SHS: Secure Hybrid Search by Combining Dynamic and Static Indexes in PEKS

Peng Xu and Hai Jin, *Senior Member, IEEE*

Abstract—With a significant advance in ciphertext searchability, Public-key encryption with keyword search (PEKS) is the first keyword searchable encryption scheme based on the probabilistic encryption, such that it is more secure than almost all previous schemes. However, there is an open problem in PEKS that its search complexity is linear with the sum of ciphertexts, such that it is inefficient for a mass of ciphertexts.

Fortunately, we find an elegant method that by adaptively taking the keyword trapdoor of each query as an index, the search complexity of the queried keywords can be decreased in a huge degree. We call this method dynamic index (DI) technique. Furthermore, for keywords having not been queried before, we employ deterministic encryption to establish indexes to decrease their first search complexity. We call this method static index (SI) technique. Consequently, we propose a secure hybrid search (SHS) system by combing DI and SI techniques in PEKS to decrease the search complexity of PEKS. At last, we demonstrate its semantic security and convergent search complexity, which is considerably lower than that of PEKS.

Index Terms—public-key encryption with keyword search, dynamic index technique, static index technique, secure hybrid search

1 Introduction

Public-key encryption with keyword search (PEKS) [1] is the first keyword searchable encryption scheme based on the probabilistic encryption. In the aspect of provable security, it particularly achieves the stronger IND-CPA (indistinguishability of ciphertexts under chosen plaintext attack) security than almost all previous schemes, who were based on the deterministic encryption or function such as [2, 3, 4]. In applications, these previous schemes, roughly speaking, needs to share a symmetric secret between each pair of sender and receiver. On the contrary, PEKS generates keyword searchable ciphertexts only by public-key encryption. Therefore, by canceling the management of symmetric secrets, PEKS is more convenient than these previous schemes.

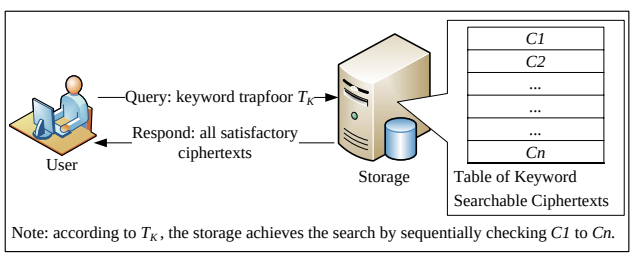


Figure 1. Query and Response of Keyword Searchable Ciphertexts in PEKS

PEKS is a more convenient and secure scheme than almost all previous schemes, however, it has a high search complexity that it is linear with the sum of ciphertexts. For example, in Figure 1, we simply describe the query procedure of PEKS and assume that the storage has stored n keyword searchable ciphertexts. For each query, because the storage can not establish indexes for $\{C1, ..., Cn\}$ by their equality, it has to sequently check all ciphertexts from $C1$ to Cn. Consequently, the search complexity of PEKS is $O(n)$ and obviously inefficient when n is very large.

1.1 Our Ideas

However, we fortunately observe that by adaptively taking the keyword trapdoor of each query as an index, the storage can improve the search performance of PEKS in a huge degree. For the convenience of following citation, we called this novel method dynamic index (DI) technique. Furthermore, DI technique can be conveniently achieved in PEKS. For each query described in Figure 1, the storage adaptively stores the keyword trapdoor T_K of keyword K and takes it as the dynamic index of a group, which contains all keyword searchable ciphertexts containing K. Subsequently, for any new received keyword searchable ciphertext from a user, such as C_{n+1}, the storage can prejudge whether C_{n+1} belongs to the group by its dynamic index. Consequently, DI technique can improve the search performance that when a user query the keyword K again, the storage can respond with all satisfactory keyword searchable ciphertexts in one step. In contrast with the original search of PEKS, DI technique obviously decrease the search complexity, because PEKS needs sequently checking

- *P. Xu and H. Jin are with Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China. E-Mail: {xupeng, hjin}@mail.hust.edu.cn.*

all keyword searchable ciphertexts for any query. In summary, DI technique can tremendously improves the search performance of keywords, which have been queried before. However, for keywords never queried, DI technique is ineffective for the first search of these keywords.

Therefore, we additionally propose a static index (SI) technique to improve the first search performance of keywords, and combine DI and SI techniques to completely improve the search performance of PEKS. Generally speaking, SI technique employs deterministic encryption (DE) [4, 5] to generate an index for each keyword searchable ciphertext. Somewhat specifically, when a user generates a keyword searchable ciphertext of keyword K, he additionally takes the concatenation of K and his own secret as the plaintext input of DE, and takes the output of DE as the static index of the ciphertext. Because all keyword searchable ciphertexts having the same static index contain the same keyword, the storage can partition all these ciphertexts of each user into different groups according to the equality of static indexes, and more efficiently achieve the first search of keywords under the help of the partition.

1.2 Necessity of Combining DI and SI Techniques

From the aspect of security, we further illuminate the reasons of combining above two techniques to accelerate the search of PEKS.

In order to keep the security of static indexes, SI technique has to take user's own secret as a partial plaintext input of DE (more reasons will be given in Section 4). So the static indexes generated by different users always are distinct. In other words, any two static indexes respectively generated by two users are unequal, even for the same keyword. So when employing SI technique in PEKS, the first search complexity of keywords will be decreased to a function of the number of users.

In contrast, DI technique is only relative to keywords. To query keywords having been queried before, its search complexity is independent with the number of users, such that it is more efficient than SI technique. But it is ineffective for the first search of keywords. Consequently, we respectively employ SI and DI techniques to respectively accelerate the first search and subsequent searches of keywords.

1.3 Our Contributions

To improve the search performance of PEKS, we novelly propose a secure hybrid search (SHS) system by combining DI and SI techniques in PEKS. SHS consists of three procedures as follows: *Initialization*, *Ciphertext Submission* and *Ciphertext Query*. In order to clearly describe these procedures, we first present six kernel algorithms based on the definitions of PEKS, public-key encryption (PKE) and DE, which are simply listed as follows:

- $Setup_{SHS}$ generates the required system parameters.
- $Trapdoor_{SHS}$ and $Encrypt_{SHS}$ respectively generate a keyword trapdoor and a keyword searchable ciphertext for an inputted keyword.
- $ESInd_{SHS}$ and $EDInd_{SHS}$ respectively establish the static and dynamic indexes for an inputted ciphertext.
- $Search_{SHS}$ finds out all ciphertexts containing the same keyword with an inputted keyword trapdoor.

Secondly, we construct the procedures based on these algorithms.

Thereafter, we demonstrate the semantic security of SHS based on the semantically secure PEKS, PKE and DE. By comparing the search complexity of SHS and PEKS, SHS is proved to be more efficient than PEKS. Moreover, the search complexity of SHS is convergent while increasing queries. On the contrary, the search complexity of PEKS is divergent.

1.4 Related Works about PEKS

PEKS was first proposed by Boneh et al. in 2004 [1], and realized based on anonymous identity-based encryption [6, 7, 8, 9]. Abdalla et al. [10] perfected the foundations of PEKS, and proposed a more secure transformation from anonymous IBE to PEKS and an extended PEKS. Baek et al. [11] freed the secure channel between the storage and users by employing public-key encryption. To achieve conjunctive keyword search, two schemes on public-key encryption with conjunctive keyword search (PECKS) [12, 13] were respectively proposed. Furthermore, Bethencourt et al. succeeded on public-key encryption with conjunctive keyword range search [14] by anonymous hierarchical IBE (HIBE) [7] in 2006, and updated their work in 2007 [15]. Boneh et al. proposed a novel technique called hidden vector encryption (HVE) to achieve conjunctive, range and subset searches [16]. Camenisch et al. [17] employed the committed two-part computation protocol to achieve the oblivious keyword in the generation of keyword trapdoor. References [18, 19] proposed several efficient PECKS by sharing a secret between sends and receivers. So far as we known, almost all related works about PEKS are focus on the versatile searchability. Our paper is the first one to improve the search complexity of PEKS.

1.5 Organization

The organization of this paper is as follows. In Section 2, some preliminary definitions will be given. In Section 3, we propose the framework of SHS. Section 4 analyzes the security of SHS. Section 5 analyzes

the search complexity of SHS. At last, conclusion is presented in Section 6.

2 PRELIMINARIES

We redefine PEKS, PKE and DE as follows.

Definition 1 (PEKS). *PEKS consists of following polynomial time algorithms:*

- $Setup_{PEKS}(k)$*: Take as input a security parameter* $k \in \mathbb{N}$*, then probabilistically generate a pair of public-and-private system parameters* $\langle Pub_{PEKS}, Pri_{PEKS}\rangle$*, in which* Pub_{PEKS} *includes the keyword space* $\mathcal{K}$*.*
- $Trapdoor_{PEKS}(Pri_{PEKS}, K)$*: Take as input a private system parameter* Pri_{PEKS} *and a keyword* $K \in \mathcal{K}$*, then probabilistically generate a keyword trapdoor* T_K*.*
- $Encrypt_{PEKS}(Pub_{PEKS}, K)$*: Take as input a public system parameters* Pub_{PEKS} *and a keyword* $K \in \mathcal{K}$*, then probabilistically generate a keyword searchable ciphertext of* K*.*
- $Test_{PEKS}(Pub_{PEKS}, T_K, C)$*: Take as input a public system parameters* Pub_{PEKS}*, a keyword trapdoor* T_K *and a keyword searchable ciphertext* $C = Encrypt_{PEKS}(Pub_{PEKS}, K')$*, then return* B*, where*

$$B = \begin{cases} 1 & if\ K' = K; \\ 0 & otherwise. \end{cases} \quad (1)$$

Moreover, PEKS satisfies the consistency that for any keyword $K' \in \mathcal{K}$ *and its keyword trapdoor* T_K*,* $Test_{PEKS}(Pub_{PEKS}, T_K, C')$ *returns '1' if and only if* $K = K'$*, in which* $C' = Encrypt_{PEKS}(Pub_{PEKS}, K')$

Definition 2 (PKE). *PKE consists of following polynomial time algorithms:*

- $Setup_{PKE}(k)$*: Take as input a security parameter* $k \in \mathbb{N}$*, then probabilistically generate a pair of public-and-private keys* $\langle Pub_{PKE}, Pri_{PKE}\rangle$*, in which* Pub_{PKE} *includes the plaintext space* $\mathcal{M}_{PKE}$*.*
- $Encrypt_{PKE}(Pub_{PKE}, M)$*: Take as input a public key* Pub_{PKE} *and a plaintext* $M \in \mathcal{M}_{PKE}$*, then probabilistically generate a ciphertext.*
- $Decrypt_{PKE}(Pri_{PKE}, C)$*: Take as input a private key* Pri_{PKE} *and a ciphertext* C*, then return a plaintext* M*.*

Moreover, PKE satisfies the consistency that for any plaintext $M \in \mathcal{M}_{PKE}$*,* $Decrypt_{PKE}(Pri_{PKE}, C)$ *always returns* M *if and only if* $C = Ecrypt_{PKE}(Pub_{PKE}, M)$*.*

Definition 3 (DE). *DE consists of following polynomial time algorithms:*

- $Setup_{DE}(k)$*: Take as input a security parameter* $k \in \mathbb{N}$*, then probabilistically generate a pair of public-and-private keys* Pub_{DE} *and* Pri_{DE}*, in which* Pub_{DE} *includes the plaintext space* $\mathcal{M}_{DE}$*.*
- $Encrypt_{DE}(Pub_{DE}, M)$*: Take as input a public system parameter* Pub_{DE} *and a plaintext* $M \in \mathcal{M}_{DE}$*, then deterministically generate a ciphertext.*
- $Decrypt_{DE}(Pri_{DE}, C)$*: Take as input a private system parameter* Pri_{DE} *and a deterministically generated ciphertext* C*, then return a plaintext* M*.*

Moreover, DE satisfies the consistency that for any plaintext $M \in \mathcal{M}_{DE}$*,* $Decrypt_{DE}(Pri_{DE}, C)$ *always returns* M *if and only if* $C = Ecrypt_{DE}(Pub_{DE}, M)$*.*

3 FRAMEWORK OF SHS

Let $||$ denote concatenation operation. Let $|T|$ denote the count of elements/records in the set/table T. Let $T[i]$ denote the i-th element/record of the set/table T. Let Pub^{TTP}_{PEKS} and Pri^{TTP}_{PEKS} respectively denote the public-and-private system parameters of PEKS generated by trusted third part (TTP). Let Pub^{TTP}_{PKE} and Pri^{TTP}_{PKE} respectively denote the public-and-private keys of PKE generated by TTP. Let Pub^{TTP}_{DE} and Pri^{TTP}_{DE} respectively denote the public-and-private keys of DE generated by TTP.

3.1 Kernel Algorithms of SHS

Based on the definitions of PEKS, PKE and DE, SHS consists of following polynomial time algorithms:

- $Setup_{SHS}$ described in Algorithm 1 generates various parameters according to an inputted state string st. It can generate the system parameters of PEKS, PKE and DE, two tables $SITab$ and $DITab$ to respectively store static and dynamic indexes, and a table $CTab_{index}$ for an index $index$ to store the relevant ciphertexts.

Algorithm 1 $Setup_{SHS}$

Input:
a security parameter, $k \in \mathbb{N}$;
a state string, $st \in \{IT, CT, PEKS, PKE, DE\}$;
Output:
1: **if** $st == IT$ **then**
2: **return** $\langle SITab, DITab\rangle$;
3: **end if**
4: **if** $st == CT$ **then**
5: **return** $\langle name, CTab_{name}\rangle$;
6: **end if**
7: **if** $st == PEKS$ **then**
8: **return** $Setup_{PEKS}(k)$;
9: **end if**
10: **if** $st == PKE$ **then**
11: **return** $Setup_{PKE}(k)$;
12: **end if**
13: **if** $st == DE$ **then**
14: **return** $Setup_{DE}(k)$;
15: **end if**

- $Trapdoor_{SHS}$ described in Algorithm 2 generates a required keyword trapdoor.
- $Encrypt_{SHS}$ described in Algorithm 3 generates a keyword searchable ciphertext by concatenating two outputs of $Encrypt_{DE}$ and $Encrypt_{PEKS}$.

Algorithm 2 $Trapdoor_{SHS}$

Input:
Pri^{TTP}_{PEKS}; Pri^{TTP}_{PKE};
a ciphertext, $C = Encrypt_{PKE}(Pub^{TTP}_{PKE}, K)$;
Output:
$K = Decrypt_{PKE}(Pri^{TTP}_{PKE}, C)$;
return $Trapdoor_{PEKS}(Pri^{TTP}_{PEKS}, K)$;

Algorithm 3 $Encrypt_{SHS}$

Input:
Pub^{TTP}_{PEKS}; Pub^{TTP}_{DE};
the secret of user, S^{User};
a keyword, K;
Output:
$SI = Encrypt_{DE}(Pub^{TTP}_{DE}, K||S^{User})$;
$SC = Encrypt_{PEKS}(Pub^{TTP}_{PEKS}, K)$
return $C = SI||SC$;

- $ESInd_{SHS}$ described in Algorithm 4 establishes a static index for an inputted ciphertext. If the static index $index$ of the inputted ciphertext C has been established, it adds C into the existing $CTab_{index}$; otherwise, it adds $index$ into $SITab$ and C into the newly created $CTab_{index}$.

Algorithm 4 $ESInd_{SHS}$

Input:
a keyword searchable ciphertext, C;
Output:
if $\langle SITab, DITab\rangle$ is not set up **then**
  $\langle SITab, DITab\rangle = Setup(None, IT)$;
end if
parsing C as $SI||SC$;
if $SI \in SITab$ **then**
  add SC into $CTab_{SI}$;
else
  $\langle name, CTab_{name}\rangle = Setup(None, CT)$;
  $name = SI$;
  add SC into $CTab_{SI}$;
end if
return 1;

- $EDInd_{SHS}$ described in Algorithm 5 establishes a dynamic index for an inputted ciphertext. For the inputted ciphertext C, if there is a $index \in DITab$ has $Test_{PEKS}(Pub^{TTP}_{PEKS}, index, C) = 1$, then it adds C into the existing $CTab_{index}$.
- $Search_{SHS}$ described in Algorithm 6 responds a query T_K. It contains two procedures: firstly, if T_K is existing in $DITab$, then it returns $CTab_{T_K}$; otherwise, for each $index \in SITab$ it sequently checks the first ciphertext of $CTab_{index}$ by T_K, and adds all successfully verified ciphertexts into the newly created $CTab_{T_K}$, finally adds T_K into $DITab$ and returns $CTab_{T_K}$.

Algorithm 5 $EDInd_{SHS}$

Input:
Pub^{TTP}_{PEKS};
a keyword searchable ciphertext, C;
Output:
if $\langle SITab, DITab\rangle$ is not set up **then**
  $\langle SITab, DITab\rangle = Setup(None, IT)$;
end if
parsing C as $SI||SC$;
for $i = 1$ to $|DITab|$ **do**
  if $Test_{PEKS}(Pub^{TTP}_{PEKS}, DITab[i], SC) == 1$ **then**
    add SC into $CTab_{DITab[i]}$;
    return 1;
  end if
end for
return 0;

Algorithm 6 $Search_{SHS}$

Input:
Pub^{TTP}_{PEKS};
a keyword trapdoor, T_K;
Output:
if $T_K \in DITab$ **then**
  return $CTab_{T_K}$;
end if
$\langle name, CTab_{name}\rangle = Setup(None, CT)$;
$name = T_K$;
add T_K into $DITab$;
for $i = 1$ to $|SITab|$ **do**
  if $Test_{PEKS}(Pub^{TTP}_{PEKS}, T_K, CTab_{SITab[i]}[1]) == 1$ **then**
    add $CTab_{SITab[i]}$ into $CTab_{T_K}$;
    delete $CTab_{SITab[i]}$ and $SITab[i]$;
  end if
end for
return $CTab_{T_K}$;

3.2 Procedures of SHS

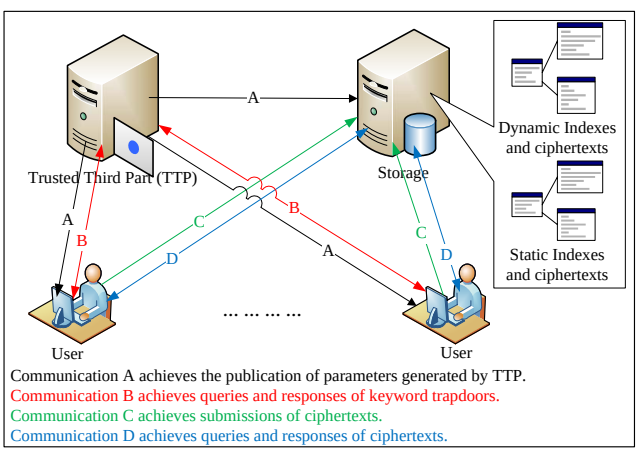
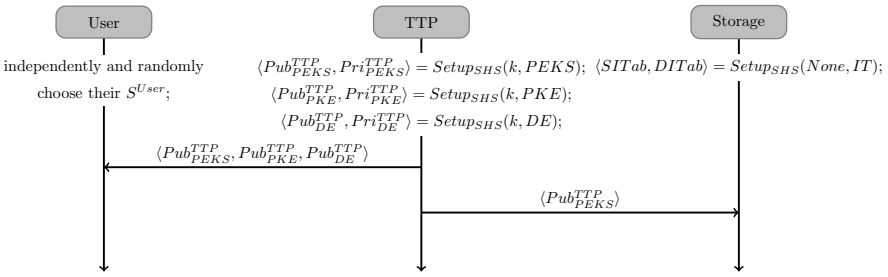
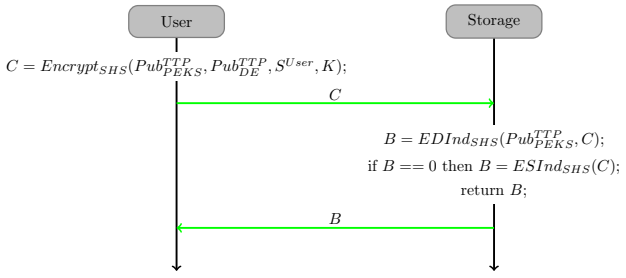


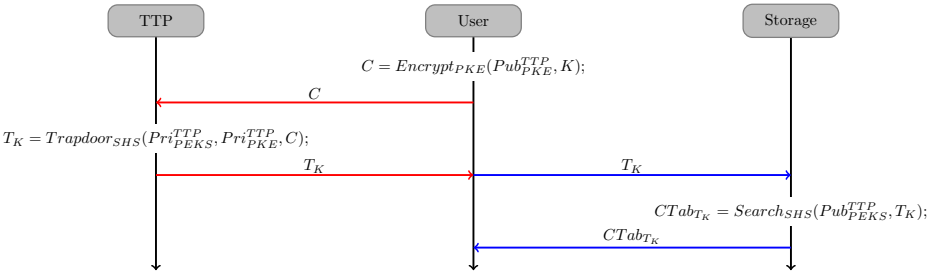
Figure 2. Communications in SHS



(a) The Procedure of Initialization.



(b) The Procedure of Ciphertext Submission.



(c) The Procedure of Ciphertext Query.

Figure 3. Three Procedures of SHS.

SHS consists of three procedures among TTP, the storage and users, as follows: *Initialization*, *Ciphertext Submission* and *Ciphertext Query*. In Figure 2, we describe the communications among TTP, the storage and users in SHS. Furthermore based on the kernel algorithms of SHS, we respectively describe above procedures as follows:

- *Initialization*. TTP, the storage and users independently generate their system parameters by $Setup_{SHS}$. TTP publishes its public system parameters to the storage and users by *Communication A* in Figure 2. The details are presented in Subfigure 3(a) of Figure 3.
- *Ciphertext Submission*. A user generates a ciphertext by $Encrypt_{SHS}$, and submits it to the storage by *Communication C* in Figure 2. The storage establishes dynamic index or static index tables for the received ciphertexts by $EDInd_{SHS}$ and $EDInd_{SHS}$. The details are presented in Subfigure 3(b) of Figure 3.
- *Ciphertext Query*. First, a user inquires TTP to respond a keyword trapdoor of a keyword K by *Communication B* in Figure 2. Secondly, the user submits keyword trapdoor to the storage and gets the relevant ciphertexts by *Communication D* in Figure 2. The details are presented in Subfigure 3(c) of Figure 3.

4 Security Analysis of SHS

Generally speaking, to realize a secure search on ciphertexts, it should be necessary to keep the semantic security of plaintexts. Specifically in SHS, it should keep the semantic security of keywords. To realize this security, we construct SHS based on PEKS, PKE and DE. Moreover, we can demonstrate that if PEKS, PKE and DE are semantically secure, then so is SHS. According to Table 1, the demonstration consists of following component conclusions:

- *If the PEKS employed in SHS is semantically secure, then* $T_K = Trapdoor_{SHS}(Pri^{TTP}_{PEKS}, K)$ *and* $Encrypt_{PEKS}(Pub^{TTP}_{PEKS}, K)$ *can keep the semantic*

Communication	Messages
B	$Encrypt_{PKE}(Pub_{PKE}^{TTP}, K)$, $T_K = Trapdoor_{SHS}(Pri_{PEKS}^{TTP}, K)$
C	$Encrypt_{DE}(Pub_{DE}^{TTP}, K\|\|S^{User})$, $Encrypt_{PEKS}(Pub_{PEKS}^{TTP}, K)$
D	$T_K = Trapdoor_{SHS}(Pri_{PEKS}^{TTP}, K)$, $CTab_{T_K} = Search_{SHS}(Pub_{PEKS}^{TTP}, T_K)$

Table 1
All Interactive Messages Containing a keyword in SHS.

Symbol	Specification
$DITab$	A dynamic index table stores all "distinct" keyword trapdoors having been queried before. (The "distinct" means that there is not any two keyword trapdoors in $DITab$ contain the same keyword.)
$SITab$	A static index table stores all inequivalent static indexes generated by $Encrypt_{DE}$.
$CTab_{index}$	A ciphertext table stores all ciphertexts belonging to $index$, in which $index \in DITab \bigcup SITab$.
$\mathcal{K}$	A keyword space $\mathcal{K}$ contains $\|\mathcal{K}\|$ keywords.

Table 2
Several Symbols of SHS and their Specifications.

security of K. According to the definition of the semantic security of PEKS [1], a semantically secure PEKS can keep the semantic security of keywords even if keyword trapdoors are public. Therefore, $T_K = Trapdoor_{SHS}(Pri_{PEKS}^{TTP}, K)$ and $Encrypt_{PEKS}(Pub_{PEKS}^{TTP}, K)$ can keep the semantic security of K, if the PEKS employed in SHS is semantically secure.

- *If the PKE employed in SHS is semantically secure, then $Encrypt_{PKE}(Pub_{PKE}^{TTP}, K)$ can keep the semantic security of K.* According to the definition of the semantic security of PKE [20], this conclusion obviously holds.
- *If the DE employed in SHS is semantically secure, then $Encrypt_{DE}(Pub_{DE}^{TTP}, K||S^{User})$ can keep the semantic security of K even under keyword guessing attacks.* Differently with probabilistic encryption, the definition of the semantic security of DE assumed that all plaintexts have enough entropy to defense guessing attacks. But in SHS, keywords would be semantical and indexed in a dictionary, such that they do not have enough entropy. Therefore, it is insecure to directly take the keyword space as the plaintext space of DE in SHS. To solve this problem, we elegantly take the concatenation of the keyword space and the random secret space of users' as the plaintexts space of DE. By this method, the entropy of the plaintexts space of DE can be sufficiently increased, such that it satisfies the assumption in the definition of the semantic security of DE, and defends keyword guessing attacks. Hence, $Encrypt_{DE}(Pub_{DE}^{TTP}, K||S^{User})$ can keep the semantic security of K even under keyword guessing attacks, if the DE employed in SHS is semantically secure.

In summary, if the PEKS, PKE and DE employed in SHS are semantically secure, then so is SHS even under keyword guessing attacks.

5 Search Complexity of SHS

To clearly present the search complexity of SHS, we first list several relevant symbols in Table 2. According to these symbols, the sum of keyword searchable ciphertexts generated by users is

$$sum = \sum_{i=1}^{|SITab|} |CTab_{SITab[i]}| + \sum_{j=1}^{|DITab|} |CTab_{DITab[j]}| \tag{2}$$

in which both $|CTab_{SITab[i]}|$ and $|CTab_{DITab[j]}|$ are ≥ 1. For each search in sum keyword searchable ciphertexts, PEKS should check ciphertexts one by one, so the search complexity of PEKS is sum. In contrast with PEKS, the search complexity of SHS is

$$O(|SITab| + |DITab|) \tag{3}$$

It is obvious that SHS always is more efficient than PEKS, except in worst case that for any $i \in [1, |SITab|]$ and $j \in [1, |DITab|]$, the equations $|CTab_{SITab[i]}|$=1 and $|CTab_{DITab[j]}|$=1 simultaneously holds. In the worst case, SHS at least has the same search complexity with PEKS.

Furthermore, the search complexity of SHS is convergent to $|\mathcal{K}|$ following the increase of $|DITab|$. On the contrary, the search complexity of PEKS is linear with sum. Assuming all keywords are uniformly chosen by users, the expected search complexity of SHS is

$$\begin{aligned} &O(|DITab| \cdot \Pr(T_K \in DITab) + \\ &\quad (|DITab| + |SITab|) \cdot \Pr(T_K \notin DITab)) \\ &= O(|DITab| + |SITab| \cdot \Pr(T_K \notin DITab)) \\ &= O(|DITab| + |SITab| \cdot (1 - \frac{|DITab|}{|\mathcal{K}|})) \end{aligned} \tag{4}$$

in which, T_K is a certain keyword trapdoors generated by a user. Following the increase of $|DITab|$, the search complexity of SHS obviously converges to $O(|DITab|)$. Furthermore, it can be alternatively presented as $O(|\mathcal{K}|)$ since it trivially has $|DITab| \leq |\mathcal{K}|$.

Under the more general assumption that all keywords are non-uniformly chosen by users, $DITab$ contains the frequently used keywords, so for any keyword trapdoor T_K it trivially has

$$\Pr(T_K \notin DITab) < (1 - \frac{|DITab|}{|\mathcal{K}|}) \tag{5}$$

Therefore, the search complexity of SHS converge to $O(|DITab|)$ faster.

In summary, the search complexity of SHS is more efficient than PEKS, except the worst case that they have the same complexity. Furthermore, storage will gradually complete a "full" dynamic index table while increasing queries, such that the search complexity is convergent. On the contrary, the search complexity of PEKS is divergent and linearly increases with the sum of keyword searchable ciphertexts.

6 Conclusion

To decrease the search complexity of PEKS, we novelly proposed SHS system by combining DI and SI techniques in PEKS. By analyzing the search complexity of SHS and PEKS, SHS is significantly more efficient than PEKS. Furthermore, it is noticeable that the search complexity of SHS converges to the sum of keywords while increasing queries. On the contrary, the search complexity of PEKS is divergent and linearly increase with the sum of keyword searchable ciphertexts, which is much larger than the sum of keywords. At last, we analyzed the security of SHS by demonstrating that SHS is semantically secure, if the PEKS, PKE and DE employed in SHS is semantically secure.

References

[1] D. Boneh, G. D. Crescenzo, and R. O. et al., "Public key encyrption with keyword search," in *Advances in Cryptology-EUROCRYPT 2004*, ser. LNCS, C. Cachin and J. Camenisch, Eds., vol. 3027. Springer-Verlag, 2004, pp. 506–522.

[2] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, Berkeley, CA , USA, 2000, pp. 44–55.

[3] E.-J. Goh, "Secure indexes," 2003, http://eprint.iacr.org/2003/216.pdf.

[4] M. Bellare, A. Boldyreva, and A. ONeill, "Deterministic and efficiently searchable encryption," in *Advances in Cryptology - CRYPTO 2007*, ser. LNCS, A. Menezes, Ed., vol. 4622. Santa Barbara, California, United States: Springer-Verlag, 2007, pp. 535–552.

[5] M. BELLARE, M. FISCHLIN, and A. O. et al., "Deterministic encryption: Definitional equivalences and constructions without random oracles," in *Advances in Cryptology-CRYPTO 08*, ser. LNCS, D. Wagner, Ed., vol. 5157. Springer-Verlag, 2008, pp. 360–378.

[6] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology-CRYPTO 2001*, ser. LNCS, J. Kilian, Ed., vol. 2139. Santa Barbara, California, United States: Springer-Verlag, 2001, pp. 213–239.

[7] X. Boyen and B. Waters, "Anonymous hierarchical identity-based encryption (without random oracles)," in *Advances in Cryptology-CRYPTO 2006*, ser. LNCS, C. Dwork, Ed., vol. 4117. Santa Barbara, California, United States: Springer-Verlag, 2006, pp. 290–307.

[8] C. Gentry, "Practical identity-based encyrption without random oracles," in *Advances in Cryptology-EUROCRYPT 2006*, ser. LNCS, S. Vaudenay, Ed., vol. 4004. Russia: Springer-Verlag, 2006, pp. 445–464.

[9] L. Ducas, "Anonymity from asymmetry: New constructions for anonymous hibe," in *The Cryptographers' Track at the RSA Conference 2010*, ser. LNCS, J. Pieprzyk, Ed., vol. 5985. San Francisco, CA, USA: Springer Berlin, 2010, pp. 148–164.

[10] M. Abdalla, M. Bellare, and D. C. et al., "Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions," in *Advances in Cryptology-CRYPTO 2005*, ser. LNCS, V. Shoup, Ed., vol. 3621. Santa Barbara, California, United States: Springer-Verlag, 2005, pp. 205–222.

[11] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Computational Science and Its Applications-ICCSA 2008*, O. Gervasi, Ed., vol. 5072. Springer-Verlag, 2008, pp. 1249–1259.

[12] D. J. Park, K. Kim, and P. J. Lee, "Public key encryption with conjunctive field keyword search," in *WISA 2004*, ser. LNCS, C. Lim and M. Yung, Eds., vol. 3325. Spring-Verlag, 2004, pp. 73–86.

[13] Y. H. Hwang and P. J. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Pairing 2007*, ser. LNCS, T. Takagi, Ed., vol. 4575. Springer-Verlag, 2007, pp. 2–22.

[14] J. Bethencourt, T.-H. H. Chan, and A. P. et al., "Anonymous multi-attribute encryption with range query and conditional decryption," Carnegie Mellon University, Tech. Rep. CMU-CS-06-135, 2006.

[15] E. Shi, J. Bethencourt, and T.-H. H. C. et al., "Multi-dimensional range query over encrypted data," Carnegie Mellon University, Tech. Rep. CMU-CS-06-135, 2007.

[16] D. Boneh and B. Waters, "conjunctive, subset, and range queries on ecrypted data," in *proceedings of TCC'07*, ser. LNCS, S. P. Vadhan, Ed., vol. 4392. Springer-Verlag, 2007, pp. 535–554.

[17] J. Camenisch, M. Kohlweiss, and A. R. et al., "Blind and anonymous identity-based encryption and authorised private searches on public key encrypted data," in *Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography: PKC '09*, ser. LNCS, vol. 5443. CA: Springer-Verlag, 2009, pp. 196–214.

[18] L. Ballard, S. Kamara, and F. Monrose, "Achiev-

ing efficient conjunctive keyword searches over encrypted data," in *ICICS 2005*, ser. LNCS, S. et al., Ed., vol. 3783. Springer-Verlag, 2005, pp. 414–426.

[19] E.-K. Ryu and T. Takagi, "Efficient conjunctive keyword-searchable encryption," in *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on*. Niagara Falls, Ontario, Canada: IEEE Computer Society, 2007, pp. 409 – 414.

[20] S. Goldwasser and S. Micali, "Probabilistic encyrption," in *Proceedings of the fourteenth annual ACM symposium on Theory of computing*. San Francisco, California, United States: ACM, 1982, pp. 365–377.