in Section 7, we will conclude the paper as a whole.

## 2. Related Work

Since the concept of SE was first introduced in [8], SE has been divided into two categories: symmetric searchable encryption (SSE) [9–11] and PEKS [4,12–15]. The two have different application scenarios and construction methods. SSE generally considers the use of a single user, which is equivalent to establishing a personal encrypted cloud disk, and relies on symmetric encryption algorithms for scheme construction; PEKS, which mainly relies on public key cryptography algorithms, usually considers multi-user scenarios, such as mail systems or multi-person file-sharing systems.

The first PEKS scheme was proposed by Boneh et al. [4], wherein the data owner could use the public key of a specified data user, so that the user could use their key to generate a trapdoor for query keywords to search encrypted data. Since then, many PEKS schemes have been proposed. Among these schemes, examples in the literature, such as [16,17], support multi-keyword search; however, these schemes cannot support multiple data owners and data users at the same time. Sun et al. propose a PEKS scheme [18] supporting multiple keyword queries and multiple users, which combines a data structure representing a keyword/identity representation (T-set) in [11], and ciphertext-policy attribute-based encryption (CP-ABE) in [19]. In their proposed scheme, the data owners grant keys to the data users, requiring each data user to maintain a set of keys in order to access data that are outsourced by different data owners. In addition, because this scheme does not deal with the relationship between the ciphertext and keywords, it brings a huge storage overhead to each entity in the scheme. In 2019, Xu et al. [20] proposed a lattice-based PEKS scheme that was derived from a blind identity-based encryption (blind IBE) [21] scheme, by substituting identities with keywords. Their construction involves an identity-based PEKS that maps the identity of the data owner to a matrix, allowing the data owner to encrypt data with the identity of the data user. In 2021, Liu [22] et al. combined a subset decision mechanism on a distributed two-trapdoor public key cryptosystem [23] to construct a PEKS scheme to be applied in distributed systems. In this scheme, both the data owner and data user have their public and private key pairs, and can generate corresponding searchable ciphertexts and trapdoors with their respective public keys. However, this scheme, similarly to all previously proposed PEKS schemes, loses the availability of data; in addition, it cannot expand the potential application scenarios of public key searchable encryption, because it does not consider keyword frequency analysis of query keywords.

In 2019, Xu et al. [24] proposed the first PEKS scheme based on blind IBE [21], which can accomplish keyword frequency analysis under the premise of protecting user privacy. Unfortunately, due to its single-server setup, this solution is not immune to KGA [25,26]. In addition, it is difficult to extend the query method from a single keyword to multiple keywords. KGA is a common type of attack facing PEKS schemes. Within KGA, an attacker can generate searchable ciphertexts for any desired keyword using the public key of the data user, and can then test these ciphertexts against the search trapdoor. To mitigate KGA, various cryptographic primitives have been proposed, including public key authenticated encryption with keyword search (PAEKS) [27–29] and public key encryption with fuzzy keyword search (PEFKS) [30]. Based on [22], we propose a PEKS that can achieve keyword frequency analysis while protecting data privacy and user privacy. In addition, this PEKS is suitable for multi-user distributed scenarios and is resistant to KGA.

## 3. Preliminaries

For ease of reading, we refer to [22] for some symbols and related terms, and present the following in Table 1, where $W$ is a universal keyword set with keyword number $\eta$, $W = \{\omega_{\eta-1}, ..., \omega_0\}$. After that, we have a decimal number $T \in \{0, ..., 2^\eta - 1\}$ to represent the relationship between each keyword set $W$ and a document $D$. This is expressed as follows:

$$T_i = \begin{cases} 1, & if\ \omega_i\ is\ contained\ in\ D, \\ 0, & otherwise \end{cases}$$

$W_T = \{\omega_i | \omega_i \in W, T_i = 1\}$ denotes the keyword set corresponding to $T$. When a data user (DU) initiates a query request, we use a decimal number $t \in \{0, ..., 2^\eta - 1\}$ to represent the relationship between each keyword set $W$ and DU's interest. This is expressed as follows:

$$t_i = \begin{cases} 1, & if\ DU\ is\ interetsed\ in\ \omega_i, \\ 0, & otherwise \end{cases}$$

$W_t = \{\omega_i | \omega_i \in W, t_i = 1\}$ denotes the keyword set corresponding to $t$. In the description that follows, $t$ matches $T$ equivalently to $W_t \subseteq W_T$.

**Table 1.** Summary of notations.

| Notations | Descriptions |
|---|---|
| $W$ | The universal keyword set |
| $D$ | A document |
| $DS$ | All documents stored on the server |
| $DS[\omega]$ | All documents that contain the keyword $\omega$ |
| $SQ$ | Keywords included in a query |
| $W_{id}$ | All keywords contained in document with identifier $id$ |
| $W_c$ | The keyword set represented by $c$ |
| $\omega$ | A keyword |
| $(c_{\eta-1}, ..., c_0)$ | The unsigned binary representation of a positive decimal $c$ |
| $c_{\eta-1}$ | The highest bit |
| $c_0$ | The least bit |
| $\eta$ | The bit length of $c$, the number of keywords in $W$ |
| $\neg c$ | The complement of $c$ |
| $T$ | A positive decimal integer is used as the plaintext representation of searchable ciphertext, and its binary representation can represent the relationship between the set $W_T$ and the universal set $W$ |
| $t$ | A positive decimal integer is used as the plaintext representation of searchable ciphertext, and its binary representation can represent the relationship between the set $W_t$ and the universal set $W$ |
| $sk_i$ | Secret key of participant $i$ |
| $pk_i$ | Public key of participant $i$ |
| $MSK_i$ | Partial master key of participant $i$ |
| $TSO$ | Timestamp specified by DUs |
| $TSN$ | Timestamp generated periodically by the time server |
| $[\cdot]_{pk}$ | The encryption of $\cdot$ under the public key $pk$ |
| $SUM$ | The decimal integer $2^\eta - 1$ |
| $L(\cdot)$ | The bit length of $\cdot$ |

### *3.1. Pseudorandom Function*

A function $F : \{0,1\}^\kappa \times \{0,1\}^l \to \{0,1\}^{l'}$ is called a pseudorandom function (PRF) [31] if for all PPT adversary $\mathcal{A}$, its advantage $|Pr[\mathcal{A}^{F(k,\cdot)}(1^\kappa) = 1] - Pr[\mathcal{A}^{R(\cdot)}(1^\kappa) = 1]| \leq v(\kappa)$, where $k \xleftarrow{\$} \{0,1\}^\kappa$, $R$ is a random function denoted as: $\{0,1\}^l \to \{0,1\}^{l'}$, and $v(\kappa)$ is negligible.

### *3.2. Subset Decision Mechanism (SDM)*

Given a universal set $W$, and two subsets and $W_T$ and $W_t$, and these sets being represented in decimal integers and binary representations, the role of the SDM [22] is to perform some calculations on these decimal integers and binary representations to determine whether $W_t \subseteq W_T$. In this paper, the SDM is used as an important component of the matching process. In the next description, we let $W$ denote the set of all keywords whose binary and decimal forms are denoted as $(1, ..., 1)$ and $SUM$, and $W_T$ denote the set of keywords corresponding to the document uploaded by the data owner (DO), which is represented in binary and decimal form as $(T_{\eta-1}, ..., T_0)$ and $T$, respectively. Furthermore,

we let $W_t$ denote the set of keywords corresponding to the data user's (DU's) query request in binary and decimal form as $(t_{\eta-1}, ..., t_0)$ and $t$, respectively.

The key for the SDM to determine $W_t \subseteq W_T$ is to ensure that there is no $i$ that satisfies $t_i = 1$ and $T_i = 0$. In the SDM, the inputs are a universal keyword set $W = \{\omega_{\eta-1}, ..., \omega_0\}$ with two subsets WT and Wt such that $W_T, W_t \subseteq W$. The outputs are a result value $\mathcal{R}$, which represents whether $W_t \subseteq W_T$. The specific process can be described as follows. We first compute the complement $(\neg T_{\eta-1}, ..., \neg T_0)$ of the binary of $T$ and the corresponding integer $\neg T$. Then, perform a bitwise addition of $\neg T_i$ and $t_i$ to obtain $\mathcal{C}_i$. Finally, if there is no $\mathcal{C}_i = 2$, then, $W_t \subseteq W_T$; otherwise, $W_t \nsubseteq W_T$. The working mechanism of the SDM is described formally in Algorithm 1.

**Algorithm 1** Subset Decision Mechanism Source: [22], Algorithm 1 Subset Decision

**Input:**
A universal set $W = \{\omega_{\eta-1}, ..., \omega_0\}$, two subsets $W_t, W_T \subseteq W$.
**Output:**
Whether $W_t \subseteq W_T$.
1: Compute the binary representations $(T_{\eta-1}, ..., T_0)$, $(t_{\eta-1}, ..., t_0)$ of $W_T$, $W_t$.
2: Compute the complement $(\neg T_{\eta-1}, ..., \neg T_0)$ of $(T_{\eta-1}, ..., T_0)$.
3: Set $i = 0, \mathcal{R} = 1$.
4: **while** $i < \eta$ **do**
5: $\quad \mathcal{C}_i = \neg t_i + T_i$;
6: $\quad \mathcal{D}_i = 2 - \mathcal{C}_i$;
7: $\quad \mathcal{R} = \mathcal{R} \cdot \mathcal{D}_i$;
8: **end while**
9: **if** $\mathcal{R} = 0$ **then**
10: $\quad$ **return** $W_t \nsubseteq W_T$
11: **else**
12: $\quad$ **return** $W_t \subseteq W_T$
13: **end if**

In addition, we are able to accelerate the execution of the SDM when $W_t \nsubseteq W_T$. That is, when $W_t \subseteq W_T$, there is always $sum = t + \neg T \leq SUM = 2^\eta - 1$. We describe this process in Algorithm 2.

**Algorithm 2** Subset Decision Mechanism With Modification. Source: [22], Algorithm 1 Subset Decision with Modification

**Input:**
A universal set $W = \{\omega_{\eta-1}, ..., \omega_0\}$, two subsets $W_t, W_T \subseteq W$.
**Output:**
Whether $W_t \subseteq W_T$.
1: We additionally compute the decimal integers $T, t$ of $W_T, W_t$, $SUM = 2^\eta - 1$ of $W$, $\neg T = SUM - T$ and $sum = \neg T + t$ after executing steps 1 and 2 in Algorithm 1.
2: **if** $sum > SUM$ **then**
3: $\quad$ **return** $W_t \nsubseteq W_T$
4: **else**
5: $\quad$ Go to Step 3 of Algorithm 1.
6: **end if**

### *3.3. Secure Bit-Decomposition (SBD)*

For a ciphertext $[c]_{pk}$ encrypted with $pk$, SBD [32] can encrypt each bit of $c$, and will not leak information about $c$ to the two parties involved in the calculation in SBD. A more standardized expression of the SBD protocol is as follows.

$$\mathbf{SBD}([c]_{pk}) \rightarrow ([c_{\eta-1}]_{pk}, ..., [c_0]_{pk}).$$

### *3.4. DT-PKC*

A distributed two-trapdoor public key cryptosystem (DT-PKC) [23] is a toolkit that can securely handle common integer operations across different cryptographic domains, meaning it will become an important part of the KFA-PEKS.

#### 3.4.1. Basic Structure

Our scheme takes a similar introduction to DT-PKC as that of [22]. Among them, since this paper needs to use the joint decryption of dual servers, the $PSDec1$ algorithm and $PSDec2$ algorithm in [23] are retained and rewritten as $PMDec1$ algorithm and $PMDec2$ algorithm in this paper. Furthermore, this paper does not involve the joint decryption between users, so the PWDec1 and PWDec2 algorithms are omitted. It is important to emphasize that this change will not affect the content that follows. The DT-PKC infrastructure consists of eight algorithms: $KeyGen$, $Enc$, $Udec$, $MDec$, $MkeyS$, $PMDec1$, $PMDec2$, and $CR$. The definition of DT-PKC accepted in [23], with slight modifications, is as follows:

- *KeyGen*: Given the security parameter $\kappa$, KGC finds two large primes $p, q$ such that $L(p) = L(q) = \kappa$, where $L(x)$ refers to the length of the parameter $x$, then computes $N = pq$, $p' = \frac{(p-1)}{2}$, $q' = \frac{(q-1)}{2}$, of which $p'$ and $q'$ are two strong primes. Simultaneously, KGC also chooses a generator of the order $\frac{(p-1)(q-1)}{2}$, and chooses a random number $\theta$ such that $\theta \in [1, \frac{N}{4}]$. Finally, KGC obtains the public key $pk_i = (N, g, h_i = g^{\theta})$ and the private key $sk_i = \theta$ for user $i$, and computes the master key $MSK = \lambda = \frac{lcm(p-1,\ q-1)}{2}$, where $lcm(a, b)$ refers to finding the least common multiple of $a$ and $b$.
- *Encryption (Enc)*: Enter the plaintext $m$ and the public key $pk_i$ and choose a random number $r$ ($r \in [1, \frac{N}{4}]$) to generate the ciphertext $[m]_{pk_i} = \{C_{i,1}, C_{i,2}\}$, where $C_{i,1} = g^{r\theta}(1 + mN) \ \ mod \ \ N^2$; $T_{i,2} = g^r \ \ mod \ \ N^2$.
- *Decryption With User's Private Key (UDec)*: Enter the ciphertext $[m]_{pk_i}$ and the private key $sk_i$ to generate the corresponding plaintext $m$ as follows:
$$m^{'} = \frac{C_{i,1}}{C_{i,2}} \ \ mod \ \ N^2,$$
$$m = \frac{m^{'}-1}{N}.$$
- *Decryption With Master Key (MDec)*: Enter the ciphertext $[m]_{pk_i}$ and the master key $MSK$ to generate the corresponding plaintext $m$ by first calculating:
$$C_{i,1}^{\lambda} \ \ mod \ \ N^2 = g^{\lambda \cdot \theta \cdot r}(1 + mN\lambda) \ \ mod \ \ N^2 = (1 + mN\lambda).$$
Then, since $gcd(\lambda, N) = 1$, we are able to obtain $m$ by the following expression:
$$m^* = C_{i,1}^{\lambda} \ \ mod \ \ N^2,$$
$$m = \frac{m^*-1}{N} \cdot \lambda^{-1} \ \ mod \ \ N.$$
- *Master Key Splitting (MkeyS)*: Enter the master key $MSK$ to generate two partial master keys $MSK_1 = \lambda_1$ and $MSK_2 = \lambda_2$ such that $\lambda_1 + \lambda_2 \equiv 0 \ \ mod \ \ \lambda$ and $\lambda_1 + \lambda_2 \equiv 1 \ \ mod \ \ N^2$.
- *Partial Decryption With Partial Master Key Step One (PMDec1)*: Enter the ciphertext $[m]_{pk_i}$ and the partial master key $MSK_1$ to generate the partial ciphertext $CT_i^1$ as follows:
$$CT_i^1 = (C_{i,1})^{\lambda_1} = g^{r\theta\lambda_1}(1 + mN\lambda_1) \ \ mod \ \ N^2.$$
- *Partial Decryption With Partial Master Key Step Two (PMDec2)*: Enter the partial ciphertext $CT_i^1$, the ciphertext $[m]_{pk_i}$ and the partial master key $MSK_2$ to generate the corresponding plaintext $m$ by first calculating:
$$CT_i^2 = (C_{i,1})^{\lambda_2} = g^{r\theta\lambda_2}(1 + mN\lambda_2) \ \ mod \ \ N^2.$$
Then, we can obtain the plaintext $m$ by computing:
$$CT = CT_i^1 \cdot CT_i^2,$$
$$m = \frac{CT-1}{N}.$$
- *Ciphertext Refresh (CR)*: Enter the ciphertext $[m]_{pk_\theta}$ and a random number $r^{'} \in \mathbb{Z}_N$ to generate another ciphertext $[m]_{pk_\theta}^{'} = \{C_{i,1}^{'}, C_{i,2}^{'}\}$, where
$$C_{i,1}^{'} = T_{i,1} \cdot h_i^{r^{'}}; \ C_{i,2}^{'} = C_{i,2} \cdot g^{r^{'}}.$$

#### 3.4.2. Sub-Protocols

DT-PKC is capable of deriving several sub-protocols, which can be found in [23]. Since the sub-protocols $SAD$ and $SMD$ of DT-PKC introduced in [22] are involved in our scheme, only the sub-protocols $SAD$ and $SMD$ are briefly described later. The following is a brief description of the sub-protocols $SAD$ and $SMD$:

- *Secure Addition Protocol across Domains (SAD)*: Enter the two ciphertexts $[m_1]_{pk_{\theta_1}}$ and $[m_2]_{pk_{\theta_2}}$, the partial master keys $MSK_1$ and $MSK_2$, and the public keys $pk_{\theta_1}$, $pk_{\theta_2}$ and $pk_{\theta_3}$ to generate the addition of two ciphertexts $[m_1 + m_2]_{pk_{\theta_3}}$ in different encryption domains.
- *Secure Multiplication Protocol across Domains (SMD)*: Enter the same inputs as the $SAD$ to generate the multiplication of two ciphertexts $[m_1 \cdot m_2]_{pk_{\theta_3}}$ in different encryption domains.

## 4. KFA-PEKS

### 4.1. System Model

The following entities are included in our scheme: a key generation center (KGC), a matching server (MS), a keyword frequency server (KFS), a time server (TS), data owners (DOs), and data users (DUs). A brief description of our system model is given in Figure 1.

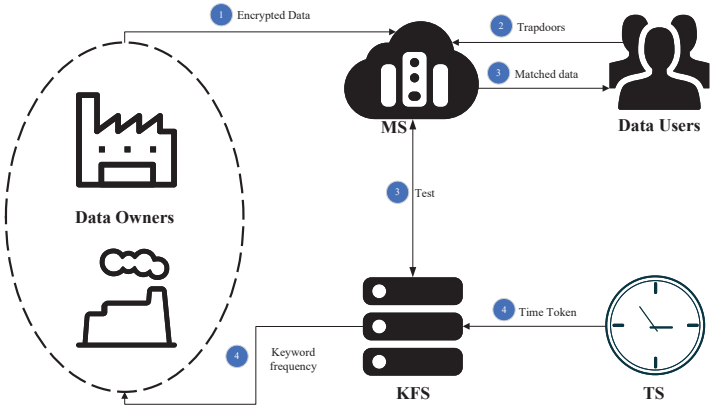


**Figure 1.** Our scheme consists of four main steps. In the first step, the DOs encrypt their data using their public key to generate searchable ciphertexts, which are then uploaded to an MS. In the second step, the DUs generate query trapdoors for the set of keywords of interest and send them to the MS. In the third step, the MS and the KFS cooperate to calculate the query results, which are then sent back to the corresponding DU. Finally, in the fourth step, the TS periodically sends time tokens to the KFS. The KFS can use time tokens to recover the keywords that satisfy the query trapdoor for the current timestamp, and then analyze these keywords. The analysis results are sent back to the DOs.

- KGC: The role of the KGC is to generate public parameters and distribute the corresponding keys to the various entities involved in the scheme.
- MS: The MS mainly provides secure file storage services for the DOs and secure file search services for the DUs.
- KFS: The KFS collaborates with the MS in processing the DUs' file query requests, and conducts keyword frequency analysis on the DUs' queries, while ensuring the privacy of the DUs, in order to provide enhanced file storage and lookup services to both the DOs and DUs.
- TS: The TS periodically sends time tokens embedded with the current timestamp to the MS, which are then used to recover keyword information from the DUs' queries in order to conduct keyword frequency analysis.
- DO: Each DO generates their own public–private key pair based on the public parameters. They then extract keywords from a file to generate searchable ciphertext, which is sent to the MS along with the file for secure storage and search services.

- DU: Each DU generates their own public–private key pair based on the public parameters and generates a query trapdoor for a set of keywords of interest. Only the DU has the ability to decrypt the queried results received from the MS in order to obtain the corresponding file index, thereby ensuring privacy and security.

### *4.2. Threat Model*

We assume that the KGC is an honest party, assigning corresponding keys to other members in KFA-PEKS.

The MS and the KFS are assumed to be a pair of non-colluding semi-honest adversaries (specific references in [33]). In KFA-PEKS, the MS will try to obtain information about trapdoors (e.g., keyword information and corresponding documentation information); the KFS will infer query keywords while analyzing the frequency of query keywords corresponding to the user information. Non-collusion means that there will be no unnecessary interaction between the MS and the KFS, except for query calculation and keyword frequency analysis.

We assume that the TS, DOs, and DUs are semi-honest, that the TS will strictly generate the correct time token based on the current time and send it to the KFS, and that the DOs and DUs will also honestly execute individual protocols in KFA-PEKS. However, they will all try to gain access to the private data of other members of the scheme.

Based on the definition of individual entities given above, we introduce an adversary $\mathcal{A}^*$ into our model that is capable of compromising some of the entities in the model. For example, $\mathcal{A}^*$ can conspire with the MS or KFS (but not both) to guess the information in the ciphertext. Furthermore, $\mathcal{A}^*$ can conspire with several DOs or DUs to obtain the decryption capability of the corresponding ciphertext or trapdoor. Such a security model has strong practical implications, and we recommend that interested readers consult [23], upon which we will not expand in this paper for brevity reasons.

### *4.3. Security Goals*

Due to the need for user privacy, even if these keywords are restored by the KFS after $TSO$, the DUs' identities should not be linked to the keyword they queried. Furthermore, these keywords should also not have been disclosed prior to the $TSO$. Specifically, the security objectives of our program should achieve the following five points:

- *Keyword privacy of DUs Before TSO*: The DUs' query operations will not leak keyword information until the current time reaches the $TSO$ specified by the DU.
- *Indistinguishability of DO's searchable ciphertext*: The searchable ciphertext uploaded by the DO does not reveal any information related to the file.
- *Keyword privacy for DUs after TSO*: After reaching the $TSO$ specified by the DU, only the KFS can recover the queried keywords, and for others, the trapdoor still retains its previous security.
- *Unlinkability of recovered keywords and DUs' identities*: The KFS is unable to locate the relevant DU through recovering the keyword.
- *Resist KGA*: The focus of this paper is on offline keyword-guessing attacks by internal attackers, that is, on preventing internal attackers, such as untrusted cloud servers, from performing exhaustive attacks on keyword information in encrypted searchable ciphertext or trapdoors.

### *4.4. Syntax*

Public key encryption with keyword search supporting keyword frequency analysis is a protocol shared among a KGC, an MS, a KFS, a TS, multiple DPs, and multiple DUs, as follows:

- $Setup(\kappa) \rightarrow (PP, r, r_2)$: Given the security parameter $\kappa$, the KGC generates the public parameter $PP$, a value shared by the KFS and DUs, and a key $r_2$, shared by the TS and DUs.
- $KeyGen_{Ser}(PP) \rightarrow (MSK_{MS}, MSK_{KFS})$: Given the public parameter $PP$, the KGC generates two keys for the MS and KFS, $MSK_{MS}$ and $MSK_{KFS}$.

- $KeyGen_{DO}(PP) \rightarrow (pk_{DO}, sk_{DO})$: Given the public parameter $PP$, the KGC generates the public–private key pair $(pk_{DO}, sk_{DO})$ for the DOs.
- $KeyGen_{DU}(PP) \rightarrow (pk_{DU}, sk_{DU})$: Given the public parameter $PP$, the KGC generates the public–private key pair $(pk_{DU}, sk_{DU})$ for the DUs.
- $KeyGen_{TS}(PP) \rightarrow (pk_{TS}, sk_{TS})$: Given the public parameter $PP$, the KGC generates the public–private key pair $(pk_{TS}, sk_{TS})$ for the TS.
- $PEKS(W, W_T, pk_{DO}) \rightarrow [T]_{pk_{DO}}$: Given the universal keyword set $W$, a keyword subset $W_T$, and the DO's public key, the DO computes the searchable ciphertext $[T]_{pk_{DO}}$.
- $Trapdoor(W, r, r_2, sk_{DU}, pk_{TS}, TSO) \rightarrow [t]_{PK}$: Given the universal keyword set $W$, the value $r$, the key $r_2$, the DU's secret key $sk_{DU}$, the TS's public key $pk_{TS}$, and a timestamp, the DU computes an encryption key $PK$ and the trapdoor $[t]_{PK}$.
- $Test([T]_{pk_{DO}}, [t]_{PK}, MSK_{MS}, MSK_{KFS}, pk_{DO}, PK) \rightarrow [1]_{PK}$ or $[0]_{PK}$: Given the searchable ciphertext $[T]_{pk_{DO}}$, trapdoor $[t]_{PK}$, the MS's secret key $MSK_{MS}$, the KFS's secret key $MSK_{KFS}$, the DO's public key $pk_{DO}$, and the encryption key $PK$, the MS and the KFS compute a test result. The DU can compute the test result and outputs 1 if $W_t \subseteq W_T$, or otherwise, 0.
- $TimeTokenGen(sk_{TS}, pk_{DU}, r_2, TSN) \rightarrow TT$: Given the TS's secret key $sk_{TS}$, the DU's public key $pk_{DU}$, the key $r_2$, and a timestamp, the TS generates a time token $TT$.
- $OpenKeyword([t]_{PK}, r, TT) \rightarrow t$: Given the encryption key $PK$, the value $r$, and a time token $TT$, the KFS can recover to obtain $t$.

### *4.5. Correctness*

The correctness of the solution KFA-PEKS is mainly reflected in two aspects: the correctness of the search results and the correctness of the keyword frequency analysis. Specifically, we express the correctness as follows:

- Correctness of search results:
  - $Test([T]_{pk_{DO}}, [t]_{PK}, MSK_{MS}, MSK_{KFS}, pk_{DO}, PK) \rightarrow 1$ *if, and only if,* $W_t \subseteq W_T$.
  - $Test([T]_{pk_{DO}}, [t]_{PK}, MSK_{MS}, MSK_{KFS}, pk_{DO}, PK) \rightarrow 0$ *if, and only if,* $W_t \nsubseteq W_T$.
- Correctness of keyword frequency analysis:
  - $OpenKeyword([t]_{PK}, r, TT) \rightarrow t$ *if, and only if,* $TSN \nsubseteq TSO$.
  - $OpenKeyword([t]_{PK}, r, TT) \rightarrow \neg t$. *if, and only if,* $TSN \subseteq TSO$.

## 5. Construction

### *5.1. The Concrete Construction*

For clarity and visibility, we denote the subset decision mechanism as the SDM in the description that follows. The KFA-PEKS is constructed as follows:

- $Setup(\kappa) \rightarrow PP$: Given the security parameter $\kappa$, the KGC finds two large primes $p$, $q$ such that $L(p) = L(q) = \kappa$, where $L(x)$ refers to the length of the parameter $x$, then computes $N = pq$, $p' = (p-1)/2$, $q' = (q-1)/2$, of which $p'$ and $q'$ are two strong primes. Simultaneously, the KGC also chooses a generator of the order $(p-1)(q-1)/2$ and initializes a set of keywords $W$, the total number of which is $\eta$. Later the KGC sends a secret value $r$ to the KFS and the DUs, and sends $PP = (W, \eta, N, g)$ to others.
- $KeyGen_{Ser}(PP) \rightarrow (MSK_{MS}, MSK_{KFS})$: The KGC first executes *KeyGen* in DT-PKC to obtain the master key $MSK = \lambda$, and then executes *MkeyS* in DT-PKC to generate two partial master keys $MSK_1 = \lambda_1$, $MSK_2 = \lambda_2$. Finally, the KGC sends $MSK_{MS} = MSK_1 = \lambda_1$ to the MS, the partial master key $MSK_{KFS} = MSK_2 = \lambda_2$ to the KFS, and keeps $MSK = \lambda$ secret.
- $KeyGen_{DO}(PP) \rightarrow (pk_{DO}, sk_{DO})$: Each DO chooses a random number $sk_{DO} = \theta_1$ and calculates $h_{DO} = g^{\theta_1}$. They then publish $pk_{DO} = (N, h_{DO}, g)$ as their public key, and keep $sk_{DO}$ as their private key.

- $KeyGen_{DU}(PP) \rightarrow (pk_{DU}, sk_{DU})$: Each DU chooses a random number $sk_{DU} = \theta_2$ and calculates $h_{DU} = g^{\theta_2}$. They then publish $pk_{DU} = (N, h_{DU}, g)$ as their public key, and keep $sk_{DU}$ as their private key.
- $KeyGen_{TS}(PP) \rightarrow (pk_{TS}, sk_{TS})$: Each TS chooses a random number $sk_{TS} = \theta_3$ and calculates $h_{TS} = g^{\theta_3}$. They then publish $pk_{TS} = (N, h_{TS}, g)$ as their public key, and keep $sk_{TS}$ as their private key.
- $PEKS(W, W_T, pk_{DO}) \rightarrow [T]_{pk_{DO}}$: Firstly, DO chooses a random number $r_1$, and generates $W_T$ based on $W$ such that $W_T \subseteq W$ and computes the corresponding decimal number $T$ according to its binary representation; then, encrypts $T$ with their public key $pk_{DO}$ to obtain the encrypted searchable ciphertext $[T]_{pk_{DO}} = (T_{DO,1}, T_{DO,2})$, which specifically is as follows:

$$T_{DO,1} = g^{r_1\theta_1}(1 + TN) \mod N^2 \quad (1)$$

$$T_{DO,2} = g^{r_1} \mod N^2 \quad (2)$$

  Finally, the DO sends $[T]_{pk_{DO}}$ to the MS.
- $Trapdoor(W, r, r_2, sk_{DU}, pk_{TS}, TSO) \rightarrow [t]_{PK}$: the DU uses their own private key $sk_{DU}$ and TS's public key $pk_{TS}$ to generate an encryption key $PK = g^{\theta_2\theta_3}$. They then generate the keyword set $W_t$ of interest based on $T$ such that $W_t \subseteq W$ and compute the corresponding decimal number $t$ according to its binary representation. At the same time, the DU runs a PRF $F$ for the specified point in order to obtain a timestamp, and shares the key $r_2$ with the TS. Subsequently, the DU encrypts $t$ with $PK$, a secret value $r$ (shared with the KFS) and their specified timestamp, and obtains the trapdoor $[t]_{PK} = t_{DU,1}, t_{DU,2}$, which specifically is as follows:

$$t_{DU,1} = g^{\theta_2 \cdot \theta_3 \cdot r \cdot F(r_2, TSO)}(1 + tN) \mod N^2 \quad (3)$$

$$t_{DU,2} = g^{\theta_3} \mod N^2. \quad (4)$$

  Finally, the DU sends $[t]_{PK}$ to the MS.
- $Test([T]_{pk_{DO}}, [t]_{PK}, MSK_{MS}, MSK_{KFS}, pk_{DO}, PK) \rightarrow [1]_{PK}$ or $[0]_{PK}$: After receiving $[T]_{pk_{DO}}$ and $[t]_{PK}$, the MS can perform the following four steps with the KFS to obtain an encrypted query result and send it to the DU.
  - step 1: After the MS receives $[T]_{pk_{DO}}$ and $[t]_{PK}$, it will execute the $SBD$ protocol with the KFS to calculate the ciphertext of each bit of $\neg T$ and $t$, i.e., $[\neg T_i]_{pk_{DO}}$ and $[t_i]_{PK}$ for $i \in \{0, ..., \eta - 1\}$.
  - step 2: After obtaining $[\neg T_i]_{pk_{DO}}$ and $[t_j]_{PK}$, the MS can calculate the ciphertext of each $\mathcal{C}_i$ and $\mathcal{D}_i$ mentioned in the SDM, i.e., $[\mathcal{C}_i]_{PK}$ and $[\mathcal{D}_i]_{PK}$ together with the KFS, and finally the MS sends the randomized matching result to the DU.
  - step 3: After obtaining $[\mathcal{D}_i]_{PK}$, the MS and KFS together calculate the matching result and the randomized value, i.e., $[\mathcal{R}]_{PK}$ and $[\mathcal{F}]_{PK}$.
  - step 4: After receiving $[\mathcal{F}]_{PK}$, the DU decrypts the final matching result. A result of 0 means that the file does not match. Any other result means that the current file does match, in which case the DU will request this file from the MS.
- $TimeTokenGen(sk_{TS}, pk_{DU}, r_2, TSN) \rightarrow TT$: The TS uses $sk_{TS}$ and $pk_{DU}$ to deal with the current timestamp $TSN$ and obtain a time token $TT = g^{\theta_2 \cdot \theta_3 \cdot F(r_2, TSN)} \mod N^2$, this is sent periodically to the KFS.
- $OpenKeyword([t]_{PK}, r, TT) \rightarrow t$: The KFS receives the $TT$, and when the timestamp in the $TT$ is consistent with the timestamp specified by the DU, then, the KFS can recover the keyword information in the trapdoor with $r$ and $TT$, so as to analyze the frequency of the queried keywords later. The recovery process is as follows:

$$t = \frac{\frac{t_{DU,1}}{(TT)^{r'}} - 1}{N} \mod N^2 \quad (5)$$

### *5.2. Process of KFA-PEKS*

Figure 2 shows specifically the workflow of the 6 members and 10 algorithms in KFA-PEKS.

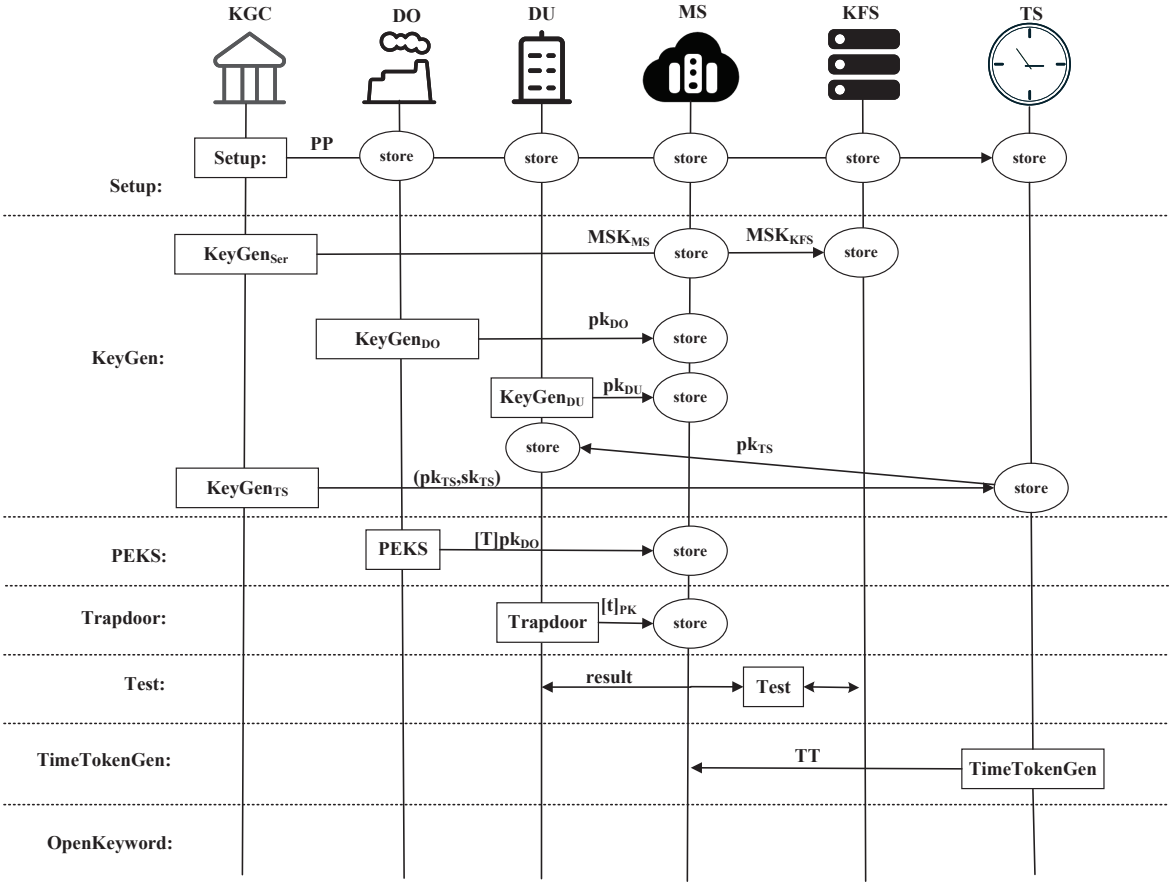


**Figure 2.** Workflow of our system.

Notably, since *TSO* in the query trapdoor is specified by the DUs, the DUs can better grasp the conditions under which the query keywords are recovered, and, therefore, can better protect the privacy of users. A DU can generate multiple trapdoors with different *TSOs*, so that the *TSO* of this trapdoor is a time interval, i.e., the time interval is between $T_1$ = "10:00:00 AM January 1, 2022 GMT" and $T_2$ = "10:00:00 AM December 1, 202 GMT". In this way, only when the current time $T$ satisfies $T_1 \leq T \leq T_2$, can the KFS open the queried trapdoors using the time token TT of the TS.

### *5.3. Security Proof*

On the premise of being correct, our scheme should also satisfy the five security definitions we mentioned earlier: *keyword privacy of DUs before TSO, indistinguishability of DO's searchable ciphertext, keyword privacy of DUs after TSO, unlinkability of recovered keywords and DUs' identities, resistance to internal and external keyword-guessing attacks.* Below, we will prove each of these five security requirements by means of the following Theorems 1–5.

**Theorem 1.** *The keyword privacy of the DU is protected until the TSO specified by the DU is reached.*

**Proof.** The DU generates a trapdoor which is sent to the MS to perform a query. Then, the MS executes the *Test* algorithm together with the KFS, and returns the test result to the DU. It is required that the test result can only be read by the DU, and the trapdoor must not reveal any information about the keywords before reaching the *TSO* specified by the DU. It

is worth noting that in our scheme, the encryption key for the trapdoor must be generated separately by the TS or the DU, while the encryption key for the searchable ciphertext is the public key of the DO; therefore, for the same keyword, the trapdoor is uncorrelated with the searchable ciphertext, meaning that common chosen keyword attacks (CKA) can be avoided in PEKS. We use $\mathcal{A}$ to denote the adversary in the indistinguishability game, and $\mathcal{C}$ to denote the challenger; this game can be represented as follows:

First, $\mathcal{A}$ chooses a time point $T^*$ to attack, and selects two different sets of keywords of interest, $W_1$ and $W_2$, which will be sent to $\mathcal{C}$; then, $\mathcal{C}$ randomly selects one of the two sets to generate a trapdoor which is then sent to $\mathcal{A}$. Finally, $\mathcal{A}$ tries to distinguish which set of keywords corresponds to the trapdoor returned by $\mathcal{C}$.

In the following, if our scheme passes the indistinguishability game of trapdoor privacy (**IND-TP**), then the scheme can be deemed **IND-TP** secure.

- **Setup:** First, $\mathcal{A}$ chooses a point in time $T^{'}$ at which to make the attack. Then, $\mathcal{C}$ executes algorithms $Setup$, $KeyGen_{Ser}$, $KeyGen_{DU}$, $KeyGen_{DO}$, and $KeyGen_{TS}$ to obtain $PP$, $(MSK_{MS}, MSK_{KFS})$, $(pk_{DU}, sk_{DU})$, $(pk_{DO}, sk_{DO})$, and $(pk_{TS}, sk_{TS})$. Furthermore, it sends $(MSK_{MS}, pk_{DU}, pk_{DO}, pk_{TS})$ to $\mathcal{A}$.
- **Phase 1:** $\mathcal{A}$ interacts with $\mathcal{C}$, which executes the $TimeTokenGen$ algorithm in an imitation of the TS, and sends the resulting time token to $\mathcal{A}$.
- **Challenge:** $\mathcal{A}$ picks two different keyword sets $W_0$, $W_1 \subseteq W$, and sends $t_0$ and $t_1$, representing the two sets, to $\mathcal{C}$. Then, $\mathcal{C}$ picks $b \subseteq_R \{0,1\}$, runs $Trapdoor(W, r, sk_{DU}, pk_{TS}, TSO) \rightarrow [t_b]_{PK}$, and sends $[t_b]_{PK}$ to $\mathcal{A}$.
- **Phase 2:** $\mathcal{A}$ can continue with the first phase of queries while also executing the algorithm $PEKS$ in an imitation of the MS interacting with the KFS.
- **Guess:** $\mathcal{A}$ starts guessing $b$. If the given guess $b^{'}$ satisfies $b^{'} = b$ , then $\mathcal{A}$ wins this game.

If $\mathcal{A}$ can win the game described above with a non-negligible probability, then the KFA-PEKS is deemed **IND-TP** secure.

Suppose there exists an adversary $\mathcal{A}$ who can win the game described above with a non-negligible probability. Let $\mathcal{C}$ be the challenger of the trapdoor privacy experiment before TSO in KFA-PEKS, and here construct a simulator $\mathcal{B}$ to exploit adversary $\mathcal{A}$ to defeat the DDH hardness problem based on $\mathbb{Z}_{N^2}^*$.

Let $(g, g^x, g^y, g^{xy}, g^z)$ be a set of examples of DDH hardness problems based on $\mathbb{Z}_{N^2}^*$. Challenger $\mathcal{C}$ first uses the generating element $g$ to construct the public parameters of the scheme $PP = (W, \eta, N, g)$. Then, challenger $\mathcal{C}$ uses $PP$ by itself to run $Setup$, $KeyGen_{Ser}$, $KeyGen_{DO}$, $KeyGen_{DU}$, and $KeyGen_{TS}$, and obtains the secret value $r$ shared by the KFS and the DU, the secret value $r_2$ shared by the TS and the DU, and some of the MS and KFS's partial master key $(MSK_{MS}, MSK_{KFS})$, the DO's public–private key pairs $(pk_{DO}, sk_{DO})$, the DU's public–private key pairs $(pk_{DU}, sk_{DU})$, and the TS's public–private key pairs $(pk_{TS}, sk_{TS})$. Then, challenger $\mathcal{C}$ sends $(PP, MSK_{MS}, pk_{DO}, pk_{DU}, pk_{TS})$ to adversary $\mathcal{A}$ and $(PP, MSK_{KFS}, pk_{DO}, pk_{DU}, pk_{TS}, sk_{TS}, r, r_2)$ to simulator $\mathcal{B}$. Adversary $\mathcal{A}$ chooses two keyword sets $W_0$ and $W_1$ and corresponding trapdoor $[t]_{PK}^b$, where $b \in \{0,1\}$. Next, $\mathcal{A}$ sends $(W_0, W_1, [t]_{PK}^b)$ to $\mathcal{B}$. Finally, $\mathcal{B}$ gives a value $b^{'}$ to guess the set of keywords $W_b^{'}$ corresponding to the trapdoor $[t]_{PK}^b$, where $b^{'} \in \{0,1\}$. Throughout the process, simulator $\mathcal{B}$ allows adversary $\mathcal{A}$ to perform the following oracle queries with the following responses:

- $\boldsymbol{Query_{Trap}(W_b, pk_{DU})}$**:** Input a keyword set $W_b$ and the DU's public key $pk_{DU}$, because simulator $\mathcal{B}$ has the TS's private key $sk_{TS}$, so it can obtain the encryption key $PK = g^{\theta_2\theta_3}$; then, simulator $\mathcal{B}$ runs the $Trapdoor$ algorithm to generate the query trapdoor $[t]_{PK}^b$ corresponding to keyword set $W_b$, and then, simulator $\mathcal{B}$ sends this query trapdoor to adversary $\mathcal{A}$.
- $\boldsymbol{Query_{TS}(TSN)}$**:** The current timestamp $TSN$ is input, and since the simulator $\mathcal{B}$ has the secret value $r$ shared by the KFS and the DU and the secret value $r_2$ shared by the TS and the DU, it can obtain the processed timestamp $g^{r \cdot F(r_2, TSN)}$ and give this value to the adversary $\mathcal{A}$.

- $\boldsymbol{Query_{Test}([T]_{pk_{DO}}, [t]^b_{PK}, MSK_{MS}, MSK_{KFS}, pk_{DO}, PK)}$**:** In this stage of interrogation, adversary $\mathcal{A}$ has the partial master key $MSK_{MS}$ of the MS, and simulator $\mathcal{B}$ uses the partial master key $MSK_{KFS}$ of the KFS to cooperate with adversary $\mathcal{A}$ to execute the *Test* algorithm and send the matching result to adversary $\mathcal{A}$. For trapdoor $[t]^b_{PK} = (t_{DU,1}, t_{DU,2})$, it satisfies:

$$t_{DU,1}=g^{\theta_2 \cdot \theta_3 \cdot r \cdot F(r_2,TSO)}(1+t_bN) \ \ mod \ \ N^2,$$
$$t_{DU,2} = g^{\theta_2\theta_3} \ \ mod \ \ N^2$$

At this point, the simulator $\mathcal{B}$ gives the guessed value $b^{'}$ by a non-negligible advantage such that $b^{'} = b$. This contradicts the DDH difficulty problem based on $\mathbb{Z}^*_{N^2}$, so the query trapdoor satisfies indistinguishability in this scheme until it reaches the *TSO* specified by the DU. □

**Theorem 2.** *The searchable ciphertext does not reveal information about its own keywords.*

**Proof.** When the document is uploaded to the MS, the DO should also attach the corresponding encrypted searchable ciphertext. It is required that the encrypted searchable ciphertext should not reveal any information about its underlying keywords. Similar to the previous security requirements, trapdoor queries and ciphertext queries are not considered. The indistinguishability of the searchable ciphertext **IND-SC** is defined by an interaction experiment between challenger $\mathcal{C}$ and adversary $\mathcal{A}$, which is represented as follows:

- **Setup:** First, $\mathcal{A}$ chooses a point in time $T^{'}$ at which to make the attack. Then, $\mathcal{C}$ executes algorithms $Setup$, $KeyGen_{Ser}$, $KeyGen_{DU}$, $KeyGen_{DO}$, and $KeyGen_{TS}$ to obtain $PP$, $(MSK_{MS}, MSK_{KFS})$, $(pk_{DU}, sk_{DU})$, $(pk_{DO}, sk_{DO})$, and $(pk_{TS}, sk_{TS})$. Furthermore, it sends $(MSK_{MS}, pk_{DU}, pk_{DO}, pk_{TS})$ to $\mathcal{A}$.
- **Challenge:** $\mathcal{A}$ picks two different keyword sets $W_0$, $W_1 \subseteq W$, and sends $T_0$, $T_1$, representing $W_0$ and $W_1$, to $\mathcal{C}$. Then, $\mathcal{C}$ picks $b \subseteq_R \{0,1\}$, runs $PEKS(W, W_T, pk_{DO}) \rightarrow [T]_{pk_{DO}}$, and sends $[T_b]_{PK}$ to $\mathcal{A}$.
- **Guess:** $\mathcal{A}$ starts guessing $b$. If the given guess $b^{'}$ satisfies $b' = b$, then $\mathcal{A}$ wins this game.

If $\mathcal{A}$ can win the game described above with a non-negligible probability, then the KFA-PEKS is deemed **IND-SC** secure.

Suppose there exists an adversary $\mathcal{A}$ who can win the game described above with a non-negligible probability. Let $\mathcal{C}$ be the challenger of the searchable ciphertext indistinguishability experiment in KFA-PEKS, and here construct a simulator $\mathcal{B}$ to exploit adversary $\mathcal{A}$ to defeat the DDH hardness problem based on $\mathbb{Z}^*_{N^2}$.

Let $(g, g^x, g^y, g^{xy}, g^z)$ be a set of examples of DDH hardness problems based on $\mathbb{Z}^*_{N^2}$. Challenger $\mathcal{C}$ first uses the generating element $g$ to construct the public parameters of the scheme $PP = (W, \eta, N, g)$. Then, challenger $\mathcal{C}$ uses $PP$ by itself to run $Setup$, $KeyGen_{Ser}$, $KeyGen_{DO}$, $KeyGen_{DU}$, and $KeyGen_{TS}$, and obtains the secret value $r$ shared by the KFS and the DU, the secret value $r_2$ shared by the TS and the DU, and some of the MS and KFS's partial master key $(MSK_{MS}, MSK_{KFS})$, the DO's public–private key pairs $(pk_{DO}, sk_{DO})$, the DU's public–private key pairs $(pk_{DU}, sk_{DU})$, and the TS's public–private key pairs $(pk_{TS}, sk_{TS})$. Then, challenger $\mathcal{C}$ sends $(PP, MSK_{MS}, pk_{DO}, pk_{DU}, pk_{TS})$ to adversary $\mathcal{A}$ and $(PP, MSK_{KFS}, pk_{DO}, pk_{DU}, pk_{TS}, sk_{TS}, r, r_2)$ to simulator $\mathcal{B}$. Adversary $\mathcal{A}$ chooses two keyword sets $W_0$ and $W_1$ and corresponding trapdoor $[t]^b_{PK}$, where $b \in \{0,1\}$. Next, $\mathcal{A}$ sends $(W_0, W_1, [t]^b_{PK})$ to $\mathcal{B}$. Finally, $\mathcal{B}$ gives a value $b^{'}$ to guess the set of keywords $W^{'}_b$ corresponding to the trapdoor $[t]^b_{PK}$, where $b^{'} \in \{0,1\}$. Throughout the process, simulator $\mathcal{B}$ allows adversary $\mathcal{A}$ to perform the following oracle queries with the following responses:

- $\boldsymbol{Query_{Cipher}(W_b, pk_{DO})}$ **:** Input the keyword set $W_b$ and the DO's public key $pk_{DO}$, and simulator $\mathcal{B}$ runs the $PEKS$ algorithm to generate the searchable ciphertext $[T]^b_{pk_{DO}} = (T_{DO,1}, T_{DO,2})$ corresponding to the keyword set $W_b$, where the ciphertext structure is as follows:

$$T_{DO,1} = g^{r_1\theta_1}(1+T_bN) \ \ mod \ \ N^2,$$

$$T_{DO,2} = g^{r_1} \ \ mod \ \ N^2$$

Then, simulator $\mathcal{B}$ sends this searchable ciphertext to adversary $\mathcal{A}$.

At this point the simulator $\mathcal{B}$ gives the guess value $b^{'}$ by a non-negligible advantage such that $b^{'} = b$. This contradicts the DDH difficulty problem based on $\mathbb{Z}^*_{N^2}$, so the searchable ciphertext in this scheme satisfies indistinguishability. □

**Theorem 3.** *Upon receipt of a time token from TS, the keyword information in trapdoor cannot be recovered by anyone except the KFS and the corresponding DU, based on the intractability of DDH's assumption over $Z^*_{N^2}$ [34] (more specific information about the hardness of DDH's assumption over $Z^*_{N^2}$ can be found in [34]).*

**Proof.** After reaching the timestamp $TSO$ specified by the DU, only the KFS can obtain the keyword information in the trapdoor. Similar to the previous security requirements, trapdoor queries and ciphertext queries are not considered. The trapdoor privacy after TSO is defined by an interaction experiment **IND-sT-TP** (the indistinguishability trapdoor privacy with a specific TSO) between challenger $\mathcal{C}$ and adversary $\mathcal{A}$, which is expressed as follows:

First, $\mathcal{A}$ chooses a time point $T^*$ to attack, and selects two different sets of keywords of interest, $W_1$ and $W_2$, which will be sent to $\mathcal{C}$; then, $\mathcal{C}$ randomly selects one of the two sets to generate a trapdoor which is then sent to $\mathcal{A}$. Finally, $\mathcal{A}$ tries to distinguish which set of keywords corresponds to the trapdoor returned by $\mathcal{C}$.

In the following, if our scheme passes the indistinguishability game of trapdoor privacy with a specific $TSO$ (**IND-sT-TP**), then the scheme can be deemed **IND-sT-TP** secure.

- **Setup:** First, $\mathcal{A}$ chooses a point in time $T^{'}$ at which to make the attack. Then, $\mathcal{C}$ executes algorithms $Setup$, $KeyGen_{Ser}$, $KeyGen_{DU}$, $KeyGen_{DO}$, and $KeyGen_{TS}$ to obtain $PP$, $(MSK_{MS}, MSK_{KFS})$, $(pk_{DU}, sk_{DU})$, $(pk_{DO}, sk_{DO})$, and $(pk_{TS}, sk_{TS})$. Furthermore, it sends $(MSK_{MS}, pk_{DU}, pk_{DO}, pk_{TS})$ to $\mathcal{A}$.
- **Phase 1:** $\mathcal{A}$ interacts with $\mathcal{C}$, which executes the $TimeTokenGen$ algorithm in an imitation of the TS, and sends the resulting time token to $\mathcal{A}$.
- **Challenge:** $\mathcal{A}$ picks two different keyword sets $W_0$, $W_1 \subseteq W$, and sends $t_0$ and $t_1$, representing the two sets, to $\mathcal{C}$. Then, $\mathcal{C}$ picks $b \subseteq_R \{0,1\}$, runs $Trapdoor(W, r, sk_{DU}, pk_{TS}, TSO) \rightarrow [t_b]_{PK}$, and sends $[t_b]_{PK}$ to $\mathcal{A}$.
- **Phase 2:** $\mathcal{A}$ can continue with the first phase of queries while also executing the algorithm $PEKS$ in an imitation of the MS interacting with the KFS.
- **Guess:** $\mathcal{A}$ starts guessing $b$. If the given guess $b^{'}$ satisfies $b^{'} = b$ , then $\mathcal{A}$ wins this game.

If $\mathcal{A}$ can win the game described above with a non-negligible probability, then the KFA-PEKS is deemed **IND-sT-TP** secure.

Suppose there exists an adversary $\mathcal{A}$ who can win the game described above with a non-negligible probability. Let $\mathcal{C}$ be the challenger of the trapdoor privacy experiment after $TSO$ in KFA-PEKS, and here construct a simulator $\mathcal{B}$ to exploit adversary $\mathcal{A}$ to defeat the DDH hardness problem based on $\mathbb{Z}^*_{N^2}$.

Let $(g, g^x, g^y, g^{xy}, g^z)$ be a set of examples of DDH hardness problems based on $\mathbb{Z}^*_{N^2}$. Challenger $\mathcal{C}$ first uses the generating element $g$ to construct the public parameters of the scheme $PP = (W, \eta, N, g)$. Then, challenger $\mathcal{C}$ uses $PP$ by itself to run $Setup$, $KeyGen_{Ser}$, $KeyGen_{DO}$, $KeyGen_{DU}$, and $KeyGen_{TS}$, and obtains the secret value $r$ shared by the KFS and the DU, the secret value $r_2$ shared by the TS and the DU, and some of the MS and KFS's partial master key $(MSK_{MS}, MSK_{KFS})$, the DO's public–private key pairs $(pk_{DO}, sk_{DO})$, the DU's public–private key pairs $(pk_{DU}, sk_{DU})$ and the TS's public–private key pairs $(pk_{TS}, sk_{TS})$. Then, challenger $\mathcal{C}$ sends $(PP, MSK_{MS}, pk_{DO}, pk_{DU}, pk_{TS})$ to adversary $\mathcal{A}$ and $(PP, MSK_{KFS}, pk_{DO}, pk_{DU}, pk_{TS}, sk_{TS}, r, r_2)$ to simulator $\mathcal{B}$. Adversary $\mathcal{A}$ chooses two keyword sets $W_0$ and $W_1$ and corresponding trapdoor $[t]^b_{PK}$, where $b \in \{0,1\}$. Next, $\mathcal{A}$ sends $(W_0, W_1, [t]^b_{PK})$ to $\mathcal{B}$. Finally, $\mathcal{B}$ gives a value $b^{'}$ to guess the set of keywords $W^{'}_b$

corresponding to the trapdoor $[t]_{PK}^{b}$, where $b^{'} \in \{0,1\}$. Throughout the process, simulator $\mathcal{B}$ allows adversary $\mathcal{A}$ to perform the following oracle queries with the following responses:

- $\boldsymbol{Query_{Trap}(W_b, pk_{DU})}$ **:** Input a keyword set $W_b$ and DU's public key $pk_{DU}$, because simulator $\mathcal{B}$ has the TS's private key $sk_{TS}$, so it can obtain the encryption key $PK = g^{\theta_2\theta_3}$; then simulator $\mathcal{B}$ runs *Trapdoor* algorithm to generate the query trapdoor $[t]_{PK}^{b}$ corresponding to keyword set $W_b$, and then simulator $\mathcal{B}$ sends this query trapdoor to adversary $\mathcal{A}$.
- $\boldsymbol{Query_{TS}(TSN)}$ **:** The current timestamp $TSN$ is input, and since the simulator $\mathcal{B}$ has the secret value $r$ shared by the KFS and the DU and the secret value $r_2$ shared by the TS and the DU, it can obtain the processed timestamp $g^{r \cdot F(r_2, TSN)}$ and give this value to the adversary $\mathcal{A}$.
- $\boldsymbol{Query_{Test}([T]_{pk_{DO}}, [t]_{PK}^{b}, MSK_{MS}, MSK_{KFS}, pk_{DO}, PK)}$ **:** In this stage of interrogation, adversary $\mathcal{A}$ has the partial master key $MSK_{MS}$ of the MS, and simulator $\mathcal{B}$ uses the partial master key $MSK_{KFS}$ of the KFS to cooperate with adversary $\mathcal{A}$ to execute the *Test* algorithm and send the matching result to adversary $\mathcal{A}$. For trapdoor $[t]_{PK}^{b} = (t_{DU,1}, t_{DU,2})$, it satisfies:

$$t_{DU,1} = g^{\theta_2 \cdot \theta_3 \cdot r \cdot F(r_2, TSO)}(1 + t_b N) \mod N^2,$$
$$t_{DU,2} = g^{\theta_2 \theta_3} \mod N^2$$

At this point the simulator $\mathcal{B}$ gives the guessed value $b^{'}$ by a non-negligible advantage such that $b^{'} = b$. This contradicts the DDH difficulty problem based on $\mathbb{Z}_{N^2}^{*}$, so the query trapdoor satisfies indistinguishability in this scheme until it reaches the $TSO$ specified by the DU. □

**Theorem 4.** *The KFS is unable to locate the relevant DU through the keyword being recovered.*

**Proof.** In Theorem 3, we prove that after reaching the time point $TSO$, specified by the DU, only the KFS can open the keyword information in the corresponding trapdoor, for the purpose of performing keyword frequency analysis. In order to ensure the privacy of DUs while performing keyword frequency analysis, our solution needs to prevent the KFS from finding the corresponding DUs through the recovered keywords. In the specific structure of our scheme, the MS is responsible for providing file storage services to the DO, and for responding to query requests initiated by the DU at the same time. When performing keyword frequency analysis, the MS sends the query trapdoor of the DU to the KFS, and then the KFS uses the time token of the TS to recover the queried keywords of the DUs. During the entire process, the KFS does not interact with the DUs. On the premise that the MS does not collude with the KFS, the KFS can only recover the query keywords, but cannot link them to the corresponding DUs through the recovery results. In this way, the security required by the scheme is achieved. □

**Theorem 5.** *Internal or external attackers cannot obtain any information about users' keywords from encrypted searchable ciphertext or query trapdoors.*

**Proof.** In the specific structure of the scheme, the result of the joint MS and KFS calculation is a secret value which is encrypted with the DU's public key. Therefore, only the DU can know the query result. In this way, even if an external attacker can construct enough encrypted searchable ciphertext to perform exhaustive attacks on query trapdoors, they have no way of knowing the results of each retrieval; thus, keyword-guessing attacks by external attackers can be avoided. Intuitively, because our solution uses a dual-server setup, the *Test* algorithm is completed by the MS and KFS in succession. Therefore, on the premise that the two servers MS and KFS do not collude, the MS or KFS cannot execute the test algorithm alone. This means that keyword-guessing attacks by internal attackers can also be avoided. □

## 6. Performance Evalution

We conducted experiments by using the Python language, based on the charm-crypto library [35], on a PC with an AMD Ryzen 5 3600 3.6 GHz processor, 4 GB of RAM, and the Ubuntu 18.04 LTS operating system. In the experiment, the security parameter was set to 1024, so that $N$ was a positive integer of 1024 bits, and $p$ and $q$ were large prime numbers of 512 bits. We tested the operating efficiency of KFA-PEKS when the maximum number of keywords was 5, 10, 15, and 20, and carried out a corresponding comparative analysis with [24].

### *6.1. Experimental Results*

Figure 3a–d depict comparisons of the average execution times of the $KeyGen_D$, $KeyGen_U$, $KeyGen_{TS}$, and $TimeTokenGen$ algorithms, respectively, between our scheme and [24]. The average execution times of these algorithms in [24] are 9.107 ms, 6.274 ms, 3.989 ms, and 8.735 ms, respectively. As a comparison, the average execution times of these algorithms in our scheme are 1.409 ms, 1.384 ms, 1.367 ms, and 1.496 m, respectively.

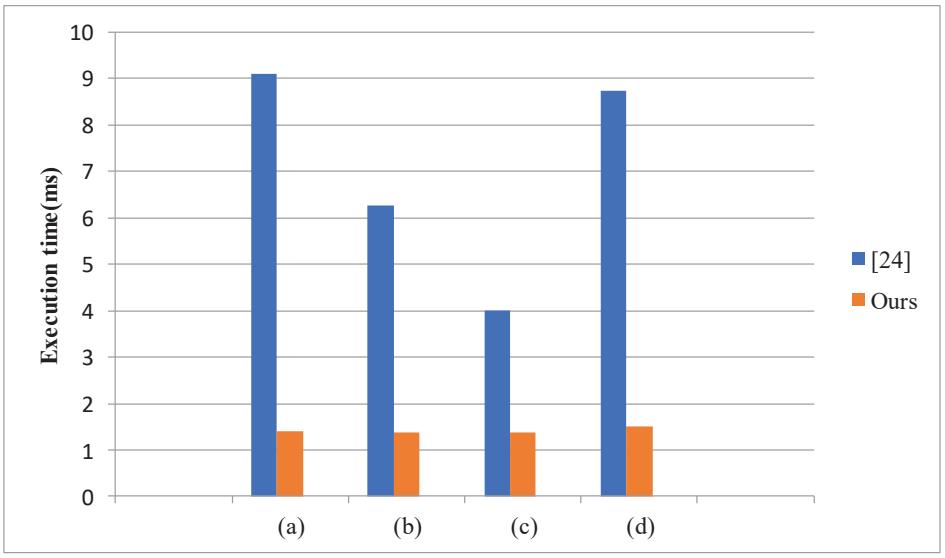


**Figure 3.** The average execution times of ours and [24].

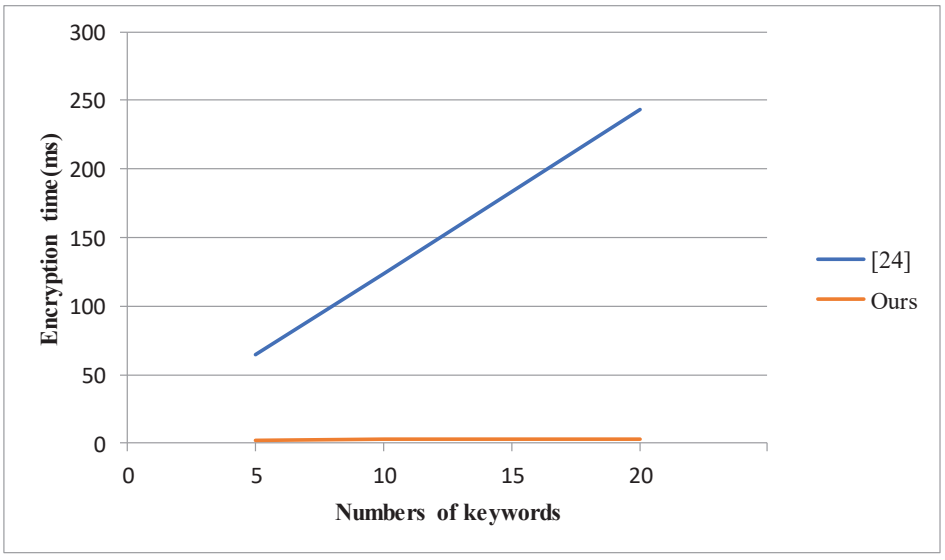


**Figure 4.** The encryption times of ours and [24].

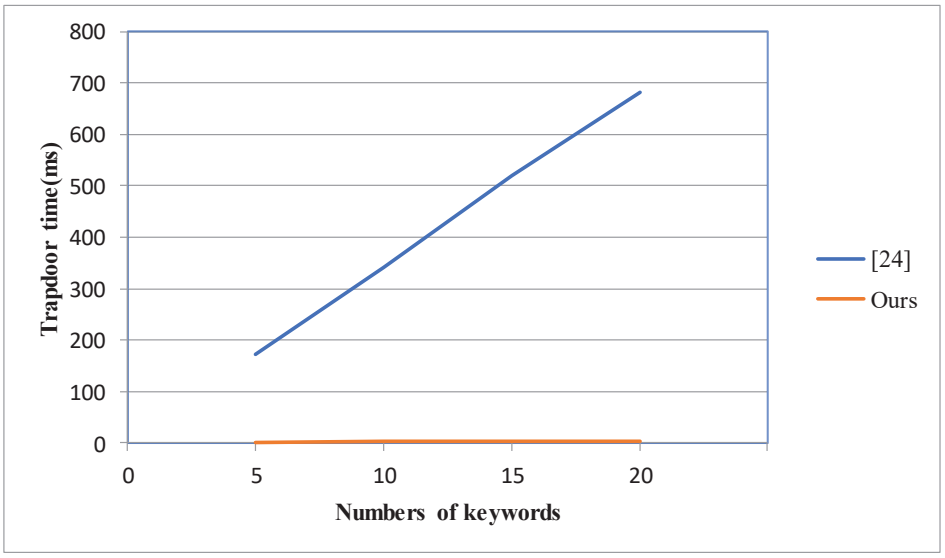


**Figure 5.** The trapdoor generation times of ours and [24].

Figure 4 shows a comparison of the time cost of *PEKS* algorithms. Since [24] is a single-keyword scheme, the time to generate ciphertexts is linearly related to the number of keywords. When the number of keywords is 20, the time taken reaches 243.65 ms. In contrast, the time cost of our ciphertext generation is only 2.739 ms, and is a figure independent of the number of keywords.

Figure 5 represents the performance comparison of the *Trapdoor* algorithm. The time cost in [24] is linearly related to the number of keywords and is approximately 681.503 ms when the number of keywords is 20. In contrast, our time cost is constant, and measures only about 4 ms.

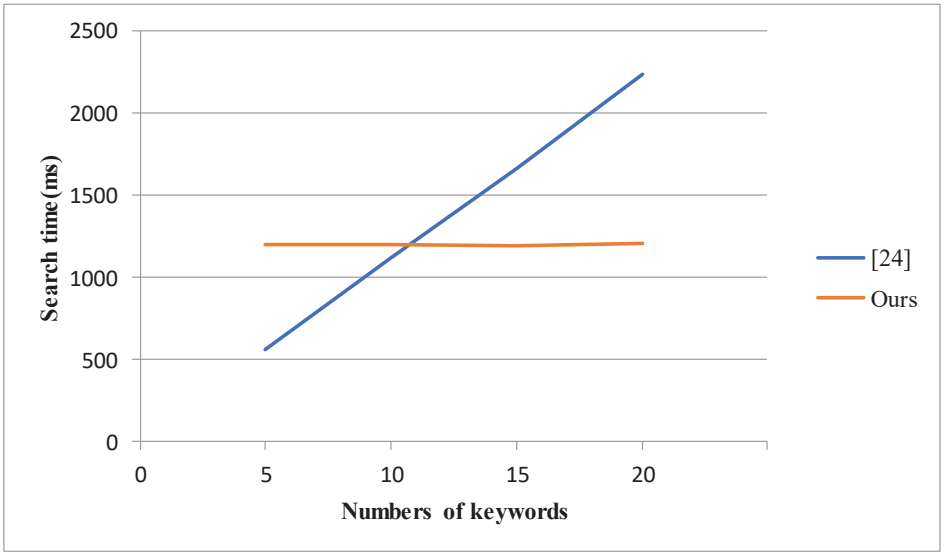


**Figure 6.** The search operation times of ours and [24].

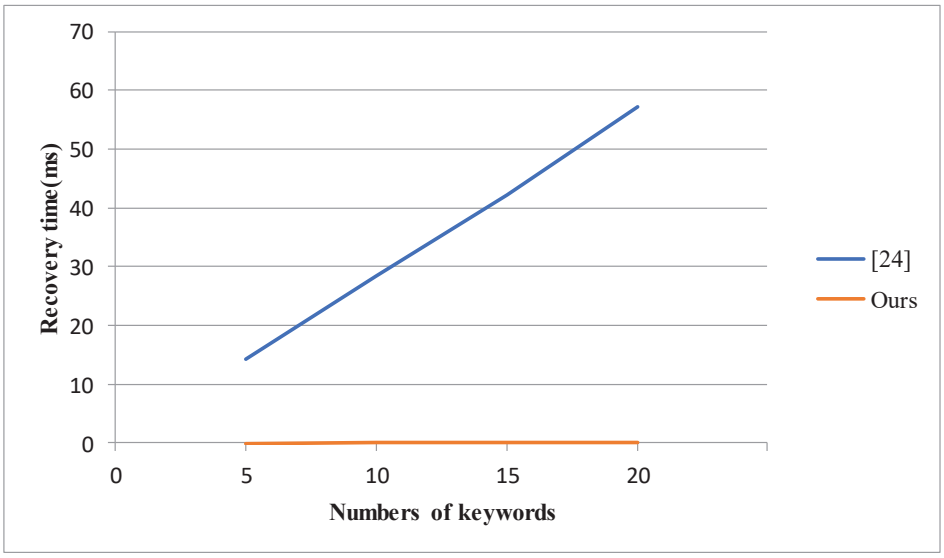


**Figure 7.** The keyword recovery times of ours and [24].

Figure 6 shows a comparison of the *Test* algorithm. It can be seen that the time cost of [24] is superior to ours when the number of keywords is relatively small, which is due to the fact that our *test* algorithm requires multiple rounds of interaction between the MS and the KFS, and the network latency affects the time consumption to a great extent. However, the performance of [24] increases with the number of keywords, and its time cost is 2234.265 ms at a keyword count of 20. In comparison, ours is about 1200 ms, and again is independent of the keyword count.

Figure 7 shows the performance comparison of the *OpenKeyword* algorithm. The time costs of [24] are 14.173 ms, 28.47 ms, 42.119 ms, and 57.199 ms for keyword counts of 5, 10, 15, and 20, respectively. As a comparison, our time cost is a constant value of about 0.03 ms.

### *6.2. Theoretical Analysis*

We show some feature comparisons between KFA-PEKS and the existing schemes in Table 2. In past research, SE schemes have been classified into four types as follows. The single-writer/single-reader (SW/SR) setting [16] is often SSE, wherein the writer and the reader act as one actor, i.e., only the writer is allowed to initiate the query. The multi-writer/single-reader (MW/SR) setting [4,24,36] is a scheme in which multiple writers can generate searchable ciphertexts for querying by a specific reader, as opposed to the single-writer/multi-reader (SW/MR) setting [18]. The multi-writer/multi-reader (MW/MR) setting [22] allows multiple users to encrypt and upload data while being able to search all users' stored encrypted data.

In the single-keyword schemes [4,24,36], the reader needs to generate multiple different query trapdoors for multiple different query keywords, i.e., the number of trapdoors is linearly related to the number of keywords. Similarly, the number of searchable ciphertexts is influenced by the number of keywords in the relevant document. The size of the trapdoor and searchable ciphertext are significant factors affecting the efficiency of keyword-searchable encryption schemes, compared to single-keyword schemes; this is related to the $|W_{id}|$ in [16]. In [18], the writer establishes the corresponding indexes for the documents to be uploaded (and the keywords contained in them) in advance, so the size of ciphertext is linearly related to $|W_{id}|$. Similarly, the search trapdoors generated by the reader when making a query are all possible indexing relationships between the query keywords and associated documents, so the number of trapdoors is linearly related to both $|DS|$ and $|SQ|$. Both the trapdoors and ciphertexts of our study and of [22] are homomorphic encryptions of a decimal integer of constant size.

**Table 2.** The feature and theoretical comparisons between our scheme and the existing schemes.

| | Multi-User | KGA Resilience | Keyword Recovery | CipNum [1] | TrapNum [2] | CipSize [3] | TrapSize [4] |
|---|---|---|---|---|---|---|---|
| [4] | × | × | × | $\sum_{\omega \in W} \lvert DS[\omega] \rvert$ | $\lvert SQ \rvert$ | $\mathcal{O}\lvert W_{id} \rvert$ | $\mathcal{O}\lvert SQ \rvert$ |
| [22] | √ | √ | × | $\lvert DS \rvert$ | 1 | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ |
| [24] | × | × | √ | $\sum_{\omega \in W} \lvert DS[\omega] \rvert$ | $\lvert SQ \rvert$ | $\mathcal{O}\lvert W_{id} \rvert$ | $\mathcal{O}\lvert SQ \rvert$ |
| [36] | × | √ | × | $\sum_{\omega \in W} \lvert DS[\omega] \rvert$ | $\lvert SQ \rvert$ | $\mathcal{O}\lvert W_{id} \rvert$ | $\mathcal{O}\lvert SQ \rvert$ |
| [16] | × | √ | × | $(\lvert DS \rvert)$ | 1 | $\mathcal{O}\lvert W_{id} \rvert$ | $\mathcal{O}\lvert W_{id} \rvert$ |
| [18] | × | √ | × | $\sum_{\omega \in W} \lvert DS[\omega] \rvert$ | $\lvert SQ \rvert$ | $\mathcal{O}\lvert W_{id} \rvert$ | $\mathcal{O}\lvert DS \rvert * \lvert SQ \rvert$ |
| ours | √ | √ | √ | $\lvert DS \rvert$ | 1 | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ |

[1] The number of searchable ciphertexts. [2] The number of trapdoors generated when a data owner executes a query $SQ$. [3] The size of the searchable ciphertext corresponding to a document. [4] The trapdoor size corresponding to the $SQ$ operation.

In PEKS, the searchable ciphertext is processed by the writer using the public key of the target reader for the query keyword; therefore, the attacker can perform KGA on the query trapdoor by forging the ciphertext. Ref. [16] is an SSE scheme in which the generation of the ciphertext and trapdoor require the user's key, so the attacker cannot generate the test ciphertext used to perform KGA. The reader in [36] shares a secret value with the writer, and the attacker cannot generate a test ciphertext to execute the KGA since they do not know the secret value. Ref. [18] is an SW/MR setting, which means that only the writer can generate the ciphertext. Our scheme and that of [22] use the partial private keys of the MS and KFS as the input for the *Test* algorithm, which means that the attacker is unable to execute the *Test* algorithm freely. Furthermore, among the above schemes, only ours and that of [24] can complete a keyword frequency analysis of the query keywords. In conclusion, the performance of our solution exceeds that of related works.

## 7. Conclusions

In this paper, we propose a PEKS scheme with secure keyword frequency analysis. Within our scheme, multiple DOs can use their public keys to encrypt their data and upload it to the MS. Then, multiple DUs can independently generate trapdoors that support a multi-keyword search for data retrieval, without interacting with the DO. The entire solution adopts a dual-server architecture, which can effectively resist KGA by internal attackers. After reaching the time node *TSO*, as specified by the DU, the KFS is able to recover the keyword information in the trapdoor for keyword frequency analysis, and cannot link the recovered keywords with their corresponding data queriers. Although the generation and utilization of a single time token are efficient for a time server, generating a common time token for all data queriers and reducing the traffic between the MS and KFS remains the direction of our future research.

**Author Contributions:** Conceptualization, Y.Z. and Y.L.; methodology, Y.Z.; software, Y.Z.; validation, B.S., D.Z. and Y.Z.; formal analysis, Y.Z., Y.L. and C.W.; investigation, Y.Z. and Y.L.; resources, Y.Z. and D.Z.; data curation, Y.Z. and B.S.; writing—original draft preparation, Y.Z.; writing—review and editing, D.Z., C.W. and B.S.; visualization, Y.Z. and Y.L.; supervision, D.Z. and C.W.; project administration, D.Z. and C.W.; funding acquisition, D.Z. and C.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** This paper was funded by the National Natural Science Foundation of China (Nos. U2001205, 61732021, 61932010), Guangdong Basic and Applied Basic Research Foundation (Nos. 2019B030302008, 2023B1515040020), Guangdong Provincial Key Laboratory of Power System Network Security (No. GPKLPSNS-2022-KF-05), and TESTBED2 (No. H2020-MSCA-RISE-2019).

**Informed Consent Statement:** Informed consent was obtained from all subjects involved in the study.

**Data Availability Statement:** All data are presented in the main text.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References


1. Lv, Z.; Singh, A.K. Big Data Analysis of Internet of Things System. *ACM Trans. Internet Technol.* **2021**, *21*, 28:1–28:15. [CrossRef]
2. Li, X.; Liu, H.; Wang, W.; Zheng, Y.; Lv, H.; Lv, Z. Big data analysis of the Internet of Things in the digital twins of smart city based on deep learning. *Future Gener. Comput. Syst.* **2022**, *128*, 167–177. [CrossRef]
3. Kamara, S.; Lauter, K.E. Cryptographic Cloud Storage. In Proceedings of the Financial Cryptography and Data Security, FC 2010 Workshops, RLCPS, WECSR, and WLC 2010, Tenerife, Canary Islands, Spain, 25–28 January 2010; Revised Selected Papers; Sion, R., Curtmola, R., Dietrich, S., Kiayias, A., Miret, J.M., Sako, K., Sebé, F., Eds.; Springer: Cham, Switzerland, 2010; Volume 6054, pp. 136–149. [CrossRef]
4. Boneh, D.; Crescenzo, G.D.; Ostrovsky, R.; Persiano, G. Public Key Encryption with Keyword Search. In Proceedings of the Advances in Cryptology-EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, 2–6 May 2004; Springer: Cham, Switzerland, 2004; Volume 3027, pp. 506–522. [CrossRef]
5. Liu, J.; Wu, M.; Sun, R.; Du, X.; Guizani, M. BMDS: A Blockchain-based Medical Data Sharing Scheme with Attribute-Based Searchable Encryption. In Proceedings of the ICC 2021-IEEE International Conference on Communications, Montreal, QC, Canada, 14–23 June 2021; pp. 1–6. [CrossRef]
6. Li, H.; Yang, Y.; Dai, Y.; Yu, S.; Xiang, Y. Achieving Secure and Efficient Dynamic Searchable Symmetric Encryption over Medical Cloud Data. *IEEE Trans. Cloud Comput.* **2020**, *8*, 484–494. [CrossRef]
7. Liu, P.; Liu, K.; Fu, T.; Zhang, Y.; Hu, J. A privacy-preserving resource trading scheme for Cloud Manufacturing with edge-PLCs in IIoT. *J. Syst. Archit.* **2021**, *117*, 102104. [CrossRef]
8. Song, D.X.; Wagner, D.; Perrig, A. Practical techniques for searches on encrypted data. In Proceedings of the Proceeding 2000 IEEE Symposium on Security and Privacy, S&P 2000, IEEE, Berkeley, CA, USA, 14–17 May 2000; pp. 44–55.
9. Curtmola, R.; Garay, J.A.; Kamara, S.; Ostrovsky, R. Searchable symmetric encryption: Improved definitions and efficient constructions. In Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, 30 October–3 November 2006; Juels, A., Wright, R.N., di Vimercati, S.D.C., Eds.; ACM: New York, NY, USA, 2006; pp. 79–88. [CrossRef]
10. Moataz, T.; Shikfa, A. Boolean symmetric searchable encryption. In Proceedings of the 8th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '13, Hangzhou, China, 8–10 May 2013; Chen, K., Xie, Q., Qiu, W., Li, N., Tzeng, W., Eds.; ACM: New York, NY, USA, 2013; pp. 265–276. [CrossRef]
11. Cash, D.; Jarecki, S.; Jutla, C.; Krawczyk, H.; Roşu, M.C.; Steiner, M. Highly-scalable searchable symmetric encryption with support for boolean queries. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2013; Springer: Cham, Switzerland, 2013; pp. 353–373.
12. Rhee, H.S.; Park, J.H.; Susilo, W.; Lee, D.H. Improved searchable public key encryption with designated tester. In Proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009, Sydney, Australia, 10–12 March 2009; Li, W., Susilo, W., Tupakula, U.K., Safavi-Naini, R., Varadharajan, V., Eds.; ACM: New York, NY, USA, 2009; pp. 376–379. [CrossRef]
13. Lu, Y.; Li, J. Efficient searchable public key encryption against keyword guessing attacks for cloud-based EMR systems. *Clust. Comput.* **2019**, *22*, 285–299. [CrossRef]
14. Senouci, M.R.; Benkhaddra, I.; Senouci, A.; Li, F. An efficient and secure certificateless searchable encryption scheme against keyword guessing attacks. *J. Syst. Archit.* **2021**, *119*, 102271. [CrossRef]
15. Gu, X.; Wang, Z.; Fu, M.; Ren, P. A Certificateless Searchable Public Key Encryption Scheme for Multiple Receivers. In Proceedings of the 2021 IEEE International Conference on Web Services, ICWS 2021, Chicago, IL, USA, 5–10 September 2021; Chang, C.K., Daminai, E., Fan, J., Ghodous, P., Maximilien, M., Wang, Z., Ward, R., Zhang, J., Eds.; IEEE: Piscataway, NJ, USA, 2021; pp. 635–641. [CrossRef]
16. Wang, P.; Wang, H.; Pieprzyk, J. Keyword field-free conjunctive keyword searches on encrypted data and extension for dynamic groups. In Proceedings of the International Conference on Cryptology and Network Security, Hong Kong, China, 2–4 December 2008; Springer: Cham, Switzerland, 2008; pp. 178–195.
17. Zhang, B.; Zhang, F. An efficient public key encryption with conjunctive-subset keywords search. *J. Netw. Comput. Appl.* **2011**, *34*, 262–267. [CrossRef]
18. Sun, S.; Liu, J.K.; Sakzad, A.; Steinfeld, R.; Yuen, T.H. An Efficient Non-interactive Multi-client Searchable Encryption with Support for Boolean Queries. In Proceedings of the Computer Security-ESORICS 2016 - 21st European Symposium on Research in Computer Security, Heraklion, Greece, 26–30 September 2016; Proceedings, Part I; Askoxylakis, I.G., Ioannidis, S., Katsikas, S.K., Meadows, C.A., Eds.; Springer: Cham, Switzerland, 2016; Volume 9878, pp. 154–172. [CrossRef]
19. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-policy attribute-based encryption. In Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP'07), IEEE, Berkeley, CA, USA, 20–23 May 2007; pp. 321–334.
20. Xu, L.; Yuan, X.; Steinfeld, R.; Wang, C.; Xu, C. Multi-writer searchable encryption: An LWE-based realization and implementation. In Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, Auckland, New Zealand, 9–12 July 2019; pp. 122–133.

21. Camenisch, J.; Kohlweiss, M.; Rial, A.; Sheedy, C. Blind and anonymous identity-based encryption and authorised private searches on public key encrypted data. In Proceedings of the International Workshop on Public Key Cryptography, Irvine, CA, USA, 18–20 March 2009; pp. 196–214.
22. Liu, X.; Yang, G.; Susilo, W.; Tonien, J.; Liu, X.; Shen, J. Privacy-preserving multi-keyword searchable encryption for distributed systems. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *32*, 561–574. [CrossRef]
23. Liu, X.; Deng, R.H.; Choo, K.K.R.; Weng, J. An efficient privacy-preserving outsourced calculation toolkit with multiple keys. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 2401–2414. [CrossRef]
24. Xu, L.; Li, W.; Zhang, F.; Cheng, R.; Tang, S. Authorized keyword searches on public key encrypted data with time controlled keyword privacy. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 2096–2109. [CrossRef]
25. Byun, J.W.; Rhee, H.S.; Park, H.A.; Lee, D.H. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In Proceedings of the Workshop on Secure Data Management, Seoul, Korea, 10–11 September 2006; pp. 75–83.
26. Yau, W.C.; Heng, S.H.; Goi, B.M. 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, Oslo, Norway, 23–25 June 2008; pp. 100–105.
27. Huang, Q.; Li, H. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. *Inf. Sci.* **2017**, *403*, 1–14. [CrossRef]
28. Qin, B.; Chen, Y.; Huang, Q.; Liu, X.; Zheng, D. Public-key authenticated encryption with keyword search revisited: Security model and constructions. *Inf. Sci.* **2020**, *516*, 515–528. [CrossRef]
29. He, D.; Ma, M.; Zeadally, S.; Kumar, N.; Liang, K. Certificateless Public Key Authenticated Encryption With Keyword Search for Industrial Internet of Things. *IEEE Trans. Ind. Inf.* **2018**, *14*, 3618–3627. [CrossRef]
30. Xu, P.; Jin, H.; Wu, Q.; Wang, W. Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack. *IEEE Trans. Comput.* **2012**, *62*, 2266–2277. [CrossRef]
31. Ghareh Chamani, J.; Papadopoulos, D.; Papamanthou, C.; Jalili, R. New constructions for forward and backward private symmetric searchable encryption. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1038–1055.
32. Samanthula, B.K.; Chun, H.; Jiang, W. An efficient and probabilistic secure bit-decomposition. In Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, Hangzhou, China, 8–10 May 2013; pp. 541–546.
33. Kamara, S.; Mohassel, P.; Raykova, M. Outsourcing Multi-Party Computation. Available online: https://eprint.iacr.org/2011/272 (accessed on 25 May 2023).
34. Bresson, E.; Catalano, D.; Pointcheval, D. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In Proceedings of the Advances in Cryptology-ASIACRYPT 2003: 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, 30 November–4 December 2003; Proceedings 9; Springer: Cham, Switzerland, 2003; pp. 37–54.
35. Akinyele, J.; Green, M.; Rubin, A. Charm-Crypto Framework. Available online: https://eprint.iacr.org/2011/617 (accessed on accessed on 25 May 2023).
36. Zheng, Y.; Xu, P.; Wang, W.; Chen, T.; Susilo, W.; Liang, K.; Jin, H. DEKS: A Secure Cloud-Based Searchable Service Can Make Attackers Pay. In Proceedings of the Computer Security-ESORICS 2022-27th European Symposium on Research in Computer Security, Copenhagen, Denmark, 26–30 September 2022; Proceedings, Part II; Atluri, V., Pietro, R.D., Jensen, C.D., Meng, W., Eds.; Springer: Cham, Switzerland, 2022; Volume 13555, pp. 86–104. [CrossRef]