Searchable Keyword-Based Encryption*

Dong Jin Park†, Juyoung Cha‡, and Pil Joong Lee‡

† Samsung Electronics Co., Ltd., Korea

djpax.park@samsung.com

‡ Information Security Laboratory, Dept. of EEE, POSTECH, Pohang, Korea

jycha@postech.ac.kr, pjl@postech.ac.kr

Abstract

To solve the problem of searching on encrypted data, many keyword search schemes have been proposed in recent years. The goal of such schemes is to enable a user to give an untrusted storage server the ability only to test whether an encrypted document contains a few keywords without learning anything else about the document. In this paper, we are concerned with decrypting the searched results as well as searching for desired documents. In the previously proposed schemes, except for the work by Waters *et al.* [26], a user decrypts searched documents using his private key, A_{priv}, or a symmetric key. Our another goal is to enable a user to give a proxy the ability to decrypt only the ciphertexts containing desired keywords, but not other ciphertexts. We propose a new mechanism, *Searchable Keyword-Based Encryption* (SKBE) which satisfies both the above goals. As a result of adding the delegation of decryption ability, our mechanism works more securely and efficiently in several applications, such as email gateways, secure audit logs, and decryption key delegation systems, than any of the previously proposed schemes. We formalize this mechanism, define its security model and propose an efficient construction whose security is proved in a random oracle model under the *Bilinear Diffie-Hellman Inversion* assumption. The scheme is constructed based on the *Public Key Encryption with Conjunctive Field Keyword Search* scheme in [21] by using a hybrid encryption technique.

keywords: Searching on encrypted data, searchable encryption, delegating decryption key, PEKS, PECK, identity-based cryptosystem

1 Introduction

Recently, there has been interest in the problem of searching on encrypted data. Consider a user with limited resources storing her data on remote and untrusted storage servers. To preserve confidentiality, it is desirable to store the data in encrypted form. However, encryption makes it hard to retrieve data selectively from the server. In other words, when a user wishes to only retrieve specific content (e.g., containing certain words), it is hard to let the server perform the search for desired data without any loss of data confidentiality. To address this problem, various forms of *searchable encryption* have been proposed [1, 3, 7, 11, 16, 17, 21, 25, 26]. For providing secure searchable encryption, most of these schemes encrypt the content of data and attach related keywords to it in encrypted form. i.e.,

$$[E_{key_E}(M), \mathsf{SKE}(key_I, (W_1, W_2, \cdots, W_m))]$$

where the first component is an encryption of message M and the second one is called a searchable keyword encryption (SKE) of keywords $(W_1, W_2, \cdots, W_m)$. A key key_E can be a symmetric key or a public key, depending on the encryption algorithm. We call a message associated with related keywords a *document*. When a user wants to retrieve documents containing a keyword W, she produces a *trapdoor for keyword* W enabling to test for the existence of the keyword within the associated encryptions and sends it to the (untrusted) remote server. The trapdoor for W reveals only which encryptions contain keyword W and no other information. Without a trapdoor, the server learns nothing about encryptions.

*This research was done during the first author was enrolled in POSTECH. This research was supported by University IT Research Center Project, the Brain Korea 21 Project, and grant No. R01-2005-000-10713-0 from the research program of KOSEF.

Related works. Song, Wagner, and Perrig introduced such a searchable encryption in [25]. They proposed a symmetric key scheme in which the same key was used to make SKEs and trapdoors. Afterwards, several schemes have been proposed to improve and extend this scheme [3, 7, 16, 17, 21, 23, 26]. In [16], Goh proposed an efficient symmetric key scheme using Bloom filters. The scheme can determine whether a document contains a keyword in a constant time. Both of these schemes [16, 25] are symmetric key schemes, so they are not applicable to a public key system such as the email gateway introduced by Boneh *et al.* [3]. For the public key systems, Boneh *et al.* proposed the *Public Key Encryption with Keyword Search* (PEKS) whose ciphertexts are created using a public key. In [26], Waters *et al.* presented an encrypted and searchable audit log (secure audit log), which is a good application of searchable encryptions, and proposed two schemes to build the system.

However, none of these schemes [3, 16, 25, 26] support a secure conjunctive keyword search, which is an indispensable requirement for the efficient and secure search in some applications, especially in secure audit logs. For example, an audit escrow agent in a secure audit log system may not give a trapdoor of a single keyword, such as "Alice", to an investigator, because the trapdoor of single keyword may match a huge number of audit logs including many unnecessary ones. There are two naive solutions for the conjunctive search, set intersection [16] and meta keyword. These methods and their limitations are well explained in [17]. However, neither solution is appropriate or practical. The set intersection makes an untrusted server learn which documents match each individual keyword, and, over time, the server may use this information for statistical analysis to deduce information about the user's documents. The meta keyword approach is impractical because it requires huge storage and searching time proportional to the number of keyword fields. The first secure conjunctive keyword search scheme was proposed by Golle *et al.* in a symmetric key setting [17] and Park *et al.* proposed the public key analogue of this scheme [21]. Recently proposed searchable encryption schemes, [23] and [7] also support the conjunctive keyword search.

Propositions. In this paper, we introduce a new mechanism called *"Searchable Keyword-Based Encryption* (SKBE)*"*, which is a natural extension of PECK. SKBE considers decrypting the searched results as well as searching for desired documents. In the previously proposed searchable encryptions, except for the work by Waters *et al.* [26], in order to decrypt searched documents a user uses a symmetric key [7, 11, 16, 17, 25] or her private key, A_{priv} [1, 3, 21]. In addition to searching ability in previous schemes, SKBE's another goal is to enable a user to give a proxy the ability to decrypt only the encryptions containing desired keywords, but not other encryptions. For providing the most powerful functionality, it is designed for supporting conjunctive keyword search in a public key setting. SKBE is a public key encryption with the following functionalities. A message is encrypted using a public key A_{pub}. Then, the ciphertext depends on the keywords associated with the message. Given certain information called a *decrypt trapdoor for specific keywords* W_i's, ciphertexts containing all of keywords W_i's can be decrypted without a private key A_{priv}. Similar to searchable encryptions, given another certain information called a *search trapdoor for specific keywords* W_i's, we can test whether a ciphertext contains keywords W_i's all, but get no other information about its original document. These trapdoors can be generated only with A_{priv}. Without a trapdoor(search trapdoor or decrypt trapdoor), a ciphertext does not reveal anything about its corresponding document.

1.1 Motivation and Applications

SKBE is of interest since it guarantees more secure and efficient operation than searchable encryption or ID-based encryption (IBE) in applications like email gateways, decryption key delegation systems, and secure audit logs.

Email gateway. Consider an email gateway. Suppose Alice wishes to read her email on a number of devices: laptop, desktop, pager, etc., and she uses the method (of PEKS or PECK) in [3, 21]. Alice's mail gateway is supposed to route an email to an appropriate device based on the keywords in the email. For example, when Bob sends an email with the keyword "urgent", the mail is routed to Alice's pager. Similarly, for "lunch", the mail is routed to Alice's desktop to be read later. For Alice to decrypt and read the mail, Alice's private key A_{priv} has to be embedded in her devices. Now, suppose an adversary succeeds in attacking her pager and learning A_{priv}. Afterwards, he can read not only the mails in the pager but all of Alice's mails in the rest of her devices. However, our SKBE can solve this problem because in our mechanism each device has a decrypt trapdoor for

its own keyword (e.g., "urgent" for pager) and decrypts the corresponding mails using this trapdoor instead of A_{priv}. Therefore, even if an adversary succeeds in learning the decrypt trapdoor of the pager, he cannot decrypt and read mails in other devices.

Delegation of decryption keys. Consider the delegation of decryption keys. Suppose Alice has several assistants, each responsible for a different task (e.g., one is "purchasing", another is "human-resources", etc.). Suppose Alice wants to delegate decryption keys to her assistants, so they can decrypt mails corresponding to their work, and she uses the method in an IBE scheme [5]. Alice plays the role of the PKG, and only she knows her master-key. Bob encrypts a mail to Alice using the subject line as an IBE encryption key (e.g., if the subject is "application for secretary", he uses "human-resources" as the key). Alice can decrypt the mail using her master-key. She gives one private key to each of her assistants corresponding to the assistant's responsibility. Each assistant can then decrypt messages whose subject line falls within his responsibilities, but cannot decrypt messages intended for other assistants. This method has two drawbacks. The first one is that Bob should know the tasks of Alice's assistants, the standard for choosing an appropriate IBE encryption key. The other is that the subject line must be known to receivers so that they can use an appropriate decryption key. Although these may be small flaws, sometimes there are situations when all information must be hidden. Moreover, if Bob does not need to know the assistants' tasks to encrypt a mail, it will be more comfortable for him to send a mail to Alice. Our SKBE can solve these flaws by giving Alice's assistants each appropriate search trapdoor and decrypt trapdoor according to their work. Bob encrypts a mail using the encryption algorithm of SKBE and sends it to Alice. In this process, he need not know the tasks of Alice's assistants, nor expose any information of the message. The assistant gives the mail server the search trapdoor, and the server can search for mails falling within the assistant's responsibility using this trapdoor. Then the server sends the resulting mails to the assistant, who can decrypt the mails with the decrypt trapdoor he received from Alice. Also, our scheme can be efficiently used when the structure of the assistants is hierarchial since the scheme supports a secure conjunctive keyword search.

Secure audit log. A similar mechanism to our scheme was suggested for building a secure audit log by Waters *et al.* in [26]. The scheme allows a designated trusted party, named the audit escrow agent, to construct trapdoors which allow (less trusted) investigators in possession of such trapdoors to search for and decrypt log entries containing a given keyword. The escrow agent can distribute a trapdoor to an investigator if he deems it appropriate. The investigator sends the trapdoor to a database server (storing encrypted log entries) and requests entries containing the keyword. The server finds and decrypts the entries, and sends them to the investigator. Here, observe that the database server must be trusted; if not, the above method is not secure or requires an extra path between the escrow agent and investigators to be secure [26]. This problem could be resolved by our SKBE because SKBE has a search trapdoor which can be used only to search for the appropriate encryptions but not decrypt them. In SKBE the escrow agent gives a search trapdoor and a decrypt trapdoor to an investigator, and the investigator sends only the search trapdoor to the database server. Therefore, the server does not need to be trusted because he can only search for the entries for the search trapdoor but can not decrypt them. In addition, we will design our SKBE for supporting a conjunctive keyword search, while the Waters *et al.*' scheme does not.

1.2 Our Contributions

Our main contribution is the proposition of the searchable keyword-based encryption scheme. For some applications, such as email gateways, decryption key delegation systems and secure audit logs, the scheme guarantees more secure and efficient functionalities than any other scheme. We formalize SKBE and construct an efficient and provably secure scheme using bilinear maps. The construction is based on the *Bilinear Diffie-Hellman Inversion* (BDHI) assumption [2]. The proposed SKBE scheme is constructed from a PECK scheme in [21] by using a hybrid encryption technique.

Overview. The rest of this paper is organized as follows. In section 2 we review a bilinear map and some complexity assumptions to be used to argue security for the construction of SKBE in this paper. We formalize

SKBE and define its security model in Section 3 and propose an efficient and provably secure construction of SKBE in Section 4. Finally, we conclude in Section 5.

2 Preliminaries

2.1 Bilinear Map

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of order p for some large prime p. A bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ between these two groups satisfies the following properties:

- Bilinear: We say that a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is bilinear if $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1$ and all $a, b \in \mathbb{Z}$.
- Non-degenerate: The map does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_1$ to the identity in $\mathbb{G}_2$. Observe that since $\mathbb{G}_1, \mathbb{G}_2$ are groups of prime order this implies that if P is a generator of $\mathbb{G}_1$ then $\hat{e}(P, P)$ is a generator of $\mathbb{G}_2$.
- Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in \mathbb{G}_1$.

We can make the bilinear map using Weil pairing or Tate pairing [5, 6, 15, 18]. In the pairings, the group $\mathbb{G}_1$ is a subgroup of the additive group of points of an elliptic curve. The group $\mathbb{G}_2$ is a subgroup of the multiplicative group of a finite field. Therefore, throughout the paper we view $\mathbb{G}_1$ as an additive group and $\mathbb{G}_2$ as a multiplicative group.

2.2 Complexity Assumptions

Let $\mathbb{G}_1$ be a bilinear group of prime order p and P be its generator. Here, we review the *Bilinear Diffie-Hellman Inversion* (BDHI) assumption [2] and the *Bilinear Collusion Attack* (BCA) assumption [8] on which the security proof of our proposed scheme is based.

Bilinear Diffie-Hellman Inversion Assumption [2] The q-BDHI problem is defined as follows: given the $(q+1)$-tuple $(P, xP, x^2P, \cdots, x^qP) \in (\mathbb{G}_1^*)^{q+1}$ as input, compute $\hat{e}(P, P)^{1/x} \in \mathbb{G}_2^*$. An algorithm $\mathcal{A}$ has advantage ϵ in solving q-BDHI in $\mathbb{G}_1$ if

$$\Pr[\mathcal{A}(P, xP, \cdots, x^qP) = \hat{e}(P, P)^{1/x}] \geq \epsilon$$

where the probability is over the random choice of generator P in $\mathbb{G}_1^*$, the random choice of x in $\mathbb{Z}_p^*$, and the random bits of $\mathcal{A}$.

Definition 1 *We say that the (t, q, ϵ)-BDHI assumption holds in $\mathbb{G}_1$ if no t-time algorithm has advantage at least ϵ in solving the q-BDHI problem in $\mathbb{G}_1$.*

The q-BDHI assumption is used to analyze the security of the proposed SKBE system in next section.

Bilinear Collusion Attack Assumption [8] The q-BCA problem is defined as follows: given the tuple $(P, xP, u_1, \cdots, u_q, \frac{1}{x+u_1}P, \cdots, \frac{1}{x+u_q}P)$ as input, compute $\hat{e}(P, P)^{1/x} \in \mathbb{G}_2^*$. An algorithm $\mathcal{A}$ has advantage ϵ in solving q-BCA in $\mathbb{G}_1$ if

$$\Pr[\mathcal{A}(P, xP, u_1, \cdots, u_q, \frac{1}{x+u_1}P, \cdots, \frac{1}{x+u_q}P) = \hat{e}(P, P)^{1/x}] \geq \epsilon$$

where the probability is over the random choice of generator P in $\mathbb{G}_1^*$, the random choices of x, $u_1, \cdots, u_q$ in $\mathbb{Z}_p^*$ and the random bits of $\mathcal{A}$, where $u_1, \cdots, u_q$ are different from each other.

Definition 2 *We say that the (t, q, ϵ)-BCA assumption holds in $\mathbb{G}_1$ if no t-time algorithm has advantage at least ϵ in solving the q-BCA problem in $\mathbb{G}_1$.*

It is known that the q-BCA assumption is equivalent to the $(q+1)$-BDHI assumption [8].

3 Searchable Keyword-Based Encryption

In this section we define a searchable keyword-based encryption (SKBE). For SKBE to have applications described in Section 1.1, we design the mechanism for a public key setting rather than a symmetric key setting. It also supports a conjunctive keyword search. That is, SKBE is a public key encryption with conjunctive keyword search and delegation decryption key ability.

We assume a document D as (M, H), where the message M is the content of D and the keywords H is associated with M. We assume that H consists of m keyword fields. For example, if documents were emails, we could define four keyword fields, such as "From", "To", "Data" and "Subject". We denote the keywords as $H = (W_1, \cdots, W_m)$, where W_i is the keyword of the document D in the i-th keyword field. For simplicity, we employ the same assumptions as in [17, 21]:

1. H does not have any two keyword fields filled in by the same keyword.
   (i.e. $H = (W_1, \cdots, W_m)$, where all W_i's are distinct from each other for $1 \leq i \leq m$.)
2. Every document has no empty keyword field of its m keyword fields.

The first requirement is satisfied by prefixing the name of keyword field to the keyword (i.e., "Data:Test" in "Data" field, "Subject:Test" in "Subject" field). The second requirement is by assigning the keyword "THE NAME OF A FIELD:NULL" to the field that does not have a valid keyword.

To search for keywords conjunctively, a query Q for requesting a trapdoor has the following form: $Q = (I_1, I_2, \cdots, I_t, \Omega_1, \Omega_2, \cdots, \Omega_t)$, where I_i is the identifier, between 1 and m, of a keyword in the i-th keyword field and Ω_i's are the keywords to search for. The corresponding *search trapdoor* T_Q^S searches for the document D whose H becomes $\{(W_{I_1} = \Omega_1) \wedge (W_{I_2} = \Omega_2) \wedge \cdots \wedge (W_{I_t} = \Omega_t)\}$. Given the corresponding *decrypt trapdoor* T_Q^D, we can retrieve the message M by decrypting the ciphertext of D whose H becomes $\{(W_{I_1} = \Omega_1) \wedge (W_{I_2} = \Omega_2) \wedge \cdots \wedge (W_{I_t} = \Omega_t)\}$. For simple description, we denote $\{(W_{I_1} = \Omega_1) \wedge (W_{I_2} = \Omega_2) \wedge \cdots \wedge (W_{I_t} = \Omega_t)\}$ as $Q \subseteq H$, and we say that H *matches* Q or that H *and* Q *match.*

We call the following defined system the *searchable keyword-based encryption* (SKBE).

Definition 3 *A searchable keyword-based encryption consists of the following polynomial time randomized algorithms:*

1. $\mathsf{KeyGen}(1^k)$*: Takes a security parameter,* 1^k*, and generates a public/private key pair* A_{pub}/A_{priv}*.*
2. $\mathsf{Encrypt}(A_{pub}, D)$*: For a public key* A_{pub} *and a document* $D = (M, H)$ *where* $M \in \mathcal{M}$*, returns a ciphertext* $C \in \mathcal{C}$ *(searchable keyword-based encryption of* M *based on its keyword fields* H*), where* $\mathcal{M}$ *is the message space and* $\mathcal{C}$ *is the ciphertext space.*
3. $\mathsf{STrapdoor}(A_{priv}, Q)$*: Given a private key* A_{priv} *and a query* Q*, produces a search trapdoor* T_Q^S*.*
4. $\mathsf{DTrapdoor}(A_{priv}, Q)$*: Given a private key* A_{priv} *and a query* Q*, produces a decrypt trapdoor* T_Q^D*.*
5. $\mathsf{Test}(A_{pub}, C, T_Q^S)$*: Given a public key* A_{pub}*, a ciphertext* $C = \mathsf{Encrypt}(A_{pub}, D) \in \mathcal{C}$*, and a* search trapdoor $T_Q^S = \mathsf{STrapdoor}(A_{priv}, Q)$*, outputs 'yes' if* $\{(W_{I_1} = \Omega_1) \wedge (W_{I_2} = \Omega_2) \wedge \cdots \wedge (W_{I_t} = \Omega_t)\}$ *and 'no' otherwise.*
6. $\mathsf{Decrypt}(A_{pub}, C, T_Q^D)$*: Given a public key* A_{pub}*, a ciphertext* $C = \mathsf{Encrypt}(A_{pub}, D) \in \mathcal{C}$*, and a* decrypt trapdoor $T_Q^D = \mathsf{DTrapdoor}(A_{priv}, Q)$*, outputs the message* $M \in \mathcal{M}$ *of the ciphertext if* $\{(W_{I_1} = \Omega_1) \wedge (W_{I_2} = \Omega_2) \wedge \cdots \wedge (W_{I_t} = \Omega_t)\}$ *and* $\perp$ *(i.e., invalid) otherwise.*

A message M is encrypted with a public key A_{pub} and the ciphertext is dependent on $H = (W_1, W_2, \cdots, W_m)$. The ciphertext value by itself does not reveal any information about the document. Given a certain *search trapdoor*, we can test whether a ciphertext contains certain keywords but get no other information. Given a certain *decrypt trapdoor*, we can decrypt only the ciphertexts containing certain keywords but not other

ciphertexts. These trapdoors can be generated only with A_{priv} and it is not known for which keywords they are intended.

We describe SKBE in operation using a secure audit log [26] as a sample application. An escrow agent runs the KeyGen algorithm to generate her public/private key pair A_{pub}/A_{priv}. An audit log server generating log entries uses the public key A_{pub} as input to the Encrypt algorithm to encrypt audit logs. At some point, when an investigator requests a search/decrypt trapdoor pair T_Q^S/T_Q^D for keywords Q, if the agent deems it appropriate, he produces T_Q^S/T_Q^D by using the STrapdoor/DTrapdoor algorithm and grants them to the investigator. The investigator gives T_Q^S to a database server storing encrypted entries and requests matching entries (sometimes, T_Q^S is given to the database server directly from the escrow agent with information identifying that this is for a certain investigator). The server uses this given trapdoor T_Q^S as input to the Test algorithm to determine which entry's H matches Q. Then he gives the results to the investigator. The investigator uses the results and T_Q^D as input to the Decrypt algorithm to get their decryptions.

Observe that for decrypting any ciphertext, even a user owning A_{priv} needs an appropriate decrypt trapdoor. Alice in email gateway or decryption key delegation described in Section 1.1 should be allowed to decrypt all mails sent to her. That is, she has to be able to make an appropriate decrypt trapdoor just given a ciphertext. This problem could be solved easily by filling a public value W_{pub} into the last keyword field of every H (i.e., if H consists of m keywords, H becomes $(W_1, W_2, \cdots, W_{m-1}, W_{pub})$). Now, Alice can make a decrypt trapdoor T_Q^D of $Q = (I_1, W_{pub})$, where I_1 is m denoting the m-th keyword field. Alice can make the T_Q^D with A_{priv}, and then always decrypt any ciphertext using T_Q^D. If necessary, we can use the public value W_{pub} as the one of the domain parameters of SKBE.

Security definition. We define security for a SKBE system in the sense of chosen ciphertext security [4, 20, 22]. We need to ensure that an Encrypt(A_{pub}, D) must not reveal any information about the document $D = (M, H)$ unless a suitable trapdoor (i.e. T_Q^S or T_Q^D, where $Q \subseteq H$) is available. When an active adversary attacks an encryption in a SKBE system, the adversary might already possess some search/decrypt trapdoors of $Q_1, \cdots, Q_n$ of her choice. The system should remain secure under such an attack. Consider an investigator in an audit log system who received some search/decrypt trapdoor pairs ($T_{Q_i}^S$, $T_{Q_i}^D$)'s for $1 \leq i \leq n$ from the agent. She is allowed to search and decrypt the encrypted log entries matching Q_i's, but it is forbidden to search or decrypt other entries. Hence the definition of chosen ciphertext security must allow the adversary to obtain the search/decrypt trapdoor for any Q_i of her choice (the restriction is mentioned later). We refer to such queries as search/decrypt trapdoor queries. In addition, similar to the IND-CCA in a public key encryption system, we allow the adversary to get access to an oracle for the decryption function. The reason is that although an investigator cannot obtain the trapdoors which are not allowed to her, she may use decryption functions of other investigators only for the period of time (e.g., for lunchtime) preceding her being given the challenge ciphertext (IND-CCA1) or even on ciphertexts chosen after obtaining the challenge ciphertext (IND-CCA2). Of course, the restriction is that the adversary may not ask for the decryption of the challenge ciphertext itself. Recall that the decryption algorithm in SKBE requires a decrypt trapdoor as an input parameter in addition to a ciphertext. Such an additional input of decryption function is required for a proxy delegated a decryption capability (i.e., decrypt trapdoor) to decrypt only ciphertexts falling within the category intended for by the decrypt trapdoor. Thus, when an adversary asks for a decryption query, she has to issue a ciphertext C_i and the desired category Q_i. Even under such attack the adversary should not be able to distinguish an encryption of a document $D_0 = (M_0, H_0)$ from an encryption of a document $D_1 = (M_1, H_1)$ for which she did not obtain suitable trapdoors. Formally, we define the chosen ciphertext security against an active adversary $\mathcal{A}$ using the following game between a challenger and the adversary.

Setup: The challenger runs the KeyGen(1^k) algorithm to generate A_{pub} and A_{prive}. It gives A_{pub} to the adversary.

Phase 1: The adversary issues queries $q_1, \cdots, q_l$ where query q_i is one of:

- Search trapdoor query $\langle Q_i \rangle$: The challenger responds by running algorithm STrapdoor to generate the search trapdoor $T_{Q_i}^S$ corresponding to the query $\langle Q_i \rangle$. It sends $T_{Q_i}^S$ to the adversary.
- Decrypt trapdoor query $\langle Q_i \rangle$: The challenger responds by running algorithm DTrapdoor to generate the decrypt trapdoor $T_{Q_i}^D$ corresponding to the query $\langle Q_i \rangle$. It sends $T_{Q_i}^D$ to the adversary.

- Decryption query $\langle Q_i, C_i \rangle$: The challenger responds by running algorithm DTrapdoor to generate the decrypt trapdoor $T^D_{Q_i}$ corresponding to Q_i. It then runs algorithm Decrypt to decrypt the ciphertext C_i using the decrypt trapdoor $T^D_{Q_i}$. It sends the resulting message M_i or $\perp$ to the adversary.

These queries may be asked adaptively, that is, each query q_i may depend on the replies to $q_1, \cdots, q_{i-1}$.

Challenge: Once the adversary decides that Phase 1 is over it outputs two different documents $D_0 = (M_0, H_0)$, $D_1 = (M_1, H_1)$ on which it wishes to be challenged. (The messages M_0 and M_1 must have equal length and could be the same value.) The restrictions are as follows: First, none of the search trapdoors asked previously in Phase 1 is distinguishing for D_0 and D_1.[1] Second, none of Q_i for the decrypt trapdoors such that Q_i matches H_0 or H_1 were issued previously in Phase 1. Lastly, neither decryption queries for $\langle Q_i \subseteq H_0, C_0 \rangle$ nor $\langle Q_i \subseteq H_1, C_1 \rangle$ were issued. The challenger picks a random $b \in \{0, 1\}$ and gives the adversary $C = \mathsf{Encrypt}(A_{pub}, D_b)$. We refer to C as the challenge ciphertext.

Phase 2: The adversary issues more queries $q_{l+1}, \cdots, q_n$ where query q_i is one of:

- Search trapdoor query $\langle Q_i \rangle$: Q_i such that the corresponding search trapdoor $T^S_{Q_i}$ distinguishing for D_0 and D_1 must not be allowed, elsewhere challenger responds as in Phase 1.
- Decrypt trapdoor query $\langle Q_i \rangle$ where $Q_i \nsubseteq H_0$ or H_1: Challenger responds as in Phase 1.
- Decryption query $\langle Q_i, C_i \rangle$ where $C_i \neq C$: Challenger responds as in Phase 1.

These queries may be asked adaptively as in Phase 1.

Guess: Finally, the adversary outputs a guess $b' \in \{0, 1\}$. The adversary wins the game if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as an IND-SKBE-CCA adversary. We define the adversary $\mathcal{A}$'s advantage in attacking a SKBE system $\mathcal{E}$ as the following function of the security parameter k: $Adv_{\mathcal{E},\mathcal{A}}(1^k) = |\Pr[b = b'] - 1/2|$. Throughout the paper we use the term *negligible function* to refer to a function $f : \mathbb{R} \to [0, 1]$ where $f(k) < \frac{1}{g(k)}$ for any polynomial g and sufficiently large k.

Definition 4 *We say that a SKBE system $\mathcal{E}$ is semantically secure against an adaptive chosen ciphertext attack if for any polynomial time IND-SKBE-CCA adversary $\mathcal{A}$ the function $Adv_{\mathcal{E},\mathcal{A}}(1^k)$ is negligible. As shorthand, we say that $\mathcal{E}$ is IND-SKBE-CCA secure.*

We do not consider an adversary who (existentially) forges a trapdoor, because if there is an adversary that can generate a forged valid trapdoor with non-negligible probability, then he can also win the IND-SKBE-CCA game with a non-negligible advantage. In other words, IND-SKBE-CCA secure SKBE means that there is no such adversary.

4 Construction

We give an efficient construction for searchable keyword-based encryption.[2] Our scheme is constructed from Park *et al.*'s PECK scheme [21], the proposed scheme 2 represented in Section 5 of their paper, using a hybrid encryption technique used in [14]. They use two groups $\mathbb{G}_1$, $\mathbb{G}_2$ of prime order p and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ between them. Let P_1, P_2 be two different generators of $\mathbb{G}_1$. In their scheme, if a ciphertext and its suitable search trapdoor are given then the tester can compute $\hat{e}(P_1, P_1)^{r_0}$ and output 'yes', where r_0 is a random element in Z^*_p. Our construction is stared from this point. If the session key to encrypt a message is hidden behind $\hat{e}(P_1, P_2)^{r_0}$, we can make a decrypt trapdoor easily by replacing P_1 with P_2 in a search trapdoor in [21]. Note that it is hard to compute $\hat{e}(P_1, P_2)^{r_0}$ from $\hat{e}(P_1, P_1)^{r_0}$, so decrypt trapdoor and search trapdoor can be separated securely.

[1] A distinguishing search trapdoor for D_0 and D_1 implies that the test results of the trapdoor with D_0 and D_1 are different each other (i.e., $\mathsf{Test}(A_{pub}, C_0, T^S_Q) \neq \mathsf{Test}(A_{pub}, C_1, T^S_Q)$). Observe that $\mathcal{A}$ succeeds trivially if a distinguishing search trapdoor for D_0 and D_1 is given to her. The security games of previously proposed conjunctive keyword search schemes [17, 21, 7, 23] also do not allowed such distinguishing search trapdoor queries.

[2] Chow[9] introduced a method attacking the scheme represented in this paper's previous version. The attack is not accomplished as their description, but it is true that the previous scheme has a problem when the decryption query in their attack is issued because SKBE allows a decryption query for $C_i \neq C$ where C is a challenge ciphertext. We notice that our present work is modified to solve that problem, and so is secure against the attack by Chow.

4.1 Description

For our scheme, we need a secure symmetric key encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ in FG (Find-Guess) sense [14] and three hash functions $H_1 : \{0,1\}^* \to \{0,1\}^{\log_2 p}, H_2 : \{0,1\}^* \to \{0,1\}^{\log_2 p}, H_3 : \{0,1\}^* \times \mathcal{K} \to \{0,1\}^{\log_2 p}$, where $\mathcal{K}$ is the key space for a secret key sk of $(\mathcal{G}, \mathcal{E}, \mathcal{D})$. H_1 and H_2 are different from each other.

- $\mathsf{KeyGen}(1^k)$: The input security parameter 1^k determines the size, p, of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$. The algorithm chooses random numbers $s_1, s_2, \cdots, s_m, s_{m+1}, s_{m+2} \in \mathbb{Z}_p^*$ and two different generators P_1, P_2 of $\mathbb{G}_1$. It outputs $A_{pub} = [P_1, P_2, Y_1 = s_1P_1, Y_2 = s_2P_1, \cdots, Y_m = s_mP_1, Y_{m+1} = s_{m+1}P_1, Y_{m+2} = s_{m+2}P_1, g = \hat{e}(P_1, P_1), h = \hat{e}(P_1, P_2)]$ and $A_{priv} = [s_1, s_2, \cdots, s_m, s_{m+1}, s_{m+2}]$.
- $\mathsf{Encrypt}(A_{pub}, D = (M, H))$: Produces a session key $sk \in \mathcal{K}$ by running the algorithm $\mathcal{G}$. The algorithm selects random numbers $r_1, r_2, \cdots, r_m \in \mathbb{Z}_p^*$ and computes $B_i = r_iY_{m+1}$ for $1 \le i \le m$. Set $r_0 = H_3(M||B_1|| \cdots ||B_m, sk)$, where each B_i is treated as a bit string and $||$ is a concatenation. It encrypts message by $E = \mathcal{E}_{sk}(M)$ and computes the following values: $A_i = r_0(Y_i + H_1(W_i)P_1) + r_iP_1$ for $1 \le i \le m$, $K = r_0Y_{m+2}$, $S = H_2(g^{r_0})$, and $R = H_2(h^{r_0}) \oplus sk$. Output the ciphertext $C = [E, A_1, A_2, \cdots, A_m, B_1, B_2, \cdots, B_m, K, S, R] \in \mathcal{C}$.
- $\mathsf{STrapdoor}(A_{priv}, Q)$: For the input $Q = (I_1, I_2, \cdots, I_t, \Omega_1, \Omega_2, \cdots, \Omega_t)$, selects a random $u \in \mathbb{Z}_p^*$ and makes the *search trapdoor* $T_Q^S = [T_1^S, T_2^S, T_3^S, I_1, \cdots, I_t]$ where

$$\begin{aligned} T_1^S &= \frac{1}{s_{I_1} + \cdots + s_{I_t} + H_1(\Omega_1) + \cdots + H_1(\Omega_t) + s_{m+2}u} P_1, \\ T_2^S &= \frac{1}{s_{m+1}} T_1^S, \\ T_3^S &= u. \end{aligned}$$

- $\mathsf{DTrapdoor}(A_{priv}, Q)$: For the input $Q = (I_1, I_2, \cdots, I_t, \Omega_1, \Omega_2, \cdots, \Omega_t)$, selects a random $v \in \mathbb{Z}_p^*$ and makes the *decrypt trapdoor* $T_Q^D = [T_1^D, T_2^D, T_3^D, I_1, \cdots, I_t]$ where

$$\begin{aligned} T_1^D &= \frac{1}{s_{I_1} + \cdots + s_{I_t} + H_1(\Omega_1) + \cdots + H_1(\Omega_t) + s_{m+2}v} P_2, \\ T_2^D &= \frac{1}{s_{m+1}} T_1^D, \\ T_3^D &= v. \end{aligned}$$

- $\mathsf{Test}(A_{pub}, C, T_Q^S)$: Checks the equality,

$$H_2\left(\frac{\hat{e}(A_{I_1} + \cdots + A_{I_t} + T_3^S K, T_1^S)}{\hat{e}(B_{I_1} + \cdots + B_{I_t}, T_2^S)}\right) = S. \tag{1}$$

  If so, outputs 'yes'; otherwise, outputs 'no'.

- $\mathsf{Decrypt}(A_{pub}, C, T_Q^D)$: Computes $\widetilde{h^{r_0}} = \frac{\hat{e}(A_{I_1} + \cdots + A_{I_t} + T_3^D K,\ T_1^D)}{\hat{e}(B_{I_1} + \cdots + B_{I_t},\ T_2^D)}$, $\widetilde{sk} = H_2(\widetilde{h^{r_0}}) \oplus R$, $\widetilde{M} = \mathcal{D}_{\widetilde{sk}}(E)$ and $\widetilde{r_0} = H_3(\widetilde{M}||B_1|| \cdots ||B_m, \widetilde{sk})$. Checks the equality,

$$h^{\widetilde{r_0}} = \widetilde{h^{r_0}}. \tag{2}$$

  If so, outputs $\widetilde{M}$; otherwise, outputs $\perp$.

The equality of $\mathsf{Test}(1)$ holds if $(W_{I_i} = \Omega_i)$ for $1 \le i \le t$. We can check as follows:

$$\begin{aligned} & H_2\left(\frac{\hat{e}(A_{I_1} + \cdots + A_{I_t} + T_3^S K, T_1^S)}{\hat{e}(B_{I_1} + \cdots + B_{I_t}, T_2^S)}\right) \\ &= H_2\left(\frac{\hat{e}(A_{I_1} + \cdots + A_{I_t} + T_3^S K, T_1^S)}{\hat{e}(r_{I_1}P_1 + \cdots + r_{I_t}P_1, T_1^S)}\right) \\ &= H_2(\hat{e}(r_0(Y_{I_1} + H_1(W_{I_1})P_1) + \cdots + r_0(Y_{I_t} + H_1(W_{I_t})P_1) + T_3^S K, T_1^S) \\ &= H_2(\hat{e}(r_0P_1, P_1)) = S. \end{aligned}$$

The equality of Decrypt(2) holds if $\{(W_{I_i} = \Omega_i)$ for $1 \leq i \leq t$. We can check as follows:

$$\begin{aligned}
\widetilde{h^{r_0}} &= \frac{\hat{e}(A_{I_1} + \cdots + A_{I_t} + T_3^D K,\ T_1^D)}{\hat{e}(B_{I_1} + \cdots + B_{I_t},\ T_2^D)} \\
&= \frac{\hat{e}(A_{I_1} + \cdots + A_{I_t} + T_3^D K,\ T_1^D)}{\hat{e}(r_{I_1} P_1 + \cdots + r_{I_t} P_1,\ T_1^D)} \\
&= \hat{e}(r_0(Y_{I_1} + H_1(W_{I_1})P_1) + \cdots + r_0(Y_{I_t} + H_1(W_{I_t})P_1) + T_3^D K, T_1^D) \\
&= \hat{e}(r_0 P_1, P_2) = h^{r_0}.
\end{aligned}$$

, $\widetilde{sk} = H_2(\widetilde{h^{r_0}}) \oplus R = H_2(h^{r_0}) \oplus R = sk$, and then $\widetilde{M} = \mathcal{D}_{\widetilde{sk}}(E) = \mathcal{D}_{sk}(\mathcal{E}_{sk}(M)) = M$ and $\widetilde{r_0} = H_3(\widetilde{M}||B_1|| \cdots ||B_m, \widetilde{sk}) = H_3(M||B_1|| \cdots ||B_m, sk) = r_0$. Therefore, $h^{\widetilde{r_0}} = h^{r_0} = \widetilde{h^{r_0}}$.

4.2 Security analysis

We prove that this scheme is IND-SKBE-CCA' secure under the q-BDHI assumption in a random oracle model. IND-SKBE-CCA' differs from IND-SKBE-CCA in that there is no common keyword sharing between D_0 and D_1 (i.e., $\forall j, H_{0,j} \neq H_{1,j}$) in the challenging stage, or an adversary cannot ask a trapdoor query for the sharing keywords. We conjecture that if a SKBE scheme is IND-SKBE-CCA secure, the scheme is IND-SKBE-CCA' secure also because in common sense it is more difficult to distinguish something from the one similar to it than the one with a distinct difference; it remains as open problem to prove this conjecture in a formal language.

Theorem 1 *Suppose the $(q_T + 1)$-BDHI assumption holds in $\mathbb{G}_1$. Then the above scheme is IND-SKBE-CCA' secure.*

Proof. Suppose $\mathcal{A}$ has advantage ϵ in attacking the proposed scheme under the IND-SKBE-CCA' game. Suppose $\mathcal{A}$ makes STrapdoor or DTrapdoor queries at most q_T times, H_1 queries at most q_{H_1} times, and H_3 queries at most q_{H_3} times. We build an adversary $\mathcal{B}$ that solves the $(q_T + 1)$-BDHI problem in $\mathbb{G}_1$ with probability at least $\epsilon' = \epsilon/(e(q_{H_1})^m q_{H_3})$, where e is the base of the natural logarithm. The running time of adversary $\mathcal{B}$ is approximately the same as $\mathcal{A}$'s.

On input $(P, xP, x^2P, \cdots, x^{q_T+1}P)$ adversary $\mathcal{B}$'s goal is to compute the value $\hat{e}(P, P)^{1/x} \in \mathbb{G}_2$ by simulating the challenger and interacts with the algorithm $\mathcal{A}$ as the following IND-SKBE-CCA' security game.

Setup: Adversary $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ selects $\delta_1, \delta_2, \cdots, \delta_{q_T} \in \mathbb{Z}_p^*$ at random and let $f(z) = \prod_{j=1}^{q_T}(z + \delta_j)$.
2. Expand the terms of f to get $f(z) = \sum_{i=0}^{q_T} c_i z^i$. Compute $U = f(x)P = \sum_{i=0}^{q_T} c_i x^i P$ and $V = xU = \sum_{i=1}^{q_T+1} c_{i-1} x^i P$.
3. $\mathcal{B}$ computes $\frac{1}{x+\delta_i}U = (f(x)/(x + \delta_i))P = \sum_{j=0}^{q_T-1} d_j x^j P$ for $1 \leq i \leq q_T$, and stores the pairs $(\delta_i, \frac{1}{x+\delta_i}U)$'s.
4. $\mathcal{B}$ selects random numbers $\alpha_0, \alpha_1, \alpha_2, \cdots, \alpha_m, \beta_1, \beta_2, \cdots, \beta_m \in \mathbb{Z}_p$ and computes $Y_i = \alpha_i V - \beta_i U$ for $1 \leq i \leq m$, $Y_{m+1} = \alpha_{m+1} U$, $Y_{m+2} = \alpha_{m+2} V$, $g = \hat{e}(U, U)$ and $h = \hat{e}(U, \alpha_0 U)$. $\mathcal{B}$ gives $\mathcal{A}$ the public key $A_{pub} = [U, \alpha_0 U, Y_1, \cdots, Y_m, Y_{m+1}, Y_{m+2}, g, h]$. The corresponding private key A_{priv} becomes $[s_1 = \alpha_1 x - \beta_1, \cdots, s_m = \alpha_m x - \beta_m, s_{m+1} = \alpha_{m+1}, s_{m+2} = \alpha_{m+2} x]$ where the $s_1, \cdots, s_m$ and s_{m+2} are unknown to $\mathcal{B}$.

H_1 **queries:** At any time adversary $\mathcal{A}$ can query the random oracle H_1. To respond to H_1 queries, $\mathcal{B}$ maintains a list of tuples $< W_i, h_i, c_i >$ called the H_1-list. The list is initially empty. When $\mathcal{A}$ queries the random oracle H_1 at a point $W_i \in \{0,1\}^*$, $\mathcal{B}$ responds as follows:

1. If the query W_i already appears in the H_1-list in a tuple $< W_i, h_i, c_i >$, then $\mathcal{B}$ responds with $H_1(W_i) = h_i$.
2. Otherwise, $\mathcal{B}$ generates a random $c_i \in \{1, \cdots, q_{H_1}\}$ so that $\Pr[c_i \leq m] = m/q_{H_1}$.

3. If $c_i > m$, $\mathcal{B}$ selects a random $h_i \in \{0,1\}^{\log_2 p}$. Otherwise, $\mathcal{B}$ makes $h_i \leftarrow \beta_{c_i}$.
4. $\mathcal{B}$ adds the tuple $< W_i, h_i, c_i >$ to the H_1-list and responds with $H_1(W_i) = h_i$.

H_2 **queries:** Similarly, at any time adversary $\mathcal{A}$ can query the random oracle H_2. To respond to H_2 queries, $\mathcal{B}$ maintains a list of tuples $< g_i, \gamma_i >$ called the H_2-list. The list is initially empty. When $\mathcal{A}$ queries the random oracle H_2 at a point $g_i \in \{0,1\}^*$, $\mathcal{B}$ responds as follows:

1. If the query g_i already appears in the H_2-list in a tuple $< g_i, \gamma_i >$, then $\mathcal{B}$ responds with $H_2(g_i) = \gamma_i$.
2. Otherwise, $\mathcal{B}$ selects a random $\gamma_i \in \{0,1\}^{\log_2 p}$ and adds the tuple $< g_i, \gamma_i >$ to the H_2-list. $\mathcal{B}$ responds with $H_2(g_i) = \gamma_i$.

H_3 **queries:** Similarly, at any time adversary $\mathcal{A}$ can query the random oracle H_3. To respond to H_3 queries, $\mathcal{B}$ maintains a list of tuples $< M_i||B_{i,1}||B_{i,2}||\cdots||B_{i,m}, sk_i, r_{0i}, X_i, Y_i >$ called the H_3-list. The list is initially empty. When $\mathcal{A}$ queries the random oracle H_3 at a point $M_i||B_{i,1}||B_{i,2}||\cdots||B_{i,m} \in \{0,1\}^*$ and $sk_i \in \mathcal{K}$, $\mathcal{B}$ responds as follows:

1. If the query $M_i||B_{i,1}||B_{i,2}||\cdots||B_{i,m}, sk_i$ already appears in the H_3-list in a tuple $< M_i||B_{i,1}||\cdots||B_{i,m}, sk_i, r_{0i}, X_i, Y_i >$, then $\mathcal{B}$ responds with $H_3(M_i||B_{i,1}||\cdots||B_{i,m}, sk_i) = r_{0i}$.
2. Otherwise, $\mathcal{B}$ selects a random $r_{0i} \in \mathbb{Z}_p^*$ and computes $X_i = H_2(g^{r_{0i}})$ and $Y_i = H_2(h^{r_{0i}}) \oplus sk_i$. $\mathcal{B}$ adds the tuple $< M_i||B_{i,1}||\cdots||B_{i,m}, sk_i, r_{0i}, X_i, Y_i >$ to the H_3-list and gives r_{0i} to $\mathcal{A}$.

Phase 1: $\mathcal{A}$ issues queries q_i where query q_i is one of:

STrapdoor queries: When $\mathcal{A}$ issues a query for the *search trapdoor* corresponding to the query $Q_i = (I_{i,1}, I_{i,2}, \ldots, I_{i,t_i}, \Omega_{i,1}, \Omega_{i,2}, \ldots, \Omega_{i,t_i})$, $\mathcal{B}$ responds as follows:

1. $\mathcal{B}$ executes the above simulation for responding to H_1 queries to obtain $h_{i,j}$'s such that $h_{i,j} = H_1(\Omega_{i,j})$. Let $< \Omega_{i,j}, h_{i,j}, c_{i,j} >$ be the corresponding tuple on the H_1-list. If $\forall j, c_{i,j} = I_{i,j}$, then $\mathcal{B}$ fails.
2. Otherwise, $\mathcal{B}$ defines $J_i = s_{I_{i,1}} + \cdots + s_{I_{i,t_i}} + h_{i,1} + \cdots + h_{i,t_i} = \Gamma_i x + \Delta_i$ and computes J_i. Observe that since J_i is equivalent to $(\alpha_{I_{i,1}} + \cdots + \alpha_{I_{i,t_i}})x - (\beta_{I_{i,1}} + \cdots + \beta_{I_{i,t_i}}) + (h_{i,1} + \cdots + h_{i,t_i})$, $\mathcal{B}$ can obtain the values Γ_i, Δ_i from $\alpha_{I_{i,j}}$'s, $\beta_{I_{i,j}}$'s, $h_{i,j}$'s which are known to $\mathcal{B}$.
3. $\mathcal{B}$ picks i-th pair $(\delta_i, \frac{1}{x+\delta_i}U)$ in a storage. $\mathcal{B}$ finds u_i, v_i satisfying the equality of $\frac{1}{x+\delta_i}U = (v_i/(\Gamma_i x + \Delta_i + \alpha_{m+2} x u_i))U$. The u_i and v_i become $(\Delta_i/\delta_i - \Gamma_i)/\alpha_{m+2}$ and Δ_i/δ_i, respectively. $\mathcal{B}$ computes $F_i = \frac{1/v_i}{x+\delta_i}U$. Observe that the value $F_i = \frac{1}{v_i(x+\delta_i)}U = \frac{1}{\Gamma_i x + \Delta_i + \alpha_{m+2} x u_i}U$ is same as $(1/(s_{i,1} + \cdots + s_{i,t_i} + h_{i,1} + \cdots + h_{i,t_i} + s_{m+2}u_i)U$. Thus, $(F_i, \frac{1}{\alpha_{m+1}}F_i, u_i)$ is the correct *search trapdoor* $T_{Q_i}^S$ corresponding to the query Q_i. $\mathcal{B}$ responds to $\mathcal{A}$ with $[F_i, \frac{1}{\alpha_{m+1}}F_i, u_i, I_{i,1}, \cdots, I_{i,t_i}]$.

DTrapdoor queries: When $\mathcal{A}$ issues a query for the *decrypt trapdoor* corresponding to the query $Q_i = (I_{i,1}, I_{i,2}, \ldots, I_{i,t_i}, \Omega_{i,1}, \Omega_{i,2}, \ldots, \Omega_{i,t_i})$, $\mathcal{B}$ responds as follows:

1. $\mathcal{B}$ executes the above simulation for responding to H_1 queries to obtain $h_{i,j}$'s such that $h_{i,j} = H_1(\Omega_{i,j})$. Let $< \Omega_{i,j}, h_{i,j}, c_{i,j} >$ be the corresponding tuple on the H_1-list. If $\forall j, c_{i,j} = I_{i,j}$, then $\mathcal{B}$ fails.
2. Otherwise, $\mathcal{B}$ defines $J_i = s_{I_{i,1}} + \cdots + s_{I_{i,t_i}} + h_{i,1} + \cdots + h_{i,t_i} = \Gamma_i x + \Delta_i$ and computes J_i. Observe that since J_i is equivalent to $(\alpha_{I_{i,1}} + \cdots + \alpha_{I_{i,t_i}})x - (\beta_{I_{i,1}} + \cdots + \beta_{I_{i,t_i}}) + (h_{i,1} + \cdots + h_{i,t_i})$, $\mathcal{B}$ can obtain the values Γ_i, Δ_i from $\alpha_{I_{i,j}}$'s, $\beta_{I_{i,j}}$'s, $h_{i,j}$'s which are known to $\mathcal{B}$.
3. $\mathcal{B}$ picks i-th pair $(\delta_i, \frac{1}{x+\delta_i}U)$ in a storage. $\mathcal{B}$ finds u_i, v_i satisfying the equality of $\frac{1}{x+\delta_i}U = (v_i/(\Gamma_i x + \Delta_i + \alpha_{m+2} x u_i))U$. The u_i and v_i become $(\Delta_i/\delta_i - \Gamma_i)/\alpha_{m+2}$ and Δ_i/δ_i, respectively. $\mathcal{B}$ computes $G_i = \frac{\alpha_0/v_i}{x+\delta_i}U$. Observe that the value $G_i = \frac{\alpha_0}{v_i(x+\delta_i)}U = \frac{\alpha_0}{\Gamma_i x + \Delta_i + \alpha_{m+2} x u_i}U$ is same as $(\alpha_0/(s_{i,1} + \cdots + s_{i,t_i} + h_{i,1} + \cdots + h_{i,t_i} + s_{m+2}u_i)U$. Thus, $(G_i, \frac{1}{\alpha_{m+1}}G_i, u_i)$ is the correct *decrypt trapdoor* $T_{Q_i}^D$ corresponding to the query Q_i. $\mathcal{B}$ responds to $\mathcal{A}$ with $[G_i, \frac{1}{\alpha_{m+1}}G_i, u_i, I_{i,1}, \cdots, I_{i,t_i}]$.

Decrypt queries: Let $\langle Q_i, C_i \rangle$ be a decryption query issued by algorithm $\mathcal{A}$. Let $C_i = [E_i, A_{i,1}, \cdots, A_{i,m}, B_{i,1}, \cdots, B_{i,m}, K_i, S_i, R_i]$. $\mathcal{B}$ responds to this query as follows:

1. Run the above algorithm for responding to H_1 queries to obtain $h_{i,j}$'s such that $h_{i,j} = H_1(\Omega_{i,j})$. Let $< \Omega_{i,j}, h_{i,j}, c_{i,j} >$ be the corresponding tuple on the H_1-list.
2. Suppose $[h_{i,1}, \cdots, h_{i,t_i}] \neq [\beta_{I_{i,1}}, \cdots, \beta_{I_{i,t_i}}]$. In this case run the algorithm for responding to decrypt trapdoor queries to obtain the decrypt trapdoor $T^D_{Q_i}$ for Q_i. Then use the decrypt trapdoor to respond to the decryption query.
3. Suppose $[h_{i,1}, \cdots, h_{i,t_i}] = [\beta_{I_{i,1}}, \cdots, \beta_{I_{i,t_i}}]$. Recall that for valid encryption, r_{0i} must have been asked to the H_3 oracle.[3] Thus, $\mathcal{B}$ can respond to the decryption query using its H_3-list as follows; $\mathcal{B}$ searches for a tuple such that $X_i = S_i$ and $Y_i = R_i$ on H_3-list.
   (a) If there are no such tuples, $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$.
   (b) Else if $\mathcal{B}$ finds out such a tuple, $\mathcal{B}$ compares $B_{i,1}, B_{i,2}, \cdots, B_{i,m}$ in C_i with the ones in the tuple on H_3-list respectively and if there is at least different one, then $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$.
   (c) Otherwise (no different one between the $B_{i,j}$'s in C_i and the ones in the tuple), $\mathcal{B}$ checks whether $E_i = \mathcal{E}_{sk_i}(M_i)$. If so, $\mathcal{B}$ sends M_i to $\mathcal{A}$, otherwise $\perp$.

Challenge: Once algorithm $\mathcal{A}$ decides that Phase 1 is over it outputs two documents, $D_0 = (M_0, H_0)$, $D_1 = (M_1, H_1)$ that she wishes to be challenged on and sends them to $\mathcal{B}$. $\mathcal{B}$ responds as follows:

1. $\mathcal{B}$ executes the above algorithm for responding to H_1 queries to obtain $h_{i,j}$'s such that $h_{i,j} = H_1(W_{i,j})$ for $i \in \{0,1\}, 1 \leq j \leq m$. Let $< W_{i,j}, h_{i,j}, c_{i,j} >$ be the corresponding tuples on the H_1-list. Unless $\forall j, c_{i,j} = j$ for at least one of i's, then $\mathcal{B}$ fails.
2. $\mathcal{B}$ picks a random $i \in \{0,1\}$ such that $\forall j, c_{i,j} = j$.
3. $\mathcal{B}$ selects random $\rho, r_1, ..., r_m \in \mathbb{Z}^*_p$, $sk \in \mathcal{K}$ and computes $E = \mathcal{E}_{sk}(M_i)$, $A_j = \frac{\rho}{x}(\beta_j U + (\alpha_j x - \beta_j)U) + r_j U = \rho\alpha_j U + r_j U$ and $B_j = r_j Y_{m+1}$ for $1 \leq j \leq m$ and $K = \rho\alpha_{m+2}U$. $\mathcal{B}$ selects random S, $R \in \{0,1\}^{\log_2 p}$ and responds to $\mathcal{A}$ with the challenge $C = [E, A_1, \cdots, A_m, B_1, \cdots, B_m, K, S, R]$.

Phase 2: $\mathcal{B}$ responds to queries q_i of $\mathcal{A}$ in the same way it did in Phase 1. The restriction is that no STrapdoor query distinguishes D_0 from D_1 and that no DTrapdoor query decrypts D_0 or D_1 and that no Decrypt query for the challenge C is answered.

Output: Finally, $\mathcal{A}$ outputs $b' \in \{0,1\}$ indicating whether the challenge is the ciphertext of D_0 or D_1.

Guess: $\mathcal{B}$ selects random r in the H_3-list's r_{oi}'s and computes h^r and $(h^r)^{1/(\alpha_0\rho)}$. $\mathcal{B}$ outputs this value as $\hat{e}(P,P)^{1/x}$.

If algorithm $\mathcal{B}$ does not abort during the simulation then algorithm $\mathcal{A}$'s view is identical to its view in the real attack. In other words, $\mathcal{B}$ succeeds in simulating the challenger in the IND-SKBE-CCA' game with the adversary $\mathcal{A}$. Now, we calculte the probability ϵ' that $\mathcal{B}$ wins the game. Adversary $\mathcal{B}$ can fail in responding to STrapdoor/DTrapdoor trapdoor queries and in preparing the challenge. We define three following events:

$\mathcal{E}_1$: $\mathcal{B}$ does not fail as the result of any $\mathcal{A}$'s STrapdoor or DTrapdoor trapdoor queries.

$\mathcal{E}_2$: $\mathcal{B}$ does not fail preparing the challenge.

$\mathcal{E}_3$: $\mathcal{B}$ selects the right value as $\hat{e}(P,P)^{1/x}$ in the H_3-list during the Guess step.

We can assume that q_T is sufficiently large, thus, $(1 - 1/q_T)^{q_T} = 1/e$. $\Pr[\mathcal{E}_1] = \prod_{i=1}^{q_T}(1 - 1/(q_{H_1})^{t_i}) \geq \prod_{i=1}^{q_T}(1 - 1/(q_T)^{t_i}) \geq \prod_{i=1}^{q_T}(1 - 1/q_T) = 1/e$, $\Pr[\mathcal{E}_2] \geq 1/(q_{H_1})^m$ and $\Pr[\mathcal{E}_3] \geq 1/q_{H_3}$. Thus, $\mathcal{B}$ breaks $(q_T + 1)$-BDHI problem with the advantage $\epsilon' > \epsilon \times \frac{1}{e} \times (\frac{1}{q_{H_1}})^m \times \frac{1}{q_{H_3}} = \frac{\epsilon}{e(q_{H_1})^m q_{H_3}}$. □

[3] If $\mathcal{A}$ asks a randomly chosen ciphertext C_i then $\mathcal{B}$ responds with $\perp$ indicating "invalid". This is because a randomly chosen ciphertext will not have the valid form with a significantly higher probability. Such a simulation of the Decrypt oracle is possible because there exists a $\lambda(k)$-knowledge extractor where $1 - \lambda(k)$ is negligible. Note that the proposed scheme is a hybrid encryption and the decrypt algorithm checks the validity of the decryption value. In such a hybrid encryption scheme, an adversary can not create a valid ciphertext C without "knowing" its underlying plaintext M [4, 14].

5 Concluding Remarks and Open Problem

We suggested a new mechanism called searchable keyword-based encryption (SKBE) that allowed the secure delegation of search capabilities and decryption capabilities. Our mechanism is the most suitable for some applications, such as email gateways, decryption key delegation systems, and secure audit logs. We defined SKBE and provided its well-formulated security model. We constructed an efficient SKBE scheme that would be IND-SKBE-CCA' secure in the random oracle model. The security is proved under the bilinear Diffie-Hellman inversion (BDHI) assumption. We notice that our proposed SKBE scheme (in Section 4) is not statistically consistent but computationally consistent [1]. However, we believe that our scheme can be modified to meet statistical consistency in a similar way that Abdalla *et al.* used in their paper to make the Boneh *et al.*'s PEKS scheme statistically consistent. We remain to deal with this problem related to consistency and construct a SKBE scheme without random oracle as future work. Also, it will be a meaningful job to find a relation between SKBE and other cryptographic primitives such as identity-based encryption or hierarchical identity-based encryption.

References

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier and Haixia Shi, "Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions," *CRYPTO 2005,* LNCS 3621, pp. 205–222, Springer-Verlag, 2005.

[2] D. Boneh and X. Boyen, "Efficient selective-ID secure identity based encryption without random oracle," *EUROCRYPT 2004,* LNCS 3027, pp. 223–238, Springer-Verlag, 2004.

[3] D. Boneh, G. Di Crescenzo, R. Ostrovxky and G. Persiano, "Public key encryption with keyword search," *EUROCRYPT 2004,* LNCS 3027, pp. 506–522, Springer-Verlag, 2004.

[4] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, "Relations among notions of security for public-key encryption schemes," *CRYPTO 1998,* LNCS 1462, pp. 26–45, Springer-Verlag, 1998.

[5] D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing," *CRYPTO 2001,* LNCS 2139, pp. 213–229, Springer-Verlag, 2001.

[6] P. S. L. M. Barreto, H. Y. Kim, B. Lynn and M. Scott, "Efficient algorithm for pairing-based cryptosystems," *CRYPTO 2002,* LNCS 2442, pp. 354–369, Springer-Verlag, 2002.

[7] L. Ballard, S. Kamara and F. Monrose, "Achieving efficient conjunctive keyword searches over encrypted data," *ICICS 2005,* to be published.

[8] L. Chen and Z. Sheng, "Security proof of Sakai-Kasahara's identity-based encryption scheme," *Cryptology ePrint Archive,* Report 2005/226, 2005, http://eprint.iacr.org/2005/226/.

[9] Sherman S. M. Chow, "Exclusion-intersection encryption and its application to searchable encrption," *Cryptology ePrint Archive,* Report 2005/377, 2005, http://eprint.iacr.org/2005/377/.

[10] C. Cocks, "An identity based encryption scheme based on quadratic residues," *IMA International Conference on Cryptography and Coding,* Royal Agricultural College, Cirencester, UK, Dec. 2001.

[11] Y. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," *ACNS 2005,* LNCS 3531, pp. 442–455, Springer-Verlag, 2005.

[12] D. Dolev, C. Dwork and M. Naor, "Non-malleable cryptography," *SIAM J. Computing,* Vol. 30(2), pp. 391–437, Springer-Verlag, 2000.

[13] Y. Dodis and A. Yampolskiy, "A verifiable random function with short proofs and keys," *PKC 2005,* LNCS 3386, pp. 416–431, Springer-Verlag, 2005.

[14] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," *CRYPTO 1999,* LNCS 1666, pp. 537–554, Springer-Verlag, 1999.

[15] S. Galbraith, K. Harrison and D. Soldera, "Implementing the Tate pairing," *ANTS-V,* LNCS 2369, pp. 324–337, Springer-Verlag, 2002.

[16] E. Goh, "Secure indexes," *Cryptology ePrint Archive,* Report 2003/216, 2003, http://eprint.iacr.org/2003/216/.

[17] P. Golle, J. Staddon and B. Waters, "Secure conjunctive keyword search over encrypted data," *ACNS 2004,* LNCS 3089, pp. 31–45, Springer-Verlag, 2004.

[18] A. Joux, "The Weil and Tate pairings as building blocks for public key cryptosystems," *ANTS-V,* LNCS 2369, pp. 20–32, Springer-Verlag, 2002.

[19] S. Mitsunani, R. Sakai and M. Kasahara, "A new traitor tracing," *IEICE Trans. Fundamentals,* Vol. E85-A, No. 2, pp. 481–484, 2002.

[20] M. Naor and M. Yung, "Public-key cryptosystems provably secure against chosen ciphertext attacks," *Proceedings of the 22nd Annual Symposium on the Theory of Computing,* ACM, 1990.

[21] D. J. Park, K. Kim and P. J. Lee, "Public key encryption with conjunctive field keyword search," *WISA 2004,* LNCS 3325, pp. 73–86, Springer-Verlag, 2004.

[22] C. Rackoff, D. Simon, "Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack," *CRYPTO 1991,* LNCS 547, pp. 433–444, Springer-Verlag, 1991.

[23] R. Sion and B. Carbunar, "Conjunctive keyword search on encrypted data with completeness and conputational privacy," *Cryptology ePrint Archive,* Report 2005/172, 2005, http://eprint.iacr.org/2005/172/.

[24] A. Shamir, "Identity based cryptosystems and signature schemes," *CRYPTO 1984,* Springer-Verlag, 1984.

[25] D. X. Song, D. Wagner and A. Perrig, "Practical techniques for searches on encrypted data," *In proceeding of IEEE Symposium on Security and Privacy,* pp. 44–55, 2000.

[26] B. Waters, D. Balfanz, G. Durfee and D. Smetters, "Buliding an encrypted and searchable audit log," *In proceedings of NDSS 2004,* pp. 205–214, 2004.

[27] F. Zhang, R. Safavi-Naini and W. Susilo, "An efficient signature scheme from bilinear pairings and its applications," *PKC 2004,* LNCS 2947, pp. 277–290, Springer-Verlag, 2004.