Secure and Efficient Searchable Public Key Encryption for Resource Constrained Environment Based on Pairings under Prime Order Group

Yu Zhang,1 Yin Li,1 and Yifan Wang2

1School of Computer and Information Technology, Xinyang Normal University, Xinyang 464000, China
2Wayne State University, 42 W Warren Ave, Detroit, MI 48202, USA

Correspondence should be addressed to Yu Zhang; willow1223@126.com

1. Introduction

In the era of big data, more and more companies and individuals are willing to share their data over the cloud platform. If the data is stored in plaintext form, the cloud service providers can easily access and analyze the data illegally, which may lead to a big threaten to some sensitive data, e.g., electronic health and personal credit records. To protect data privacy, encrypting data before outsourcing it to the cloud server is a common approach. But, this brings a new issue of how to perform keyword search over these unreadable ciphertext. A straightforward solution is to download all the encrypted data to the client and decrypt it. After obtaining all unencrypted data, users can utilize the common information retrieval technique used on the plaintext to perform keywords query. However, this approach requires a very high communication, storage, and computing costs, which are not applicable for large scale networks. This issue put forward a new challenge: how to effectively search for encrypted data without decrypting it first.

Searchable encryption (SE) is a promising method to address this problem. SE can be classified into two categories according to its applications: searchable symmetric key encryption (SSE) and searchable public key encryption (SPE). Based on many researchers’ work these years, SSE can support advanced query conditions like Boolean keywords search, multikey words rank search, and top-k similarity keywords search. Works [1–3] dealt with multikeywords search, where the query condition contains more than one keyword. Considering the fact that it is not practical to return all the search results, more recent works like [4–6] focused on the multikeywords rank search. Sometimes, users want to search documents related to their query keywords rather than the documents that precisely match those keywords. The SSE schemes supporting similarity keywords search were created to address this issue [7, 8].

Conversely, SPE is difficult to support advanced search function without sacrificing security as anyone who has the public key (pk) can create encrypted index or ciphertext, and easily perform chosen keywords attack. In addition, the SSE scheme is generally more efficient than the SPE scheme.
Although the SPE scheme has these weaknesses, it still holds some advantages for many applications. Specifically, for the application in which there are many data senders and only one data receiver (Figure 1), SSE needs that all senders and the receiver hold the same symmetric key to generate the encrypted data. So, if one of the senders is compromised by an adversary, then the data for all senders will be leaked. Compared with SSE, in SPE, the public key can be accessed by all senders, and the private key is only given to the receiver. Thus, SPE can naturally solve the security problem of key management in this application [9] and is an irreplaceable solution.

The first SPE named as public key encryption with keyword search (PEKS) was introduced in [10], in which senders who hold $pk$ can send their encrypted documents with encrypted indexes to a server, and the receiver holding secret key ($sk$) can make a keyword search over the encrypted data. However, this scheme only supports a single keyword search. How to construct a public key encryption scheme supporting a complex query conditions such as the Boolean keyword search is a significant problem. Specifically, as two typical Boolean keyword search schemes, conjunctive and disjunctive keyword search over encrypted data in the public key setting are needed to be developed.

Park et al. [11] first proposed the public key encryption with conjunctive keyword search (PECK) scheme that supports keywords search like $w_1 \land w_2 \land w_3$, where $w_1, w_2,$ and $w_3$ are query words. After that, many PECK schemes [12, 13] were proposed to improve the efficiency and security. The first public key encryption with disjunctive keyword search (PEDK) scheme was proposed by Katz et al. [14]. In their work, they proposed an inner product encryption (IPE) scheme and converted this IPE scheme into a PEDK scheme, which supports keywords search like $w_1 \lor w_2 \lor w_3$. In an IPE scheme, the secret key corresponding to a predicate vector $\vec{v}$ can decrypt the ciphertext associated with an attribute vector $\vec{x}$ if and only if $\vec{x} \cdot \vec{v} = 0$. According to this property, we know that, by converting index and query keyword set into an attribute and a predicate vector, respectively, an IPE scheme can be changed into a SPE scheme supporting multi-keyword search.

In the practical cloud environment, the user usually perform conjunctive and disjunctive keywords search simultaneously. In [15], Zhang and Lu proposed an approach of converting an IPE scheme into a public key encryption with conjunctive and disjunctive keyword search (PECDK) scheme and gave a concrete scheme. In their scheme, the size of each document’s index, the size of a trapdoor, and the time cost of pairing operations in the test process are all $O(N)$. Since $N$ is the number of keywords in a dictionary, the previous PECDK scheme is impractical if $N$ is a large integer. Inspired by this problem, we proposed a new approach to build an efficient PECDK scheme with less time and space overhead [16]. However, these two PECDK schemes are based on the IPE scheme [17], which is a costly cryptographic prototype. In this paper, we aim to build an efficient PECDK scheme using an alternative method.

Main Works. In the scheme introduced in [17], the user first converts an index and a query keyword set into an attribute and a predicate vector, respectively. After this, the user can construct encrypted index and trapdoor by applying the attribute and predicate vectors to the encryption and key generation algorithms of IPE. The keyword search process can be executed by adopting the decryption algorithm of IPE. However, this scheme is constructed by utilizing many group operations due to being based on the IPE scheme. The
IPE scheme adopts a technique called dual pairing vector space (DPVS), which generates \( n^2 \) group elements to create the public key and secret key. By using these keys, both the storage size of the obtained indices and the time cost of the search process are linear with \( O(n^2) \). To address this issue, we aim to create an efficient PECDK scheme without using IPE as the infrastructure. We take advantage of the conversion method similar to the one introduced in [16] for creating a group of vectors. Then, by applying these vectors to the prime order pairing groups, a PECDK scheme with a better time and space complexity than the previous schemes is presented. Security of our construction is based on decision linear Diffie-Hellman (DLIN) assumption in the random oracle model. In addition, we design an experiment that verifies the efficiency of the proposed scheme and give a detailed comparison with the previous scheme. The experiment result shows that the key generation, index building, trapdoor generation, and testing algorithms in our scheme are more efficient than those in the previous one. Besides, the space cost for indices in our scheme is also less than that in the previous one. In practice, client device, e.g., a mobile device, has less storage space and limited computation capacity. Thus, compared with the previous scheme, ours is much more suitable for the resource constrained environment.

Related Work. There are two classes of searchable encryption schemes in terms of different cryptography primitives: searchable symmetric key encryption (SSE) and searchable public key encryption (SPE).

SSE. Song et al. [18] introduced the definition of searching on encrypted data and gave a concrete scheme. Then, Goh [19] defined a formal security definition for SSE and presented an effective conjunctive keywords search scheme by using a technique called Bloom Filter. Soon afterward, many works [20, 21] were constructed to improve the search efficiency and security level. These schemes failed to achieve a sublinear search speed. To address this issue, by taking advantage of tree structures, such as r-tree and kd-tree, SSE schemes with sublinear search time were released in [22, 23]. The SSE scheme supporting Boolean keywords search with sublinear search time was given in [24]. In order to support queries with rich expressiveness, SSE schemes supporting media data and similarity keywords search are introduced in [25, 26]. More important, these works also support dynamic update. In practical application, the search engine should return the top-\( k \) related documents to users instead of returning all the search results. Some SSE schemes, supporting rank search, are proposed in [4–6] to obtain more accurate query results.

SPE. The first SPE scheme is called public key encryption with keyword search (PEKS), which was proposed by Boneh et al. [10]. Then, Abdalla et al. [27] defined the computational and statistical consistency in PEKS and presented a statistically consistent scheme. However, this work only supports a single keyword search. The concept of PECK was first proposed in [11]. They defined the security model of this mechanism and gave two constructions. One requires many bilinear pairing operations, while the other needs more private keys. Hwang and Lee [28] designed a more efficient scheme in a multiple users setting. However, all these constructions use the keyword field as additional information for keyword search. The keyword field may leak more information to the public. To eliminate the keyword field, Boneh and Waters [29] presented a public key encryption scheme called hidden vector encryption (HVE), which supports a conjunctive search, comparison queries (\( x \geq a \)), and subset queries \( (x \in S) \) on encrypted data.

In order to create a SPE scheme supporting disjunctive keyword search, Katz et al. [14] proposed a public key encryption scheme called inner product encryption (IPE), which is related to the attribute-based encryption (ABE) [30, 31]. Recently, in order to better utilize the encrypted data and improve the security, some public key schemes which can support verifiable attribute-based search over outsourced encrypted data are proposed in [32–37]. For supporting advanced keywords search function, Zhang and Lu proposed the first PECDK scheme [15] by combining an efficient IPE scheme [17] and a keyword conversion method. However, their scheme is a costly solution due to using an IPE scheme. In fact, the PECDK scheme based on an IPE scheme needs more computation and storage cost than the one without utilizing an IPE scheme as a foundation. Therefore, in this paper, we will find a new method irrelevant to IPE to construct a more efficient PECDK scheme.

Organization. The remainder of the paper is organized as follows: The model and security definitions of PECDK are defined in Section 2. In Section 3, we propose our PECDK scheme and give the security proof for the scheme. The analysis of the presented PECDK scheme is described in Section 4. Section 5 presents some conclusions.

2. Preliminaries

In this section, we will give a formal definition of the PECDK model and the corresponding security model. Our scheme is based on this model and proven to be secure under this security model. Moreover, we also briefly introduce some basic ingredients used to create our scheme, including the bilinear pairing and the complexity assumption.

2.1. Model of PECDK. In PECDK, let \( pk, sk \) be the receiver's public key and secret key. When a sender wants to send a message \( M \) to a receiver with keywords \( w_1, w_2, \ldots, w_n \), she uses the standard public key encryption system to encrypt \( M \) and uses \( pk \) and keywords \( w_1, w_2, \ldots, w_n \) to construct the encrypted index of \( M \). After that, she sends the encrypted message \( M \) and the encrypted index of \( M \) to the server. When the receiver wants to retrieve messages including a specific list of keywords, he takes the \( sk \), a symbol, and keyword set \( Q \) as input to construct a trapdoor \( T_Q \) and sends \( T_Q \) to the server. Note that the symbol is used to refer to the type of query, e.g., conjunctive form or disjunctive form. When the server receives the trapdoor, it tests each encrypted index by using the trapdoor and returns the matched messages to the receiver.
According to the description above, we give the model of PECDK based on the previous definition introduced in [16].

**Definition 1.** There are four probabilistic polynomial time algorithms in the PECDK scheme. There are KenGen, IndexBuild, Trapdoor, and Test:

1. **KeyGen(1^y):** choosing a security parameter y, this algorithm outputs the key pair (pk, sk) in which pk is the public key and sk is the secret key.
2. **IndexBuild(pk, W):** the algorithm is performed by the sender to produce an encrypted index I_W by using a keyword set W = \{w_1, w_2, ..., w_n\} and pk.
3. **Trapdoor(sk, |Q, sym|):** the receiver runs this algorithm to produce a trapdoor. It takes sk, sym ∈ \{∨, ∧\}, and the keyword query Q = \{q_1, q_2, ..., q_m\} as input to generate a trapdoor T_Q, where m ≤ n. ∨ and ∧ represent disjunctive and conjunctive keywords search, respectively.
4. **Test(pk, T_Q, I_W):** it takes the trapdoor T_Q = \{t_Q, sym\}, the secure index I_W, and the public key pk as input. If sym is ∨, it means that the trapdoor is for conjunctive keywords search. In this case, the output is 1 if Q ⊆ W, or 0 otherwise. If sym is ∧, disjunctive keywords search should be performed. In this case, if Q ∩ W ≠ ∅, it outputs 1, or 0 otherwise.

**Correctness Property.** Let |Q, sym| and W be a query and a keyword set, respectively. Suppose that pk, sk, I_W, and T_Q are correctly generated by KeyGen(y), IndexBuild(pk, W), and Trapdoor(sk, |Q, sym|), respectively. The correctness property of PECDK involves two situations described as follows.

1. When there is a conjunctive keywords search (sym is ∧), Test(pk, T_Q, I_W) = 1 holds if Q ⊆ W. Otherwise, it holds with negligible probability.
2. When there is a disjunctive keywords search (sym is ∨), Test(pk, T_Q, I_W) = 1 holds if Q ∩ W ≠ ∅. Otherwise, it holds with negligible probability.

In practical use, if a sender would like to send a message M with keyword set W, a ciphertext with the form of \{Enc(pk, M), IndexBuild(pk, W)\} will be created, where Enc(·) is a frequently used public key encryption scheme, e.g., RSA. Similar with other related works [10, 11, 28], the proposal only concentrate on searchable encryption part, which is PECDK in this paper.

**2.2. Security Definition of PECDK.** Since previous PECDK schemes are based on the IPE scheme, the security definition also depends on the security of IPE. For the PECDK scheme without using IPE, we need to give a new security definition. Inspired by the previous security definition called indistinguishability of ciphertext from random (IND-CCR-CKA) introduced in [28], we define a security game under the PECDK model.

The security game is described as follows.

1. Setup: the challenger C runs the KeyGen(1^y) algorithm to generate the public key pk and the secret key sk. Then, C gives pk to the attacker A.
2. Phase 1: the attacker A can adaptively ask the challenger C for the trapdoor T_Q for any query Q of his choice.
3. Challenge: A selects a target keyword set W^* and sends it to C. After receiving W^*, C selects a random keyword set R and sets W^0 = W^* and W^1 = R. Suppose that \{Q_1, sym_1\}, \{Q_2, sym_2\}, ..., \{Q_t, sym_t\} are queries which are asked to construct trapdoors in Phase 1, for each i ∈ [1, t], the only restriction is that |Q_i ∩ W^0| = 0 = |Q_i ∩ W^1|, where the number of the keywords in Q_i ∩ W^0 and Q_i ∩ W^1 is denoted by |Q_i ∩ W^0| and |Q_i ∩ W^1|, respectively. Then, C picks a random bit \(\beta \in \{0, 1\}\) produces I_\beta = IndexBuild(pk, W^\beta), and sends \{I_\beta, W^0, W^1\} to A.
4. Phase 2: A can continue to ask for trapdoor T_Q for any query Q of his choice, and subject to the same restriction as before.
5. Response: A outputs \(\beta' \in \{0, 1\}\) and wins the game if \(\beta' = \beta\).

A’s advantage can be expressed as a function of the security parameter (1^y):

\[
Adv_A^{Game}(1^y) = \Pr[\beta' = \beta] - \frac{1}{2}
\]  

(1)

Then, we have the following definition.

**Definition 2.** According to the game above, a PECDK is semantic secure if, for any probabilistic polynomial time attacker A, the function \(Adv_A^{Game}(1^y)\) is negligible.

**2.3. Bilinear Pairings.** Let G_1, G_2 be two cyclic groups of prime order q. There are three properties in the bilinear pairings map \(\hat{e}: G_1 \times G_1 \rightarrow G_2\).

1. Bilinear: \(\hat{e}(a^u, b^v) = \hat{e}(a, b)^{uv}\), where a, b ∈ G_1 and u, v ∈ \(Z_q^*\).
2. Nondegenerate: if g ∈ G_1, then \(\hat{e}(g, g) \in G_2\).
3. Computable: for any a, b ∈ G_1, \(\hat{e}(a, b)\) can be efficiently computable.

An efficient bilinear map can be obtained by applying the Weil pairing or the Tate pairing [38].

**2.4. Complexity Assumption.** We review the complexity assumption named Decision I Bilinear Diffie-Hellman Inversion (Decision I-BDHI) assumption [11], which is related to the bilinear pairing. The security of our scheme is based on this assumption.
Decision l-BDHI Assumption: Given a tuple $D = \{g, g^x, g^{x^2}, \ldots, g^{x^l}\}$ with the random choice of $x \in Z^*_p$ and $g \in G_1$, it is argued that an algorithm $C$ outputting $b \in \{0, 1\}$ has advantage $\varepsilon$ in solving the decision l-BDHI problem in $G_1$ if

$$\text{Adv}^D_{l-\text{BDHI}}$$

$$= \left| \Pr [B(D, \widetilde{g}(g, g)^{y/x}) = 1] - \Pr [B(D, R) = 1] \right| \geq \varepsilon$$

where the probability $(\Pr[\cdot])$ is over the random choice $R \in G_2$ and random bits chosen by $C$.

Definition 3. The Decision l-BDHI assumption holds in $G_1$, if no probabilistic polynomial time algorithm has advantage at least $\varepsilon$ in solving the Decision l-BDHI problem in $G_1$.

3. PECDK Scheme

Based on previous definitions and notations, in this section, we give our PECDK scheme without using IPE. Our scheme mainly involves three steps. Initially, we utilize a keyword conversion method inspired by the approach introduced in [16] to convert the index and query keywords into a set of vectors. Moreover, by applying the bilinear pairing under the prime order group, we encrypt these vectors to construct encrypted index and trapdoor. Eventually, a test algorithm is achieved. Specifically, for the conjunctive keyword search, the matched documents are returned if and only if all the keywords in the query are included in the index keyword set. For the disjunctive logic query, the matched documents are returned if and only if the intersection between the index and the query keyword set is not an empty set.

3.1. Keywords Conversion Method. The keywords in query and index are strings; thus we need to change all the keywords to a group of integers. Moreover, in order to verify whether the query matches the index or not, the integers between the query and index must hold some relationships. According to these concerns, we introduce a method that can convert the index and query set into a matrix and a vector.

This method is based on the relationship between the roots and coefficients in an equation of degree $n$ with one unknown. Specifically, the roots and coefficients are used to create a matrix of index keywords and a vector of query keywords, respectively. By adopting a prime order group to encrypt the matrix and vector, an encrypted index and query can be built. The relationship between the matrix and vector can be utilized to construct the test algorithm.

Suppose that any keyword $w$ can be expressed as $\{0, 1\}^*$; we define a function $H_1 : \{0, 1\}^* \rightarrow Z^*_p$. Since $p$ is a large prime and is larger than the number of all words, $H_1$ can be collision-resistant. This means that, if $i \neq j$, then $H_1(w_i) \neq H_1(w_j)$, where $w_i$ and $w_j$ are two distinct keywords. Let $W = \{w_1, w_2, \ldots, w_n\}$ and $Q = \{q_1, q_2, \ldots, q_m\}$ be two keyword sets, where $m < n$. The method can be listed into three steps and described as follows.

1. For the keyword set $W = \{w_1, w_2, \ldots, w_n\}$, we construct a matrix for $W$:

   $\begin{pmatrix}
   H_1(w_1)^0 & H_1(w_1)^1 & \cdots & H_1(w_1)^n \\
   H_1(w_2)^0 & H_1(w_2)^1 & \cdots & H_1(w_2)^n \\
   \vdots & \vdots & \ddots & \vdots \\
   H_1(w_n)^0 & H_1(w_n)^1 & \cdots & H_1(w_n)^n
   \end{pmatrix}$

   Note that $H_1(w_1), H_1(w_2), \ldots, H_1(w_n)$ can be seen as the roots of equation:

   $f(x) = (x - H_1(w_1))(x - H_1(w_2))\cdots(x - H_1(w_n))$ (4)

2. For the query $\{q_1, q_2, \ldots, q_m\}$, we create an equation:

   $f(x) = (x - H_1(q_1))(x - H_1(q_2))\cdots(x - H_1(q_m))$

   $= a_m x^m + a_{m-1} x^{m-1} + \ldots + a_0 x^0$ (5)

According to the coefficient of the $f(x)$, a vector $\overrightarrow{a} = [a_0, a_1, \ldots, a_n]$ can be obtained.

3. Suppose that $\overrightarrow{w_i} = [H_1(w_i)^0, H_1(w_i)^1, \ldots, H_1(w_i)^m]$ is from the $i$-th row of the matrix (3), we can infer that if there is a keyword $w_i \in W$ such that $w_i \in Q$, where $i \in [1, m]$, according to (5), it is not difficult to verify that $\overrightarrow{a} \cdot \overrightarrow{w_i} = 0$. Based on this property, we can build the test algorithm.

3.2. Construction of PECDK. By combining the keyword conversion method and the bilinear pairing under prime order group, we propose a PECDK scheme which allows the user to perform conjunctive and disjunctive keyword search over encrypted data. Our key idea is as follows. First, by adopting the keyword conversion method introduced in Section 3.1, we change the index and query keyword set into an index matrix and a query vector, respectively. Then, by using a set of elements randomly chosen from the prime order group, we encrypt the index matrix to the secure index and the query vector to the trapdoor, respectively. Specifically, in order to guarantee the scheme’s security, the encryption process relies strictly on the complexity assumption given in Section 2.4. Finally, by taking advantage of the properties of the bilinear pairing, both conjunctive and disjunctive keywords search can be achieved. Specifically, for the conjunctive keywords search, the matched documents are returned if and only if all the keywords in the query are included in the index keyword set. For the disjunctive logic query, the matched documents are returned if and only if the intersection between the index and the query keyword set is not an empty set.

According to the above ideas, the concrete construction is proposed in the following paragraphs. Because of making use of the prime order group, the proposed scheme is more efficient than the previous scheme adopting the IPE cryptosystem on the time and space complexities. Experimental results showing the advantages of our scheme can be found in Section 4.
(i) KeyGen($1^λ$): given a security parameter $λ$, the algorithm generates two cyclic groups $G_1,G_2$ of prime order $q$ and a bilinear pairing $e : G_1 × G_1 → C$. It picks a random generator $g$ of $G_1$ and two hash functions $H_1 : \{0,1\}^* → Z_q^*$ and $H_2 : G_2 → \{0,1\}^{\log_q q}$. Choosing $2n + 3$ random numbers $α_0, α_1, ..., α_n, β_0, β_1, ..., β_n, θ ∈ Z_q^*$, it computes $X_0 = g^{α_0}, X_1 = g^{α_1}, ..., X_n = g^{α_n}, Y_0 = g^{β_0}, Y_1 = g^{β_1}, ..., Y_n = g^{β_n}, Z = g^{θ}, μ = e(g, g)$. After this, it outputs the public key $pk = \{X_0, X_1, ..., X_n, Y_0, Y_1, ..., Y_n, Z, μ, g, H_1, H_2, e\}$ and the secret key $sk = \{α_0, α_1, ..., α_n, β_0, β_1, ..., β_n, θ\}$, where $pk$ is also open to the public.

(ii) IndexBuild($pk, W$): the algorithm generates $n^2 + 2n$ random numbers $r_1, r_2, ..., r_n$ and $u_j ∈ Z_q^*$, where $i ∈ [1, n], j ∈ [0, n]$. Given a keyword set $W = \{w_1, w_2, ..., w_q\}$, it constructs the public key $pk$ for each word $w_i ∈ W$, it computes $A_{ij} = X_i^j × g^{r_i H_i(w_i) + u_j}, B_{ij} = Y_i^j, C_j = Z^j = g^{θ j}$, and $D_i = H_2(g^{θ i})$, where $i ∈ [1, n]$ and $j ∈ [0, n]$. The index of the keyword set $W$ is

$$I_W = \begin{pmatrix}
A_{10} & A_{11} & \cdots & A_{1n} & B_{10} & B_{11} & \cdots & B_{1n} & C_1 & D_1 \\
A_{20} & A_{21} & \cdots & A_{2n} & B_{20} & B_{21} & \cdots & B_{2n} & C_2 & D_2 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
A_{n0} & A_{n1} & \cdots & A_{nn} & B_{n0} & B_{n1} & \cdots & B_{nn} & C_n & D_n
\end{pmatrix}$$

Note that the secure index can be seen as an encrypted version of matrix (3).

(iii) Trapdoor($sk, \{Q, sym\}$): when the algorithm receives a keyword query $Q = \{q_1, q_2, ..., q_m\}$, where $m ≤ n$, it will construct a vector $\vec{d} = \{d_m, d_{m-1}, ..., d_0\}$ by utilizing (5). Given a vector $\vec{d}$ and $sk$, $t_1j = g^{α_j / (Σ^n j=0 α_j + Σ^n j=0 β_j)}$ and $t_2j = t_1j^{β_j}$ can be computed, where $u$ is a random number in $Z_q$ and $j ∈ [0, m]$. Let $t_3 = u$ and $sym ∈ \{1, n\}$, the trapdoor for the keyword query $Q$ is $T_Q = (t_1, t_{11}, ..., t_{1m}, t_{20}, t_{21}, ..., t_{2m}, t_{3}, sym)$.

(iv) Test($pk, T_Q, I_W$): when the algorithm receives a trapdoor $T_Q$ and a secure index $I_W$, it works as follows:

1. If the symbol in $T_Q$ is $\vee$,
   (a) The algorithm chooses a counter $i$ and sets $i = 1$.
   (b) If $i > n$, then it goes to step (c); otherwise it checks whether $H_2(t_{1j} \cdot test_{t_2}) = D_j$, where $test_{t_2} = Π^m_{i=0} e(t_{1j}, A_i C_i), test_{t_2} = Π^m_{i=0} e(t_{2j}, B_j)$. If so, the output algorithm 1 and ends. Otherwise, it sets $i = i + 1$ and goes to step (b).
   (c) The algorithm outputs 0 and ends.

2. If the symbol in $T_Q$ is $\wedge$,
   (a) The algorithm chooses two counters $i$ and $j$ and sets $i = 1$ and $j = 1$.
   (b) If $i > n$, then it goes to step (c); otherwise it tests whether $H_2(t_{1j} / test_{t_2}) = D_j$, where $test_{t_2} = Π^m_{i=0} e(t_{1j}, A_i C_i), test_{t_2} = Π^m_{i=0} e(t_{2j}, B_j)$. If so, then the algorithm sets $j = j + 1$. Otherwise, it does nothing. After that, it sets $i = i + 1$ and goes to step (b).
   (c) If $j = m$, the algorithm outputs 1 and ends. Otherwise, it outputs 0 and ends.

Correctness. The key idea of disjunctive keywords search is to determine whether there exists an $i ∈ [1, n]$ such that $w_i ∈ Q$. For the conjunctive keywords search, we need to check whether there are m keywords $w_{φ(1)}, w_{φ(2)}, ..., w_{φ(n)}$ such that $w_{φ(i)} ∈ Q$, where $φ(j) ∈ [1, n]$ and $j ∈ [1, m]$. According to these ideas, we know that the essence of test algorithm is to verify whether $w_i ∈ Q$. In the proposed scheme, it can be verified that $test_{t_1} / test_{t_2} = e(g, g)^{r_{φ(i)} Σ^n j=0 α_j H_i(w_j) / (Σ^n j=0 α_j + Σ^n j=0 β_j)}$. If $w_i ∈ Q$, there is $Σ^n j=0 α_j H_i(w_j) = 0$. Thus, it must be $test_{t_1} / test_{t_2} = e(g, g)^{r_{φ(i)} 1}$ and $H_2(t_{1j} / test_{t_2}) = D_j$. Based on this, we argue that our scheme is correct.

3.3. Security Analysis

Theorem 4. The proposed PECKD scheme is secure according to security definition in the random model if Decision $(q_T + 1)$-BDHI is hard.

Proof. Suppose that an algorithm $A$ has advantage $ε$ in breaking the PECKD under the security definition. Suppose $A$ makes at most $q_T$ trapdoor queries, we can build an algorithm $C$ which solves the decision $(q_T + 1)$-BDHI assumption with a probability at least $ε’ = ε(ε(q_T + 1)$, where $ε$ is the base of the natural logarithm, and $n$ is the number of keywords in an index. Let $g$ be a generator of $G_1$. Given $g, g^x, g^{x^2}, ..., g^{x^{2n}}$, $R$, if $R = e(g, g)^{1/x}$, the algorithm $C$ outputs 1,0 otherwise. The interaction between the algorithm $C$ and the algorithm $A$ is as follows:

(i) Setup: the algorithm $C$ works as follows:

1. The algorithm $C$ randomly chooses $q_T$ numbers $p_1, p_2, ..., p_{q_T} ∈ Z_q^*$ and returns $f(z) = Π^{q_T}_{i=1} (z + p_i) = Σ^{q_T}_{i=1} c_i z^i$.

2. $C$ computes $U = g^{f'(x)} = g^{Σ^{q_T}_{i=1} c_i x^i}, V = g^{f''(x)} = g^{Σ^{q_T}_{i=1} c_i x^{i+1}}$, and $f(z) = (1/(z + p_i)) f(z) = Σ^{q_T}_{i=1} d_i z^i$, where $i ∈ [1, q_T]$. Then, $C$ gets $U'/x + p_i) = g^{f'(x)} = g^{Σ^{q_T}_{i=1} c_i x^i}$ and stores the pairs $(p_i, g^{f'(x)})$ in a list named $S$-list, where $i ∈ [1, q_T]$.
(3) \( C \) computes \( (f(z) - c_k)/z = \sum_{i=0}^{q_1} c_i z^{-i} \) and sets \( R_U = a \tilde{e}(g^\xi x^{-1}, g^{c_0} g^{x^j y_{R1}}) \cdot R^2 \). If \( R = \tilde{e}(g,g)^{1/x}, R_U = \tilde{e}(g^\xi x^{-1}, g^{c_0} g^{x^j y_{R1}}) \cdot R^2 = \tilde{e}(g,g)^{(f(z)/x) - c_0/x} = \tilde{e}(g,g)^{(f(z)/x)^{-1}} \cdot x = \tilde{e}(g,g)^{(f(z)/x)^{-1}} \cdot x = \tilde{e}(U,U)^{1/x} \).

(4) \( C \) randomly chooses \( 2n + 4 \) numbers \( s_0, s_1, \ldots, s_n, t_0, t_1, \ldots, t_n, \eta, \lambda \in Z^*_n \) and constructs \( X_j = V^{x} \cdot U^y, Y_i = U^y \) for each \( i \in [0, n] \). Let \( Z = V^\lambda \), \( C \) gives \( A \) the public key \( pk = \{ U, X_0, X_1, \ldots, X_n, Y_0, Y_1, \ldots, Y_n, Z, \mu \} = \tilde{e}(U, U), g, H_1, H_2, \tilde{e} \). The corresponding sk is \( \{s_0 x - \eta^0, s_1 x - \eta^1, \ldots, s_n x - \eta^n, t_0 x, t_1 x, \ldots, t_n x, \lambda x \} \), where the values \( s_i x - \eta^i, s_i x - \eta^i, \ldots, s_n x - \eta^n, \lambda x \) are unknown to \( C \).

(ii) \( H_1, H_2 \)-queries: the random oracles \( H_1 \) or \( H_2 \) can be queried by \( A \) at any time. To answer \( H_1 \)-queries, \( C \) keeps a list of tuples \( (w_i, h_i, \sigma_i) \) called \( H_1 \)-list which is empty initially. When \( A \) queries the random oracle \( H_1 \) at a point \( u_i \in \{0, 1\}^n \), algorithm \( C \) answers as follows:

\( H_1 \)-queries:

(1) If the query \( w_i \) is already in a tuple \( (w_i, h_i, \sigma_i) \) on \( H_1 \)-list, then \( C \) responds with \( H_1(w_i) = h_i \), where \( h_i \in Z^*_n \).

(2) Otherwise, \( C \) generates a random coin \( \sigma_i \in \{0, n_{q_T} \} \) so that \( Pr[\sigma_i = 0] = 1/(n_{q_T} + 1) \).

(3) If \( \sigma_i = 0 \), \( C \) sets \( h_i = \eta \). Otherwise, \( C \) picks a random \( a_i \in Z^*_q \) and sets \( h_i = a_i \).

(4) \( C \) adds the tuple \((w_i, h_i, \sigma_i)\) to \( H_1 \)-list and answers \( A \) with \( H_1(w_i) = h_i \).

\( H_2 \)-queries are similar to \( H_1 \)-queries. To answer \( H_2 \) queries from \( A \), \( C \) holds a list of tuples \( (\psi_i, \psi_j) \) called \( H_2 \)-list which is initially empty. When \( A \) queries the random oracle \( H_2 \) at a point \( \psi_i \in G_2 \), algorithm \( C \) responds as follows:

\( H_2 \)-queries:

(1) If the query \( \psi_i \) already appears on \( H_2 \)-list in a tuple \( (\psi_i, \psi_j) \), then \( C \) responds with \( H_2(\psi_i) = \psi_j \), where \( \psi_j \in \{0, 1\}^{\log^2 q} \).

(2) Otherwise, \( C \) picks a random \( b_i \in \{0, 1\}^{\log^2 q} \) and sets \( \psi_j = b_j \).

(3) \( C \) adds the tuple \((\psi_i, \psi_j)\) to \( H_2 \)-list and responds \( A \) with \( H_2(\psi_i) = \psi_j \).

(iii) Trapdoor queries: when \( A \) queries a trapdoor of the keyword set \( Q_i = \{ q_{ij1}, q_{ij2}, \ldots, q_{im} \} \) and \( \text{sym} \), the algorithm \( C \) answers as follows:

(1) \( C \) runs \( H_1 \)-queries algorithm to obtain \( h_{ij} \), where \( h_{ij} = H_1(q_{ij}) \) for each \( j \in [1, m] \). Let \((w_{ij}, h_{ij}, \sigma_{ij})\) be the corresponding tuples on \( H_1 \)-list, for each \( j \in [1, m] \); if \( \sigma_{ij} = 0 \), then \( C \) reports failure and terminates.

(2) Otherwise, \( C \) constructs a trapdoor for query \( \{Q_i, \text{sym} \} \). \( C \) first computes \( f'(x') = \prod_{i=1}^{m} (x' + h_{ij}) \) = \( a_i x^{m+1} + a_{m-1} x^{m-1} + \ldots + a_1 x' + a_0 \). Then, for each \( j \in [0, m] \), \( C \) can compute \( l_j = -(\sum_{i=0}^{m} a_i l_i) / a_j \) and \( \nu_j = -(\sum_{i=0}^{m} a_i \nu_i + \rho_j \sum_{i=0}^{m} a_i l_j) / \rho \sum_{i=0}^{m} a_i l_j \) through an equality of \( 1/(x' + \rho_j) = a_{1/\rho_j}/(\sum_{i=0}^{m} a_i (s_i x' - \eta^j) + \nu_j \sum_{i=0}^{m} a_i l_j) \). Usually, \( U^{1/(x' + \rho_j) l_i}, U^{1/(x' + \rho_j) l_{ij}}, \ldots, U^{1/(x' + \rho_j) l_{ij} \nu_j}, U^{1/(x' + \rho_j) l_{ij} \nu_j, \psi_j, \text{sym} } \) is the trapdoor for the keyword query \( Q_i \).

(3) \( C \) searches \( S \)-list to obtain the tuple \( \{b_i, g^{f'(x')} \} \). Then \( C \) sends \( g^{f'(x')} l_0, \ldots, g^{f'(x')} l_m, g^{f'(x')} b_{ij}, g^{f'(x')} b_{ij} l_i, \ldots, g^{f'(x')} b_{ij} l_i \nu_j, \psi_j, \text{sym} \) to the algorithm \( A \) as the correct trapdoor for the query \( Q_i \).

(iv) Challenge: algorithm \( A \) produces a keyword set \( W^* = \{ w_{01}, w_{02}, \ldots, w_{bn} \} \), which is what \( A \) wants to challenge. Then, \( A \) sends the target keyword set \( W^* \) to \( C \). \( C \) chooses a random keyword set \( R^* = \{ w_{11}, w_{12}, \ldots, w_{bn} \} \) and sets \( W^0 = W^* \) and \( W^1 = R^* \). The only restriction is that the previous trapdoor queries cannot distinguish \( W^0 \) and \( W^1 \). If there is any previous trapdoor that can break the restriction, \( C \) regenerates the keyword set \( W^1 \). Then \( C \) selects a random bit \( \beta \in \{0, 1\} \) and runs the above algorithm for responding to \( H_1 \)-queries to obtain the values \( h_{1j1}, h_{2j1}, \ldots, h_{bn} \), where \( H_i(w_{\beta j}) = h_{\beta j}, w_{\beta j} \in W^{\beta}, i \in [1, n] \). Let \((w_{\beta j}, h_{\beta j}, \sigma_{\beta j})\) be the corresponding tuples on \( H_1 \)-list, if there is no \( \sigma_{\beta j} = 0 \) for all \( i \in [1, n] \), then \( C \) terminates. Otherwise, \( C \) generates the challenge PECDK index \( I_\beta \) of \( W^\beta \) as follows:

(1) If \( \sigma_{\beta j} = 0 \), \( C \) computes \( A_{\beta j} = U^{f(x') \psi_{w_{\beta j}}} \), \( B_{\beta j} = U^{f(x') \psi_{w_{\beta j}}} \), \( \beta_{\beta j} = U^{f(x') \psi_{w_{\beta j}}} \), and \( D_{\beta j} = H_3(\tilde{e}(U, U)^{1/x}) \), where \( j \in [0, n] \). Observe that if \( R = \tilde{e}(g, g)^{1/x} \), then \( R_U = \tilde{e}(U, U)^{1/x} \). So, the above values are equivalent to \( A_{\beta j} = Y_j^{1/x} \cdot U^{f(x') \psi_{w_{\beta j}}} \), \( B_{\beta j} = Y_j^{1/x} \cdot U^{f(x') \psi_{w_{\beta j}}} \), \( \beta_{\beta j} = U^{f(x') \psi_{w_{\beta j}}} \), and \( D_{\beta j} = H_3(\tilde{e}(U, U)^{1/x}) \), where \( j \in [0, n] \). It means that this is a valid PECDK encryption of keyword \( w_{\beta j} \) when \( R = \tilde{e}(g, g)^{1/x} \).

(2) Otherwise, \( C \) computes \( A_{\beta j} = V^{1/x} \cdot U^{f(x') \psi_{w_{\beta j}}} \), \( B_{\beta j} = U^{1/x} \), \( \beta_{\beta j} = V^{1/x} \), and \( D_{\beta j} = H_3(\tilde{e}(U, U)^{1/x}) \), where \( j \in [0, n] \).
(3) $C$ sends

$$I_\beta = \begin{pmatrix}
A_{p_10} & A_{p_11} & \ldots & A_{p_1n} & B_{p_1} & B_{p_11} & \ldots & B_{p_1n} & C_{p_1} & D_{p_1} \\
A_{p_20} & A_{p_21} & \ldots & A_{p_2n} & B_{p_2} & B_{p_21} & \ldots & B_{p_2n} & C_{p_2} & D_{p_2} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
A_{p_k0} & A_{p_k1} & \ldots & A_{p_kn} & B_{p_k} & B_{p_k1} & \ldots & B_{p_kn} & C_{p_k} & D_{p_k}
\end{pmatrix}$$

(7)

and two keyword sets $W^0$ and $W^1$ to $A$.

(v) More queries: $A$ continues to inquire trapdoor queries. The restriction is similar to the challenge phase.

(vi) Response: $A$ outputs a guess $\beta' \in \{0, 1\}$. $C$ will output 1 if $\beta' = \beta$. Otherwise, $C$ will output 0.

Now, we show that the algorithm $C$ can solve the decision $(q_T + 1)$-BDHI assumption with probability at least $\epsilon' = \epsilon/e(nq_T + 1)$. First of all, we should analyze the probability that $C$ does not abort in trapdoor queries phase and challenge phase. We define two events:

$\omega_1$: $C$ does not abort due to any of $A$’s trapdoor queries.

$\omega_2$: $C$ does not abort in challenge phase for generating $I_\beta$.

Suppose that $q_T$ is large enough. Therefore, the probability of the event $\omega_1$ is at least $(1 - 1/(nq_T + 1))^{3mT} \geq \epsilon$. The probability of the event $\omega_2$ is $1/(nq_T + 1)$.

If $R = \tilde{e}(g, g)^{1/x}$, $A$’s view is identical to its view in a real attack game. Since we assume that $A$ can break the PECDK scheme with an advantage $\epsilon$, it must satisfy $\Pr[\beta' = \beta] - 1/2 \geq \epsilon$. If $R$ is a random number, where $R \in G_2$, then $\Pr[\beta' = \beta] = 1/2$ since $A$ made a random guess.

Therefore, let $D = \{g, g^x, g^{x^2}, \ldots, g^{x^{n-1}}\}$; we have that

$$\frac{\Pr \{ B(D, \tilde{e}(g, g)^{1/x}) = 1 \} - \Pr \{ B(D, R) = 1 \}}{\epsilon/nq_T + 1} \geq \epsilon$$

(8)

Then we conclude the theorem.

\[ \square \]

4. Performance Analysis

In this section, we first give a detailed theoretical analysis for the previous scheme and then compare it with the proposed one. After that, we set up an experiment to show that our scheme has better performance than the previous one. For simplicity, we denote the proposed scheme and the previous one by PECDK-1 and PECDK-2, respectively.

4.1. Theoretical Analysis. In an IPE scheme, each ciphertext associated with an attribute vector $\overrightarrow{\alpha}$ can be decrypted by the secret keys corresponding to the predicate vector $\overrightarrow{\nu}$ if and only if $\overrightarrow{\nu} \cdot \overrightarrow{\alpha} = 0$. The PECDK-2 is based on the approach of converting an IPE scheme into a PECDK scheme. In order to analyze the efficiency of the previous one, we will introduce the conversion method, which is described as follows.

Suppose that $Q = \{q_1, q_2, \ldots, q_m\}$ and $W = \{w_1, w_2, \ldots, w_l\}$ are two keyword sets. By utilizing an IPE scheme, $(c_1, c_2, \ldots, c_o)$ is the index of $W$, where $c_i$ is a ciphertext of the vector $\{w_1, w_2, \ldots, w_l\}$ and $i \in [1, n]$. Suppose that $f(x) = a_0x^n + a_{m-1}x^{m-1} + \ldots + a_1x + a_0 = (x - H_1(q_1))(x - H_1(q_2)) \ldots (x - H_1(q_m))$; the trapdoor of $Q$ is $\{k_a, \text{sym}\}$, where $k_a$ is a secret key of the vector $\{w_1, w_2, \ldots, w_l\}$. Let the decryption algorithm be $DEC(pk, CT, KEY)$, where $pk$ is the public key, $CT$ is the ciphertext, and $KEY$ is the secret key. For each $i \in [1, n]$, the test algorithm can be performed by implementing $DECI(pk, c_i, k_a)$. According to different search function, there are two situations:

(1) For $\text{sym} = \text{v}$, if there is at least one $i \in [1, n]$ such that the decryption algorithm $DECI(pk, c_i, k_a)$ outputs 1, the test algorithm outputs 1, otherwise, 0.

(2) For $\text{sym} = \land$, the test algorithm will verify whether there are $m$ $c_i$ that can be decrypted by the decryption algorithm $DECI(pk, c_i, k_a)$. If so, the test algorithm outputs 1, otherwise, 0.

PECDK-2 is based on the IPE presented in [17]. For this IPE scheme, the time cost of setup, encryption, key generation, and decryption are all linear with $O(N^2)$, $O(N^2)$, and $O(N)$, respectively, where $N$ is the length of the predicate or attribute vector. Moreover, the space overhead of public key, master secret key ($msk$), ciphertext, and secret key is linear with $O(N^2)$, $O(N^2)$, $O(N)$, and $O(N)$, respectively. According to the conversion method mentioned above, we have the following results.

(1) The $pk$ and $sk$ in the previous scheme are the same as the $pk$ and $msk$ in the IPE scheme. Thus, the time and space cost of $pk$ and $sk$ are both linear with $o(n^2)$.

(2) Since the index of the previous scheme contains $n$ ciphertexts of IPE, the time and space cost of index are linear with $O(n^5)$ and $O(n^5)$, respectively. Because the algorithm of trapdoor generating is similar with that of the index building, the time and space cost of trapdoor is also linear with $O(n^5)$ and $O(n^5)$, respectively.

(3) The test algorithm needs to perform $n$ times decryption algorithm. Thus, the time cost of test is proportional to the square of $n$.

For our scheme, $pk$ contains $2n + 3$ elements in group $G_1$ and one element in $G_2$, and $sk$ involves $2n + 3$ elements in $Z_p^*$. Thus, we know that the time and space cost of KeyGen algorithm in our scheme are both linear with $O(n)$. Because the IndexBuild algorithm generates $2n + 3n$ elements in group $G_1$ and $n$ elements in $G_2$, the time and space cost of IndexBuild algorithm are both linear with $O(n^2)$. The time cost and space cost of Trapdoor algorithm are both $O(mn)$ since this algorithm creates $2m + 2$ elements in group $G_1$. Whether $\text{sym} = \land$ or $\text{sym} = \lor$, the test algorithm executes at most $O(mn)$ pairing operations. Therefore, we can reckon that the time cost of test algorithm is less than $O(mn)$.

According to the above analysis, we will give two tables to demonstrate the comparison. Let $|T_1|$ be the time cost for
a pairing operation [15] on $G_1$, and $|T_{G_1}|$ and $|T_{G_2}|$ be the time cost for the exponential operation on $G_1$ and $G_2$, where $G_1$ and $G_2$ are two groups of a prime order. For evaluating the time complexity, we only take these two operations into account since the time cost of these two operations is much more than the time cost of other operations like group add operation. The theoretical analysis of time complexity is shown in Table 1.

We denote the size of an element of $Z_p$, $G_1$, and $G_2$ by $|Z_p|$, $|G_1|$, and $|G_2|$, respectively. The comparison result of space complexity is shown in Table 2.

4.2. Experimental Results. We implement our construction in JAVA with Java Pairing Based Cryptography (JPBC) library [39]. In our implementation, the bilinear map is instantiated as Type A pairing (base field size is 128 bits), which offers a level of security equivalent to 1024-bit DLOG [16]. Our experiment was run on Intel(R) Core(TM) i7-4570 CPU at 3.60GHz processor with 8GB memory size. Such an experiment was run on Intel(R) Core(TM) i7-4570 CPU at 3.60GHz processor with 8GB memory size. Such an experiment was run on Intel(R) Core(TM) i7-4570 CPU at 3.60GHz processor with 8GB memory size.

### Table 1: Comparison with the previous PECDK scheme in time complexity.

|                  | PECDK-1                                                                 | PECDK-2                                                                 |
|------------------|--------------------------------------------------------------------------|--------------------------------------------------------------------------|
| Key Generation   | $O(n^2) | T_{G_1} | + | T_i |$ | $(2n+3) | T_{G_1} | + | T_i |$ |
| Index Building   | $(4n^2 + 2n) | T_{G_1} | + | n | T_{G_1} |$ | $(3n^2 + 4n) | T_{G_1} | + | n | T_{G_1} |$ |
| Trapdoor Generation | $(4m + 2) | T_{G_1} |$ | $(2m + 2) | T_{G_1} |$ |
| Testing          | $2n(2m + 1) | T_i |$ | $2n (m + 1) | T_i |$ |

### Table 2: Comparison with the previous PECDK scheme in space complexity.

|                  | PECDK-1                                                                 | PECDK-2                                                                 |
|------------------|--------------------------------------------------------------------------|--------------------------------------------------------------------------|
| PK size          | $O(n^2) | G_1 | + | G_2 |$ | $(2n + 3) | G_1 | + | G_2 |$ |
| SK size          | $O(n^2) | G_1 |$ | $(2n + 3) | Z_p |$ |
| Trapdoor size    | $(4m + 2) | G_1 |$ | $(2m + 2) | G_1 | + | Z_p |$ |
| Index size       | $(4n^2 + 2n) | G_1 | + | G_2 |$ | $(2n^2 + 4n) | G_1 | + | G_2 |$ |

Figure 3 shows the impact of $m$ on the time consumption. Because the key generation and index algorithms are not related to the parameter $m$, we only focus on the trapdoor generation and test algorithms. The comparison is stated as follows.

(1) The time costs of trapdoor generation in PECDK-1 and PECDK-2 are both linear with $O(m)$, which is shown in Figure 3(a). However, the time cost of creating trapdoor in PECDK-2 is nearly 10 times than that in PECDK-1. The reason for this is that PECDK-2 needs more exponential operations on $G_1$.
Figure 2: Impact of $n$ on the time cost of key generation (a), index building (b), testing (c), and trapdoor generation (d) (blue and red scale are for PECDK-1 and PECDK-2, respectively, and $D = 100$, $m = 5$, $Q = 10$, $n = \{5;10;15;20;25\}$).

(2) Figure 3(b) shows that the time cost of testing in PECDK-1 is linear with $O(m)$, while that in PECDK-2 is not related with $m$. The reason is that the predicate vector and attribute vector must have the same dimension in an IPE scheme. So, we must convert the query keyword set into a predicate vector having the same length as the attribute vector generated from the index keyword set. According to this, the time cost of testing in PECDK-2 is independent to $m$ actually. In addition, the time cost in DKS is also slightly less than that in CKS since DKS needs less test operations.

4.2.2. Storage Overhead. The storage cost of $pk$ and $sk$, indices, and trapdoors are illustrated in Figure 4. The comparison results are listed as follows.

(1) Figures 4(a) and 4(b) show that the storage cost of $pk$ and $sk$ in PECDK-2 increases with $O(n^2)$, while that in PECDK-1 is linear with $O(n)$. Moreover, the storage cost in PECDK-2 is more than 100 times than that in PECDK-1. The reason is that the PECDK-2 needs to store two matrices.

(2) In Figure 4(c), we know that the storage costs of indices in PECDK-1 and PECDK-2 are both linear with $O(n^2)$. However, the storage cost in PECDK-2 is also nearly two times than that in PECDK-1 since PECDK-2 needs to store more $G_1$ elements than PECDK-1.

(3) In Figure 4(d), the storage cost of trapdoors in PECDK-1 is independent to $n$, while that in PECDK-2
Figure 3: Impact of $m$ on the time cost of trapdoor generation (a) and testing (b) (blue and red scale are for PECDK-1 and PECDK-2, respectively, and $D = 100$, $n = 25$, $Q_t = 10$, $m = \{5; 10; 15; 20; 25\}$).

Table 3: Impact of $m$ on storage consumption of trapdoor generation ($n=25$, $D=100$, $Q_t = 10$).

| Size of $m$ | 5  | 10 | 15 | 20 | 25 |
|------------|----|----|----|----|----|
| PECDK-1(kb)| 6  | 10 | 14 | 19 | 23 |
| PECDK-2(kb)| 46 | 46 | 46 | 46 | 46 |

is linear with $O(n)$. The storage cost in PECDK-2 is still more than that in PECDK-1.

Because the size of $pk$, $sk$, and indices are not related to the parameter $m$, we only analyze the impact of parameter $m$ on the storage cost of trapdoor. According to Table 3, we find that the size of trapdoor in PECDK-2 is independent with $m$ since the query keywords must be converted into a predicate vector owing the same length to the attribute vector generated from the index keywords. The trapdoor size in PECDK-1 is linear with $m$ and less than that in PECDK-2.

According to the experimental results described above, we can find that these results are consistent with our theoretical analysis.

4.2.3. More Comments. Generally speaking, the number of keywords in a document ($n$) is usually only 3 ~ 5 (e.g., the research paper), and the number of keywords in a query ($m$) is often less than 10 [40]. According to above results, we can reckon that both $n$ and $m$ are less than 10 in the actual process of retrieval. According to the experiment result with $n = 10$ and $m = 5$, for the PECDK-1, the generation time of a single index and a single trapdoor is nearly 100 ms and 20 ms, respectively, and the test time of a single document is 15 ms. Compared with the PECDK-2, it is more practical. Moreover, the time and storage cost of trapdoor in PECDK-1 scheme is very low, which means that the proposed scheme is fit for the mobile setting where the client has limited computation and storage resources.

Many applications require that the client cannot only add a group of documents to the server but also remove a set of documents from the server. Considering this situation, we need that the proposed scheme supports dynamic update. It means that users can update the encrypted indices and files securely. Many previous works can achieve this goal [24, 25]. In our scheme, we adopt the forward index structure in which each file has its index. Thus, it can support dynamic update naturally. For example, when a client wants to add a document, he/she will generate an encrypted document and its corresponding secure index and send them to the server. If a client wants to delete a document, he/she will send a request to the server. The server will remove the related document and the associated index according to the request.

In addition, because each document has its own encrypted index, we can easily accelerate the search process by utilizing the technique of parallel computation [6]. Thus, we argue that our scheme is more practical in the cloud platform, which has strong computing power.

5. Conclusion

In this paper, we propose an efficient PECDK scheme based on a prime order group, which has better performance than the previous PECDK scheme. Our scheme is proven to be secure under the security definition.
To justify the efficiency of the proposed scheme, we present detailed theoretical analysis and experimental results. These results show that the proposed scheme is more practical than the most efficient PECDK scheme based on the IPE scheme. In the future, we will focus on building a SPE scheme supporting more advanced search function.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

We gratefully acknowledge the support of the National Natural Science Foundation of China under Grant Nos. 61402393 and 61601396 and Shanghai Key Laboratory of Integrated Administration Technologies for Information Security (no. AGK201607).

References

[1] Y. Zhu, D. Ma, and S. Wang, “Secure data retrieval of outsourced data with complex query support,” in Proceedings of the 2012 32nd International Conference on Distributed Computing Systems Workshops (ICDCS Workshops), pp. 481–490, IEEE Computer Society, Macau, China, June 2012.
D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, “Predicate encryption with an underlying public-key infrastructure,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 9, pp. 2546–2559, 2016.

C. Wang, N. Cao, K. Ren, and J. Zhou, “Towards efficient conjunctive keyword search over encrypted cloud data,” in *Proceedings of the IEEE 28th International Conference on Data Engineering (ICDE ’12)*, pp. 1156–1167, IEEE, Washington, DC, USA, April 2012.

Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, “Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2706–2716, 2016.

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

D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, “Public key encryption with keyword search,” in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques EUROCRYPT 2004*, vol. 3027 of *Lecture Notes in Computer Science*, pp. 506–522, Springer, Interlaken, Switzerland, May 2004.

D. Park, K. Kim, and P. Lee, “Public key encryption with conjunctive field keyword search,” in *Proceedings of the International Workshop on Information Security Applications WISA 2004*, vol. 3325 of *Lecture Notes in Computer Science*, pp. 73–86, Springer, Jeju Island, Korea, August 2004.

B. Zhang and F. Zhang, “An efficient public key encryption with conjunctive-subset keywords search,” *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 262–267, 2011.

C. Song, X. Liu, and Y. Yan, “Efficient public key encryption with field-free conjunctive keywords search,” in *Proceedings of the 6th International Conference on Trusted Systems (INTRUST ’14)*, vol. 9473 of *Lecture Notes in Computer Science*, pp. 394–406, Springer International Publishing, Beijing, China, December 2014.

J. Katz, A. Sahai, and B. Waters, “Predicate encryption supporting disjunctions, polynomial equations, and inner products,” in *Proceedings of the 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, vol. 4965 of *Lecture Notes in Computer Science*, pp. 146–162, Springer, Istanbul, Turkey, April 2008.

Y. Zhang and S. Lu, “POSTER: efficient method for disjunctive and conjunctive keyword search over encrypted data,” in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pp. 1535–1537, Hong Kong, China, December 2014.
[30] J. Li, Y. Wang, Y. Zhang, and J. Han, “Full verifiability for outsourced decryption in attribute based encryption,” IEEE Transactions on Services Computing, p. 1, 2017.

[31] J. Li, W. Yao, J. Han, Y. Zhang, and J. Shen, ”User collusion avoidance CP-ABE with efficient attribute revocation for cloud storage,” IEEE Systems Journal, vol. 12, no. 2, pp. 1767–1777, 2018.

[32] J. Li, Q. Yu, Y. Zhang, and J. Shen, ”Key-policy attribute-based encryption against continual auxiliary input leakage,” Information Sciences, vol. 470, pp. 175–188, 2019.

[33] J. G. Li, W. Yao, Y. C. Zhang, H. L. Qian, and J. G. Han, “Flexible and fine-grained attribute-based data storage in cloud computing,” IEEE Transactions on Services Computing, vol. 10, no. 5, pp. 785–796, 2017.

[34] J. Li, H. Yan, and Y. Zhang, ”Certificateless public integrity checking of group shared data on cloud storage,” IEEE Transactions on Services Computing, p. 1, 2017.

[35] H. L. Qian, J. G. Li, Y. C. Zhang, and J. G. Han, ”Privacy-preserving personal health record using multi-authority attribute-based encryption with revocation,” International Journal of Information Security, vol. 14, no. 6, pp. 487–497, 2015.

[36] J. G. Li, X. N. Lin, Y. C. Zhang, and J. G. Han, ”KSF-OABE: outsourced attribute-based encryption with keyword search function for cloud storage,” IEEE Transactions on Services Computing, vol. 10, no. 5, pp. 715–725, 2017.

[37] J. G. Li, Y. R. Shi, and Y. C. Zhang, ”Searchable ciphertext-policy attribute-based encryption with revocation in cloud storage,” International Journal of Communication Systems, vol. 30, no. 1, Article ID e2942, 2017.

[38] A. Joux, ”The Weil and Tate pairings as building blocks for public key cryptosystems,” in Proceedings of the Algorithmic Number Theory, Lecture Notes in Computer Science, C. Fieker and D. R. Kohel, Eds., vol. 2369 of Lecture Notes in Computer Science, pp. 20–32, Springer, Sydney, Australia, July 2002.

[39] A. D. Caro, ”The java pairing based cryptography library (jpb),” 2013, http://gas.dia.unisa.it/projects/jpbc/.

[40] H. Cui, Z. Wan, R. H. Deng et al., ”Efficient and expressive keyword search over encrypted data in the cloud,” IEEE Transactions on Dependable and Secure Computing, vol. PP, no. 99, p. 1, 2016.
