information

MDPI

Article

Efficient Public Key Encryption with Disjunctive Keywords Search Using the New Keywords Conversion Method

Yu Zhang [1,*], Yin Li [1] and Yifan Wang [2]

[1] School of Computer and Information Technology, Xinyang Normal University, Xinyang 464000, China; yufeiyangli@gmail.com
[2] Department of Computer Science, Wayne State University, Detroit 48202, USA; stacie0630@gmail.com
* Correspondence: willow1223@126.com; Tel.: +86-376-639-0765

Received: 8 October 2018; Accepted: 29 October 2018; Published: 1 November 2018

Abstract: Public key encryption with disjunctive keyword search (PEDK) is a public key encryption scheme that allows disjunctive keyword search over encrypted data without decryption. This kind of scheme is crucial to cloud storage and has received a lot of attention in recent years. However, the efficiency of the previous scheme is limited due to the selection of a less efficient converting method which is used to change query and index keywords into a vector space model. To address this issue, we design a novel converting approach with better performance, and give two adaptively secure PEDK schemes based on this method. The first one is built on an efficient inner product encryption scheme with less searching time, and the second one is constructed over composite order bilinear groups with higher efficiency on index and trapdoor construction. The theoretical analysis and experiment results verify that our schemes are more efficient in time and space complexity as well as more suitable for the mobile cloud setting compared with the state-of-art schemes.

Keywords: searchable encryption scheme; public key system; disjunctive keyword search; mobile security; data privacy

1. Introduction

Searchable encryption has attracted tremendous research attention in recent years. This field can be applied in many situations, such as email system, database management system and document management system. Constructing such a scheme supporting complex query conditions like boolean keyword search is an important issue. For creating a public key encryption system supporting boolean keyword search, we need to study public key encryption with conjunctive keyword search (PECK) and public key encryption with disjunctive keyword search (PEDK) first. The concept and security models of PECK are first defined by Park et al. [1]. In their work, they also gave two constructions based on these models. After this, Hwang and Lee [2] designed a more efficient PECK scheme under the multi-users setting. However, all of these schemes need keyword fields. To eliminate the keyword fields, a hidden vector encryption (HVE) scheme supporting conjunctive keyword search and range search over the encrypted data was proposed [3].

For the conjunctive keyword search, designers only need to consider whether all query keywords are included in the index keyword set, which means that the search algorithm only needs to consider the situation of keyword matching. However, for the disjunctive keyword search, designers should consider whether a part of keywords in the query are contained in the index keyword set, which means that the search algorithm needs to recognize the situation of both keyword matching and keyword non-matching at the same time. As a result, constructing an efficient PEDK scheme is more difficult

than a PECK one. The first solution of PEDK was proposed by Katz et al. [4]. They proposed a function encryption paradigm called inner product encryption (IPE) that supports more advanced search functions including disjunctive keyword search. In the IPE scheme, each secret key and ciphertext are associated with a predicate vector $\vec{v}$ and an attribute vector $\vec{x}$, respectively. If and only if $\vec{v} \cdot \vec{x} = 0$, the secret key can decrypt the corresponding ciphertext. By applying a keyword conversion method which changes the index and query keywords into attribute and predicate vectors respectively, a PEDK scheme can be created according to IPE. However, the PEDK scheme introduced in their work is not very practical. Specifically, the keyword conversion method used in [4] relies on a polynomial $F = \prod_{i=1}^{n}(x_i - y_1)(x_i - y_2)\ldots(x_i - y_m)$ which contains $(n+1)^m$ terms, where $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_m\}$ are denoted as an index and a query keyword set, respectively. Note that each term in F consists of two parts: the attribute part, which is the product of some elements in $X \cup \{1\}$, and the predicate part, which is the product of some elements in $Y \cup \{1\}$. By using the attribute part and the predicate part in each term, an attribute vector $\overrightarrow{x}$ and a predicate vector $\overrightarrow{y}$ can be created. Obviously, if $X \cap Y \neq \emptyset$, there is $F = 0$, which means $\overrightarrow{x} \cdot \overrightarrow{y} = 0$. Since the dimensions of predicate and attribute vector are $(n+1)^m$, the space and time complexity of the obtained PEDK scheme increases in exponential order. To construct a more efficient PEDK scheme supporting conjunctive keyword search simultaneously, Zhang and Lu proposed an approach that changes an IPE scheme to a scheme called public key encryption with conjunctive and disjunctive keyword search (PECDK), and gave a concrete construction [5]. In this scheme, the size of trapdoor and each document's index are both linear with $O(N_D)$. Moreover, the number of times of pairing operations in the search algorithm is still linear with $O(N_D)$, where N_D is the number of keywords in all indices. Thus, there is still a great room to improve the efficiency for disjunctive keyword search.

In addition, IPE is not designed only for creating a searchable encryption scheme supporting disjunctive keyword search. Thus, in order to obtain a highly efficient PEDK scheme, one should utilize alternative methods.

In this paper, we aim to create efficient PEDK schemes with less time and space cost. The contributions are listed as follows.

(1) A new method for converting the index and query keywords into a vector space model is proposed. The dimension of the vectors generated by the proposed method is more small than that generated by the previous methods. Moreover, our scheme is based on an equation of degree n with one unknown. The coefficients of this equation is composed of the index keyword set, while the roots of this equation is composed of the query keyword set. The coefficients and roots create the attribute vectors and predicate vectors, respectively. Thus, our method can easily be combined with other techniques, such as IPE scheme and composite bilinear order groups. By combining the new approach with an efficient IPE scheme, we propose a more efficient IPE-based PEDK scheme with a better time and space complexity (We denote this PEDK scheme by PEDK-1).

(2) We also demonstrate that a construction of PEDK does not rely on IPE. Applying techniques of dual system encryption and composite order group, a new PEDK scheme without IPE, denoted by PEDK-2, is given. We show that this scheme can largely reduce the complexity of index building and key generation compared with our former proposal.

Moreover, we design a experiment to show the efficiency of the previous PEDK schemes and two proposed schemes. The experiment results show that the efficiency of proposed schemes is more practical than the previous ones. We also give a detailed comparison between PEDK-1 and PEDK-2. The theoretical analysis and experiment results show that the key generation, index building and trapdoor generation operations in PEDK-2 are more efficient than that in PEDK-1, except for the test operation. In addition, the space cost in PEDK-2 is less than that in PEDK-1. In practice, client device, e.g., a mobile device, has less storage space and limited computation capacity. Thus, compared with PEDK-1, PEDK-2 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: public key system and symmetric key system.

Song et al. first introduced the definition of searching symmetric encryption and gave a specific scheme [6]. Then, Goh gave the concept and security definition of multi-keyword query on encrypted data [7], and gave a more practical scheme by using a Bloom filter. Based on this concept, improved schemes [8,9] were proposed to reduce computation and communication costs. However, the time cost of search in these schemes is linear with the number of documents. To optimize the query speed, some works utilize tree structure, such as r-tree and kd-tree, to obtain a sub-linear search efficiency [10,11]. Since the query results in these works are not sorted, all related documents will be returned, which will lead to a large network traffic problem. Based on the order-preserving encryption (OPE) scheme [12], rank search schemes [13,14] were proposed, which can quickly search top-k related documents. However, works mentioned above only support single keyword query. Recently, some schemes were proposed to achieve multi-keywords rank search [15,16].

The first searchable public key encryption solution called public-key encryption with keyword search (PEKS) is designed by Boneh et al. [17], which is related to the identity-based encryption (IBE) proposed in [18]. Based on this, Abdalla et al. gave the computational and statistical consistency of PEKS, and gave a concrete scheme [19]. However, these works fail to support multi-keyword search. The framework and security model of PECK are proposed by Park et al. [1]. They also gave two schemes in their work. One needs more bilinear pairing operations, while the other needs more private keys. Then, Hwang and Lee designed a more effective solution for multi-user setting [2]. The PECK schemes mentioned above were using keyword field as an additional information which are not practical in many applications. In order to avoid using a keyword field, a hidden vector encryption (HVE) scheme supporting conjunctive keyword search and range search over the encrypted data was proposed [3]. To achieve disjunctive keyword search, Katz et al. constructed the first IPE scheme [4], which is related to the attribute-based encryption (ABE) [20]. Fully secure IPE schemes with better decryption efficiency were proposed in [21,22].

In recent years, the work of searchable public-key encryption (SPE) has focused on two aspects. On the one hand, it focuses on improving the efficiency of traditional SPE; on the other hand, it adds special abilities on the traditional SPE, such as extra security mechanism and faster search rate. We use Table 1 to show main works for SPE in the last ten years. According to Table 1, we found that PEDK studies are relatively few, so this paper is devoted to building a more efficient PEDK scheme.

Table 1. Summary of previous searchable public key encryption schemes.

Type	Ref.	Query Condition	Additional Security Measures	Fast Search
Standard SPE	[23]	Conjunctive keyword search	-	-
	[24]	Conjunctive keyword search	-	-
	[22]	Disjunctive keyword search	-	-
	[3]	Range, conjunctive keywords, subset search	-	-
	[5]	Conjunctive and disjunctive keyword search	-	-
	[25]	Range search	-	-
Special SPE	[26]	Conjunctive keyword search	Verifiable	-
	[27]	Conjunctive keyword search	Verifiable	-
	[28]	Single keyword search	Verifiable and Access control	-
	[29]	Single keyword search	-	Yes
	[30]	Single keyword search	-	Yes
	[31]	Fuzzy keyword search	Access control	-

The query condition represents the search mode supported by the scheme; the additional security mechanism added in the scheme involves access control and the verification of query results; fast search means that the work is committed to building a scheme whose search speed is similar to the searchable symmetric encryption scheme.

Organization. We organize this paper as follows: in Section 2, the model of PEDK and its security model are defined, and briefly review the concept of composite order bilinear groups and complexity assumption. In Section 3, we introduce our keyword conversion method, and then propose two concrete PEDK scheme based on our approach. The security proof of proposed schemes are given in Section 4. The theoretical and experimental analysis are given in Section 5. Section 6 covers the conclusion.

2. Preliminaries

In this section, we first introduce the definition of PEDK's framework. Then, we present the security definition of PEDK. Finally, we briefly review some techniques adopted in our work, including complexity assumption and bilinear groups of composite order.

2.1. Model of PEDK

We suppose that pk is the public key, and sk is the secret key, where pk can be accessed by anyone and sk can be only held by the receiver. A sender can send an encrypted plaintext M with an encrypted index generated by using keywords $w_1, w_2, \ldots, w_n$ of M and pk to a server. When the receiver would like to retrieve the messages containing a specific list of keywords, the receiver can use sk and query keywords to construct a trapdoor, and sends the trapdoor to the server. After receiving the trapdoor, the server adopts the test algorithm to determine which documents match the trapdoor, and returns the matched documents to the receiver. The architecture of this process is described in Figure 1.

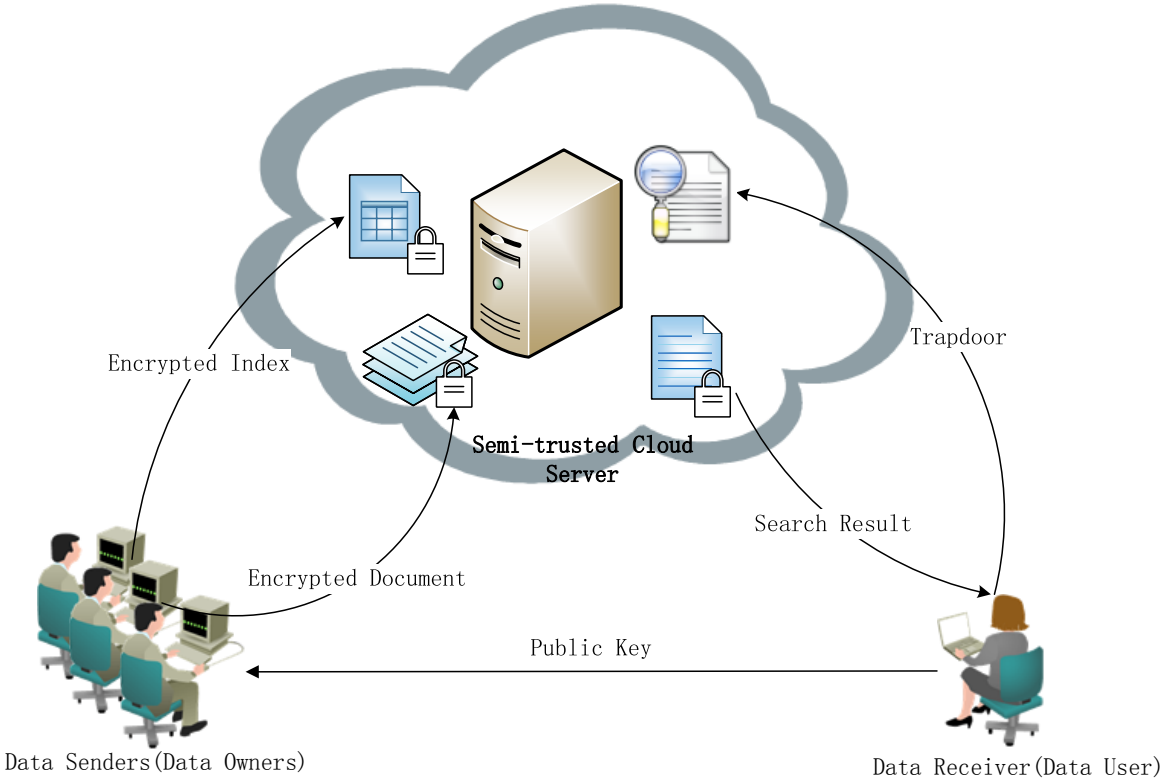

Figure 1. Architecture of the search over encrypted cloud data.

According to this architecture, we give the formal model of PEDK inspired by the model proposed in [1] as follows.

Definition 1. *The PEDK scheme involves four polynomial time algorithms, which are KenGen, IndexBuild, Trapdoor and Test:*

(1) KeyGen(γ): The algorithm takes a security parameter γ as input, and outputs a key pair (pk, sk), where "pk" and "sk" represent public key and secret key, respectively.

(2) IndexBuild(pk,W): By taking advantage of "pk" and a keyword set $W = \{w_1, w_2, \ldots, w_n\}$, the sender applies the algorithm to create an encrypted index I_W.

(3) Trapdoor(sk,Q): By utilizing the keyword query $Q = \{q_1, q_2, \ldots, q_m\}$ and "sk", the receiver adopts this algorithm to generate a trapdoor T_Q, where $m \leq n$.

(4) Test(pk, T_Q, I_W): The server executes this algorithm to test whether the encrypted index I_W and the trapdoor T_Q contain at least one same keyword. It takes T_Q, I_W and "pk" as input. If $Q \cap W \neq \phi$, then it outputs 1; otherwise, 0 .

2.2. Security Definition of PEDK

The proposed schemes must be proven to be secure under a formal security definition. Similar to the definition introduced in [1], we present the security definition as follows.

Definition 2. *If an PEDK scheme can resist chosen plaintext attacks and is adaptively index-hiding, then it must have that, for any probabilistic polynomial-time (PPT) adversary* A*, under a security parameter k, the* A*'s advantage for winning the following game is negligible:*

(1) Setup: The challenger C *performs the KeyGen(1^γ) algorithm to generate pk and sk. Then,* C *sends pk to the attacker* A*.*

(2) Phase 1: The attacker A *can adaptively ask* C *for any trapdoor T_Q of query Q he wants.*

(3) Challenge: A *randomly chooses two keyword sets W^0 and W^1, and gives them to* C*. Suppose that $Q_1, Q_2, \ldots, Q_t$ are the keyword sets which are queried to construct trapdoors in phase 1, where t is the number of trapdoors queried in phase 1, there is a restriction in which $Q_i \cap W^0 = \emptyset$ and $Q_i \cap W^1 = \emptyset$ for each $i \in [1, t]$. Then,* C *picks a random bit $\beta \in \{0, 1\}$, and creates $I_\beta = IndexBuild(pk, W^\beta)$. After that,* A *sends $\{I_\beta, W^0, W^1\}$ to* A*.*

(4) Phase 2: Under the restriction mentioned above, A *can continue to issue any query* Q *he wants.* C *responds the corresponding trapdoor.*

(5) Response: A *outputs $\beta' \in \{0, 1\}$. If $\beta' = \beta$, then* A *wins the game.*

According to the game mentioned above, *A*'s advantage in the above game is defined as:

$$Adv^A_{Game} = |Pr[\beta' = \beta] - \tfrac{1}{2}|.$$

Generally speaking, the key is to ensure that the encrypted form of W^0 and that of W^1 are computationally indistinguishable to the adversary.

2.3. Composite Order Bilinear Groups and Complexity Assumption

Boneh et al. [32] firstly uses the composite order bilinear groups to create cryptographic algorithms. In our paper, we will adopt groups of order N that is a product of four (distinct) prime. Moreover, by using a security parameter 1^k, we apply a generator $\mathfrak{g}$ to generate a description $I = (p_1, p_2, p_3, p_4, G, G_T, \hat{e})$; where p_1, p_2, p_3, p_4 are distinct primes, G_T and G are cyclic groups of order $N = p_1 p_2 p_3 p_4$, and $\hat{e} : G \times G \to G_T$ is a non-degenerate bilinear map such that:

1. Bilinear: $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$,where $g, h \in G$ and $a, b \in Z_N$;
2. Non-degenerate: $\hat{e}(g, g)$ is a generator of G_T if g is a generator of G;
3. Computable: For any $g, h \in G$, an efficient algorithm must exist to compute $\hat{e}(g, h)$.

In addition, we also further require that the group operations in G and G_T is computable in deterministic polynomial time under the security parameter k. Furthermore, we suppose that the

descriptions of G and G_T contain generators of G and G_T, respectively. For $S \subseteq \{1,2,3,4\}$, the subgroup of order $\prod_{i\in S} p_i$ is denoted by $G_{\prod_{i\in S} p_i}$. Suppose that $h_1 \in G_{\prod_{i\in S_1} p_i}$ and $h_2 \in G_{\prod_{i\in S_2} p_i}$, where $S_1, S_2 \subseteq \{1,2,3,4\}$, it is easy to verify that $\hat{e}(h_1,h_2) = 1$ if $gcd(\prod_{i\in S_1} p_i \times \prod_{i\in S_2} p_i | N^2, N) = N$. We call this property as the orthogonality property, which is very important in our construction.

For proving the security of our construction, we introduce a complexity assumption called General Subgroup Decision (GSD) Assumption [33] as follows.

GSD Assumption. Let $S_0, S_1, S_2, \ldots, S_k$ be non-empty subsets of $\{1,2,3,4\}$ in which, for each $j \in [2,k]$, either $S_j \cap S_0$, $S_j \cap S_1$ are both empty or $S_j \cap S_0$, $S_j \cap S_1$ are both non-empty. Choosing a group generator $\mathfrak{g}$, we can define the distribution as follows:

$$\begin{gathered}\mathbb{G} = (N = p_1p_2p_3p_4, G, G_T, \hat{e}) \xleftarrow{R} \mathfrak{g},\\ T_0 \xleftarrow{R} G_{\prod_{i\in S_0} p_i}, T_1 \xleftarrow{R} G_{\prod_{i\in S_1} p_i},\\ Z_2 \xleftarrow{R} G_{\prod_{i\in S_2} p_i}, \ldots, Z_k \xleftarrow{R} G_{\prod_{i\in S_k} p_i},\\ D = (\mathbb{G}, Z_2, Z_3, \ldots, Z_k).\end{gathered}$$

The algorithm A's advantage in breaking GSD Assumption is defined as:

$$Adv_{\mathfrak{g},A}(k) = Pr[A(D,T_0) = 1] - Pr[A(D,T_1) = 1]. \tag{1}$$

According to the description above, the definition of GSD assumption is given as follows.

Definition 3. *For all PPT algorithms A, if the function $Adv_{\mathfrak{g},A}(k)$ is negligible of k, then we can say that, for, generator $\mathfrak{g}$, GSD Assumption holds.*

3. Proposed PEDK Schemes

In this section, we first introduce the conversion method that changes an index and a query keyword set into a attribute vector and a set of predicate vectors, respectively. Then, we combine this method into an efficient IPE scheme to construct the PEDK-1 scheme. Finally, we build the PEDK-2 scheme based on a composite bilinear order group.

3.1. Conversion Method

The key idea of this method is to first construct an equation of degree n with one unknown by using the index and query keyword sets, where n is the number of keywords. Then, by using the relationship between the roots and coefficients in this equation, an attribute vector for the index keyword set and a set of predicate vectors for the query keyword set can be created. The concrete steps are shown below.

Suppose that any keyword w can be expressed as a string in $\{0,1\}^*$, and we define a function $H_1 : \{0,1\}^* \to Z_p^*$. Since p is a large prime and is larger than the number of the all words, H_1 can be collision-resistance. 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.

We first construct an equation of degree n with one unknown by using the index and query keyword sets. After that, we use the roots and coefficients of the equation to create a set of query vectors and an index vector. Let $W = \{w_1, w_2, \ldots, w_n\}$ and $Q = \{q_1, q_2, \ldots, q_m\}$ are two keyword sets, where $m < n$. The approach is described as follows:

(1) For the keyword set $W = \{w_1, w_2, \ldots, w_n\}$, constructing a function:

$$\begin{aligned}f(x) &= (x - H_1(w_1))(x - H_1(w_2)) \ldots (x - H_1(w_n))\\ &= a_n x^n + a_{n-1}x^{n-1} + \ldots + a_0 x^0.\end{aligned} \tag{2}$$

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

(2) For each keyword q_i in the keyword set Q, we construct $\vec{q_i} = \{H_1(q_i)^0, H_1(q_i)^1, \ldots, H_1(q_n)^n\}$, and output a vector set $\{\vec{q_1}, \vec{q_2}, \ldots, \vec{q_m}\}$.

Note that, if there is a keyword $q_i \in Q$ such that $q_i \in W$, where $i \in [1, m]$, according to the Equations (2), it is not difficult to verify that $\vec{a} \cdot \vec{q_i} = 0$.

As a result, we can test each $q_i \in Q$ with W to make a disjunctive keyword search. If $Q \cap W \neq \emptyset$, then it must exist an $i \in [1, m]$ such that $\vec{a} \cdot \vec{q_i} = 0$. Based on this, two concrete PEDK schemes will be proposed in the rest of this section.

3.2. PEDK-1

By using the method given in Section 3.1, we can give a new PEDK scheme based on the IPE scheme.

Construction. In the IPE scheme, there are four algorithms: $Setup_{IPE}$, $Enc_{IPE}(pk_{IPE}, \vec{x}, M)$, $KeyGen_{IPE}(pk_{IPE}, msk_{IPE}, \vec{v})$, and $Dec_{IPE}(pk_{IPE}, c, sk_{\vec{v}})$. We denote the public key and the master secret key by pk_{IPE} and msk_{IPE}, respectively. $\vec{x}$ and $\vec{v}$ represent the attribute and predicate vectors, respectively. The ciphertext and secret key are denoted by c and $sk_{\vec{v}}$, respectively. By combining the conversion method and an efficient IPE scheme introduced in [22], the PEDK-1 scheme works as follows:

- **KeyGen**: it runs $Setup_{IPE}$ to generate a key pair $\{pk_{IPE}, msk_{IPE}\}$, and then sets $pk = pk_{IPE}$ and $sk = msk_{IPE}$.
- **IndexBuild**: For the keyword set W, it uses the Equation (2) to create an vector $\vec{a}$. After this, it outputs $I_W = Enc_{IPE}(pk, \vec{a}, message)$ as the index of W. Note that it sets the *message* to 1.
- **Trapdoor**: For the query Q, it generates a group of vectors $\{\vec{q_1}, \vec{q_2}, \ldots, \vec{q_m}\}$. By applying $KeyGen_{IPE}$ to each vector, it creates a trapdoor $T_Q = \{t_1, t_2, \ldots, t_m\}$, where $t_i = KeyGen_{IPE}(pk, msk, \vec{q_i})$ and $i \in [1, m]$.
- **Test**: Given I_W and T_Q, it runs $Dec_{IPE}(I_W, t_i, pk)$ for each $i \in [1, m]$. If there is at least one $i \in [1, m]$ such that Dec_{IPE} outputs 1, then it outputs 1. Otherwise, it outputs 0.

Correctness. If $q_i \in W$, then there is $\vec{a} \cdot \vec{q_i} = 0$. Let I_W and T_Q be as the above. It exists that t_i can decrypt the index I_W. As a result, according to the property of IPE, the Dec_{IPE} algorithm outputs 1 (Note that the *message* is set to be 1 in the IndexBuild algorithm.). This means that the Test algorithm also outputs 1.

Security. The security of this scheme depends on that of the IPE scheme. The detailed security proof is given in Section 1.

3.3. PEDK-2

The previous PEDK schemes are based on the IPE scheme. Considering the reason that IPE is a general encryption prototype which is not designed only for disjunctive keyword search on encrypted data, in this subsection, we aim to create a non-IPE PEDK scheme by using the composite order bilinear groups.

Construction. The PEDK-2 scheme works as follows:

- **KeyGen**: Randomly selecting a bilinear group G of order $N = p_1p_2p_3p_4$ (where p_1,p_2,p_3,p_4 are distinct primes), $\alpha_i, \beta_i \in Z_N^*$, $U_1, A_1 \in G_{p_1}$ and A_{4i}, B_{4i}, U_4 and $g_4 \in G_{p_4}$, where $i \in [0, n]$, pk is published as:

$$pk = \{N, U_1U_4, {A_1}^{\beta_i}A_{4i}, {A_1}^{\alpha_i}B_{4i}, g_4, H_1\}.$$

The secret key sk is $\{\alpha_i, \beta_i, U_1, A_1, g_3\}$, where g_3 is a generator of G_{p_3}.

- **IndexBuild**: Given a keyword set $W = \{w_1, w_2, \ldots, w_n\}$, the algorithm uses the Equation (2) to create an vector $\vec{a} = \{a_0, a_1, a_2, \ldots, a_n\}$. Choosing $n+3$ random elements $s, c_0, c_1, c_2, \ldots, c_n, c_{n+1} \in Z_N^*$, for the vector $\vec{a}$, the encryption algorithm creates the index $I_W = (C_{10}, C_{11}, C_{12}, \ldots, C_{1n}, C_2)$ as:

$$\begin{aligned} C_{1i} &= (A_1{}^{\beta_i} A_{4i})^{sa_i} \times (A_1{}^{\alpha_i} B_{4i})^s \times g_4{}^{c_i} \\ &= A_1{}^{s(\beta_i a_i + \alpha_i)} C_{4i}, \end{aligned}$$

$$\begin{aligned} C_2 &= (U_1 U_4)^s \times g_4^{c_{n+1}}, \\ &= U_1^s C_4 \end{aligned}$$

where $C_{4i} = A_{4i}^{sa_i} B_{4i}^s g_4{}^{c_i}$, $C_4 = U_4^s g_4^{c_{n+1}}$ and $i \in [0, n]$.

- **Trapdoor**: Given a keyword set $Q = \{q_1, q_2, \ldots, q_m\}$ where $m \leq n$, the trapdoor generation algorithm chooses $r \in Z_N^*$ and generates random elements R_{3ji} where $i \in [0, n]$, $j \in [1, m]$ and R_{3j} by using g_3 and raising it to the random exponents modulo N. Then, it chooses random elements $r_1, r_2, \ldots, r_m \in Z_N^*$ and two random orthogonal vector bases $B = (\overrightarrow{b_1}, \ldots, \overrightarrow{b_m})^T$, and $B^* = (\overrightarrow{b_1}^*, \ldots, \overrightarrow{b_m}^*)^T \in Z_N^{m \times m}$ in which $\overrightarrow{b_i} \cdot \overrightarrow{b_j}^* = 0 \ (mod\ N)$ whenever $i \neq j$ and $\overrightarrow{b_i} \cdot \overrightarrow{b_i}^* = \lambda \ (mod\ N)$ for all $i \in m$ where λ is a random elements in Z_N^*. Suppose that $\overrightarrow{b_t} = \{b_{t1}, b_{t2}, \ldots, b_{tm}\}$ and $\overrightarrow{b_t}^* = \{b_{t1}^*, b_{t2}^*, \ldots, b_{tm}^*\}$ where $t \in [1, m]$, it computes:

$$K_{ji} = U_1 \frac{r_1 b_{1j} H_1(q_1)^i + r_2 b_{2j} H_1(q_2)^i + \ldots + r_m b_{mj} H_1(q_m)^i}{\beta_i} R_{3ji},$$

$$K_j = A_1 \sum_{i=0}^{n} \frac{\alpha_i (r_1 b_{1j} H_1(q_1)^i + r_2 b_{2j} H_1(q_2)^i + \ldots + r_m b_{mj} H_1(q_m)^i)}{\beta_i} R_{3j},$$

where $i \in [0, n]$, $j \in [1, m]$. The trapdoor of keyword set Q is :

$$T_Q = \begin{pmatrix} K_{10} & K_{11} & \ldots & K_{1n} & K_1 & \overrightarrow{b_1}^* \\ K_{20} & K_{21} & \ldots & K_{2n} & K_2 & \overrightarrow{b_2}^* \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ K_{m0} & K_{m1} & \ldots & K_{mn} & K_m & \overrightarrow{b_m}^* \end{pmatrix}.$$

- **Test**: After receiving a trapdoor T_Q and a secure index I_W, the algorithm works as follows:

  (a) The algorithm computes $M_j = \frac{\prod_{i=0}^{n} \hat{e}(C_{1i}, K_{ji})}{\hat{e}(C_2, K_j)}$ for each $j \in [1, m]$.

  (b) Choosing a counter k, and setting $k = 1$.

  (c) If $k > m$, then go to step d), otherwise the algorithm computes: $D_k = \prod_{j=1}^{m} M_j^{b_{kj}^*}$. If $D_k = 1$, then the algorithm outputs 1 and ends. Otherwise, it sets $k = k + 1$ and goes to the step c).

  (d) The algorithm outputs 0 and ends.

Correctness. Let I_W and T_Q be as the above. Then, we have the following two equations:

$$\begin{aligned} \hat{e}(C_2, K_j) &= \hat{e}(U_1^s C_4, A_1 \sum_{i=0}^{n} \frac{\alpha_i (r_1 b_{1j} H_1(q_1)^i + \ldots + r_m b_{mj} H_1(q_m)^i)}{\beta_i} R_3) \\ &= \hat{e}(U_1, A_1)^{s \sum_{i=0}^{n} \frac{\alpha_i (r_1 b_{1j} H_1(q_1)^i + \ldots + r_m b_{mj} H_1(q_m)^i)}{\beta_i}}, \end{aligned} \tag{3}$$

$$\prod_{i=0}^{n} \hat{e}(C_{1i}, K_{ji}) = \hat{e}(A_1, U_1)^{s\sum_{i=0}^{n} \frac{\alpha_i (r_1 b_{1j} H_1(q_1)^i + \ldots + r_m b_{mj} H_1(q_m)^i)}{\beta_i}} \times \hat{e}(A_1, U_1)^{s(r_1 b_{1j} \sum_{i=0}^{n} a_i H_1(q_1)^i + \ldots + r_m b_{mj} \sum_{i=0}^{n} a_i H_1(q_m)^i)}. \tag{4}$$

According to the Equations (3) and (4), we know:

$$M_j = \hat{e}(A_1, U_1)^{s(r_1 x_1 b_{1j} + r_2 x_2 b_{2j} + \ldots + r_m x_m b_{mj})}. \tag{5}$$

In Equation (5), we can find:

$$x_k = \sum_{i=0}^{n} a_i H_1(q_k)^i, \ where \ k \in [1, m]. \tag{6}$$

Since $\overrightarrow{b_i} \cdot \overrightarrow{b_j}^* = 0 \ (mod \ N)$ whenever $i \neq j$ and $\overrightarrow{b_i} \cdot \overrightarrow{b_i}^* = \lambda \ (mod \ N)$ for all $i \in m$, there is:

$$\begin{aligned} D_k &= \prod_{j=1}^{m} M_j^{b_{kj}^*} \\ &= \hat{e}(A_1, U_1)^{s(r_1 x_1 \overrightarrow{b_1} \overrightarrow{b_k}^* + \ldots + r_k x_k \overrightarrow{b_k} \overrightarrow{b_k}^* + \ldots + r_m x_m \overrightarrow{b_m} \overrightarrow{b_k}^*)} \\ &= \hat{e}(A_1, U_1)^{s r_k x_k \lambda}. \end{aligned}$$

Note that there is $x_k = 0$ if the keyword q_k in the trapdoor is also contained in the index. Thus, $D_k = 1$ if $q_k \in W$.

Security. The security of this scheme depends on the assumption introduced in Section 2.3. The detailed security proof is given in Section 4.2.

4. Security Analysis

4.1. Security of PEDK-1

The PEDK-1 scheme is based on the fully secure IPE scheme. Thus, we give the following proposition.

Proposition 1. *The PEDK-1 scheme is secure if the IPE scheme that PEDK-1 is based on is secure.*

Proof Sketch. If it exists an PPT algorithm $\mathbb{A}$ that can break the PEDK-1 scheme, then the IPE scheme on which PEDK-1 is based can be broken by $\mathbb{A}$. For the setup phase, $\mathbb{C}$ applies the $Setup_{IPE}$ algorithm to creat pk_{IPE} and sk_{IPE}, and sets $pk = pk_{IPE}$, $sk = sk_{IPE}$. For the phase 1, $\mathbb{A}$ can adaptively query trapdoors of keyword set $\{Q_1, Q_2, \ldots, Q_t\}$. These trapdoors are composed of a set of decryption keys of IPE. For the challenge phase, under a constraint that $Q_i \cap W^0 = \emptyset$ and $Q_i \cap W^1 = \emptyset$, $\mathbb{A}$ randomly chooses two challenge keyword sets W^0 and W^1, where $i \in [1, t]$. After this, $\mathbb{C}$ randomly chooses a $\beta \in \{0, 1\}$, and sends an index I_{W^β} to $\mathbb{A}$. This index is composed of a set of challenge ciphertexts of IPE. For the phase 2, $\mathbb{A}$ still asks for trapdoors which he wants under the restriction mentioned above. For the response phase, $\mathbb{A}$ issues a guess β'. If $\mathbb{A}$ can break the PEDK-1 scheme, the value of $|Pr[\beta' = \beta] - \frac{1}{2}|$ can not negligible. It means that the two challenge indices can be distinguished. Because the challenge indices in the PEDK-1 scheme is equal to the challenge ciphertexts in the IPE scheme, we can reckon that $\mathbb{A}$ can break the IPE scheme according to the security definition for IPE.

4.2. Security of PEDK-2

To prove the security of the PEDK-2 system, according to the dual system encryption, we first introduce the concept of semi-functional trapdoor and index. The semi-functional trapdoor and index will be adopted in the proof, but not in the real PEDK-2 system. This is similar to those introduced in [34].

Semi-functional index: We denote the generator of the subgroup G_2 by g_2, and create the Semi-functional index as follows. The encryption algorithm generates a normal index $C'_{10}, C'_{11}, C'_{12}, \ldots, C'_{1n}, C'_2$. Choosing $n+2$ random elements $x, z_{c0}, z_{c1}, \ldots, z_{cn}$ and $z_c \in Z_N$, C_{1i} is set to be $C'_{1i} g_2^{x z_{ci}}$ for each $i \in [0, n]$ and C_2 is set to be $C'_2 g_2^{x z_c}$. The semi-functional index is $\{C_{10}, C_{11}, \ldots, C_{1n}, C_2\}$.

Semi-functional trapdoor: We denote the generator of the subgroup G_2 by g_2, and create the semi-functional trapdoor as follows. A normal trapdoor $K'_{j0}, K'_{j1}, \ldots, K'_{jn}, K'_j$ is constructed by the encryption algorithm, where $j \in [1, m]$. Choosing $(n+2)m+1$ random elements $\xi, z_{kj0}, z_{kj1}, \ldots, z_{kjn}$ and $z_{kj} \in Z_N$, K_{ji} is set to be $K'_{ji} g_2^{\xi z_{kji}}$ for each $i \in [0, n]$ and K_j is set to be $K'_j g_2^{\xi z_{kj}}$. The semi-functional trapdoor is $\{K_{j0}, K_{j1}, \ldots, K_{jn}, K_j\}$.

If we want to decrypt the semi-functional index by utilizing the semi-functional trapdoor, an additional factor $M'_j = \hat{e}(g_2, g_2)^{x\xi(z_c z_{kj} - \sum_{i=0}^{n} z_{ci} z_{kji})}$ will be generated for each $j \in [1, m]$.

The security of PEDK-2 depends on GSD Assumption. The proof process is based a hybrid method in which a set of games will be proven to be undistinguishable. We list these games as follows.

1. $Game_{Real}$: This game is the original security game.
2. $Game_{Restricted}$: Suppose that the keyword set $W = \{w_1, w_2, \ldots, w_n\}$ is one of the challenge keyword sets. We construct an n-degree polynomial $f(x) = (x - H_1(w_1))(x - H_1(w_2)) \ldots (x - H_1(w_n)) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$. This game is identical to the real game except that the attacker is not allowed to obtain a trapdoor in which the corresponding keyword set does not contain any keyword w which satisfies $\sum_{i=0}^{n} a_i H_1(w)^i = 0 mod p_2$. This restriction will be kept throughout the following games.
3. $Game_k$: For each $k \in [0, q]$, we define $Game_k$ identical to $Game_{Restricted}$ except that A is given the semi-functional index. Moreover, the first k trapdoors are semi-functional and the rest are normal. In $Game_0$, the trapdoors sent to A are normal, but the index is semi-functional. In $Game_q$, both trapdoor and index are in the semi-functional form.
4. $Game_{Final_k}$: Letting the keyword set $W = \{w_1, w_2, \ldots, w_n\}$ be one of the challenge keyword sets, we construct an n-degree polynomial $f(x)$ mentioned above. Compared with $Game_q$, the index in this game is a semi-functional encryption of a challenge vector that its first k elements are random and the rest of the elements are $\{a_k, a_{k+1}, \ldots, a_n\}$, where $k \in [0, n]$.

Obviously, $Game_{Final_0}$ is a game in which the index is a semi-functional encryption of a normal keyword set, while $Game_{Final_n}$ is a game in which the index is a semi-functional encryption of a random keyword set. The essence of the security proof is applying the following lemmas to verify that these games are indistinguishable.

Lemma 1. *If there is a probabilistic polynomial time algorithm A such that* $Adv^A_{Game_{Real}} - Adv^A_{Game_{Restricted}} = \epsilon$, *then a PPT algorithm B with advantage* $\geq \frac{\epsilon}{3}$ *in breaking GSD Assumption can be created.*

Lemma 2. *If there is a PPT algorithm A such that* $Adv^A_{Game_{Restricted}} - Adv^A_{Game_0} = \epsilon$, *then a PPT algorithm B with advantage* ϵ *in breaking a GSD Assumption can be created.*

Lemma 3. *For each* $k \in [0, q]$, *if there is a PPT algorithm A such that* $Adv^A_{Game_{k-1}} - Adv^A_{Game_k} = \epsilon$, *then a PPT algorithm B with advantage* ϵ *in breaking the GSD Assumption can be created.*

Lemma 4. *For each* $k \in [0, n]$, *suppose that there exists a PPT algorithm A such that* $Adv^A_{Game_{Final_{k-1}}} - Adv^A_{Game_{Final_k}} = \epsilon$. *Then, a PPT algorithm B with advantage* ϵ *in breaking GSD Assumption can be created.*

Considering the length of the article and the coherence of the article structure, the proofs of Lemmas 1–4 are given in Appendix A.

Theorem 1. *PEDK-2 scheme is secure if Assumptions* 1, 2, 3, *and* 4 *hold.*

Proof. If Assumptions 1, 2, 3, and 4 hold, $Game_{Real}$ is indistinguishable from $Game_{Final_n}$ based on the previous lemmas that have been proved. In $Game_{Final_n}$, β value is information-theoretically concealed from the attacker. According to this, we argue that the attacker fails to obtain any advantage in breaking PEDK-2 scheme. □

5. Performance Evaluation

5.1. Theoretical Analysis

According to Table 1, there are two previous PEDK schemes needed to be compared. One can be regarded as a combination of the conversion approach described in Section 1 and the most efficient IPE scheme given in [22] (For simplicity, we denote this scheme with PEDK-0.). The other PECDK is introduced in [5]. In the rest of this subsection, we will compare the proposed schemes (PEDK-1 and PEDK-2) with PEDK-0 and PECDK.

Let $|T_e|$ and $|T_{4e}|$ be the time cost for a pairing operation [33] on G and $G4$, and $|T_G|$ and $|T_{4G}|$ be the time cost for the power operation on G and $G4$, where G and $G4$ are a group of a prime order and a composite group of an order $N_4 = p_1p_2p_3p_4$, respectively. 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 that of other operations like group add operation. The theoretical analysis of time complexity is shown in Table 2.

Table 2. Comparison with the previous schemes in time complexity.

	PEDK-0	PECDK	PEDK-1	PEDK-2
Key Generation	$O((n+1)^m)\|T_G\| + \|T_e\|$	$O(N_D^2)\|T_G\|$	$O(n^2)\|T_G\| + \|T_e\|$	$O(n)\|T_{4G}\| + \|T_{4e}\|$
Index Building	$O((n+1)^m)\|T_G\|$	$O(N_D)\|T_G\|$	$O(n^2)\|T_G\|$	$O(n)\|T_{4G}\|$
Trapdoor Generation	$O((n+1)^m)\|T_G\|$	$O(N_D)\|T_G\|$	$O(n^2 \cdot m)\|T_G\|$	$O(n \cdot m)\|T_{4G}\|$
Testing	$O((n+1)^m)\|T_e\|$	$O(N_D)\|T_G\|$	$O(n^2)\|T_e\|$	$O(n)\|T_{4e}\|$

We denote the size of an element of G and $G4$ by $|G|$ and $|G4|$, and that of G_T and G_{4T} by $|G_T|$ and $|G_{4T}|$ respectively. The comparison result of space complexity is shown in Table 3.

Table 3. Comparison with the previous schemes in space complexity.

	PEDK-0	PECDK	PEDK-1	PEDK-2
PK size	$O((n+1)^m)\|G\| + \|G_T\|$	$O(N_D^2)\|G\|$	$O(n^2)\|G\| + \|G_T\|$	$O(n)\|G4\| + \|G_{4T}\|$
SK size	$O((n+1)^m)\|G\|$	$O(N_D^2)\|G\|$	$O(n^2)\|G\|$	$O(n)\|G4\|$
Trapdoor size	$O((n+1)^m)\|G\|$	$O(N_D)\|G\|$	$(4n+2)\|G\|$	$(n+1)\|G4\|$
Index size	$O((n+1)^m)\|G\| + \|G_T\|$	$O(N_D)\|G\|$	$(4n+2)\|G\| + \|G_T\|$	$(n+1)\|G4\| + \|G_{4T}\|$

According to the Tables 2 and 3, it is easy to find that the efficiency of the proposed schemes is better than PEDK-0. Note that N_D is a large integer and seen as the number of keywords in a dictionary, and N_D is much bigger than n in general. Therefore, we argue that the proposed schemes has better performance on time and space complexity than PECDK one.

5.2. Experimental Results

In our experiments, we build a group of artificial keyword sets with a different number of keywords in each set (i.e., $n = 5; 10; 15; 20; 25$), where each keyword set can be seen as an index of a document. In each keyword set, we denote each keyword as a unique integer in the range $[0, 5000]$, where 5000 can be regarded as the number of different words in artificial keyword sets. We encrypted each keyword set with PEDK-1 and PEDK-2, respectively, and stored these encrypted indices on

our machine. After this, we randomly performed some queries over the stored indices. In addition, we use the IPE scheme introduced in [22] to quantify the efficiency of PEDK-0 and PECDK. We applied the Java Pairing Based Cryptography library (JPBC) [35] to realize our schemes. The experimental environment is under 8 GB memory and Intel(R) Core(TM) i7-3520M@2.90GHz CPU (Xinyang, Henan province, China). Moreover, our construction is based on the Type A pairing whose base field size is 512-bit. The security level is identical to 1024-bit DLOG [35].

A. Performance of PEDK-0 and PECDK.

According to the analysis in Section 1, we know that the main method for constructing PEDK-0 is converting the index and query keywords into a predicate and attribute vector, respectively. By using predicate and attribute vectors, we can apply an IPE scheme to build a PEDK-0 scheme. Note that the length of predicate and attribute vectors is linear with $O((n+1)^m)$. Even if $m = 5$ and $n = 5$, the length of vector is 6^5. In addition, the PECDK scheme [5] is also based on the IPE scheme, where the the length of predicate and attribute vectors is linear with $O(N_D)$. For the sake of simplicity, we test the performance of the IPE scheme [22] under $N = \{5, 10, 15, \ldots, 50\}$, where N denotes the dimension of the predicate and attribute vectors, and then inferred that the efficiency of PEDK-0 and PECDK needed to be improved.

From Figure 2a, we find that the time cost of key generation, index building and trapdoor generation is linear with $O(N^2)$, and that of testing is linear with $O(N)$. From Figure 2b, the space cost of indices and trapdoors is linear with $O(N)$, and that of keys is linear with $O(N)$. Even if N is very small, the time and space consumptions cannot be ignored. When n and m are very large, N will become very large (Note that N is linear with $O((n+1)^m)$), which makes PEDK-0 ineffective. Moreover, the vocabulary size (N_D) is commonly linear with $O(10^6)$ [36]. Considering that $N = N_D$ in this case, we think the efficiency of PECDK is relatively low.

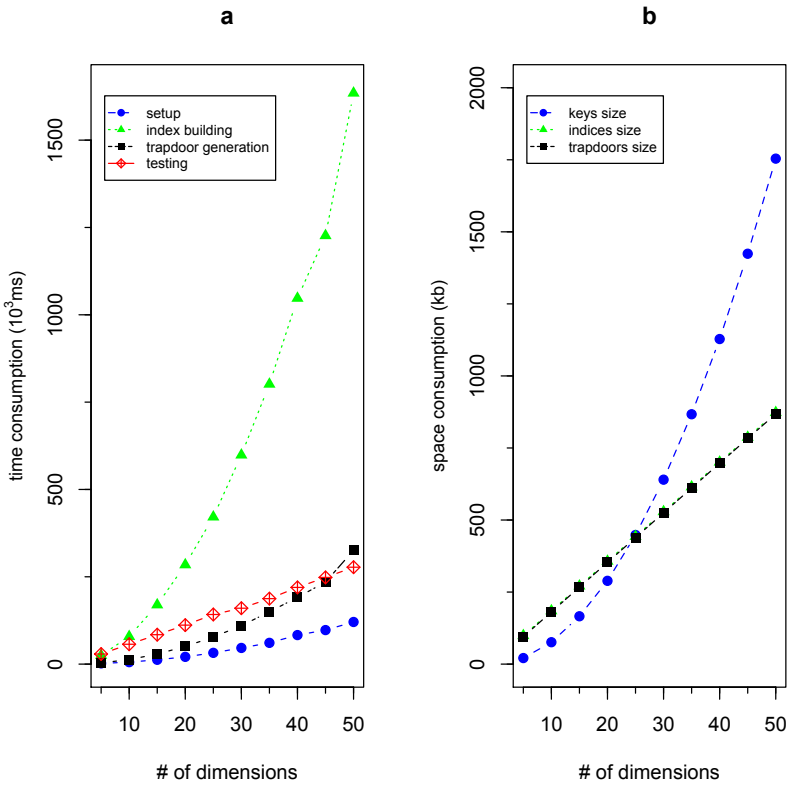

Figure 2. Impact of N on the time cost of key generation, index construction, trapdoor generation and testing in PEDK-0 and PECDK (**a**); and impact of N on the storage cost of the size of keys, indices and trapdoors in PEDK-0 and PECDK (**b**). ($D = 100, N = \{5, 10, 15, 20, 25\}$).

B. Performance Comparison between PEDK-1 and PEDK-2

B.1: Time Overhead. Impact of the keyword size (n). For a query with five keywords, Figure 3 shows that:

(1) Figure 3a–c show that the execution time of key generation, index building and trapdoor generation is linear with $O(n^2)$ in PEDK-1, while $O(n)$ in PEDK-2. Since the PEDK-1 scheme is based on the dual pairing vector space DPVS, the PEDK-2 has a better performance in the key generation, index building and trapdoor generation phase; and

(2) Figure 3d indicate that the running time of testing in PEDK-1 and PEDK-2 is linear with $O(n)$, and the time cost in PEDK-2 is nearly four times more than that in PEDK-1 since the pairing operations in a composite group are more time-consuming than that in a prime one.

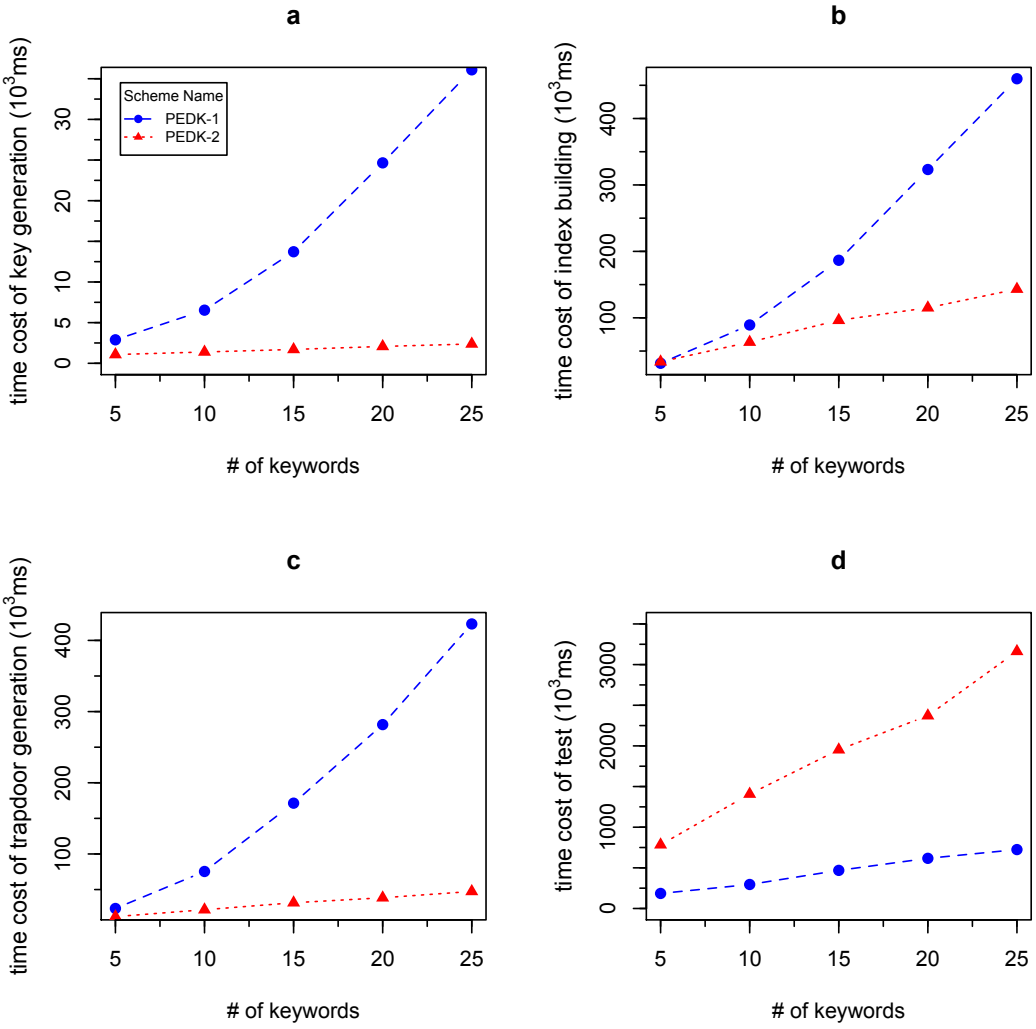

Figure 3. Impact of n on the time cost of key generation (**a**), index construction (**b**), trapdoor generation (**c**) and testing (**d**) in PEDK-1 and PEDK-2. ($D = 100, m = 5, n = \{5, 10, 15, 20, 25\}$).

Impact of the keyword size (m). According to the analysis in Section 5.1, we know that m only affects algorithms of trapdoor generation and testing. For an index with 25 keywords ($n = 25$), Figure 4 shows that the time consumption in trapdoor generation and testing are linear with $O(m)$. Specifically, the execution time of trapdoor generation in the PEDK-2 scheme is less than that in the PEDK-1 scheme, while the time cost for testing in the PEDK-2 scheme is more than that in the PEDK-1 scheme.

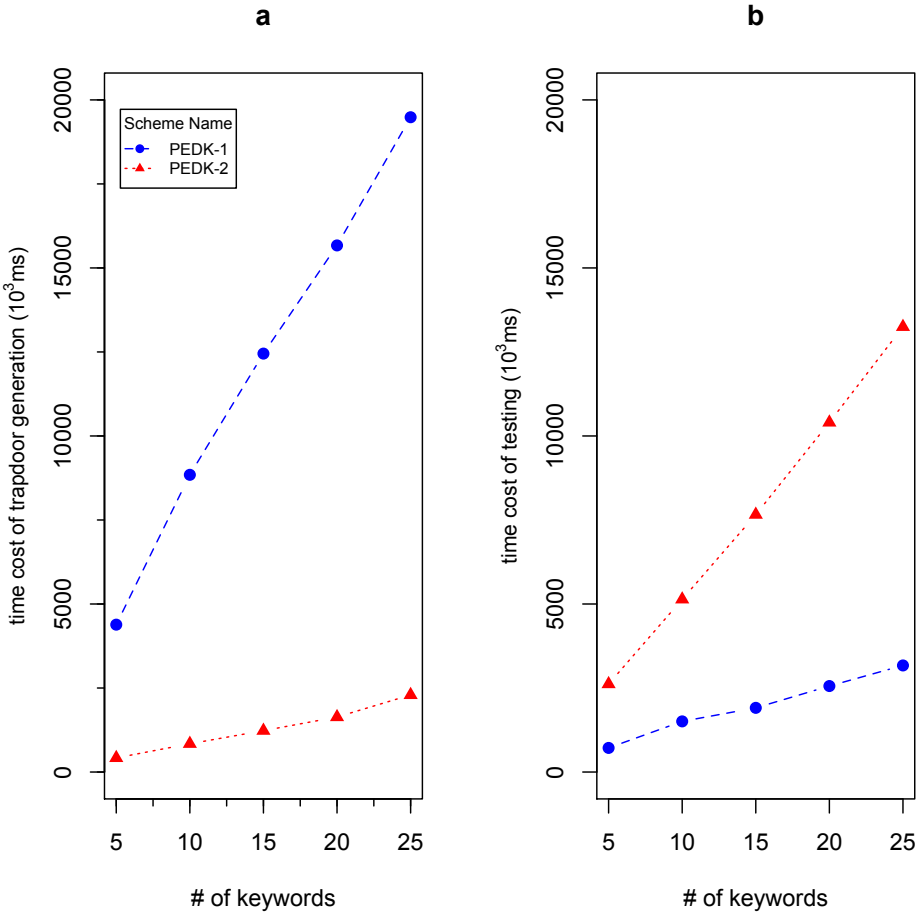

Figure 4. Impact of m on the time cost of trapdoor generation (**a**) and testing (**b**) in PEDK-1 and PEDK-2. ($D = 100, n = 25, m = \{5, 10, 15, 20, 25\}$).

B.2: Storage Overhead.

As shown in Figure 5, we can argue that:

(1) Figure 5a–c show the impact of n on the storage size of keys (pk and sk), indices and trapdoors. Figure 5a verifies that the storage size of keys is linear with $O(n^2)$ in PEDK-1, while $O(n)$ in PEDK-2. Given a fixed parameter m, Figure 5b,c show that the storage size of indices and trapdoors is linear with $O(n)$, and PEDK-2 needs less space overhead than PEDK-1 since PEDK-2 needs less group elements in the index and trapdoor.

(2) Because the parameter m only affects the phases of trapdoor and test, we only present Figure 5d to show the impact of m on the storage size of trapdoors. Given a fixed parameter n, both the trapdoor size in PEDK-1 and that in PEDK-2 are linear with $O(m)$. Due to owning less elements in the trapdoor, the space consumption in PEDK-2 is still less than that in PEDK-1.

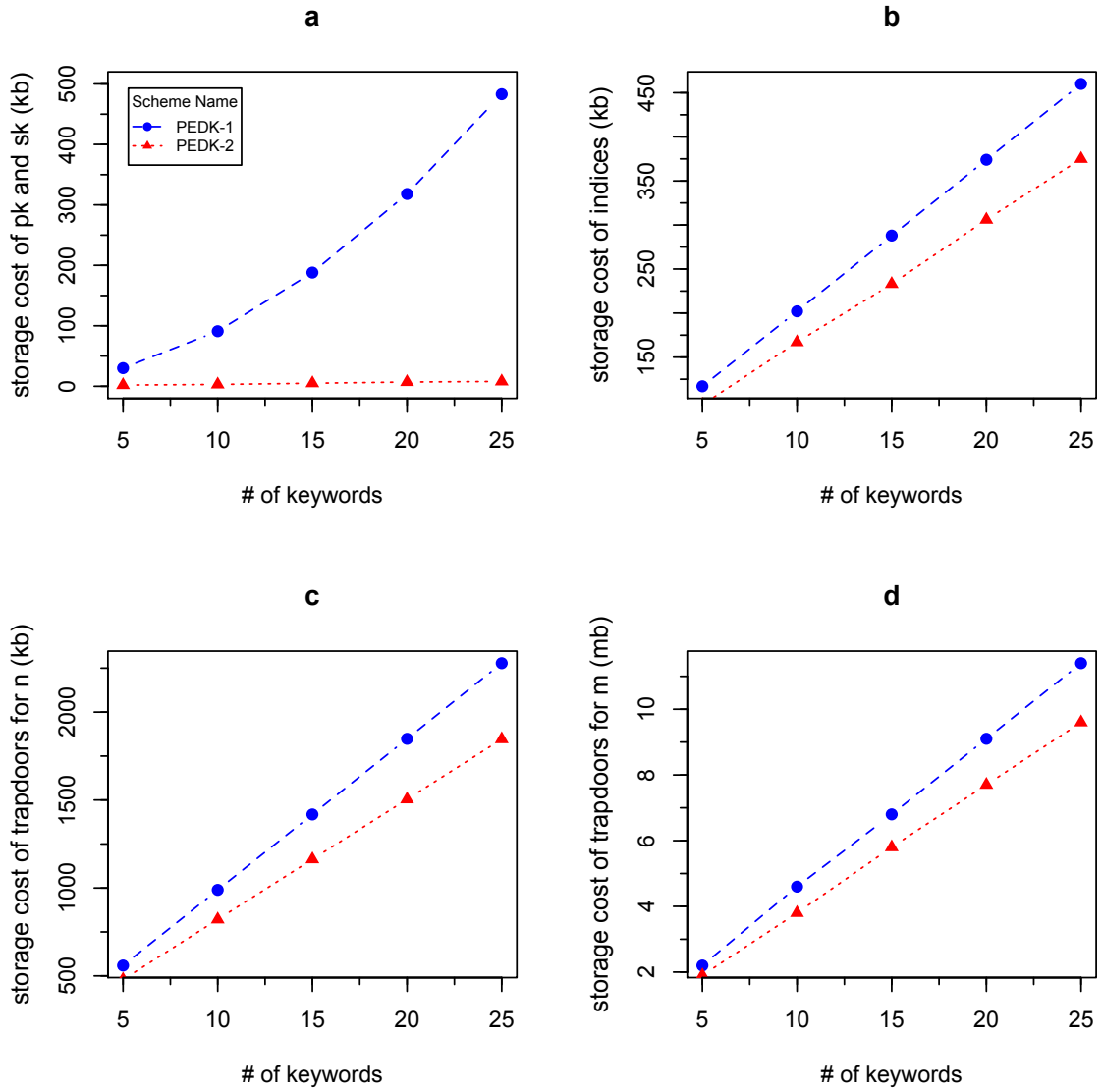

Figure 5. Impact of n on the storage cost of keys (**a**), indices (**b**) and trapdoors (**c**) in PEDK-1 and PEDK-2 ($D = 100, m = 5, n = \{5, 10, 15, 20, 25\}$); impact of m on the storage cost of trapdoors (**d**) in PEDK-1 and PEDK-2 ($D = 100, n = 5, m = \{5, 10, 15, 20, 25\}$).

6. Conclusions

In this paper, we proposed a new approach that can convert the operation of disjunctive keyword search into inner product operations among vectors. Based on this approach, we give two concrete schemes which are better than previous schemes and proven to be secure under an adaptive security model.

To justify the efficiency of the proposed schemes, we present detailed theoretical analysis and experimental results. These results show that: (1) compared with previous PEDK schemes, the proposed two schemes are more efficient; (2) the first proposed scheme based on the IPE has better performance in testing phase; and (3) the second one achieves better time and space complexities on key generation, index building and trapdoor generation by taking advantage of the composite order bilinear groups. Moreover, because each document has its own encrypted index, we can easily accelerate the search process by utilizing the technique of parallel computation. The search process is

preformed by the cloud server, which has strong computing power. Considering this actual situation and the experimental result, we argue that our scheme is more practical in the cloud platform.

In conclusion, our proposed schemes are beneficial for applications with computation and memory limitations. The future work is to create efficient encryption schemes supporting more complex query condition, e.g., simple boolean keyword search like $q_1 \vee q_2 \wedge q_3$.

Author Contributions: Conceptualization, Y.Z. and Y.L.; Methodology, Y.Z. and Y.L.; Software, Y.Z. and Y.W.; Validation, Y.Z., Y.L. and Y.W.; Formal Analysis, Y.Z. and Y.L.; Investigation, Y.Z. and Y.L.; Resources, Y.Z. and Y.L.; Data Curation, Y.Z. and Y.W.; Writing—Original Draft Preparation, Y.Z.; Writing—Review and Editing, Y.Z., Y.L. and Y.W.; Visualization, Y.Z.; Supervision, Y.Z.; Project Administration, Y.Z.; Funding Acquisition, Y.Z. and Y.L.

Funding: This research was funded by the National Natural Science Foundation of China under Grant No. 61402393, 61601396, and Nanhu Scholars Program for Young Scholars of XYNU.

Acknowledgments: The authors gratefully acknowledge the National Natural Science Foundation of China under Grant No. 61402393, 61601396, and Nanhu Scholars Program for Young Scholars of XYNU.

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

Abbreviations

The following abbreviations are used in this manuscript:

PECK	Public key encryption with conjunctive keyword search
PEDK	Public key encryption with disjunctive keyword search
PECDK	Public key encryption with conjunctive and disjunctive keyword search
IPE	Inner product encryption
SE	Searchable encryption
SSE	Searchable symmetric key encryption
SPE	Searchable public key encryption
PK	Public key
SK	Secret key
DPVS	Dual pairing vector space
XYNU	Xinyang normal university

Appendix A

Proof of Lemma 1.

Proof. Given $D = (N = p_1p_2p_3p_4, G, G_T, \hat{e}, g_1 \in G_{p_1}, g_3 \in G_{p_3}, g_4 \in G_{p_4})$, B simulate $Game_{Real}$ with A. With probability ϵ, A can generate a keyword w and $\{a_0, a_1, \ldots, a_n\}$ such that $\sum_{i=0}^{n} a_i H_1(w)^i mod N \neq 0$ and $\sum_{i=0}^{n} a_i H_1(w)^i mod p_2 = 0$. By computing $a = gcd(\sum_{i=0}^{n} a_i H_1(w)^i, N)$, B uses w and $\{a_0, a_1, \ldots, a_n\}$ to generate a non-trivial factor of N . Let $b = \frac{N}{a}$. Considering that p_2 divides a and $N = ab = p_1p_2p_3p_4$, we focus on three cases:

- Case 1: p_1 divides b,
- Case 2: p_1 can not divide b and p_4 can divide b,
- Case 3: $a = p_1p_2p_4$ and $b = p_3$.

The probability of at least one of these cases occurring is larger than $\frac{\epsilon}{3}$. In case 1, given D and T where $T \in G_{p_1}$ or $T \in G_{p_1p_2}$, B computes T^b. If T^b is the identity element of G_T, then $T \in G_{p_1}$. Otherwise, $T \in G_{p_1p_2}$. Therefore, B can break the GSD assumption.

Case 2 is the same as Case 1 except that $T \in G_{p_4}$ or $T \in G_{p_2p_4}$. B computes T^b. If T^b is the identity element of G_T, then $T \in G_{p_4}$. Otherwise, $T \in G_{p_2p_4}$.

In case 3, given $D = (N = p_1p_2p_3p_4, G, G_T, \hat{e}, D_1D_2 \in G_{p_1p_2}, B_2B_3 \in G_{p_2p_3}, g_1 \in G_{p1}, g_3 \in G_{p3}, g_4 \in G_{p4})$ and T where $T \in G_{p_1p_3}$ or $T \in G_{p_1p_2p_3}$, B computes $\hat{e}(T, (B_2B_3)^b)$. If $\hat{e}(T, (B_2B_3)^b)$ is the identity element of G_T, then $T \in G_{p_1p_3}$. Otherwise, $T \in G_{p_1p_2p_3}$. □

Proof of Lemma 2.

Proof. Given $D = (N = p_1p_2p_3p_4, G, G_T, \hat{e}, g_1 \in G_{p1}, g_3 \in G_{p3}, g_4 \in G_{p4})$, and T where $T \in G_{p_1}$ or $T \in G_{p_1p_2}$, B can simulate $Game_0$ or $Game_{Real}$ with A. To generate the public key, B chooses the function H_1 and random exponents a, c, u, a_i, b_i, α_i and $\beta_i \in Z_N$ for each $i \in [0, n]$ and sets $A_1{}^{\beta_i}A_{4i} = g_1{}^{a\beta_i}g_4{}^{b_i}$, $A_1{}^{\alpha_i}B_{4i} = g_1{}^{a\alpha_i}g_4{}^{a_i}$ and $U_1U_4 = g_1^u g_4^c$ for each $i \in [0, n]$. Obviously, $U_1 = g_1^u$. B sends public key $\{N, H_1, U_1U_4, A_1{}^{\beta_i}A_{4i}, A_1{}^{\alpha_i}B_{4i}, g_4\}$ to A. When A asks B for a key of a keyword set $Q = \{q_1, q_2, \ldots, q_m\}$ where $m \leq n$, B chooses random exponents $r_j, k_{j0}, k_{j1}, \ldots, k_{jn}, k_j \in Z_N$ and two random vector bases $B = (\overrightarrow{b_1}, \ldots, \overrightarrow{b_m})^T, B^* = (\overrightarrow{b_1}^*, \ldots, \overrightarrow{b_m}^*)^T$. Suppose that $\overrightarrow{b_t} = \{b_{t1}, b_{t2}, \ldots, b_{tm}\}$ and $\overrightarrow{b_t}^* = \{b^*_{t1}, b^*_{t2}, \ldots, b^*_{tm}\}$ where $t \in [1, m]$, $K_{ji} = g_1^{u\frac{r_1b_{1j}H_1(q_1)^i + r_2b_{2j}H_1(q_2)^i + \ldots + r_mb_{mj}H_1(q_m)^i}{\beta_i}} g_3^{k_{ji}}$ and $K_j = g_1{}^{a\sum_{i=0}^{n}\frac{\alpha_i(r_1b_{1j}H_1(q_1)^i + r_2b_{2j}H_1(q_2)^i + \ldots + r_mb_{mj}H_1(q_m)^i)}{\beta_i}} g_3^{k_j}$ can be obtained, where $i \in [0, n], j \in [1, m]$.

After that, A sends B two challenge keyword sets, $W^{(0)} = \{w_1^{(0)}, w_2^{(0)}, \ldots, w_n^{(0)}\}$ and $W^{(1)} = \{w_1^{(1)}, w_2^{(1)}, \ldots, w_n^{(1)}\}$. B randomly chooses $\beta \in \{0, 1\}$ and computes a n-degree polynomial $f(x)$ mentioned above. Given $C_{40}, C_{41}, \ldots, C_{4n}$ and $C_4 \in G_{p_4}$ randomly, B generates the index as follows:

$$C_{1i} = T^{a(\beta_i x_i^{(\beta)} + \alpha_i)}C_{4i} \text{ and } C_2 = T^u C_4 \text{ where } i \in [0, n].$$

If $T \in G_{p_1p_2}$, suppose that $T = g_1^s g_2^{s'}$, then this is a semi-functional index with $z_{ci} = s'a(\beta_i x_i^{(\beta)} + \alpha_i)$ and $z_c = s'u$. Since a, u, α_i, β_i are chosen randomly modulo N, we can find that the value of z_{ci} and z_c module p_2 is unrelated with the value of a, u, α_i, β_i module p_1 where $i \in [0, n]$. The index is normal if $T \in G_{p_1}$. Therefore, if A can distinguish the $Game_{Restricted}$ from $Game_0$ with advantage ϵ, then the advantage of B using the output of A to break Assumption 1 is ϵ. □

Proof of Lemma 3.

Proof. Given $D = (N = p_1p_2p_3p_4, G, G_T, \hat{e}, g_1 \in G_{p1}, g_3 \in G_{p3}, g_4 \in G_{p4}, D_1D_2 \in G_{p_1p_2}, B_2B_3 \in G_{p_2p_3})$ and T where $T \in G_{p_1p_3}$ or $T \in G_{p_1p_2p_3}$, B can simulate $Game_{k-1}$ or $Game_k$ with A. Choosing random exponents a, u, c, a_i, b_i, α_i and $\beta_i \in Z_N$ for each $i \in [0, n]$, B sets $A_1{}^{\beta_i}A_{4i} = g_1{}^{a\beta_i}g_4{}^{b_i}$, $A_1{}^{\alpha_i}B_{4i} = g_1{}^{a\alpha_i}g_4{}^{a_i}$ and $U_1U_4 = g_1^u g_4^c$ for each $i \in [0, n]$, and sends $pk = \{N, H_1, A_1{}^{\beta_i}A_{4i}, A_1{}^{\alpha_i}B_{4i}, g_4, U_1U_4\}$ to A. Any time A asks the l-th trapdoor for a keyword set $Q^{(l)} = (q_1^{(l)}, q_2^{(l)}, \ldots, q_m^{(l)})$, where $m \leq n$, B generates the normal trapdoor or the semi-function trapdoor for the keyword set $Q^{(l)}$.

For $l < k$, B creates a semi-function trapdoor. Choosing random exponents $r_j^{(l)}, z_j^{(l)}, t_{ji}^{(l)} \in Z_N$ and two random vector bases $B^{(l)} = (\overrightarrow{b_1^{(l)}}, \ldots, \overrightarrow{b_m^{(l)}})^T$ and $B^{(l)*} = (\overrightarrow{b_1^{(l)}}^*, \ldots, \overrightarrow{b_m^{(l)}}^*)^T$, where $i \in [0, n], j \in [1, m]$, suppose that $\overrightarrow{b_t^{(l)}} = \{b_{t1}^{(l)}, b_{t2}^{(l)}, \ldots, b_{tm}^{(l)}\}$ and $\overrightarrow{b_t^{(l)}}^* = \{b_{t1}^{(l)*}, b_{t2}^{(l)*}, \ldots, b_{tm}^{(l)*}\}$, B computes:

$$K_{ji} = g_1^{u\frac{r_1^{(l)}b_{1j}^{(l)}H_1(q_1^{(l)})^i + r_2^{(l)}b_{2j}^{(l)}H_1(q_2^{(l)})^i + \ldots + r_m^{(l)}b_{mj}^{(l)}H_1(q_m^{(l)})^i}{\beta_i}} (B_2B_3)^{t_{ji}^{(l)}} \text{ and}$$

$$K_j = g_1{}^{a\sum_{i=0}^{n}\frac{\alpha_i(r_1^{(l)}b_{1j}^{(l)}H_1(q_1^{(l)})^i + r_2^{(l)}b_{2j}^{(l)}H_1(q_2^{(l)})^i + \ldots + r_m^{(l)}b_{mj}^{(l)}H_1(q_m^{(l)})^i)}{\beta_i}} (B_2B_3)^{z_j^{(l)}}.$$

It is identical to the semi-functional trapdoor with $g_2^{\xi z_{kj}} = B_2{}^{z_j^{(l)}}$ and $g_2^{\xi z_{kji}} = B_2{}^{t_{ji}^{(l)}}$.

For $l > k$, B generates a normal trapdoor. Randomly selecting exponents $r_j^{(l)}, z_j^{(l)}$ and $t_{ji}^{(l)} \in Z_N$ and two random vector bases $B^{(l)} = (\overrightarrow{b_1^{(l)}}, \ldots, \overrightarrow{b_m^{(l)}})^T, B^{(l)*} = (\overrightarrow{b_1^{(l)}}^*, \ldots, \overrightarrow{b_m^{(l)}}^*)^T$, where $i \in [0, n], j \in [1, m]$, suppose that $\overrightarrow{b_t^{(l)}} = \{b_{t1}^{(l)}, b_{t2}^{(l)}, \ldots, b_{tm}^{(l)}\}$ and $\overrightarrow{b_t^{(l)}}^* = \{b_{t1}^{(l)*}, b_{t2}^{(l)*}, \ldots, b_{tm}^{(l)*}\}$, B sets:

$$K_{ji} = g_1^{u\frac{r_1^{(l)}b_{1j}^{(l)}H_1(q_1^{(l)})^i + r_2^{(l)}b_{2j}^{(l)}H_1(q_2^{(l)})^i + \ldots + r_m^{(l)}b_{mj}^{(l)}H_1(q_m^{(l)})^i}{\beta_i}} (g_3)^{t_{ji}^{(l)}},$$

$$K_j = g_1^{a\sum_{i=0}^{n} \frac{\alpha_i(r_1^{(l)}b_{1j}^{(l)}H_1(q_1^{(l)})^i + r_2^{(l)}b_{2j}^{(l)}H_1(q_2^{(l)})^i + \ldots + r_m^{(l)}b_{mj}^{(l)}H_1(q_m^{(l)})^i)}{\beta_i}} (g_3)^{z_j^{(l)}}.$$

To create the k-th requested trapdoor, B chooses random exponents $r_j^{(k)}$, $z_j^{(k)}$ and $t_{ji}^{(k)} \in Z_N$ and two random vector bases $B^{(k)} = (\overrightarrow{b_1^{(k)}}, \ldots, \overrightarrow{b_m^{(k)}})^T$, $B^{(k)*} = (\overrightarrow{b_1^{(k)}}_*, \ldots, \overrightarrow{b_m^{(k)}}_*)^T$, where $i \in [0,n]$, $j \in [1,m]$. Suppose that $\overrightarrow{b_t^{(k)}} = \{b_{t1}^{(k)}, b_{t2}^{(k)}, \ldots, b_{tm}^{(k)}\}$ and $\overrightarrow{b_t^{(k)}}_* = \{b_{t1}^{(k)*}, b_{t2}^{(k)*}, \ldots, b_{tm}^{(k)*}\}$, B sets:

$$K_{ji} = T^{u\frac{r_1^{(k)}b_{1j}^{(k)}H_1(q_1^{(k)})^i + r_2^{(k)}b_{2j}^{(k)}H_1(q_2^{(k)})^i + \ldots + r_m^{(k)}b_{mj}^{(k)}H_1(q_m^{(k)})^i}{\beta_i}} (g_3)^{t_{ji}^{(k)}},$$

$$K_j = T^{a\sum_{i=0}^{n} \frac{\alpha_i(r_1^{(k)}b_{1j}^{(k)}H_1(q_1^{(k)})^i + r_2^{(k)}b_{2j}^{(k)}H_1(q_2^{(k)})^i + \ldots + r_m^{(k)}b_{mj}^{(k)}H_1(q_m^{(k)})^i)}{\beta_i}} (g_3)^{z_j^{(k)}}.$$

Obviously, it can be found that:

$$z_{kji} = u\frac{r_1^{(k)}b_{1j}^{(k)}H_1(q_1^{(k)})^i + r_2^{(k)}b_{2j}^{(k)}H_1(q_2^{(k)})^i + \ldots + r_m^{(k)}b_{mj}^{(k)}H_1(q_m^{(k)})^i}{\beta_i},$$

$$z_{kj} = a\sum_{i=0}^{n} \frac{\alpha_i(r_1^{(k)}b_{1j}^{(k)}H_1(q_1^{(k)})^i + r_2^{(k)}b_{2j}^{(k)}H_1(q_2^{(k)})^i + \ldots + r_m^{(k)}b_{mj}^{(k)}H_1(q_m^{(k)})^i)}{\beta_i}.$$

After the trapdoor request phase, A sends B two challenge keyword sets, $W^{(0)} = \{w_1^{(0)}, w_2^{(0)}, \ldots, w_n^{(0)}\}$ and $W^{(1)} = \{w_1^{(1)}, w_2^{(1)}, \ldots, w_n^{(1)}\}$. B randomly chooses a $\beta \in \{0,1\}$ and generates the semi-functional index. B computes a n-degree polynomial $f(x)$ mentioned above. Given C_{40}, C_{41}, …, C_{4n} and $C_4 \in G_{p_4}$ randomly, B generates the index as follows:

$$C_{1i} = (D_1D_2)^{a(\beta_i a_i^{(\beta)} + \alpha_i)} C_{4i} \text{ and } C_2 = (D_1D_2)^u C_4,$$

where $i \in [0,n]$.

Note that this sets $z_c = u$ and $z_{ci} = a(\beta_i x_i^{(\beta)} + \alpha_i)$. Since a, u, α_i, β_i are chosen randomly modulo N, we can find that the value of z_{ci} and z_c module p_2 is unrelated with the value of a, u, α_i, β_i module p_1, where $i \in [0,n]$.

Let $M'_j = \sum_{i=0}^{n} z_{ci}z_{kji} - z_c z_{kj} = au(\sum_{i=0}^{n} r_1^{(k)}b_{1j}^{(k)}a_i^{(\beta)}H_1(q_1^{(k)})^i + \sum_{i=0}^{n} r_2^{(k)}b_{2j}^{(k)}a_i^{(\beta)}H_1(q_2^{(k)})^i + \ldots + \sum_{i=0}^{n} r_m^{(k)}b_{mj}^{(k)}a_i^{(\beta)}H_1(q_m^{(k)})^i)$. If it exists $y \in [1,m]$ such that $\sum_{i=0}^{n} a_i^{(\beta)}H_1(q_y^{(k)})^i = 0 mod p_2$, there is $\prod_{j=1}^{m} M_j'^{b_{yj}^{(k)*}} = 0 mod p_2$. It means that, if $\sum_{i=0}^{n} a_i^{(\beta)}H_1(q_y^{(k)})^i = 0 mod p_2$, then A has queried an invalid key according to the additional modular restriction. Therefore, according to the $Game_{Restricted}$, as long as it does not exist $j \in [1,m]$ such that $\sum_{i=0}^{n} a_i^{(\beta)}H_1(q_j^{(k)})^i = 0 mod p_2$, we can find that z_{kj}, z_c, z_{kji} and z_{ci} are randomly distributed to A for each $i \in [0,n]$, $j \in [1,m]$.

In addition, we observe that, if B tries to verify whether the k-th trapdoor is semi-functional by generating a semi-functional index of a keyword set $W^{(k)} = \{w_1^{(k)}, w_2^{(k)}, \ldots, w_n^{(k)}\}$ such that $W^{(k)} \cap Q^{(k)} \neq \emptyset$, then B can find that the test algorithm can still work whether the k-th key is semi-functional or not due to $\prod_{j=1}^{m} M_j'^{b_{yj}^{(k)*}} = 0 mod p_2$, if $Q_j^{(k)} \in W^{(k)}$.

Therefore, we argue that, if $T \in G_{p_1p_3}$, then B has simulated $Game_{k-1}$ properly. If $T \in G_{p_1p_2p_3}$, then B has simulated $Game_k$ properly . Thus, we can find that, if A can distinguish the $Game_{k-1}$ from $Game_k$ with advantage ϵ, then the advantage of B using the output of A to break Assumption 1 is ϵ. □

Proof of Lemma 4.

Proof. Given $D = (N = p_1p_2p_3p_4, G, G_T, \hat{e}, g_2 \in G_{p_2}, g_3 \in G_{p_3}, g_4 \in G_{p_4}, A_1A_4 \in G_{p_1p_4}, E_1E_2 \in G_{p_1p_2})$ and T where $T \in G_{p_2p_4}$ or $T \in G_{p_1p_2p_4}$, B can simulate $Game_{Final_{k-1}}$ or $Game_{Final_k}$ with A. B chooses the function H_1 and random exponents c, u, α, a_i, b_i, α_i, $\beta_i \in Z_N$ for $i \in [0,n]$, and generates $A_1{}^{\beta_i}A_{4i} = (A_1A_4)^{\beta_i}g_4^{a_i}$, $A_1{}^{\alpha_i}B_{4i} = (A_1A_4)^{\alpha_i}g_4^{b_i}$, $U_1U_4 = (A_1A_4)^u g_4^c$. This implicitly sets $U_1 = A_1^u$. Then, B sends public key $\{N, H_1, U_1U_4, A_1{}^{\beta_i}A_{4i}, A_1{}^{\alpha_i}B_{4i}, g_4\}$ to A. Whenever A asks B for a key of a keyword set $Q = \{q_1, q_2, \ldots, q_m\}$ where $m \leq n$, B chooses random exponents r_j,

$k_{j0}, k_{j1}, \ldots, k_{jn}, k_j, w_{j0}, w_{j1}, \ldots, w_{jn}, w_j \in Z_N$ and two random vector bases $B = (\overrightarrow{b_1}, \ldots, \overrightarrow{b_m})^T, B^* = (\overrightarrow{b_1}^*, \ldots, \overrightarrow{b_m}^*)^T$. Suppose that $\overrightarrow{b_t} = \{b_{t1}, b_{t2}, \ldots, b_{tm}\}$, $\overrightarrow{b_t}^* = \{b^*_{t1}, b^*_{t2}, \ldots, b^*_{tm}\}$, where $t \in [1, m]$, B creates the semi-functional trapdoor as follows:

$$K_{ji} = (E_1 E_2)^{u \frac{r_1 b_{1j} H_1(q_1)^i + r_2 b_{2j} H_1(q_2)^i + \ldots + r_m b_{mj} H_1(q_m)^i}{\beta_i}} g_2^{w_{ji}} g_3^{k_{ji}},$$

$$K_j = (E_1 E_2)^{a \sum_{i=0}^{n} \frac{\alpha_i (r_1 b_{1j} H_1(q_1)^i + r_2 b_{2j} H_1(q_2)^i + \ldots + r_m b_{mj} H_1(q_m)^i)}{\beta_i}} g_2^{w_j} g_3^{k_j},$$

where $i \in [0, n], j \in [1, m]$.

At some point, A sends B two challenge keyword sets, $W^{(0)} = \{w_1^{(0)}, w_2^{(0)}, \ldots, w_n^{(0)}\}$ and $W^{(1)} = \{w_1^{(1)}, w_2^{(1)}, \ldots, w_n^{(1)}\}$. B randomly chooses $\beta \in \{0, 1\}$ and computes a n-degree polynomial $f(x)$ mentioned above. Given C_{40}, C_{41}, …, C_{4n} and $C_4 \in G_{p_4}$ randomly and random elements $x_i, y_i, r_2, s \in Z_N$ where $i \in [0, n]$, B generates the index as follows:

For each $i < k$:

$$C_{1i} = (A_1 A_4)^{x_i} g_2^{y_i} C_{4i} = A_1^{x_i} C_{24}^{(i)},$$

where $C_{24}^{(i)} = g_2^{y_i} C_{4i} A_4^{x_i}$.

For each $i > k$:

$$C_{1i} = (A_1{}^{\beta_i} A_{4i})^{s a_i^{(\beta)}} \times (A_1{}^{\alpha_i} B_{4i})^s \times g_2^{y_i} C_{4i} = A_1^{s(\beta_i a_i^{(\beta)} + \alpha_i)} C_{24}^{(i)},$$

where $C_{24}^{(i)} = g_2^{y_i} A_{4i}^{s a_i^{(\beta)}} B_{4i}^s C_{4i}$.

For $i = k$:

$$C_{1k} = (A_1{}^{\beta_k} A_{4k})^{s a_k^{(\beta)}} \times (A_1{}^{\alpha_k} B_{4k})^s \times g_2^{y_k} C_{4i} \times T = A_1^{s(\beta_k a_k^{(\beta)} + \alpha_k)} C_{24}^{(k)} T,$$

where $C_{24}^{(k)} = g_2^{y_k} A_{4k}^{s a_k^{(\beta)}} B_{4k}^s C_{4k}$.

For all i:

$$C_2 = (U_1 U_4)^s \times g_2^{r_2} C_4 = U_1^s R_{24},$$

where $R_{24} = g_2^{r_2} C_4 U_4^s$.

Since $A_1 A_4$ is chosen randomly in G_1 and $\alpha_i, \beta_i, y_i, u, s$ are chosen randomly in Z_N, we can find that the value of z_{ci} and z_c module p_2 is unrelated with the value of s, u, α_i, β_i module p_1, where $i \in [0, n]$.

If $T \in G_{p_2 p_4}$, there is a properly distributed semi-functional index with a challenge keyword set that its first $k-1$ elements are random while the rest elements are normal. In this case, B has properly simulated $Game_{Final_{k-1}}$. If $T \in G_{p_1 p_2 p_4}$, there is a properly distributed semi-functional index with a challenge keyword set that its first k elements are random and the rest elements are normal. In this case, B has simulated $Game_{Final_k}$ properly.

Therefore, we can find that, if A can distinguish the $Game_{Final_{k-1}}$ from $Game_{Final_k}$ with advantage ϵ, then the advantage of B using the output of A to break Assumption 1 is ϵ. □

References

1. Park, D.J.; Kim, K.; Lee, P.J. Public Key Encryption with Conjunctive Field Keyword Search. In Proceedings of the 5th International Workshop, WISA 2004, Jeju Island, Korea, 23–25 August 2004; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3325, pp. 73–86.
2. Hwang, Y.H.; Lee, P.J. Public Key Encryption with Conjunctive Keyword Search and Its Extension to a Multi-user System. In Proceedings of the First International Conference, Tokyo, Japan, 2–4 July 2007; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4575, pp. 2–22.
3. Boneh, D.; Waters, B. Conjunctive, Subset, and Range Queries on Encrypted Data. In Proceedings of the 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, 21–24 February 2007; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4392, pp. 535–554.

4. Katz, J.; Sahai, A.; Waters, B. Predicate encryption supporting disjunctions, polynomial equations, and inner products. *J. Cryptol.* **2013**, *26*, 191–224. [CrossRef]
5. Zhang, Y.; Lu, S. POSTER: Efficient Method for Disjunctive and Conjunctive Keyword Search over Encrypted Data. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, 3–7 November 2014; pp. 1535–1537.
6. Song, D.; Wagner, D.; Perrig, A. Practical Techniques for Searching on Encrypted Data. In Proceedings of the IEEE Symposium on Research in Security and Privacy 2000, Berkeley, CA, USA, 14–17 May 2000; IEEE Computer Society Press: Los Alamitos, CA, USA, 2000; pp. 44–55.
7. Goh, E.J. Secure indexes. *IACR Cryptol. ePrint Arch.* **2003**, *2003*, 216.
8. Li, J.; Lin, X.; Zhang, Y.; Han, J. KSF-OABE: Outsourced attribute-based encryption with keyword search function for cloud storage. *IEEE Trans. Ser. Comput.* **2017**, *10*, 715–725. [CrossRef]
9. Li, J.; Shi, Y.; Zhang, Y. Searchable Ciphertext-Policy Attribute-Based Encryption with Revocation in Cloud Storage. *Int. J. Commun. Syst.* **2017**, *30*, e2942. [CrossRef]
10. 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 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2013; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2013; pp. 353–373.
11. Cash, D.; Jaeger, J.; Jarecki, S.; Jutla, C.S.; Krawczyk, H.; Rosu, M.C.; Steiner, M. Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation. *Netw. Distrib. Syst. Secur. Symp.* **2014**, *14*, 23–26.
12. Boldyreva, A.; Chenette, N.; Lee, Y.; O'neill, A. Order-Preserving Symmetric Encryption. In Proceedings of the 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, 26–30 April 2009; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; pp. 224–241.
13. Zerr, S.; Olmedilla, D.; Nejdl, W.; Siberski, W. Zerber+R:top-k retrieval from a confidential index. In Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, Saint Petersburg, Russia, 24–26 March 2009; pp. 439–449.
14. Wang, C.; Cao, N.; Ren, K.; Lou, W. Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data. *IEEE Trans. Parallel Distrib. Syst.* **2012**, *23*, 1467–1479. [CrossRef]
15. Fu, Z.; Sun, X.; Liu, Q.; Zhou, L.; Shu, J. Achieving Efficient Cloud Search Services: Multi-Keyword Ranked Search over Encrypted Cloud Data Supporting Parallel Computing. *IEICE Trans. Commun.* **2015**, *98*, 190–200. [CrossRef]
16. Xia, Z.; Wang, X.; Sun, X.; Wang, Q. A Secure and Dynamic Multi-Keyword Ranked Search Scheme over Encrypted Cloud Data. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 340–352. [CrossRef]
17. Boneh, D.; Di Crescenzo, G.; Ostrovsky, R.; Persiano, G. Public Key Encryption with Keyword Search. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, 2–6 May 2004; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3027, pp. 506–522.
18. Dan, B.; Franklin, M. Identity-Based Encryption from the Weil Pairing. In *Society for Industrial and Applied Mathematics*; Springer: Berlin/Heidelberg, Germany, 2001.
19. Abdalla, M.; Bellare, M.; Catalano, D.; Kiltz, E.; Kohno, T.; Lange, T.; Malone-Lee, J.; Neven, G.; Paillier, P.; Shi, H. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. In Proceedings of the 25th Annual International Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2005; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3621, pp. 205–222.
20. Li, J.; Yao, W.; Zhang, Y.; Qian, H.; Han, J. Flexible and fine-grained attribute-based data storage in cloud computing. *IEEE Trans. Ser. Comput.* **2017**, *10*, 785–796. [CrossRef]

21. Lewko, A.; Okamoto, T.; Sahai, A.; Takashima, K.; Waters, B. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, France, 30 May–3 June 2010; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6110, pp. 62–91.
22. Okamoto, T.; Takashima, K. Achieving short ciphertexts or short secret-keys for adaptively secure general inner-product encryption. *Des. Codes Cryptogr.* **2015**, *77*, 138–159. [CrossRef]
23. Zhang, B.; Zhang, F. An efficient public key encryption with conjunctive-subset keyword search. *J. Netw. Comput. Appl.* **2011**, *34*, 262–267. [CrossRef]
24. Song, C.; Liu, X.; Yan, Y. Efficient Public Key Encryption with Field-Free Conjunctive Keywords Search. In Proceedings of the 6th International Conference, INTRUST 2014, Beijing, China, 16–17 December 2014; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2014; pp. 394–406.
25. Wang, B.; Hou, Y.; Li, M.; Wang, H.; Li, H. Scalable multidimensional range search over encrypted cloud data with tree-based index. In Proceedings of the ACM Symposium on Information, Computer and Communications Security, Kyoto, Japan, 4–6 June 2014; pp. 111–122.
26. Ding, M.; Gao, F.; Jin, Z.; Zhang, H. An efficient Public Key Encryption with Conjunctive Keyword Search scheme based on pairings. In Proceedings of the 2012 3rd IEEE International Conference on Network Infrastructure and Digital Content, Beijing, China, 21–23 September 2013; pp. 526–530.
27. Hwang, M.S.; Hsu, S.T.; Lee, C.C. A New Public Key Encryption with Conjunctive Field Keyword Search Scheme. *Inf. Technol. Control.* **2014**, *43*, 277–288. [CrossRef]
28. Miao, Y.; Ma, J.; Liu, X.; Zhang, J.; Liu, Z. VKSE-MO: Verifiable keyword search over encrypted data in multi-owner settings. *Inf. Sci.* **2017**, *60*, 122105. [CrossRef]
29. Xu, P.; Wu, Q.; Wang, W.; Susilo, W.; Domingo-Ferrer, J.; Jin, H. Generating Searchable Public-Key Ciphertexts With Hidden Structures for Fast Keyword Search. *IEEE Trans. Inf. Forensics Secur.* **2017**, *10*, 1993–2006.
30. Xu, P.; He, S.; Wang, W.; Susilo, W.; Jin, H. Lightweight Searchable Public-key Encryption for Cloud-assisted Wireless Sensor Networks. *IEEE Trans. Ind. Inform.* **2017**, *14*, 3712–3723. [CrossRef]
31. Zhu, H.; Mei, Z.; Wu, B.; Li, H.; Cui, Z. Fuzzy Keyword Search and Access Control over Ciphertexts in Cloud Computing. In Proceedings of the 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, 3–5 July 2017; Lecture Notes in Computer Science; Springer: Cham, Germany, 2017; pp. 248–265.
32. Boneh, D.; Goh, E.; Nissim, K. Evaluating 2-dnf formulas on ciphertexts. In Proceedings of the Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, 10–12 February 2005; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3378, pp. 325–342.
33. Bellare, M.; Waters, B.; Yilek, S. Identity-based encryption secure against selective opening attack. In Proceedings of the 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, 28–30 March 2011; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011.
34. Waters, B. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Proceedings of the 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, 16–20 August 2009; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5677, pp. 619–636.
35. Caro, A.D. The Java Pairing Based Cryptography Library (JPBC). 2013. pp. 2–24. Available online: http://gas.dia.unisa.it/projects/jpbc/ (accessed on 4 December 2013).
36. Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; Hovy, E. Hierarchical Attention Networks for Document Classification. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, USA, 12–17 June 2017; pp. 1480–1489.