Hindawi
Mobile Information Systems
Volume 2020, Article ID 2904861, 15 pages
https://doi.org/10.1155/2020/2904861

Research Article

Efficient Boolean Keywords Search over Encrypted Cloud Data in Public Key Setting

Yu Zhang, Wei He, and Yin Li

School of Computer and Information Technology, Xinyang Normal University, Xinyang 464000, China

Correspondence should be addressed to Yin Li; yunfeiyangli@gmail.com

Received 5 November 2019; Revised 14 July 2020; Accepted 11 August 2020; Published 26 August 2020

Academic Editor: Laurence T. Yang

Searchable public key encryption- (SPE-) supporting keyword search plays an important role in cloud computing for data confidentiality. The current SPE scheme mainly supports conjunctive or disjunctive keywords search which belongs to very basic query operations. In this paper, we propose an efficient and secure SPE scheme that supports Boolean keywords search, which is more advanced than the conjunctive and disjunctive keywords search. We first develop a keyword conversion method, which can change the index and Boolean keywords query into a group of vectors. Then, through applying a technique so-called dual pairing vector space to encrypt the obtained vectors, we propose a concrete scheme proven to be secure under chosen keyword attack. Finally, we put forward a detailed theoretical and experimental analysis to demonstrate the efficiency of our scheme.

1. Introduction

Currently, thousands of information retrieval systems, such as e-mail systems, database management systems, and document management systems, are operating successfully in both the government and private sectors. As the data stored in these systems increase rapidly, more and more people want to migrate these data to cloud. To keep data privacy, users often encrypt these data before uploading them to the cloud. Since the encrypted data are difficult to retrieve, how to execute keyword search over encrypted data has attracted tremendous research attention over the past few years. Among these research studies, the searchable encryption (SE) is one of the most important techniques to address the issue of searching over encrypted data [1, 2].

The SE enables data users to retrieve the encrypted data of interest from a cloud server without decrypting the data. Commonly, SE is divided into two categories: one is searchable symmetric key encryption (SSE); the other is searchable public key encryption (SPE). During recent years, many SSE schemes have been proposed to support keyword search over encrypted data [3–6]. The key of SSE for encrypting data is the same as the key for generating search trapdoor. By contrast, the key of SPE for encrypting data is open to public, while the key for generating search trapdoor is only given to the authorized data receivers. Compared with SSE, SPE is more suitable for the situation in which there are many data senders and only a few data receivers, e.g., e-mail system [7], personal health record [8], and wireless sensor network [9]. As illustrated in Figure 1, in the scenario of e-mail system, the security requirements can be summarized as follows: (1) any data senders can generate encrypted e-mail data; (2) only data receiver can query and decrypt the encrypted e-mail data; (3) except the data receiver, none of the other entities, including the cloud server, can know the content of the encrypted e-mail data. Since security characteristics of SPE satisfy all these requirements in the above scenario, it is argued that SPE is very suitable for this application. Therefore, how to construct an efficient and secure SPE scheme supporting keyword search is always a hotspot in the field of SE.

1.1. Motivation. The very first SPE scheme supporting keyword search was introduced by Boneh et al., and it is so-called public key with keyword search (PEKS) [7]. However, their work only supports a single keyword search. In order to support more expressive query, many SPE schemes [10–12, 16] were proposed to realize advanced search, for

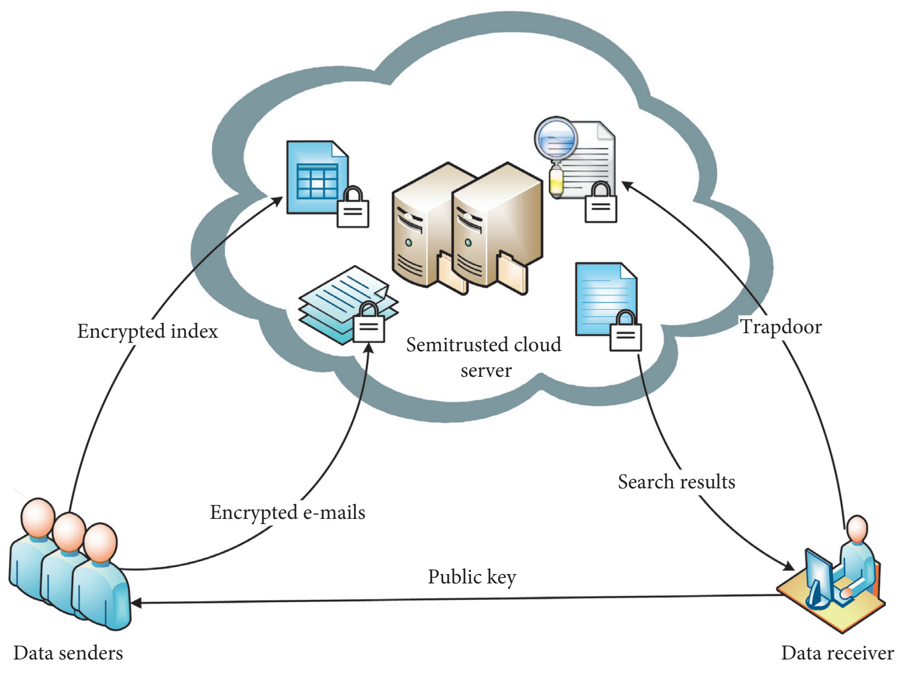

FIGURE 1: An example of the scenario of SPE: e-mail system.

example, conjunctive and disjunctive keywords search. In practice, most of the applications need more advanced keywords search function than the conjunctive and disjunctive keywords search. More precisely, many applications require Boolean keywords search. For example, in an e-mail system, users want to make a query like $(A \wedge B) \vee (C \wedge D)$, where A, B, C, and D are keywords. A naive thought is that a Boolean query can be obtained by remoulding a PECK or PEDK scheme, i.e., by combining the query results of conjunctive or disjunctive keywords search. However, we argue this simple method has many drawbacks. To better illustrate our motivation, based on a PEDK or PECK scheme, we construct a naive scheme supporting the Boolean keywords search like $q_1 \vee q_2 \wedge q_3$, where q_1, q_2, and q_3 are three keywords. We then briefly review the simple solution and explain why it is unsatisfactory.

The approach is that we first execute the query q_1 and the query $q_2 \wedge q_3$ by making use of the PECK scheme, respectively, and obtain the union of the results of q_1 query and $q_2 \wedge q_3$ query. However, this method will leak the trapdoors of q_1 and $q_2 \wedge q_3$. By utilizing the trapdoors, the search results of q_1 and $q_2 \wedge q_3$ are also leaked. Over time, the adversary may combine this information to derive the contents of user's documents. In addition, we also can execute the query of $q_1 \vee q_2$ and q_3 by making use of the PEDK scheme, respectively, and then obtain the intersection of the results of the query $q_1 \vee q_2$ and the query q_3. However, this method carries the same drawback.

1.2. Contribution. In this paper, we seek to construct a secure and efficient SPE scheme supporting Boolean keyword search which is not based on the PECK and PEDK schemes. We define a Boolean keywords search Q as a combination of conjunctive normal form (CNF) and disjunctive normal form (DNF), denoted by $Q = (\mathrm{CNF}_1) \vee (\mathrm{CNF}_2) \vee \cdots \vee (\mathrm{CNF}_m)$, where CNF_i is defined as $q_{i1} \wedge q_{i2} \wedge \cdots \wedge q_{in_i}$. Here, q_{ij} is a keyword, $i \in [1, m]$ and $j \in [1, n_i]$. This Boolean keywords search is more expressive than the conjunctive and disjunctive keywords search. The contributions of our work are summarized as follows:

(1) Inspired by the keyword conversion method introduced in [17], we create a novel keyword conversion method which can transform the index keyword set and Boolean query into an attribute and a predicate vector, respectively. These vectors can efficiently realize Boolean keywords search by an inner product operation. Moreover, the vector dimension is much less than that generated by adopting the previous method.

(2) Through elaborately applying the existing technique called dual pairing vector space (DPVS) to encrypt the attribute and predicate vectors, we propose a secure and efficient SPE scheme supporting Boolean keywords search (SPE-BKS), which can accomplish Boolean keywords search over encrypted data with a better search efficiency than the previous schemes.

Moreover, for security concern, we introduce a formal security definition for SPE-BKS and give a detailed proof to demonstrate that our scheme is secure against chosen keyword attack. To verify the efficiency of the proposed scheme, we conduct an experiment for comparing our scheme with some recent schemes over a real-world dataset (Enron Email Dataset).

1.3. Related Work. The first SPE scheme supporting keyword search was introduced by Boneh et al. [7]. They called it as public key encryption with keyword search (PEKS), which only supports a single keyword search. To support multi-keyword search, Park et al. proposed an SPE scheme supporting conjunctive keyword search, which is called public key encryption with conjunctive keywords search (PECK) [10]. In their scheme, each keyword is associated with a keyword field. The mechanism of the keyword field is based on two assumptions: one is that the keywords in a keyword field must be arranged in a preset order; the other is that the same keyword never appears in two different keyword fields of the same document. However, in many applications, the keyword field will make the multikeyword search unpractical. For instance, in an e-mail system, the keyword fields usually contain "From," "To," and "Title." Many e-mails may have the same keyword in different keyword fields, e.g., "From: LeBron James" and "To: James Harden." Moreover, the keywords in the keyword field "Title" may be organized in an alphabet order. To address this issue, the subsequence work is to create a PECK scheme without keyword field. In [11], Boneh and Waters proposed a public key encryption scheme called hidden vector encryption, which can efficiently support conjunctive keywords search without keyword field. After this, some efficient PECK schemes with better performance were proposed in [12–15]. To support disjunctive keyword search over encrypted data without keyword field, Katz et al. introduced a novel encryption scheme called predicate encryption supporting inner product, which is also named as inner product encryption (IPE) [16]. Through changing the index and query into an attribute and a predicate vector, respectively, a public key encryption with disjunctive keywords search (PEDK) scheme can be built based on the IPE scheme. Considering that the previous SPE schemes cannot use one trapdoor to realize conjunctive and disjunctive keywords search simultaneously, Zhang et al. proposed two public key encryption with conjunctive and disjunctive keyword search (PECDK) schemes [17, 18], which can efficiently support conjunctive and disjunctive keyword search at the same time. In order to support expressive query over encrypted data, based on the Paillier cryptosystem with threshold decryption (PCTD) [19], Yang et al. proposed an SPE scheme supporting versatile search query patterns, such as the range, conjunctive, disjunctive, and Boolean keywords search [20]. Miao et al. presented a hybrid keyword-field search scheme that supports both keyword search and range search simultaneously [21]. In addition, their scheme also provides an efficient key management mechanism to reduce the storage cost of keys. For the issue of fuzzy keyword search, Yang et al. designed a method to segment keyword according to the position of wildcards and proposed an SPE scheme supporting wildcard keyword search by combining the segmentation method and PCTD [22]. To support keyword search over arbitrary languages, Yang et al. realized a general method which can convert a variety of languages into a uniform big integer. By utilizing this conversion method and PCTD, they can carry out an SPE scheme supporting multikeyword rank search in arbitrary language [23]. To add the access control mechanism to SE, Li et al. created an attribute-based encryption (ABE) scheme which supports not only keyword search but also update operations for users ciphertext and secret key [24]. Then, they presented an outsourced ABE scheme supporting keyword search, which can transfer operations of decryption and key issuing to the cloud server partially [25]. He et al. proposed an SPE scheme which can control user's search permission according to an access control policy [26]. Miao et al. proposed an attribute-based keyword search scheme under a shared multiowner setting [27]. Zhang et al. proposed an SPE scheme achieving both Boolean keywords search and fine-grained search permission [28]. For the problem of tensor decomposition over encrypted data, by elaborately combining homomorphic encryption and block chain techniques, Feng et al. designed several schemes to implement different types of tensor decomposition, such as high-order Bi-Lanczos and Tucker decomposition [29–31]. To improve the efficiency of SPE, Hwang et al. created a more efficient SPE scheme, by replacing the operation of bilinear pairing with ElGamal encryption system [32]. Lu et al. proposed a certificate-less encryption supporting keyword search under a multi-recipient setting [33]. In order to obtain a better efficiency, their scheme avoids using a costly operation called bilinear pairing. Considering the scenario in which devices have limited resources, two secure and efficient energy-saving platforms were proposed to protect user's sensitive data [34, 35]. To resist the DoS attack, Li et al. gave an efficient remote user authentication and privacy-preserving scheme by adopting the technique called extended chaotic maps [36]. In order to improve search accuracy, Zhang et al. proposed an SPE scheme supporting semantic keywords search by adopting a method called "Word2vec" [37].

1.4. Organization. This paper is organized as follows. In Section 2, we give the framework of SPE-BKS and its security definition. Some basic tools are also provided in the section. In Section 3, the construction of SPE-BKS is given, and its security proof is also presented. The experimental and theoretical analysis is provided in Section 4. We conclude this paper in Section 5.

2. Preliminaries

In this section, we will give a formal definition of the framework and security model of SPE-BKS. In addition, we also briefly introduce some basic ingredients used in our scheme, including dual pairing vector space (DPVS), two important lemmas, and complexity assumption.

2.1. Framework of SPE-BKS. The SPE-BKS consists of three roles: data sender, data receiver, and cloud server. The responsibilities of these three roles are listed as follows:

(1) Data receiver generates the public key (pk) and secret key (sk) and sends the pk to the public. Data receiver also generates the trapdoor for any query of his/her interest and sends the trapdoor to the cloud server.

(2) For a message M with a keyword set W, data sender encrypts W to create the encrypted index I_W by using pk. Moreover, data sender will produce the encrypted message C for M. After this, data sender sends I_W and C to the cloud server.

(3) When the cloud server receives the trapdoor generated by the data receiver, the server tests the trapdoor against each encrypted index and returns the matched messages to the receiver.

According to the responsibilities of these three roles, we give a formal definition of the framework of SPE-BKS.

Definition 1. SPE-BKS consists of four polynomial-time algorithms (KeyGen, IndexBuild, Trapdoor, and Test) as follows:

(1) KeyGen (λ): this algorithm is run by the data receiver. It takes a security parameter λ as input and outputs pk and sk.

(2) IndexBuild (pk, W): this algorithm is executed by the data sender to encrypt the keyword set $W = \{w_1, w_2, \ldots, w_n\}$. It produces a searchable encrypted index I_W by using pk and W.

(3) Trapdoor (pk, sk, and Q): the algorithm is executed by the receiver to construct a trapdoor of Q. It takes pk, sk, and Q as input and outputs a trapdoor T_Q.

(4) Test (pk, I_W, and T_Q): for the query $Q = (\mathrm{CNF}_1) \vee (\mathrm{CNF}_2) \vee \cdots \vee (\mathrm{CNF}_m)$ and the index keyword set W, we define the function $f(W, Q)$ as follows: if there exists some $i \in [1, m]$ such that the keyword set in CNF_i is a subset of W, then $f(W, Q) = 1$. Otherwise, $f(W, Q) = 0$. This algorithm is run by the cloud server. It takes a trapdoor T_Q, a secure index I_W, and pk as input and outputs 1 if $f(W, Q) = 1$, or 0 otherwise.

2.1.1. Correctness.

For a query Q and a keyword set W, for pk, sk, I_W, and T_Q correctly generated by the algorithms KeyGen (λ), IndexBuild (pk, W), and Trapdoor (pk, sk, Q), respectively, the correctness property asks that the following two situations are needed to be met:

(1) If $f(W, Q) = 1$, Test (pk, I_W, T_Q) outputs 1

(2) If $f(W, Q) = 0$, Test (pk, I_W, T_Q) outputs 1 with negligible probability

In practice, data senders will send a message M with a keyword set W. The above algorithms aim to construct a secure and searchable index for W. For the message M, we can apply the symmetric encryption scheme, e.g., AES and triple DES, to protect the security of M. Like the previous SPE schemes, we only concentrate on searchable encryption part.

2.2. Security Definition of the SPE-BKS.

In this section, we present a formal definition for SPE-BKS, which defines a group of adversaries who can adaptively query the trapdoors of chosen keyword sets, and issue two challenge ciphertexts. The essential of the security of SPE-BKS is that the adversaries fail to distinguish these two ciphertexts based on the given trapdoors. Depending on the above description, inspired by the security definition of the previous SPE schemes, the security definition of SPE-BKS is given as follows.

Definition 2. An SPE-BKS scheme is adaptively index-hiding against chosen keyword attack if for all probabilistic polynomial-time (PPT) adversaries $\mathscr{A}$, the advantage of $\mathscr{A}$ in the following game is negligible for the security parameter λ:

(1) Setup: the challenger $\mathscr{C}$ runs the KeyGen (λ) algorithm to generate pk and sk and gives pk to the attacker $\mathscr{A}$.

(2) Phase 1: the attacker $\mathscr{A}$ can adaptively ask the challenger $\mathscr{C}$ for the trapdoor T_Q for any query Q of his choice.

(3) Challenge: $\mathscr{A}$ first selects two keyword sets $W^{(0)}$ and $W^{(1)}$ and sends them to $\mathscr{C}$. Suppose that $Q^{(1)}$, $Q^{(2)}$, $\ldots$, $Q^{(t)}$, $Q^{(2)}, \ldots, Q^{(t)}$ are the keyword queries which are queried to construct trapdoors in Phase 1; the only restriction is that these queries cannot distinguish these two challenge keyword sets. Then, $\mathscr{C}$ randomly chooses a bit $\beta \in \{0, 1\}$ and generates $I_\beta = \mathbf{IndexBuild}(pk, W^{(\beta)})$. Finally, $\{I_\beta, W^{(0)}, W^{(1)}\}$ are sent to $\mathscr{A}$.

(4) Phase 2: $\mathscr{A}$ continues to ask for trapdoor T_Q for any query Q of his/her choice under the restriction mentioned in the Challenge phase.

(5) Response: the attacker $\mathscr{A}$ outputs $\beta' \in \{0, 1\}$ and wins the game if $\beta' = \beta$.

Based on the above game, the advantage of $\mathscr{A}$ is defined as follows:

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

2.3. Prime Order Bilinear Group.

Let G, G_T be two cyclic groups of prime order p. There are three properties in the bilinear pairings map $\overline{e}$: $G \times G \longrightarrow G_T$ as follows:

(1) Bilinear: $\overline{e}(a^u, b^v) = \overline{e}(a, b)^{uv}$, where $a, b \in G$ and $u, v \in Z_p^*$

(2) Nondegenerate: if $g \in G$, then $\overline{e}(g, g) \in G_T$

(3) Computable: for any a, $b \in G$, $\overline{e}(a, b)$ can be efficiently computable

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

2.4. Dual Pairing Vector Space.

Suppose that $\overrightarrow{v} = (v_1, v_2, \ldots, v_l) \in Z_p^l$ and $g \in G$; we have $g^{\overrightarrow{v}} = (g^{v_1}, g^{v_2}, \ldots, g^{v_l})$. We can perform the scalar multiplication and vector addition in the exponent. For any $a \in Z_p$ and $\overrightarrow{v}$, $\overrightarrow{w} \in Z_p^l$, we have $g^{a\vec{v}} = (g^{av_1}, g^{av_2}, \ldots, g^{av_l})$ and $g^{\vec{v}+\vec{w}} = (g^{v_1+w_1}, g^{v_2+w_2}, \ldots, g^{v_l+w_l})$. We can also have $\bar{e}(g^{\vec{v}}, g^{\vec{w}}) = \bar{e}(g, g)^{\vec{v}\cdot\vec{w}}$ and $(g^{\vec{v}})^{\vec{w}} = g^{\vec{v}\cdot\vec{w}}$. Here, the dot product is taken as modulo p.

We will employ the concept of DPVS which is introduced in [39]. The notation used to describe DPVS is introduced in [40]. Suppose that $B = (\vec{b_1}, \vec{b_2}, \ldots, \vec{b_l})$ and $B^* = (\vec{b_1}^*, \vec{b_2}^*, \ldots, \vec{b_l}^*)$ are two random bases of Z_p^l, where l is a fixed dimension; if $\vec{b_i} \cdot \vec{b_j}^* = 0 \bmod p$ whenever $i \neq j$ and $\vec{b_i} \cdot \vec{b_i}^* = \lambda \pmod p$ for all $i \in n$, where λ is a random elements in Z_p, then we call B and B^* dual orthonormal bases. Obviously, for a generator $g \in G$, $\bar{e}(g^{\vec{b_i}}, g^{\vec{b_j}^*}) = 1$ whenever $i \neq j$, where 1 can be seen as the identity element of G_T.

2.5. Two Important Lemmas. We will introduce two important lemmas used in the security proof of our scheme. The first lemma is presented in [40]. To describe the lemma formally, first of all, we give some notations and definitions which are also introduced in [40]. Let t, l be two fixed positive integers where $t < l$, $A \in Z_p^{t\times t}$ be an invertible matrix and $S_t \subseteq \{1, 2, \ldots, l\}$ be a subset of size t. Suppose that B and B^* are random dual orthonormal bases; a new pair of dual orthonormal bases B_A and B_A^* was defined as follows.

Let B_t be a $l \times t$ matrix over Z_p whose columns are the vectors $\vec{b_i} \in B$ such that $i \in S_t$. We can easily find that $B_t A$ is also a $l \times t$ matrix. By keeping all of the vectors $\vec{b_i} \in B$ for $i \notin S_t$ and exchanging $\vec{b_i} \in B$ for $i \in S_t$ with the columns of $B_t A$, B_A is then constructed. Because $B_t^* (A^{-1})^T$ is also a $l \times t$ matrix, B_A^* also can be constructed by using the same method.

For a fixed dimension l and prime p, we denote randomly choosing a pair of dual orthonormal bases B and B^* by $(B, B^*) \xleftarrow{R} \mathrm{Dual}(Z_p^l)$. $\mathrm{Dual}(Z_p^l)$ can be viewed as a dual orthonormal bases set.

The first lemma is described as follows.

Lemma 1. *For any fixed positive integers $t < l$, any fixed invertible $A \in Z_p^{t\times t}$ and set $S_t \subseteq \{1, 2, \ldots, l\}$ of size t, if $(B, B^*) \xleftarrow{R} \mathrm{Dual}(Z_p^l)$, (B_A, B_A^*) is also distributed as a random sample from $\mathrm{Dual}(Z_p^l)$. In particular, the distribution of (B_A, B_A^*) is independent of A.*

The second lemma introduced in [39] (Lemma 23) is described as follows.

Lemma 2. *Let $C = \{(\vec{x}, \vec{v}) \mid \vec{x} \cdot \vec{v} \neq 0\} \subset V \times V^*$, where V is l-dimensional vector space, and F_p^l and V^* are its dual. For all $(\vec{x}, \vec{v}) \in C$, $(\vec{r}, \vec{w}) \in C$,*

$$\Pr_{Z \xleftarrow{R} F_p^{l\times l}, \rho, \tau \xleftarrow{R} F_p^{\times}} [\vec{x}(\rho U) = \vec{r} \wedge \vec{v}(\tau Z) = \vec{w}] = \frac{1}{s}, \quad (2)$$

where $U = (Z^{-1})^T$ and $s = \#C = (p^l - 1)(p^l - p^{l-1})$.

2.6. Complexity Assumption. In order to prove our scheme's security, subspace complexity assumption introduced in [40] is needed. This validity of this assumption is also given in [40].

For a fixed dimension $n' \geq 3$ and a prime p, the dual orthonormal bases B and B^* which are randomly chosen are denoted by $(B, B^*) \xleftarrow{R} \mathrm{Dual}\,(Z_p^l)$. $\mathrm{Dual}\,(Z_p^{n'})$ can be seen as a dual orthonormal bases set. For a positive integer $k \leq (n'/3)$, the definition of this assumption is described as follows.

Definition 3 (subspace complexity). Given a group generator $\mathcal{G}$, we define the following distribution:

$$\begin{gathered}(p, G, G_T, e) \xleftarrow{R} \mathcal{G}, \\ (B, B^*) \xleftarrow{R} \mathrm{Dual}(Z_p^l), \\ g \xleftarrow{R} G(\eta, \beta, \tau_1, \tau_2, \tau_3, \mu_1, \mu_2, \mu_3) \xleftarrow{R} Z_p, \end{gathered}$$
$$\begin{aligned}
U_1 &= g^{\mu_1 \vec{b_1} + \mu_2 \vec{b_{k+1}} + \mu_3 \vec{b_{2k+1}}}, \\
U_2 &= g^{\mu_1 \vec{b_2} + \mu_2 \vec{b_{k+2}} + \mu_3 \vec{b_{2k+2}}}, \ldots, U_k = g^{\mu_1 \vec{b_k} + \mu_2 \vec{b_{2k}} + \mu_3 \vec{b_{3k}}}, \\
V_1 &= g^{\tau_1 \eta \vec{b_1}^* + \tau_2 \beta \vec{b_{k+1}}^*}, \\
V_2 &= g^{\tau_1 \eta \vec{b_2}^* + \tau_2 \beta \vec{b_{k+2}}^*}, \ldots, V_k = g^{\tau_1 \eta \vec{b_k}^* + \tau_2 \beta \vec{b_{2k}}^*}, \\
W_1 &= g^{\tau_1 \eta \vec{b_1}^* + \tau_2 \beta \vec{b_{k+1}}^* + \tau_3 \vec{b_{2k+1}}^*}, \\
W_2 &= g^{\tau_1 \eta \vec{b_2}^* + \tau_2 \beta \vec{b_{k+2}}^* + \tau_3 \vec{b_{2k+2}}^*}, \ldots, W_k = g^{\tau_1 \eta \vec{b_k}^* + \tau_2 \beta \vec{b_{2k}}^* + \tau_3 \vec{b_{3k}}^*}, \\
D &= \Big(g^{\vec{b_1}}, g^{\vec{b_2}}, \ldots, g^{\vec{b_{2k}}}, g^{\vec{b_{3k+1}}}, g^{\vec{b_{3k+2}}}, \ldots, g^{\vec{b_{n'}}}, g^{\eta \vec{b_1}^*}, \\
&\quad g^{\eta \vec{b_2}^*}, \ldots, g^{\eta \vec{b_k}^*}, g^{\beta \vec{b_{k+1}}^*}, g^{\beta \vec{b_{k+2}}^*}, \ldots, g^{\beta \vec{b_{2k}}^*}, \\
&\quad g^{\vec{b_{2k+1}}^*}, g^{\vec{b_{2k+2}}^*}, \ldots, g^{\vec{b_{n'}}^*}, U_1, U_2, \ldots, U_k, \mu_3 \Big).
\end{aligned} \quad (3)$$

We assume that, for any PPT algorithm A with output in $\{0, 1\}$, the advantage of $\mathcal{A}$ defined by $\mathrm{Adv}_{\mathrm{Game}}^{\mathcal{A}} = |\Pr[\mathcal{A}(D, V_1, V_2, \ldots, V_k) = 1] - \Pr[\mathcal{A}(D, W_1, W_2, \ldots, W_k) = 1]|$ is negligible in the security parameter λ.

3. The Proposed SPE-BKS Scheme

In this section, we first introduce a keyword conversion method which converts the index and query keywords into a group of vectors. Then, through taking advantage of DPVS to encrypt these vectors, the construction of SPE-BKS is given. Finally, the security proof of our scheme is presented.

3.1. Keyword Conversion Method. Before describing the method, some notations will be introduced. Suppose that any keyword w can be expressed as a string in $\{0, 1\}^*$, we define a function H_1: $\{0, 1\}^* \longrightarrow Z_p^*$. 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.

For the index keyword set $W = \{w_1, w_2, \ldots, w_n\}$, we construct an equation of degree n with one unknown:

$$f(x) = (x - H_1(w_1))(x - H_1(w_2)), \dots, (x - H_1(w_n)) \\ = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 x^0. \tag{4}$$

According to the coefficient of the $f(x)$, the vector $\overrightarrow{W} = \{a_0, a_1, \dots, a_n\}$ for W is obtained.

For the query $Q = (\mathrm{CNF}_1) \vee (\mathrm{CNF}_2) \vee \cdots \vee (\mathrm{CNF}_m)$, we first split Q into a group of keyword sets. For each $\mathrm{CNF}_i = q_{i1} \wedge q_{i2} \wedge \cdots \wedge q_{in_i}$, we obtain a keyword set $Q_i = \{q_{i1}, q_{i2}, \dots, q_{in_i}\}$, where $i \in [1, m]$. For each Q_i, we can create a vector:

$$\overrightarrow{Q_i} = \left\{ \sum_{j=1}^{n_i} H_1(q_{ij})^0, \sum_{j=1}^{n_i} H_1(q_{ij})^1, \dots, \sum_{j=1}^{n_i} H_1(q_{ij})^n \right\}. \tag{5}$$

Note that if it exists some i such that $Q_i \subseteq W$, where $i \in [1, m]$, according to (4) and (5), it is not difficult to verify that $\overrightarrow{W} \cdot \overrightarrow{Q_i} = 0$.

As a result, we can test each Q_i in Q against W to make a Boolean keywords search. If $f(W, Q) = 1$, there is at least an $i \in [1, m]$ such that $\overrightarrow{W} \cdot \overrightarrow{Q_i} = 0$. Based on this property, a concrete SPE-BKS scheme will be proposed in the next section.

3.2. Construction.

3.2. Construction. According to Definition 1, we present a concrete construction of our SPE-BKS scheme:

(i) KeyGen: choosing a bilinear group G of a prime order p and setting $n' = 3n + 3$, the algorithm randomly selects a pair of dual orthonormal bases (B, B^*) from the dual orthonormal bases set $\mathrm{Dual}(Z_p^{n'})$, where $B = (\overrightarrow{b_1}, \overrightarrow{b_2}, \dots, \overrightarrow{b_{3n+3}})$, $B^* = (\overrightarrow{b_1}^*, \overrightarrow{b_2}^*, \dots, \overrightarrow{b_{3n+3}}^*)$ and $\overrightarrow{b_i} \cdot \overrightarrow{b_i}^* = \psi$ (mod p), where $i \in [1, 3n + 3]$. The algorithm outputs pk and sk as follows:

$$pk := \left\{ G, p, H_1, g^{\overrightarrow{c_i}} = g^{\overrightarrow{b_{i+1}}}, g^{\overrightarrow{d_i}} = g^{\overrightarrow{b_{i+n+2}}} \right\}, \\ sk := \left\{ g^{\overrightarrow{c_i}^*} = g^{\overrightarrow{b_{i+1}}^*}, g^{\overrightarrow{d_i}^*} = g^{\overrightarrow{b_{i+n+2}}^*} \right\}, \tag{6}$$

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

(i) IndexBuild: given a keyword set $W = \{w_1, w_2, \dots, w_n\}$, the algorithm constructs an n-degree polynomial $f(x) = (x - H_1(w_1))(x - H_1(w_2)), \dots, (x - H_1(w_n)) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 x^0$, where $H_1(w_1), H_1(w_2), \dots, H_1(w_n)$ are n roots of the equation $f(x) = 0$. Choosing two random elements $s_1, s_2 \in Z_p$, for the vector $\overrightarrow{W} = \{a_0, a_1, \dots, a_n\}$, this algorithm creates the index I_W as follows:

$$I_W = g^{s_1 \left(a_0 \overrightarrow{c_0} + a_1 \overrightarrow{c_1} + \cdots + a_n \overrightarrow{c_n}\right) + s_2 \left(a_0 \overrightarrow{d_0} + a_1 \overrightarrow{d_1} + \cdots + a_n \overrightarrow{d_n}\right)}. \tag{7}$$

(i) Trapdoor: given a query Q, this algorithm first generates a group of vectors $\overrightarrow{Q_1}, \overrightarrow{Q_2}, \dots, \overrightarrow{Q_m}$, $\overrightarrow{Q_2}, \dots, \overrightarrow{Q_m}$ by using the keyword conversion method introduced in Section 3.1. Then, it randomly chooses $r_1, r_2, \dots, r_m, t_1, t_2, \dots, t_m \in Z_p$ and an invertible matrix α. Suppose that $\alpha = (\overrightarrow{\alpha_1}, \overrightarrow{\alpha_2}, \dots, \overrightarrow{\alpha_m})^T$ and $\alpha^{-1} = (\overrightarrow{\alpha_1}^*, \overrightarrow{\alpha_2}^*, \dots, \overrightarrow{\alpha_m}^*)$ in which $\overrightarrow{\alpha_j} = (\alpha_{j1}, \alpha_{j2}, \dots, \alpha_{jm})^T$ and $\overrightarrow{\alpha_j}^* = (\alpha_{j1}^*, \alpha_{j2}^*, \dots, \alpha_{jm}^*)^T$, where $j \in [1, m]$, for each j, the trapdoor generation algorithm computes

$$K_j = g^{\sum_{i=0}^{n} \left(\sum_{\phi=1}^{m} \alpha_{\phi j} r_\phi \sum_{\theta=1}^{n_\phi} H_1(q_{\phi\theta})^i \right) \overrightarrow{c_i}^* + \sum_{i=0}^{n} \left(\sum_{\phi=1}^{m} \alpha_{\phi j} t_\phi \sum_{\theta=1}^{n_\phi} H_1(q_{\phi\theta})^i \right) \overrightarrow{d_i}^*}. \tag{8}$$

(i) The trapdoor of Q is $T_Q = \{K_1, K_2, \dots, K_m, \alpha^{-1}\}$.

(ii) Test: the test algorithm first computes $M_j = \overline{e}(I_W, K_j)$ for each $j \in [1, m]$. Suppose that $\overrightarrow{M} = \{M_1, M_2, \dots, M_m\}^T$; it outputs

$$\overrightarrow{M} = \overline{e}(g, g)^{\psi\left[(s_1 r_1 + s_2 t_1)\overrightarrow{\alpha_1} x_1 + (s_1 r_2 + s_2 t_2)\overrightarrow{\alpha_2} x_2 + \cdots + (s_1 r_m + s_2 t_m)\overrightarrow{\alpha_m} x_m\right]}, \tag{9}$$

where $x_j = \sum_{\theta=1}^{n_j} \sum_{i=0}^{n} a_i H_1(q_{j\theta})^i$ and $j \in [1, m]$. Based on $\overrightarrow{M}$, the test algorithm works as follows:

(1) Choose a counter v, and set $v = 1$.

(2) If $v > m$, then go to step (3); otherwise, the algorithm computes $D_j = \overrightarrow{M}^{\overrightarrow{\alpha_j}^*}$. If $D_j = 1$, the algorithm outputs 1 and ends. Otherwise, it sets $v = v + 1$ and goes to step (2).

(3) The algorithm outputs 0 and ends.

3.2.1. Correctness.

3.2.1. Correctness. Suppose that I_W and T_Q are correctly generated by the "IndexBuild" and "Trapdoor" algorithms, respectively, then we have the following equation:

$$M_j = \bar{e}\left(I_W, K_j\right) = \bar{e}(g,g)^{s_1 \sum_{i=0}^{n} \left[\left(a_i \sum_{\phi=1}^{m} \alpha_{\phi j} r_\phi \sum_{\theta=1}^{n_\phi} H_1 (q_{\phi\theta})^i\right) \overrightarrow{c_i} \cdot \overrightarrow{c_i}^*\right]} \times \bar{e}(g,g)^{s_1 \sum_{i=0}^{n} \left[\left(a_i \sum_{\phi=1}^{m} \alpha_{\phi j} r_\phi \sum_{\theta=1}^{n_\phi} H_1 (q_{\phi\theta})^i\right) \overrightarrow{d_i} \cdot \overrightarrow{d_i}^*\right]}$$
$$= \bar{e}(g,g)^{\psi\left[(s_1 r_1 + s_2 t_1)\alpha_{1j} x_1 + (s_1 r_2 + s_2 t_2)\alpha_{2j} x_2 + \cdots + (s_1 r_m + s_2 t_m)\alpha_{mj} x_m\right]}, \quad (10)$$

where $x_j = \sum_{\theta=1}^{n_j} \sum_{i=0}^{n} a_i H_1 (q_{j\theta})^i$ and $j \in [1, m]$.

Owing to $\overrightarrow{M} = \{M_1, M_2, \ldots, M_m\}^T$, based on the equation above, we have the following equation:

$$\overrightarrow{M} = \{M_1, M_2, \ldots, M_m\}^T$$
$$= \bar{e}(g,g)^{\psi\left[(s_1 r_1 + s_2 t_1)\overrightarrow{\alpha_1} x_1 + (s_1 r_2 + s_2 t_2)\overrightarrow{\alpha_2} x_2 + \cdots + (s_1 r_m + s_2 t_m)\overrightarrow{\alpha_m} x_m\right]}. \quad (11)$$

If there exists some $j \in [1, m]$ such that $Q_j = \left\{q_{j1}, q_{j2}, \ldots, q_{jn_j}\right\} \subseteq W$, it has $x_{j=0}$, which makes $D_j = \overrightarrow{M}^{\overrightarrow{\alpha_j}^*} = \bar{e}(g,g)^{\psi(s_1 r_j + s_2 t_j) x_j} = 1$, and, thus, the test algorithm outputs 1.

3.2.2. Application. According to the user's identity, the proposed scheme works as follows:

(1) *Data Receiver.* Data receiver runs the "KeyGen" function to generate *pk* and *sk*, and *pk* is open to the public. When data receiver wants to perform Boolean keywords search, the "Trapdoor" function is called to generate a trapdoor by using *sk* and a Boolean query condition. After this, the trapdoor is sent to the cloud server.

(2) *Data Sender.* For a document set, the data sender builds the secure index by calling the "IndexBuild" function and sends the index to the cloud server.

(3) *Cloud Server.* Upon receiving a trapdoor generated by the data receiver, the cloud server launches the "Test" function and returns documents associated with the query to the data receiver.

In the real world, any practical application that needs ciphertext retrieval can integrate our scheme to realize the function of searching on encrypted data.

3.3. Security. To prove the security of our SPE-BKS system, we adopt the dual system encryption method proposed in [41, 42]. According to this method, we give the construction of semifunctional index and trapdoor in our scheme. The semifunctional index and trapdoor will not be implemented in the real system but used in the proof:

(i) *Semifunctional Index.* Let $\overrightarrow{e_i} = \overrightarrow{b_{2n+3+i}}$, where $i \in\in [0, n]$ and $\overrightarrow{b_{2n+3+i}}$ is introduced in "KeyGen" algorithm. A normal index I_W' is constructed by the "IndexBuild" algorithm. Choosing random values $y_0, y_1, \ldots, y_n \in Z_p$, the semifunctional index is created as follows:

$$I_W = I_W' \times g^{y_0 \overrightarrow{e_0} + y_1 \overrightarrow{e_1} + \cdots + y_n \overrightarrow{e_n}}. \quad (12)$$

(i) *Semifunctional Trapdoor.* Let $\overrightarrow{e_i}^* = \overrightarrow{b_{2n+3+i}}^*$, where $i \in\in [0, n]$. A normal trapdoor $T_Q' = \{K_1', K_2', \ldots, K_m', \alpha^{-1}\}$ is constructed by the "Trapdoor" algorithm. Choosing random values $z_{j0}, z_{j1}, \ldots, z_{jn} \in Z_p$ where $j \in [1, m]$, the semifunctional trapdoor is created as follows:

$$K_j = K_j' \times g^{z_{j0} \overrightarrow{e_0}^* + z_{j1} \overrightarrow{e_1}^* + \cdots + z_{jn} \overrightarrow{e_n}^*}. \quad (13)$$

When using the semifunctional trapdoor to test the semifunctional index, the additional factors $M_j' = \bar{e}(g,g)^{z_{j0} y_0 + z_{j1} y_1 + \cdots + z_{jn} y_n}$ will be generated, where $j \in [1, m]$.

The security proof of our SPE-BKS scheme relies on subspace complexity assumption which is presented in Section 2.6. We will prove security by using a hybrid method which consists of a sequence of games. These games are described as follows:

(1) $\text{Game}_{\text{Real}}$: this game is the real security game.

(2) Game_k: for each $k \in [0, q]$, Game_k is similar to $\text{Game}_{\text{Real}}$ except that the index given to $\mathscr{A}$ is semifunctional and the first k trapdoors are semifunctional. The remaining trapdoors are normal. In Game_0, all the trapdoors given to $\mathscr{A}$ are normal and the index is semifunctional. In Game_q, the index and all trapdoors are semifunctional.

(3) $\text{Game}_{\text{Final}_k}$: suppose that a keyword set $W = \{w_1, w_2, \ldots, w_n\}$ is the challenge keyword set; we construct an n-degree polynomial $f(x) = (x - H_1(w_1))(x - H_1(w_2)), \ldots, (x - H_1(w_n)) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 x^0$ by using the function H_1, where $H_1(w_1), H_1(w_2), \ldots, H_1(w_n)$ are n roots of the equation $f(x) = 0$. Then, we define this game. For each $k \in [-1, n]$, $\text{Game}_{\text{Final}_k}$ is similar to Game_q except that index is a semifunctional encryption of a vector in which the first $k+1$ elements are random and the remaining elements are $\{a_{k+1}, a_{k+2}, \ldots, a_n\}$. $\text{Game}_{\text{Final}_{(-1)}}$ is a game such that the index is a semifunctional encryption of a real challenge keyword set, which is identical to Game_q. $\text{Game}_{\text{Final}_n}$ is a game such that the index is a semifunctional encryption of a random keyword set. We will show that these games are indistinguishable in the following lemmas.

Lemma 3. *Suppose that there exists a PPT algorithm $\mathscr{A}$ such that $Adv_{Game_{Real}}^{\mathscr{A}} - Adv_{Game_0}^{\mathscr{A}}$ is nonnegligible. Then, we can build a PPT algorithm $\mathscr{C}$ with nonnegligible advantage in breaking subspace complexity assumption, with $n' = 3n + 3$, $k = n + 1$.*

Lemma 4. *Suppose that there exists a PPT algorithm $\mathscr{A}$ such that $Adv_{Game_{k-1}}^{\mathscr{A}} - Adv_{Game_k}^{\mathscr{A}}$ is nonnegligible. Then, we can* *build a PPT algorithm $\mathscr{C}$ with nonnegligible advantage in breaking subspace complexity assumption, with $n' = 3n + 3$, $k = n + 1$.*

Lemma 5. *Suppose that there exists a PPT algorithm $\mathscr{A}$ such that $Adv^{\mathscr{A}}_{Game_{Final_{k-1}}} - Adv^{\mathscr{A}}_{Game_{Final_k}}$ is nonnegligible. Then, we can build a PPT algorithm $\mathscr{C}$ with nonnegligible advantage in breaking subspace complexity assumption, with $n' = 6$, $k = 2$.*

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

Theorem 1. *If subspace complexity assumption holds, then our SPE-BKS scheme is secure.*

Proof. If subspace complexity assumption holds, the real security game is indistinguishable from $\text{Game}_{\text{Final}_n}$ based on the previous lemmas. In $\text{Game}_{\text{Final}_n}$, the value of β is information-theoretically hidden from the attackers. Hence, we can state that the attackers can attain no advantage in breaking our SPE-BKS scheme. □

4. Performance Evaluation

In this section, we present a detailed experiment to demonstrate that our scheme can efficiently perform Boolean keywords search over the encrypted data. We implement our scheme in JAVA with Java Pairing-Based Cryptography (JPBC) Library [43]. 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 [43]. Our experiment is run on Intel® Core™ i7 CPU at 2.90 GHz processor and 16 GB memory size and is over a real-world e-mail dataset called Enron Email Dataset [44]. In our experiment, we randomly choose 1000 e-mails from the Enron Email Dataset and denote the number of documents by d ($d = 1000$). To show the efficiency of our scheme, we compare our scheme to three previous SPE schemes in terms of key generation, index building, trapdoor generation, and search. For simplicity, we denote these three schemes introduced in [17, 18, 20] by PECDK-1, PECDK-2, and YY18. These three SPE schemes can perform conjunctive, disjunctive, and Boolean keywords search over encrypted data.

4.1. Key Generation. From Figure 2(a), the time costs of key generation in PECDK-1 and our scheme are both linear with, while that in PECDK-2 is linear with $O(n)$. The reason for this phenomenon is the case that both our scheme and PECDK-1 adopt DPVS to generate group elements in G. Because the dimension of DPVS in our scheme is $3n$ while that in PECDK-1 is $4n$, the time cost of key generation in our scheme is less than that in PECDK-1. In addition, since the key generation algorithm in YY18 is independent of n, the time cost of key generation is not related to n. Although the time cost of key generation in our scheme is higher than that in PECDK-2 and YY18, it has little impact on our practical application since this algorithm only runs when system initialization and key pair replacement are carried out.

As shown in Figures 3(a) and 3(b), because both pk and sk contain group elements in G, the space cost for key pair in our scheme and PECDK-1 are both linear with the square of n. By contrast, the space cost for key pair in PECDK-2 is linear with $O(n)$. Besides, for YY18, since both pk and sk contain constant big integers, the space cost for key pair is not related to n. Though the storage cost of keys in our scheme is more than that in the other three schemes, our scheme still does not need much space to store the keys as these keys are stored only a few copies.

4.2. Index Building. From Figure 2(b), the time costs of index building in PECDK-1, PECDK-2, and our scheme are all linear with, while that in YY18 is linear with $O(n)$. For PECDK-2, the index building algorithm needs to convert the keywords into a matrix and then needs exponentiation computation of G to encrypt the keywords. For the proposed scheme and PECDK-1, they also require exponentiation computation of G owing to DPVS. More precisely, compared to PECDK-1, our scheme needs less time cost in index building since the dimension of DPVS in our scheme is less than that in PECDK-1. Besides, the time cost of index building in our scheme is slightly higher than that in PECDK-2 since our scheme needs exponentiation computations while PECDK-2 requires exponentiation computations. The reason for this phenomenon is that, compared to PECDK-2, our scheme needs more group elements to support more complex search function. Compared with YY18, our scheme needs more index building time since our scheme needs exponentiation computations while YY18 only runs the encryption algorithm of PCTD n times.

For the storage cost of indices, the group elements on G in the index for our scheme are linear with n. For YY18, since each document's index contains n ciphertexts generated by PCTD, the space cost of index building is linear with $O(n)$. By contrast, the group elements in the index for PECDK-1 and PECDK-2 are both linear with the square of n since the index structures for PECDK-1 and PECDK-2 are both a matrix. As shown in Figure 3(c), the storage costs of indices in our scheme and YY18 are linear with $O(n)$ while those in PECDK-1 and PECDK-2 are both linear with.

4.3. Trapdoor Generation. As shown in Figure 2(c), the time costs of trapdoor generation in PECDK-1, PECDK-2, YY18, and the proposed scheme are linear with m, m, and respectively. More precisely, for PECDK-1, the keywords in the query are first converted to be a vector, whose dimension is n. Then, this vector will be encrypted by using DPVS. Since the encryption operation needs exponentiation computations of G, the time cost of trapdoor generation in PECDK-1 is linear with. For PECDK-2, suppose that the number of keywords in the query is m, the query is converted to be a vector whose dimension is m, and each dimension needs one exponentiation computation on G. Thus, the time cost of trapdoor generation in PECDK-2 is linear with m. For YY18, if the query contains m keywords, the trapdoor algorithm will perform encryption

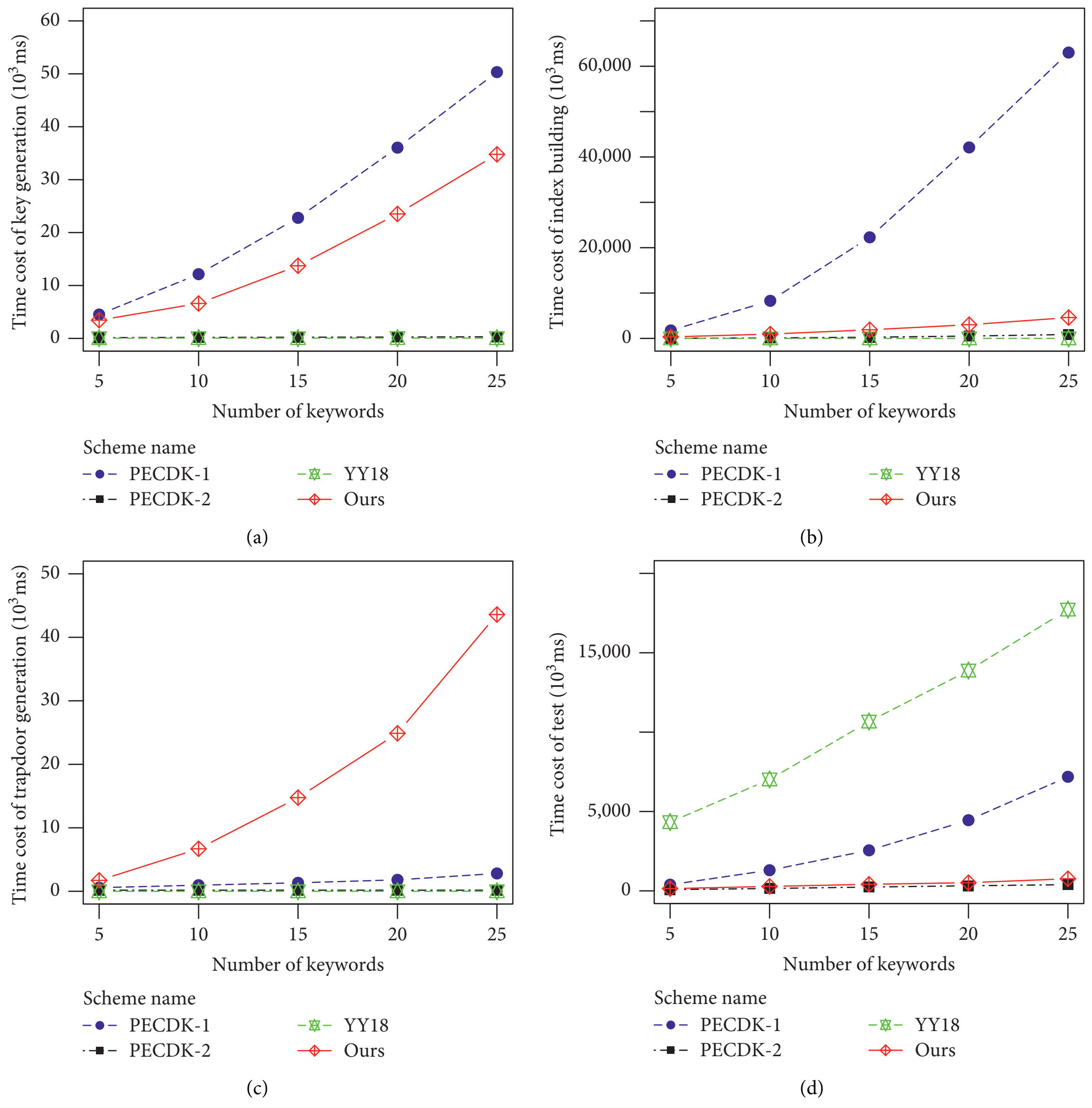

Figure 2: Impact of n on the time cost of key generation (a), index building (b), testing (c), and trapdoor generation (d) ($d = 1000$, $m = 5$, and $n = \{5;\ 10;\ 15;\ 20;\ 25\}$).

algorithm of PCTD n times, so the time cost of trapdoor generation is linear with $O(m)$. For the proposed scheme, the query is converted to be m vectors in which each vector's dimension is n. After this, each vector is encrypted by making use of DPVS, and thus, the time consumption of trapdoor generation in our scheme is linear with.

From Figure 3(d), the space costs for PECDK-1, PECDK-2, YY18, and our scheme are linear with n, n, n, and mn, respectively. The reason for this phenomenon is that the trapdoors in PECDK-1, PECDK-2, and our scheme contain n, m, and mn group elements on G, respectively, and the trapdoor in YY18 involves m ciphertexts of PCTD.

4.4. Search. As shown in Figure 2(d), the time cost of search in PECDK-1 is linear with, while that in PECDK-2, YY18, and our scheme is linear with mn. More precisely, for PECDK-1, the index of W contains n ciphertexts, and each ciphertext needs n pairing operations. For PECDK-2, the index is a matrix, and the trapdoor is a vector whose dimension is m. The test algorithm in PECKD-2 performs mn pairing operations between the first m rows of the matrix and the vector. For YY18, since the index and trapdoor hold n and m ciphertext of PCTD, respectively, the test algorithm will run secure less or equal (SLE) protocol and secure multiplication protocol across domains (SMD) mn times. For the proposed scheme, the trapdoor has m items, and the test algorithm in our scheme performs n pairing operations between each item and the index. Thus, total pairing operations in our scheme are mn. Since PECDK-1, PECDK-2, and our scheme need nearly 2 mn and 3 mn pairing operations,

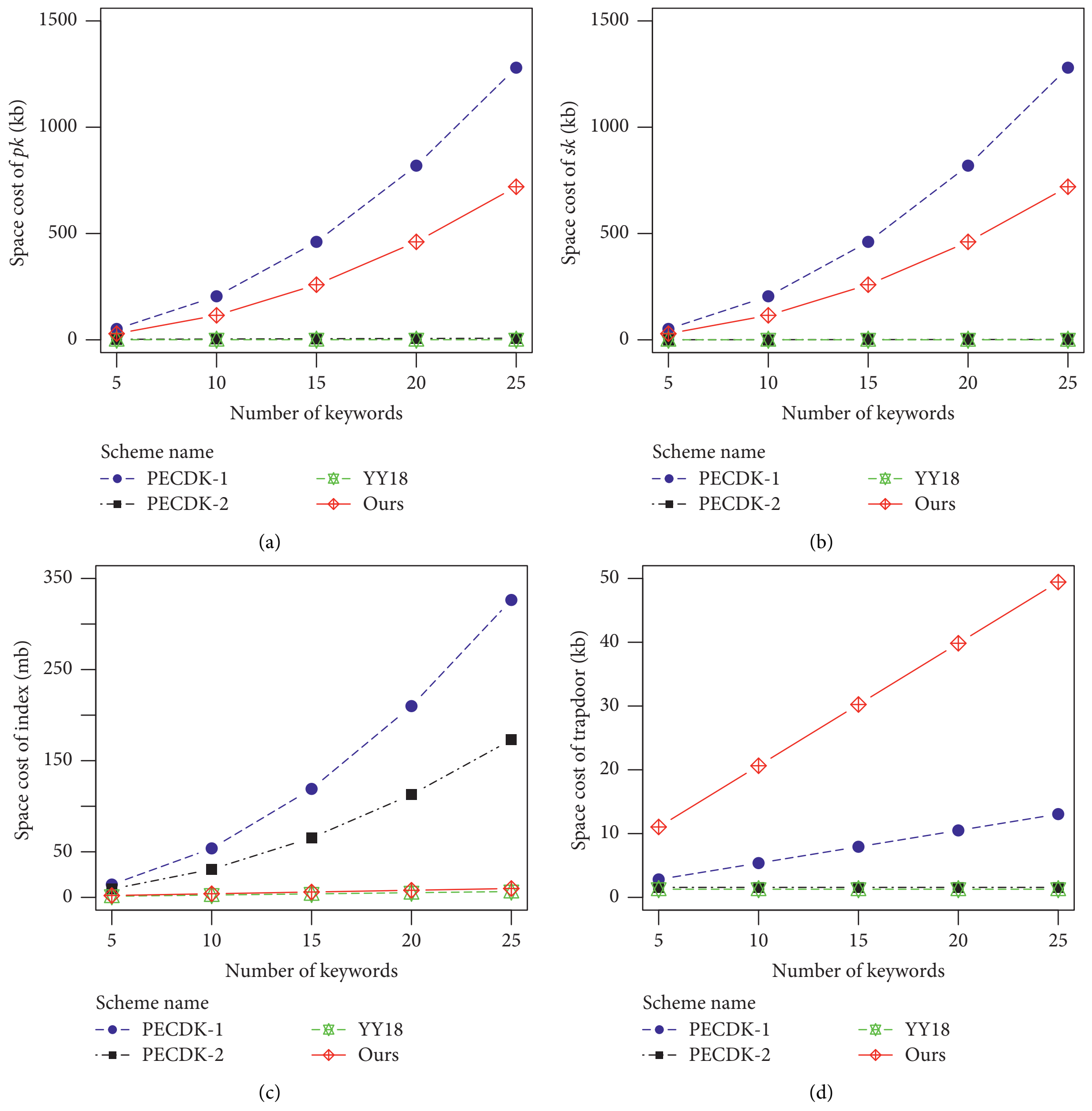

FIGURE 3: Impact of n on the space cost of pk (a), sk (b), index (c), and trapdoor (d) ($d = 1000$, $m = 5$, and $n = \{5; 10; 15; 20; 25\}$).

respectively, the time consumption in our scheme is slightly more than that in PECDK-2 and is less than that in PECDK-1. Moreover, since the time cost of a pairing operation is less than that of SLE and SMD, our scheme is more efficient than YY18 in test phase.

4.5. More Comments. As shown in the experimental results, when $n = 5$, $d = 1000$, and $m = 5$, the time cost of index building in our scheme is 331 s, the generation time of a single trapdoor is 1.7 s, and the search time is 142 s. According to the statistical data given in [17, 45], the number of keywords in a document (n) is usually less than 20, e.g., only 3~5 keywords in the scientific paper, and the number of keywords in a query (m) is often less than 10. We can argue that our scheme is suitable for the applications with fewer keywords, such as the keywords in the scientific literature, e-mail title and summaries, medical data summaries, and so on.

Although Figure 2 shows that the time complexity of our scheme is as good as that of PECDK-2, our scheme can support Boolean keywords search, which is much advanced than the conjunctive and disjunctive keywords search. Compared with YY18 that supports Boolean keywords search, our scheme needs less search time, despite the fact that it increases index building time. In practice, the index building in real-world application is usually a one-time activity, while queries are frequently performed. Thus, we reckon that it is worth sacrificing index building time to reduce retrieval time. For the space complexity, from Figure 3, our scheme needs less space for index storage, though requiring more storage space for the trapdoor and keys. Considering the fact that trapdoor and keys often require much less storage space than the index, we argue that our scheme is practicable in the real world.

5. Conclusions

In this paper, by applying DPVS and the bilinear pairing, we proposed a searchable public key encryption scheme supporting Boolean keyword search, which is proven to be secure under chosen keyword search attack. Compared to previous SPE schemes supporting conjunctive and disjunctive keywords search, the proposed scheme can support more advanced search function. Moreover, through a detailed experiment over a real-world dataset, we can argue that the efficiency of our scheme is suitable for practical applications with fewer keywords. Considering that the efficiency in our scheme still needed to be improved, we will construct a more efficient scheme in the forthcoming work.

Appendix

A. Proof of Lemma 3

Proof. Given $D = (g^{\overrightarrow{b_1}}, g^{\overrightarrow{b_2}}, \ldots, g^{\overrightarrow{b_{2n+2}}}, g^{\eta\overrightarrow{b_1}^*}, g^{\eta\overrightarrow{b_2}^*}, \ldots, g^{\eta\overrightarrow{b_{n+1}}^*}, g^{\beta\overrightarrow{b_{n+2}}^*}, g^{\beta\overrightarrow{b_{n+3}}^*}, \ldots, g^{\beta\overrightarrow{b_{2n+2}}^*}, g^{\overrightarrow{b_{2n+3}}^*}, g^{\overrightarrow{b_{2n+4}}^*}, \ldots, g^{\overrightarrow{b_{3n+3}}^*}, U_1, U_2, \ldots, U_{n+1}, \mu_3), T_1, T_2, \ldots, T_n$ and T_{n+1}, C needs to decide whether $T_1, T_2, \ldots, T_n$ and T_{n+1} are $g^{\tau_1\eta\overrightarrow{b_1}^* + \tau_2\beta\overrightarrow{b_{n+2}}^*}$, $g^{\tau_1\eta\overrightarrow{b_2}^* + \tau_2\beta\overrightarrow{b_{n+3}}^*}$ distributed as $g^{\tau_1\eta\overrightarrow{b_1}^* + \tau_2\beta\overrightarrow{b_{n+2}}^*}$, $g^{\tau_1\eta\overrightarrow{b_2}^* + \tau_2\beta\overrightarrow{b_{n+3}}^*}, \ldots, g^{\tau_1\eta\overrightarrow{b_i}^* + \tau_2\beta\overrightarrow{b_{n+1+i}}^*}, \ldots, g^{\tau_1\eta\overrightarrow{b_{n+1}}^* + \tau_2\beta\overrightarrow{b_{2n+2}}^*}$ or $g^{\tau_1\eta\overrightarrow{b_1}^* + \tau_2\beta\overrightarrow{b_{n+2}}^* + \tau_3\overrightarrow{b_{2n+3}}^*}$, $g^{\tau_1\eta\overrightarrow{b_2}^* + \tau_2\beta\overrightarrow{b_{n+3}}^* + \tau_3\overrightarrow{b_{2n+4}}^*}, \ldots, g^{\tau_1\eta\overrightarrow{b_i}^* + \tau_2\beta\overrightarrow{b_{n+1+i}}^* + \tau_3\overrightarrow{b_{2n+2+i}}^*}, \ldots, g^{\tau_1\eta\overrightarrow{b_{n+1}}^* + \tau_2\beta\overrightarrow{b_{2n+2}}^* + \tau_3\overrightarrow{b_{3n+3}}^*}$.

By using $T_1, T_2, \ldots, T_n$ and T_{n+1}, C can simulate Game$_0$ or Game$_{\text{Real}}$ with $\mathscr{A}$. To create pk, firstly, $\mathscr{C}$ randomly selects an invertible matrix $A \in Z_p^{(n+1)\times(n+1)}$. Then, we define a dual orthonormal bases F and F^* by $\overrightarrow{c_0} = \eta\overrightarrow{b_1}^*$, $\overrightarrow{c_1} = \eta\overrightarrow{b_2}^*, \ldots, \overrightarrow{c_n} = \eta\overrightarrow{b_{n+1}}^*$, $\overrightarrow{d_0} = \beta\overrightarrow{b_{n+2}}^*$, $\overrightarrow{d_1} = \beta\overrightarrow{b_{n+3}}^*, \ldots, \overrightarrow{d_n} = \beta\overrightarrow{b_{2n+2}}^*$, $\overrightarrow{f_0} = \overrightarrow{b_{2n+3}}^*$, $\overrightarrow{f_1} = \overrightarrow{b_{2n+4}}^*, \ldots, \overrightarrow{f_n} = \overrightarrow{b_{3n+3}}^*$ and $\overrightarrow{c_0}^* = \eta^{-1}\overrightarrow{b_1}$, $\overrightarrow{c_1}^* = \eta^{-1}\overrightarrow{b_2}, \ldots, \overrightarrow{c_n}^* = \eta^{-1}\overrightarrow{b_{n+1}}$, $\overrightarrow{d_0}^* = \beta^{-1}\overrightarrow{b_{n+2}}$, $\overrightarrow{d_1}^* = \beta^{-1}\overrightarrow{b_{n+3}}, \ldots, \overrightarrow{d_n}^* = \beta^{-1}\overrightarrow{b_{2n+2}}$, $\overrightarrow{f_0}^* = \overrightarrow{b_{2n+3}}$, $\overrightarrow{f_1}^* = \overrightarrow{b_{2n+4}}, \ldots, \overrightarrow{f_n}^* = \overrightarrow{b_{3n+3}}$.

C implicitly sets $E = F_A = \{\overrightarrow{c_0}, \overrightarrow{c_1}, \ldots, \overrightarrow{c_n}, \overrightarrow{d_0}, \overrightarrow{d_1}, \ldots, \overrightarrow{d_n}, \overrightarrow{e_0}, \overrightarrow{e_1}, \ldots, \overrightarrow{e_n}\}$ and $E^* = F_A^* = \{\overrightarrow{c_0}^*, \overrightarrow{c_1}^*, \ldots, \overrightarrow{c_n}^*, \overrightarrow{d_0}^*, \overrightarrow{d_1}^*, \ldots, \overrightarrow{d_n}^*, \overrightarrow{e_0}^*, \overrightarrow{e_1}^*, \ldots, \overrightarrow{e_n}^*\}$ where the matrix A is applied as a change of basis matrix to $\overrightarrow{f_0}, \overrightarrow{f_1}, \ldots, \overrightarrow{f_n}$ and $(A^{-1})^T$ is applied as a change of basis matrix to $\overrightarrow{f_0}^*, \overrightarrow{f_1}^*, \ldots, \overrightarrow{f_n}^*$, as described in Section 2.5. Note that the first $2n+2$ basis vectors are unchanged. According to Lemma 2, E and E^* are properly distributed.

Choosing a function H_1, C computes $pk = \{p, H_1, g^{\overrightarrow{c_0}}, g^{\overrightarrow{c_1}}, \ldots, g^{\overrightarrow{c_n}}, g^{\overrightarrow{d_0}}, g^{\overrightarrow{d_1}}, \ldots, g^{\overrightarrow{d_n}}\}$ and sends it to A. Each time A asks C to provide a key for a keyword query Q, C creates a normal trapdoor of Q. Choosing $r_1', r_2', \ldots, r_m', t_1', t_2', \ldots, t_m' \in Z_p$ and an invertible matrix $\alpha = (\overrightarrow{\alpha_1}, \overrightarrow{\alpha_2}, \ldots, \overrightarrow{\alpha_m})^T$ and $\alpha^{-1} = (\overrightarrow{\alpha_1}^*, \overrightarrow{\alpha_2}^*, \ldots, \overrightarrow{\alpha_m}^*)$, C computes

$$K_j = g^{\sum_{i=0}^{n}\left(\sum_{\phi=1}^{m}\alpha_{\phi j}r_\phi'\sum_{\theta=1}^{n_\phi}H_1\left(q_{\phi\theta}\right)^i\right)\overrightarrow{b_{l+1}} + \sum_{i=0}^{n}\left(\sum_{\phi=1}^{m}\alpha_{\phi j}t_\phi'\sum_{\theta=1}^{n_\phi}H_1\left(q_{\phi\theta}\right)^i\right)\overrightarrow{b_{l+2+n}}} = g^{\sum_{i=0}^{n}\left(\sum_{\phi=1}^{m}\alpha_{\phi j}r_\phi'\sum_{\theta=1}^{n_\phi}H_1\left(q_{\phi\theta}\right)^i\right)\overrightarrow{c_l}^* + \sum_{i=0}^{n}\left(\sum_{\phi=1}^{m}\alpha_{\phi j}t_\phi'\sum_{\theta=1}^{n_\phi}H_1\left(q_{\phi\theta}\right)^i\right)\overrightarrow{d_l}^*} \tag{A.1}$$

and sends $T_Q = \{K_1, K_2, \ldots, K_m, \alpha^{-1}\}$ to A, where $r_j = r_j'\eta$, $t_j = t_j'\beta$, $j \in [1, m]$.

At some point, A sends C two challenge keyword sets, $W^{(0)} = \{w_1^{(0)}, w_2^{(0)}, \ldots, w_n^{(0)}\}$ and $W^{(1)} = \{w_1^{(1)}, w_2^{(1)}, \ldots, w_n^{(1)}\}$. By randomly choosing $\beta \in [0, 1]$ and computing an n-degree polynomial $f(x) = a_nx^n + a_{n-1}x^{n-1} + \cdots + a_0x^0 = (x - H_1(w_1^{(\beta)}))(x - H_1(w_2^{(\beta)})), \ldots, (x - H_1(w_n^{(\beta)}))$, C sets

$$I_W = T_1^{a_0}T_2^{a_1}, \ldots, T_{n+1}^{a_n}, \tag{A.2}$$

where C implicitly sets $\tau_1 = s_1$, $\tau_2 = s_2$.

Then, C gives the index I_W to A . If $T_1, T_2, \ldots, T_n$ and T_{n+1} are equal to $g^{\tau_1\eta\overrightarrow{b_1}^* + \tau_2\beta\overrightarrow{b_{n+2}}^*}$, $g^{\tau_1\eta\overrightarrow{b_2}^* + \tau_2\beta\overrightarrow{b_{n+3}}^*}, \ldots, g^{\tau_1\eta\overrightarrow{b_l}^* + \tau_2\beta\overrightarrow{b_{n+1+l}}^*}, \ldots, g^{\tau_1\eta\overrightarrow{b_{n+1}}^* + \tau_2\beta\overrightarrow{b_{2n+2}}^*}$, then this is a properly distributed normal index. In this case, C has properly simulated Game$_{\text{Real}}$.If $T_1, T_2, \ldots, T_n$ and T_{n+1} are equal to $g^{\tau_1\eta\overrightarrow{b_1}^* + \tau_2\beta\overrightarrow{b_{n+2}}^* + \tau_3\overrightarrow{b_{2n+3}}^*}$, $g^{\tau_1\eta\overrightarrow{b_2}^* + \tau_2\beta\overrightarrow{b_{n+3}}^* + \tau_3\overrightarrow{b_{2n+4}}^*}, \ldots, g^{\tau_1\eta\overrightarrow{b_l}^* + \tau_2\beta\overrightarrow{b_{n+1+l}}^* + \tau_3\overrightarrow{b_{2n+2+l}}^*}, \ldots, g^{\tau_1\eta\overrightarrow{b_{n+1}}^* + \tau_2\beta\overrightarrow{b_{2n+2}}^* + \tau_3\overrightarrow{b_{3n+3}}^*}$, then there is an additional term of $\tau_3(a_0\overrightarrow{b_{2n+3}}^* + a_1\overrightarrow{b_{2n+4}}^* + \cdots + a_n\overrightarrow{b_{3n+3}}^*)$ in the exponent part of the index. The coefficients in the basis $\overrightarrow{b_{2n+3}}^*, \overrightarrow{b_{2n+4}}^*, \ldots, \overrightarrow{b_{3n+3}}^*$ are the vector $\tau_3(a_0, a_1, \ldots, a_n)$. In order to acquire the coefficients in the basis $\overrightarrow{e_0}, \overrightarrow{e_1}, \ldots, \overrightarrow{e_n}$, we multiply the matrix A^{-1} by the transpose of these vectors and obtain $\tau_3A^{-1}(a_0, a_1, \ldots, a_n)$. Since A is random, these coefficients are uniformly random. Therefore, in this case, C has properly simulated Game$_0$. So, if A can distinguish Game$_{\text{Real}}$ from Game$_0$ with nonnegligible advantage, then C can use the output of A to break subspace assumption with nonnegligible advantage. □

B. Proof of Lemma 4

Proof. Given $D = (g^{\overrightarrow{b_1}}, g^{\overrightarrow{b_2}}, \ldots, g^{\overrightarrow{b_{2n+2}}}, g^{\eta\overrightarrow{b_1}^{*}}, g^{\eta\overrightarrow{b_2}^{*}}, \ldots, g^{\eta\overrightarrow{b_{n+1}}^{*}}, g^{\beta\overrightarrow{b_{n+2}}^{*}}, g^{\beta\overrightarrow{b_{n+3}}^{*}}, \ldots, g^{\beta\overrightarrow{b_{2n+2}}^{*}}, g^{\overrightarrow{b_{2n+3}}^{*}}, g^{\overrightarrow{b_{2n+4}}^{*}}, \ldots, g^{\overrightarrow{b_{3n+3}}^{*}}, U_1, U_2, \ldots, U_n+1, \mu_3)$, $T_1, T_2, \ldots, T_n$ and T_{n+1}, $\mathscr{C}$ needs to decide whether $T_1, T_2, \ldots, T_n$ and T_{n+1} are distributed as follows: $g^{\tau_1\eta\overrightarrow{b_1}^{*}+\tau_2\beta\overrightarrow{b_{n+2}}^{*}}$, $g^{\tau_1\eta\overrightarrow{b_2}^{*}+\tau_2\beta\overrightarrow{b_{n+3}}^{*}}$, $\ldots$, $g^{\tau_1\eta\overrightarrow{b_i}^{*}+\tau_2\beta\overrightarrow{b_{n+1+i}}^{*}}$, $\ldots$, $g^{\tau_1\eta\overrightarrow{b_{n+1}}^{*}+\tau_2\beta\overrightarrow{b_{2n+2}}^{*}}$ or $g^{\tau_1\eta\overrightarrow{b_1}^{*}+\tau_2\beta\overrightarrow{b_{n+2}}^{*}+\tau_3\overrightarrow{b_{2n+3}}^{*}}$, $g^{\tau_1\eta\overrightarrow{b_2}^{*}+\tau_2\beta\overrightarrow{b_{n+3}}^{*}+\tau_3\overrightarrow{b_{2n+4}}^{*}}$, $\ldots$, $g^{\tau_1\eta\overrightarrow{b_i}^{*}+\tau_2\beta\overrightarrow{b_{n+1+i}}^{*}+\tau_3\overrightarrow{b_{2n+2+i}}^{*}}$, $\ldots$, $g^{\tau_1\eta\overrightarrow{b_{n+1}}^{*}+\tau_2\beta\overrightarrow{b_{2n+2}}^{*}+\tau_3\overrightarrow{b_{3n+3}}^{*}}$.

By using $T_1, T_2, \ldots, T_n$ and T_{n+1}, $\mathscr{C}$ can simulate Game$_{k-1}$ or Game$_k$ with $\mathscr{A}$. To create pk, firstly, $\mathscr{C}$ randomly selects an invertible matrix $A \in Z_p^{(n+1)\times(n+1)}$ and implicitly sets $E = B_A = \{\overrightarrow{c_0}, \overrightarrow{c_1}, \ldots, \overrightarrow{c_n}, \overrightarrow{d_0}, \overrightarrow{d_1}, \ldots, \overrightarrow{d_n}, \overrightarrow{e_0}, \overrightarrow{e_1}, \ldots, \overrightarrow{e_n}\}$, and $E^{*} = B_A^{*} = \{\overrightarrow{c_0}^{*}, \overrightarrow{c_1}^{*}, \ldots, \overrightarrow{c_n}^{*}, \overrightarrow{d_0}^{*}, \overrightarrow{d_1}^{*}, \ldots, \overrightarrow{d_n}^{*}, \overrightarrow{e_0}^{*}, \overrightarrow{e_1}^{*}, \ldots, \overrightarrow{e_n}^{*}\}$, where A is applied as a change of basis matrix to $\overrightarrow{b_{2n+3}}, \overrightarrow{b_{2n+4}}, \ldots, \overrightarrow{b_{3n+3}}$ and $(A^{-1})^T$ is applied as a change of basis matrix to $\overrightarrow{b_{2n+3}}^{*}, \overrightarrow{b_{2n+4}}^{*}, \ldots, \overrightarrow{b_{3n+3}}^{*}$, as described in Section 2.5. Then, $\overrightarrow{c_i} = \overrightarrow{b_{i+1}}$, $\overrightarrow{c_i}^{*} = \overrightarrow{b_{i+1}}^{*}$, $\overrightarrow{d_i} = \overrightarrow{b_{i+n+2}}$, $\overrightarrow{d_i}^{*} = \overrightarrow{b_{i+n+2}}^{*}$ for $i \in [0, n]$. According to Lemma 2, E and E^{*} are properly distributed.

Choosing a function H_1, $\mathscr{C}$ computes $pk = \{p, H_1, g^{\overrightarrow{c_0}}, g^{\overrightarrow{c_1}}, \ldots, g^{\overrightarrow{c_n}}, g^{\overrightarrow{d_0}}, g^{\overrightarrow{d_1}}, \ldots, g^{\overrightarrow{d_n}}\}$ and sends it to $\mathscr{A}$.

When $\mathscr{A}$ requests the lth trapdoor query, $\mathscr{C}$ generates the normal trapdoor or the semifunctional trapdoor as follows:

For $l < k$, choosing $r'_{l1}, r'_{l2}, \ldots, r'_{lm}, t'_{l1}, t'_{l2}, \ldots, t'_{lm}, z_{lji} \in Z_p$ and implicitly setting $r_{lj} = r'_{lj}\eta$, $t_{lj} = t'_{lj}\beta$, where $j \in [1, m]$ and $i \in [0, n]$, $\mathscr{C}$ can produce semifunctional trapdoor by using $g^{\eta\overrightarrow{b_1}^{*}}, g^{\eta\overrightarrow{b_2}^{*}}, \ldots, g^{\eta\overrightarrow{b_{n+1}}^{*}}, g^{\beta\overrightarrow{b_{n+2}}^{*}}, g^{\beta\overrightarrow{b_{n+3}}^{*}}, \ldots, g^{\beta\overrightarrow{b_{2n+2}}^{*}}, g^{\overrightarrow{b_{2n+3}}^{*}}, g^{\overrightarrow{b_{2n+4}}^{*}}, \ldots, g^{\overrightarrow{b_{3n+3}}^{*}}$.

For $l > k$, $\mathscr{C}$ runs the normal trapdoor generation algorithm to produce the normal trapdoor.

To create the kth requested trapdoor, $\mathscr{C}$ firstly chooses $r''_1, r''_2, \ldots, r''_m, r'_1, r'_2, \ldots, r'_m, t'_1, t'_2, \ldots, t'_m \in Z_p$ and an invertible matrix α. Let $\alpha = (\overrightarrow{\alpha_1}, \overrightarrow{\alpha_2}, \ldots, \overrightarrow{\alpha_m})^T$ and $\alpha^{-1} = (\overrightarrow{\alpha_1}^{*}, \overrightarrow{\alpha_2}^{*}, \ldots, \overrightarrow{\alpha_m}^{*})$. Then, for each $j \in [1, m]$, $\mathscr{C}$ computes

$$\begin{aligned}K_j &= \prod_{i=1}^{n+1} T_i^{\sum_{\phi=1}^{m}\alpha_{\phi j}r''_{\phi}\sum_{\theta=1}^{n_\phi}H_1(q_{\phi\theta})^i} \times g^{\sum_{i=0}^{n}\left(\sum_{\phi=1}^{m}\alpha_{\phi j}r'_{\phi}\sum_{\theta=1}^{n_\phi}H_1(q_{\phi\theta})^i\right)\eta\overrightarrow{b_{i+1}}^{*}+\sum_{i=0}^{n}\left(\sum_{\phi=1}^{m}\alpha_{\phi j}t'_{\phi}\sum_{\theta=1}^{n_\phi}H_1(q_{\phi\theta})^i\right)\beta\overrightarrow{b_{i+2+n}}^{*}} \\ &= g^{\sum_{i=0}^{n}\left(\sum_{\phi=1}^{m}\alpha_{\phi j}r''_{\phi}\sum_{\theta=1}^{n_\phi}H_1(q_{\phi\theta})^i\tau_3\overrightarrow{b_{2n+3+i}}^{*}\right)+\sum_{i=0}^{n}\left(\sum_{\phi=1}^{m}\alpha_{\phi j}r'_{\phi}\sum_{\theta=1}^{n_\phi}H_1(q_{\phi\theta})^i\right)\overrightarrow{c_i}^{*}+\sum_{i=0}^{n}\left(\sum_{\phi=1}^{m}\alpha_{\phi j}t'_{\phi}\sum_{\theta=1}^{n_\phi}H_1(q_{\phi\theta})^i\right)\overrightarrow{d_i}^{*}}.\end{aligned} \tag{B.1}$$

The above equation implicitly sets $r_j = (r'_j + \tau_1 r''_j)\eta$ and $t_j = (t'_j + \tau_2 r''_j)\beta$, where $j \in [1, m]$. If $T_1, T_2, \ldots, T_n$ and T_{n+1} are equal to $g^{\tau_1\eta\overrightarrow{b_1}^{*}+\tau_2\beta\overrightarrow{b_{n+2}}^{*}}$, $g^{\tau_1\eta\overrightarrow{b_2}^{*}+\tau_2\beta\overrightarrow{b_{n+3}}^{*}}$, $\ldots$, $g^{\tau_1\eta\overrightarrow{b_i}^{*}+\tau_2\beta\overrightarrow{b_{n+1+i}}^{*}}$, $\ldots$, $g^{\tau_1\eta\overrightarrow{b_{n+1}}^{*}+\tau_2\beta\overrightarrow{b_{2n+2}}^{*}}$; then this is a properly distributed normal trapdoor. If $T_1, T_2, \ldots, T_n$ and T_{n+1} are equal to $g^{\tau_1\eta\overrightarrow{b_1}^{*}+\tau_2\beta\overrightarrow{b_{n+2}}^{*}+\tau_3\overrightarrow{b_{2n+3}}^{*}}$, $g^{\tau_1\eta\overrightarrow{b_2}^{*}+\tau_2\beta\overrightarrow{b_{n+3}}^{*}+\tau_3\overrightarrow{b_{2n+4}}^{*}}$, $\ldots$, $g^{\tau_1\eta\overrightarrow{b_i}^{*}+\tau_2\beta\overrightarrow{b_{n+1+i}}^{*}+\tau_3\overrightarrow{b_{2n+2+i}}^{*}}$, $\ldots$, $g^{\tau_1\eta\overrightarrow{b_{n+1}}^{*}+\tau_2\beta\overrightarrow{b_{2n+2}}^{*}+\tau_3\overrightarrow{b_{3n+3}}^{*}}$, then this is a properly distributed semifunctional trapdoor. For each $j \in [1, m]$, K_j's exponent vector contains the item $V_j = \sum_{i=0}^{n}(\sum_{\phi=1}^{m}\alpha_{\phi j}r''_{\phi}\sum_{\theta=1}^{n_\phi}H_1(q_{\phi\theta})^i\tau_3\overrightarrow{b_{2n+3+i}}^{*})$.

At some point, $\mathscr{A}$ sends $\mathscr{C}$ two challenge keyword sets, $W^{(0)} = \{w_1^{(0)}, w_2^{(0)}, \ldots, w_n^{(0)}\}$ and $W^{(1)} = \{w_1^{(1)}, w_2^{(1)}, \ldots, w_n^{(1)}\}$. By randomly choosing $\beta \in [0, 1]$ and computing an n-degree polynomial $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0 x^0 = (x - H_1(w_1^{(\beta)}))(x - H_1(w_2^{(\beta)})), \ldots, (x - H_1(w_n^{(\beta)}))$, where $H_1(w_1^{(\beta)}), H_1(w_2^{(\beta)}), \ldots, H_1(w_n^{(\beta)})$, are n roots of the equation $f(x) = 0$, $\mathscr{C}$ sets

$$I_W = U_1^{a_0}U_2^{a_1}, \ldots, U_{n+1}^{a_n}, \tag{B.2}$$

where $\mathscr{C}$ implicitly sets $\mu_1 = s_1$ and $\mu_2 = s_2$.

After that, $\mathscr{C}$ sends the semifunctional index I_W to $\mathscr{A}$. Obviously, I_W contains the exponent vector $V = \tau_3(a_0\overrightarrow{b_{2n+3}} + a_1\overrightarrow{b_{2n+4}} + \cdots + a_n\overrightarrow{b_{3n+3}})$. The authors observe that if $\mathscr{C}$ attempts to test whether the kth trapdoor of Q is semifunctional by creating a semifunctional index of keyword set W which satisfies $f(W, Q) = 1$, then $\mathscr{C}$ can find that test algorithm can still work whether the kth key is semifunctional or not, since V_j and V will be eliminated when $f(W, Q) = 1$. Therefore, we can say that the kth key is a nominally semifunctional key.

In view of this, for each $j \in [1, m]$, V_j and V are distributed as random vectors in the spans of $\overrightarrow{b_{2n+3}}^{*}, \overrightarrow{b_{2n+4}}^{*}, \ldots, \overrightarrow{b_{3n+3}}^{*}$ and $\overrightarrow{b_{2n+3}}, \overrightarrow{b_{2n+4}}, \ldots, \overrightarrow{b_{3n+3}}$. In V_j, the coefficients in the basis $\overrightarrow{b_{2n+3}}^{*}, \overrightarrow{b_{2n+4}}^{*}, \ldots, \overrightarrow{b_{3n+3}}^{*}$ are the vector $\tau_3(b_0^{(j)}, b_1^{(j)}, \ldots, b_n^{(j)})$, where $b_i^{(j)} = \alpha_{1j}r''_1\sum_{\theta=1}^{n_1}H_1(q_{1\theta})^i + \alpha_{2j}r''_2\sum_{\theta=1}^{n_2}H_1(q_{2\theta})^i + \cdots + \alpha_{mj}r''_m\sum_{\theta=1}^{n_m}H_1(q_{m\theta})^i$ and $i \in [0, n]$, $j \in [1, m]$. In order to acquire the coefficients in the basis $\overrightarrow{e_0}^{*}, \overrightarrow{e_1}^{*}, \ldots, \overrightarrow{e_n}^{*}$, we multiply the matrix A^t by the transpose of these vectors and obtain $E_j = \tau_3 A^t(b_0^{(j)}, b_1^{(j)}, \ldots, b_n^{(j)})$). Since $\mathscr{A}$ is random and $\overrightarrow{b}^{(i)} \neq \overrightarrow{b}^{(j)}$ if $i \neq j$, we can say that $E_1, E_2, \ldots, E_m$ are uniformly random. In V, the coefficients in the basis $\overrightarrow{b_{2n+3}}, \overrightarrow{b_{2n+4}}, \ldots, \overrightarrow{b_{3n+3}}$ are the vector $\mu_3(a_0, a_1, \ldots, a_n)$. In order to acquire the coefficients in the basis $\overrightarrow{e_0}$, $\overrightarrow{e_1}, \dots, \overrightarrow{e_n}$, we multiply the matrix A^{-1} by the transpose of these vectors and obtain $E_j = \mu_{(\beta j)} A^{-1}(a_0, a_1, \dots, a_n)$. Since $\mathscr{A}$ is random and $\overrightarrow{a} \cdot \overrightarrow{b}^{(\beta j)} \neq 0$, the coefficients E_j and mentioned above are uniformly random according to Lemma 2, where $j \in [1, m]$ and $\overrightarrow{a} = (a_0, a_1, \dots, a_n)$, $\overrightarrow{b}^{(j)} = (b_0^{(j)}, b_1^{(j)}, \dots, b_n^{(j)})$.

According to the above analysis, we conclude that if $T_1, T_2, \dots, T_n$ and T_{n+1} are distributed as $g^{\tau_1 \eta \overrightarrow{b_1}^* + \tau_2 \beta \overrightarrow{b_{n+2}}^*}$, $g^{\tau_1 \eta \overrightarrow{b_2}^* + \tau_2 \beta \overrightarrow{b_{n+3}}^*}, \dots, g^{\tau_1 \eta \overrightarrow{b_i}^* + \tau_2 \beta \overrightarrow{b_{n+1+i}}^*}, \dots, g^{\tau_1 \eta \overrightarrow{b_{n+1}}^* + \tau_2 \beta \overrightarrow{b_{2n+2}}^*}$, $\mathscr{C}$ has properly simulated Game$_{k-1}$. If $T_1, T_2, \dots, T_n$ and T_{n+1} are equal to $g^{\tau_1 \eta \overrightarrow{b_1}^* + \tau_2 \beta \overrightarrow{b_{n+2}}^* + \tau_3 \overrightarrow{b_{2n+3}}^*}$, $g^{\tau_1 \eta \overrightarrow{b_2}^* + \tau_2 \beta \overrightarrow{b_{n+3}}^* + \tau_3 \overrightarrow{b_{2n+4}}^*}, \dots, g^{\tau_1 \eta \overrightarrow{b_i}^* + \tau_2 \beta \overrightarrow{b_{n+1+i}}^* + \tau_3 \overrightarrow{b_{2n+2+i}}^*}, \dots, g^{\tau_1 \eta \overrightarrow{b_{n+1}}^* + \tau_2 \beta \overrightarrow{b_{2n+2}}^* + \tau_3 \overrightarrow{b_{3n+3}}^*}$, $\mathscr{C}$ has properly simulated Game$_k$. Thus, we argue that if $\mathscr{A}$ can distinguish Game$_{k-1}$ from Game$_k$ with nonnegligible advantage, then $\mathscr{C}$ can use the output of $\mathscr{A}$ to break subspace complexity assumption with nonnegligible advantage. □

C. Proof of Lemma 5

Proof. Given $D = (g^{\overrightarrow{b_1}}, g^{\overrightarrow{b_2}}, g^{\overrightarrow{b_3}}, g^{\overrightarrow{b_4}}, g^{\eta \overrightarrow{b_1}^*}, g^{\eta \overrightarrow{b_2}^*}, g^{\beta \overrightarrow{b_3}^*}, g^{\beta \overrightarrow{b_4}^*}, g^{\overrightarrow{b_5}^*}, g^{\overrightarrow{b_6}^*}, U_1, U_2, \mu_3)$, T_1 and T_2, C needs to decide whether T_1 and T_2 are distributed as $g^{\tau_1 \eta \overrightarrow{b_1}^* + \tau_2 \beta \overrightarrow{b_3}^*}$ and $g^{\tau_1 \eta \overrightarrow{b_2}^* + \tau_2 \beta \overrightarrow{b_4}^*}$ or as $g^{\tau_1 \eta \overrightarrow{b_1}^* + \tau_2 \beta \overrightarrow{b_3}^* + \tau_3 \overrightarrow{b_5}^*}$ and $g^{\tau_1 \eta \overrightarrow{b_2}^* + \tau_2 \beta \overrightarrow{b_4}^* + \tau_3 \overrightarrow{b_6}^*}$, respectively.

By using T_1 and T_2, for $k \in [0, n]$, C can simulate Game$_{\text{Final}_{k-1}}$ or Game$_{\text{Final}_k}$ with A. To construct pk, C implicitly sets $E = (\overrightarrow{e_0} = \eta \overrightarrow{b_1}^*, \overrightarrow{e_1} = \eta \overrightarrow{b_2}^*, \overrightarrow{e_2} = \beta \overrightarrow{b_3}^*, \overrightarrow{e_3} = \beta \overrightarrow{b_4}^*, \overrightarrow{c_k} = \overrightarrow{b_5}^*, \overrightarrow{d_k} = \overrightarrow{b_6}^*, \dots, \overrightarrow{c_l} = \overrightarrow{b_{2l+5}}^*, \overrightarrow{d_l} = \overrightarrow{b_{2l+6}}^*, \dots, \overrightarrow{c_0} = \overrightarrow{b_{2k+5}}^*, \overrightarrow{d_0} = \overrightarrow{b_{2k+6}}^*, \dots, \overrightarrow{e_4} = \overrightarrow{b_{2n+7}}^*, \overrightarrow{e_5} = \overrightarrow{b_{2n+8}}^*, \dots, \overrightarrow{e_n} = \overrightarrow{b_{3n+3}}^*)$ and $E^* = (\overrightarrow{e_0}^* = \eta^{-1} \overrightarrow{b_1}, \overrightarrow{e_1}^* = \eta^{-1} \overrightarrow{b_2}, \overrightarrow{e_1}^* = \eta^{-1} \overrightarrow{b_2}, \overrightarrow{e_3}^* = \beta^{-1} \overrightarrow{b_4}, \overrightarrow{c_k}^* = \overrightarrow{b_5}, \overrightarrow{d_k}^* = \overrightarrow{b_6}, \dots, \overrightarrow{e_4}^* = \overrightarrow{b_{2n+7}}, \overrightarrow{e_5}^* = \overrightarrow{b_{2n+8}}, \dots, \overrightarrow{e_n}^* = \overrightarrow{b_{3n+3}})$, where $i \in [0, k]$. Apparently, E and E^* are properly distributed dual orthonormal bases.

Because C can obtain $g^{\overrightarrow{b_5}^*}, g^{\overrightarrow{b_6}^*}, \dots, g^{\overrightarrow{b_{2n+6}}^*}$, pk can be easily created. Each time A asks C to provide a key for a keyword query Q, C creates a semifunctional trapdoor of Q. Choosing $r_1, r_2, \dots, r_m, t_1, t_2, \dots, t_m, z_{ji} \in Z_p$ and an invertible matrix $\alpha = (\overrightarrow{\alpha_1}, \overrightarrow{\alpha_2}, \dots, \overrightarrow{\alpha_m})^T$ and $\alpha^{-1} = (\overrightarrow{\alpha_1}^*, \overrightarrow{\alpha_2}^*, \dots, \overrightarrow{\alpha_m}^*)$, C computes

$$\begin{aligned} K_j = {} & g^{\sum_{i \in [0,n]/k} \left(\sum_{\phi=1}^{m} \alpha_{\phi j} r_\phi \sum_{\theta=1}^{n_\phi} H_1 (q_{\phi\theta})^i \right) \overrightarrow{b_{l+1}} + \sum_{i \in [0,n]/k} \left(\sum_{\phi=1}^{m} \alpha_{\phi j} t_\phi \sum_{\theta=1}^{n_\phi} H_1 (q_{\phi\theta})^i \right) \overrightarrow{b_{l+2+n}}} \times U_1^{\mu_3^{-1} \sum_{\phi=1}^{m} \alpha_{\phi j} r_\phi \sum_{\theta=1}^{n_\phi} H_1 (q_{\phi\theta})^i} \\ & \times U_2^{\mu_3^{-1} \sum_{\phi=1}^{m} \alpha_{\phi j} r_\phi \sum_{\theta=1}^{n_\phi} H_1 (q_{\phi\theta})^i} \times g^{z_{j0} \overrightarrow{b_1} + z_{j1} \overrightarrow{b_2} + z_{j2} \overrightarrow{b_3} + z_{j3} \overrightarrow{b_4} + \sum_{i=4}^{n} z_{ji} \overrightarrow{b_{2n+3+l}}}, \end{aligned} \tag{C.1}$$

and sends $T_Q = \{K_1, K_2, \dots, K_m, \alpha^{-1}\}$ to A, where $j \in [1, m]$.

At some point, A sends C two challenge keyword sets, $W^{(0)} = \{W_1^{(0)}, W_2^{(0)}, \dots, W_n^{(0)}\}$ and $W^{(1)} = \{W_1^{(1)}, W_2^{(1)}, \dots, W_n^{(1)}\}$. By randomly choosing $\beta \in [0, 1]$, $s_1, s_2 \in Z_p$, two random vectors $\overrightarrow{x} = \{x_0, x_1, \dots, x_n\}$, $\overrightarrow{w} = \{w_0, w_1, \dots, w_n\}$, and computing an n-degree polynomial $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 x^0 = (x - H_1(w_1^{(\beta)}))(x - H_1(w_2^{(\beta)})), \dots, (x - H_1(w_n^{(\beta)}))$ by using the function H_1, where $H_1(w_1^{(\beta)}), H_1(w_2^{(\beta)}), \dots, H_1(w_n^{(\beta)})$ are n roots of the equation $f(x) = 0$, then C sets

$$I_W = g^{s_1 \left(\sum_{i=0}^{k-1} x_i \overrightarrow{c_l} + \sum_{i=k}^{n} a_i \overrightarrow{c_l} \right) + s_2 \left(\sum_{i=0}^{k-1} x_i \overrightarrow{d_l} + \sum_{i=k}^{n} a_i \overrightarrow{d_l} \right)} \times g^{\sum_{i=0}^{n} w_i \overrightarrow{e_l}} T_1^{s_1} T_2^{s_2}. \tag{C.2}$$

Then, C gives the indexI_w to A. If T_1 and T_2 are equal to $g^{\tau_1 \eta \overrightarrow{b_1}^* + \tau_2 \beta \overrightarrow{b_3}^*}$ and $g^{\tau_1 \eta \overrightarrow{b_2}^* + \tau_2 \beta \overrightarrow{b_4}^*}$, then this is a properly distributed semifunctional index of the vector $\{x_0, x_1, \dots, x_{k-1}, a_k, a_{k+1}, \dots, a_n\}$. In this case, C has properly simulated Game$_{\text{Final}_{k-1}}$. If T_1 and T_2 are equal to $g^{\tau_1 \eta \overrightarrow{b_1}^* + \tau_2 \beta \overrightarrow{b_3}^* + \tau_3 \overrightarrow{b_5}^*}$ and $g^{\tau_1 \eta \overrightarrow{b_2}^* + \tau_2 \beta \overrightarrow{b_4}^* + \tau_3 \overrightarrow{b_6}^*}$, respectively, then this is a properly distributed semifunctional index of the vector $\{x_0, x_1, \dots, x_{k-1}, b_k, a_{k+1}, \dots, a_n\}$, where $b_k = a_k + \tau_3$. In this case, C has properly simulated Game$_{\text{Final}_k}$. So, if A can distinguish Game$_{\text{Final}_{k-1}}$ from Game$_{\text{Final}_k}$ with nonnegligible advantage, then C can use the output of A to break subspace assumption with nonnegligible advantage. □

Data Availability

The data used to support the findings of this study are available from the website http://www.cs.cmu.edu/~./enron/.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors gratefully acknowledge the support of the National Natural Science Foundation of China under Grant nos. 61402393 and 61601396 and Nanhu Scholars Program for Young Scholars of XYNU.

References

[1] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber + r: top-*k* retrieval from a confidential index," in *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pp. 439–449, Saint Petersburg, Russia, March 2009.

[2] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340–352, 2016.

[3] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 9, pp. 2546–2559, 2016.

[4] 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.

[5] C. Guo, R. Zhuang, C.-C. Chang, and Q. Yuan, "Dynamic multi-keyword ranked search based on bloom filter over encrypted cloud data," *IEEE Access*, vol. 7, pp. 35826–35837, 2019.

[6] Q. Jiang, Y. Qi, S. Qi, W. Zhao, and Y. Lu, "Pbsx: a practical private boolean search using Intel SGX," *Information Sciences*, vol. 521, pp. 174–194, 2020.

[7] 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*, pp. 506–522, Berlin, Germany, May 2004.

[8] 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*, pp. 481–490, June 2012.

[9] 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, 2017.

[10] D. J. Park, K. Kim, and P. J. Lee, "Public key encryption with conjunctive field keyword search," in *Proceedings of the International Workshop on Information Security Applications*, pp. 73–86, Berlin, Germany, August 2004.

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

[12] 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.

[13] C. C. Lee, S. T. Hsu, and M. S. Hwang, "A study of conjunctive keyword searchable schemes," *International Journal of Network Security*, vol. 15, no. 5, pp. 321–330, 2013.

[14] M. S. Hwang, S. T. Hsu, and C. C. Lee, "A new public key encryption with conjunctive field keyword search scheme," *Information Technology and Control*, vol. 43, no. 3, pp. 277–288, 2014.

[15] Y. Zhang, Y. Li, and Y. Wang, "Efficient conjunctive keywords search over encrypted e-mail data in public key setting," *Applied Sciences*, vol. 9, no. 18, p. 3655, 2019.

[16] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 146–162, Berlin, Germany, April 2008.

[17] Y. Zhang, Y. Li, and Y. Wang, "Conjunctive and disjunctive keyword search over encrypted mobile cloud data in public key system," *Mobile Information Systems*, vol. 2018, Article ID 3839254, 11 pages, 2018.

[18] Y. Zhang, Y. Li, and Y. Wang, "Secure and efficient searchable public key encryption for resource constrained environment based on pairings under prime order group," *Security and Communication Networks*, vol. 2019, Article ID 5280806, 14 pages, 2019.

[19] X. Liu, R. H. Deng, K. K. R. Choo et al., "An efficient privacy-preserving outsourced calculation toolkits with multiple keys," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2401–2414, 2016.

[20] Y. Yang, X. Liu, and R. Deng, "Expressive query over outsourced encrypted data," *Information Sciences*, vol. 442-443, pp. 33–53, 2018.

[21] Y. Miao, X. Liu, R. H. Deng et al., "Hybrid keyword-field search with efficient key management for industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3206–3217, 2019.

[22] Y. Yang, X. Liu, R. H. Deng, and J. Weng, "Flexible wildcard searchable encryption system," *IEEE Transactions on Services Computing*, vol. 13, no. 3, pp. 464–477, 2020.

[23] Y. Yang, X. Liu, and R. H. Deng, "Multi-user multi-keyword rank search over encrypted data in arbitrary language," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 2, pp. 320–334, 2020.

[24] J. Li, Y. Shi, and Y. 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.

[25] J. Li, X. Lin, Y. Zhang, and J. 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.

[26] K. He, J. Guo, J. Weng, J. Weng, J. K. Liu, and X. Yi, "Attribute-based hybrid Boolean keyword search over outsourced encrypted data," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2018.

[27] Y. Miao, X. Liu, K. K. R. Choo et al., "Privacy-preserving attribute-based keyword search in shared multi-owner setting," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2019.

[28] K. Zhang, M. Wen, R. Lu, and K. Chen, "Multi-client sublinear boolean keyword searching for encrypted cloud storage with owner-enforced authorization," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2020.

[29] J. Feng, L. T. Yang, Q. Zhu, and K. K. R. Choo, "Privacy-preserving tensor decomposition over encrypted data in a federated cloud environment," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 4, pp. 857–868, 2020.

[30] J. Feng, L. T. Yang, and R. Zhang, "Practical privacy-preserving high-order Bi-lanczos in integrated edge-fog-cloud

architecture for cyber-physical-social systems," *ACM Transactions on Internet Technology*, vol. 19, no. 2, pp. 1–18, 2019.

[31] J. Feng, L. T. Yang, R. Zhang, and B. S. Gavuna, "Privacy preserving tucker train decomposition over Blockchain-based encrypted industrial IoT data," *IEEE Transactions on Industrial Informatics*, p. 1, 2020.

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

[33] Y. Lu, J. Li, and Y. Zhang, "Privacy-preserving and pairing-free multi-recipient certificateless encryption with keyword search for cloud-assisted IIoT," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2553–2562, 2020.

[34] S. Singh, P. K. Sharma, S. Y. Moon, and J. H. Park, "EH-GC: an efficient and secure architecture of energy harvesting Green cloud infrastructure," *Sustainability*, vol. 9, no. 4, p. 673, 2017.

[35] K.-S. Lim, J. Park, and J. Park, "An energy-efficient virtualization-based secure platform for protecting sensitive user data," *Sustainability*, vol. 9, no. 7, p. 1250, 2017.

[36] C.-T. Li, C.-C. Lee, and C.-Y. Weng, "An extended chaotic maps based user authentication and privacy preserving scheme against DoS attacks in pervasive and ubiquitous computing environments," *Nonlinear Dynamics*, vol. 74, no. 4, pp. 1133–1143, 2013.

[37] Y. Zhang, Y. Wang, and Y. Li, "Searchable public key encryption supporting semantic multi-keywords search," *IEEE Access*, vol. 7, pp. 122078–122090, 2019.

[38] A. Joux, "The Weil and Tate pairings as building blocks for public key cryptosystems," in *Proceedings of the International Algorithmic Number Theory Symposium*, pp. 20–3, Berlin, Germany, July 2002.

[39] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 62–91, Berlin, Germany, May 2009.

[40] A. Lewko, "Tools for simulating features of composite order bilinear groups in the prime order setting," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 318–335, Berlin, Germany, April 2012.

[41] A. Lewko and B. Waters, "New techniques for dual system encryption and fully secure HIBE with short ciphertexts," in *Proceedings of the Theory of Cryptography Conference*, pp. 455–479, Berlin, Germany, February 2010.

[42] B. Waters, "Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions," in *Proceedings of the Annual International Cryptology Conference*, pp. 619–636, Berlin, Germany, August 2009.

[43] A. D. Caro, "The java pairing based cryptography library (JPBC)," 2013, http://gas.dia.unisa.it/projects/jpbc/laatstnagekekenop.

[44] W. W. Cohen: Enron e-mail dataset, http://www.cs.cmu.edu/~./enron/.

[45] H. Cui, Z. Wan, R. H. Deng, G. Wang, and Y. Li, "Efficient and expressive keyword search over encrypted data in cloud," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 3, pp. 409–422, 2018.