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@oberon.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 data 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.* [24], the user decrypts the searched documents using his private key, A_{priv}. Our another goal is to enable the 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 the 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* (PECK) scheme in [19] by using a hybrid encryption technique.

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

1 Introduction

Recently, there has been interest in the problem of searching on encrypted data. Song *et al.*[23] addressed this problem and proposed the solution working as follows. Consider a user that stores encrypted documents on an untrusted data storage server. Documents and keywords are encrypted in a way that allows the untrusted server only to test whether the document contains a certain keyword W after receiving from a user a piece of information called *a trapdoor for keyword* W without learning anything else about the document. Without a trapdoor, the server learns nothing about encrypted documents. For such a scheme, they proposed a symmetric key scheme in which the same key was

*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.

used to encrypt a document and make a trapdoor. Afterwards, several schemes have been proposed to improve and extend this scheme [3, 14, 15, 19, 24]. We call them "*Searchable Keyword Encryption* (SKE)". In [14], 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 [14, 23] 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 the public key. In [24], Waters *et al.* also presented another application, an encrypted and searchable audit log (secure audit log), and proposed two schemes to build the system. However, none of these schemes support a secure conjunctive keyword search, which becomes an indispensable requirement for an efficient and secure search function in some applications, such as a secure audit log. For example, an audit escrow agent in the secure audit log system may not give a trapdoor of a single keyword, such as "Alice", to an investigator, because the trapdoor of the single keyword may match a huge number of audit logs including many unnecessary and even unauthorized ones. There are two naive solutions for the conjunctive search, set intersection [14] and meta keyword. These methods and their limitations are explained well in [15]. However, neither solution is appropriate. The set intersection makes an untrusted server learn which documents match each individual keyword, and, over time, the server may combine this information with a knowledge of statistically likely searches to infer 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 conjunctive keyword search scheme was proposed by Golle *et al.* in a symmetric key setting [15]. Park *et al.* proposed the public key analogue of this scheme [19]. Recently proposed searchable encryption schemes, [21] and [7] also support the conjunctive keyword search.

The *Identity-Based Encryption* (IBE) scheme is another cryptographic primitive that has received much attention. The idea of IBE was formulated by Shamir [22] in 1984, aimed at simplifying certificate management in e-mail systems. An IBE scheme is an asymmetric system where the public key is effectively replaced by a user's publicly available identity information or any arbitrary string derived from the user's identity. The private key is computed using a master-key of a trusted third party called *Private Key Generator* (PKG) and the identity of the user. However, constructing a practical IBE scheme remained an open problem until the work by Boneh and Franklin [5]. They proposed a practical IBE scheme using bilinear maps, with some additional applications such as key escrow/recovery, revocation of public keys and delegation of decryption keys. In the same year, another IBE system whose security is based on quadratic residues was proposed by Cocks [9].

In this paper, we introduce a new mechanism called "Searchable Keyword-Based Encryption (SKBE)", which is a public key encryption with the following functionalities. A plaintext is encrypted using a public key A_{pub}. Then, the ciphertext depends on the keywords of the plaintext. Given certain information called *a decrypt trapdoor for specific keywords W_i's*, ciphertexts containing keywords W_i's can be decrypted without a private key A_{priv}. Similarly to SKE, 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 but get no other information about the ciphertext. These trapdoors can be generated only with A_{priv}. Without a trapdoor, a ciphertext does not reveal anything about the plaintext.

Our Contributions. Our main contribution is the proposition of the searchable keyword-based encryption scheme. The scheme guarantees more secure and efficient functionalities than any other scheme for some applications, such as email gateways, decryption key delegation systems and secure audit logs. We will illustrate this fact in the following paragraph. 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 *Public Key Encryption with Conjunctive Field Keyword Search* (PECK) scheme in [19] by using a hybrid encryption technique.

Motivation and Applications. SKBE is of interest since it guarantees more secure and efficient operation than SKE or 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 in [3, 19]. Alice's mail gateway is supposed to route an email to the 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 the 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 the 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 [24]. 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 [24]. This problem could be resolved by our SKBE because it 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 the entries for the search trapdoor but can not decrypt them. In addition, we will design our SKBE to support conjunctive keyword search, while the Waters *et al.*' scheme does not.

Overview. The rest of this paper is organized as follows. We first formalize SKBE and define its security model in Section 2 and cover complexity assumptions in Section 3. In Section 4 we propose an efficient and provably secure construction of SKBE. Finally, this paper concludes in Section 5.

2 Searchable Keyword-Based Encryption Scheme

We assume a document D as (M, H). The message M is the content of D and the keywords H is associated with M. We assume 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 document D in the i-th keyword field. For simplicity, we employ the same assumptions as in [15, 19]:

1. H does not have any two keyword fields with 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.

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

Definition 1 *A searchable keyword-based encryption scheme $\mathcal{E}$ consists of the following polynomial time randomized algorithms:*

1. *KeyGen*(1^k)*: Takes a security parameter,* 1^k*, and generates a public/private key pair* A_{pub}/A_{priv}*.*

2. *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*).* $\mathcal{M}$ *is the message space and* $\mathcal{C}$ *is the ciphertext space.*

3. *STrapdoor*(A_{priv}, Q)*: Given a private key* A_{priv} *and a query* Q*, produces a search trapdoor* T_Q^S*.*

4. *DTrapdoor*(A_{priv}, Q)*: Given a private key* A_{priv} *and a query* Q*, produces a decrypt trapdoor* T_Q^D*.*

5. *Test*(A_{pub}, C, T_Q^S)*: Given a public key* A_{pub}*, a ciphertext* $C =$ *Encrypt*$(A_{pub}, D) \in \mathcal{C}$*, and a* search trapdoor $T_Q^S =$ *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. *Decrypt*(A_{pub}, C, T_Q^D)*: Given a public key* A_{pub}*, a ciphertext* $C =$ *Encrypt*$(A_{pub}, D) \in \mathcal{C}$*, and a* decrypt trapdoor $T_Q^D =$ *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* $\perp$ *otherwise.*

A message M is encrypted depending on $H = (W_1, W_2, \cdots, W_m)$ with a public key A_{pub}. 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 that they are intended for which keywords.

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 decrypt the message M. 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 describe SKBE in operation using a secure audit log [24] as a sample application. The escrow agent runs the KeyGen algorithm to generate her public/private key pair A_{pub}/A_{priv}. The 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 uses the STrapdoor/DTrapdoor algorithm to generate T_Q^S/T_Q^D and grants them to the investigator. The investigator gives T_Q^S to database server 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 entries' 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 the user owning A_{priv} needs an appropriate decrypt trapdoor. Alice in email gateway [3] and decryption key delegation [5] should be allowed to decrypt all mails sent to her. That is, she has to be able to make the appropriate decrypt trapdoor just

by looking at the 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 decrypt any ciphertext always using T_Q^D. We denote such a trapdoor and query as $T_{Q_D}^D$ and Q_D, respectively. We use the public value W_{pub} as the one of the domain parameters of SKBE.

Informally, a SKBE scheme is secure if the ciphertext $C = \mathsf{Encrypt}(A_{pub}, D)$ does not reveal any information about the document D when a suitable trapdoor T_Q^S or T_Q^D for $Q \subseteq H$ is not available. Now, we formalize this notion of security in the sense of chosen ciphertext security. Chosen ciphertext security (IND-CCA) is the standard notion of security for a public key encryption scheme [4, 10, 20]. Hence, it is natural to require that a searchable keyword-based encryption scheme also satisfy this strong notion of security. However, the definition of chosen ciphertext security must be strengthened a bit. The reason is that an active adversary might obtain trapdoors T_Q^S or T_Q^D for any Q of his choice. We refer to such queries as search/decrypt queries. Even under such attack the adversary should not be able to distinguish a ciphertext C_0 of a document $D_0 = (M_0, H_0)$ from a ciphertext C_1 of a document $D_1 = (M_1, H_1)$ when he did not obtain suitable trapdoors for them. For the rest, following [15], we say a search trapdoor T_Q^S distinguishes a document D_0 from a document D_1 if the Test's outputs of their ciphertexts with T_Q^S are different from each other. Observe that $\mathcal{A}$ succeeds trivially if it is given a search trapdoor distinguishing D_0 from D_1 or a decrypt trapdoor matching D_0 or D_1. Formally, we define the chosen ciphertext security against an active adversary $\mathcal{A}$ using the following game between a challenger and the adversary.

Definition 2 The *IND-SKBE-CCA security game* is as follows:

Setup: *The challenger runs the* $\mathsf{KeyGen}(1^k)$ *algorithm to generate* A_{pub} *and* A_{priv}. *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:*

- $\mathsf{STrapdoor}$ *query* $< Q_i >$: *The challenger responds by running algorithm* $\mathsf{STrapdoor}$ *to generate the search trapdoor* T_Q^S *corresponding to the query* $< Q_i >$. *It sends* T_Q^S *to the adversary.*
- $\mathsf{DTrapdoor}$ *query* $< Q_i >$: *The challenger responds by running algorithm* $\mathsf{DTrapdoor}$ *to generate the decrypt trapdoor* T_Q^D *corresponding to the query* $< Q_i >$. *It sends* T_Q^D *to the adversary.*
- $\mathsf{Decrypt}$ *query* $< C_i >$: *The challenger responds by running algorithm* $\mathsf{Decrypt}$ *to decrypt the ciphertext* $C_i \in \mathcal{C}$ *using the decrypt trapdoor* $T_{Q_D}^D$. *It sends the resulting message* $M_i \in \mathcal{M}$ *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* $\mathsf{Phase\ 1}$ *is over it outputs two 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 *could be the same value. The restrictions are as follows: First, none of the search trapdoors asked previously in* $\mathsf{Phase\ 1}$ *trivially distinguish* D_0 *and* D_1. *Second, none of* Q_i *for the decrypt trapdoors such that* Q_i *matches* H_0 *or* H_1 *were issued previously in* $\mathsf{Phase\ 1}$. *Lastly, neither decryption queries*

for C_0 nor C_1 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:

STrapdoor query $< Q_i >$: Q_i such that the corresponding search trapdoor T_Q^S is trivially distinguishing for D_0 and D_1 must not be allowed, elsewhere challenger responds as in Phase 1.

DTrapdoor query $< Q_i >$ where $Q_i \notin H_0, H_1$: Challenger responds as in Phase 1.

Decrypt query $< C_i >$ 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 $f : \mathbb{R} \rightarrow [0,1]$ where $f(k) < \frac{1}{g(k)}$ for any polynomial g and sufficiently large k.

Definition 3 *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.

3 Complexity Assumptions

Let $\mathbb{G}_1$ be a bilinear group of prime order p, and let P be its generator. Here, we review the *Bilinear Diffie-Hellman Inversion* (BDHI) assumption [2, 17] and the *Bilinear Collusion Attack* (BCA) assumption [8, 17].

3.1 Bilinear Map

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

- Bilinear: We say that a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ 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 es an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in \mathbb{G}_1$.

We can make the bilinear map using the Weil pairing or the Tate pairing [5, 6, 13, 16]. 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.

3.2 Bilinear Diffie-Hellman Inversion Assumption

The q-BDHI problem[2, 17] 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 4 *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 a proposed SKBE system in the next section.

3.3 Bilinear Collusion Attack Assumption

The q-BCA problem[8, 17] 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 $\Pr[\mathcal{A}(P, xP, u_1, \cdots, u_q, \frac{1}{1+u_1}P, \cdots, \frac{1}{1+u_q}P) = \hat{e}(P, P)^{1/x}] \geq \epsilon$ 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 5 *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, 17].

4 Construction

We construct an efficient searchable keyword-based encryption scheme $\mathcal{SKBE}$ that is IND-SKBE-CCA secure based on the Bilinear Diffie-Hellman Inversion assumption from Section 3.2. The security is proved in the random oracle model. This scheme is constructed from a public key encryption with conjunctive field keyword search (PECK) scheme in [19] using a hybrid encryption technique (likely in [12]).

Let $\mathbb{G}_1$ be a bilinear group of prime order p, and let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear map. For our scheme, we need 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 the secret key of a symmetric key encryption scheme $(\mathcal{E}_{sk}, \mathcal{D}_{sk})$ secure in FG (Find-Guess) [12]. H_1 and H_2 are different from each other.

1. $\mathsf{KeyGen}(1^k)$: 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 $A_{priv} = [s_1, s_2, \cdots, s_m, s_{m+1}, s_{m+2}]$.

2. $\mathsf{Encrypt}(A_{pub}, D = (M, H))$: Randomly select a secret key $sk \in \mathcal{K}$ for symmetric key encryption of M, $\mathcal{E}_{sk}(M)$. Obtain the hash value $H_3(M, sk)$, and set it to r_0. Compute the following values. The ciphertext $C \in \mathcal{C}$ is $[U, A_1, A_2, \cdots, A_m, B_1, B_2, \cdots, B_m, C, S, R]$ where

$$\begin{aligned} U &= \mathcal{E}_{sk}(M), \\ A_i &= r_0(Y_i + H_1(W_i)P_1) + r_iP_1, \\ B_i &= r_iY_{m+1}, \\ C &= r_0Y_{m+2}, \\ S &= H_2(g^{r_0}), \\ R &= H_2(h^{r_0}) \oplus sk. \end{aligned}$$

3. $\mathsf{STrapdoor}(A_{priv}, Q)$: For the input $Q = (I_1, I_2, \cdots, I_t, \Omega_1, \Omega_2, \cdots, \Omega_t)$, select a random $u \in \mathbb{Z}_p^*$ and make the *search trapdoor* $T_Q^S = [T_1^S, T_2^S, T_3^S, I_1, \cdots, I_t]$ where

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

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

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

5. $\mathsf{Test}(A_{pub}, C, T_Q^S)$: Check the equality,

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

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

6. $\mathsf{Decrypt}(A_{pub}, C, T_Q^D)$: Compute $\widetilde{h^{r_0}} = \frac{\hat{e}(A_{I_1} + \cdots + A_{I_t} + T_3^D C,\ T_1^D)}{\hat{e}(B_{I_1} + \cdots + B_{I_t},\ T_2^D)}$, $\widetilde{sk} = H_2(\widetilde{h^{r_0}}) \oplus R$, $\widetilde{M} = \mathcal{D}_{\widetilde{sk}}(U)$ and $\widetilde{r_0} = H_3(\widetilde{M}, \widetilde{sk})$. Check the equality,

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

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

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

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

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

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

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

The security of this scheme is proved under the q-BDHI assumption in the random oracle model.

Theorem 1 *Suppose the q-BDHI assumption holds in $\mathbb{G}_1$. Then the above scheme $\mathcal{SKBE}$ is IND-SKBE-CCA secure.*

Proof. See Appendix A.

Abdalla *et al.* proposed the first statistically consistent PEKS scheme in [1]. We notice that our proposed SKBE scheme is not statistically consistent but computationally consistent like *BDOP-PEKS* [3]. However, our scheme can be modified to meet statistical consistency in a way that Abdalla *et al.* used in their paper to make statistically consistent PEKS scheme based on *BDOP-PEKS*. Due to the limited page, we do not deal with this problem related to consistency in this paper but we will treat the method to make the statistical consistent SKBE scheme in future work.

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 assumption. In future works, we will study generic construction methods for SKBE. Furthermore, we will analyze SKBE in various directions, for example its relations with other cryptographic primitives such as identity-based encryption or hierarchial identity-based encryption.

References

[1] M. Abdalla *et al.*, "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," *CRYTO 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] 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.

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

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

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

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

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

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

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

[17] K. Kim, D. J. Park and P. J. Lee, "ID-based encryption scheme without admissible encoding," (*In Korean*), *CISC-S04,* pp. 30–39, 2004.

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

[19] 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.

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

[21] 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/.

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

[23] 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.

[24] 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.

[25] 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.

A Proof of Theorem 1

Proof. Suppose $\mathcal{A}$ has advantage ϵ in attacking the proposed scheme under the IND-SKBE-CCA game. Suppose $\mathcal{A}$ makes $\mathsf{STrapdoor}$ or $\mathsf{DTrapdoor}$ 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 $\epsilon' = \epsilon/(e(q_{H_1}/m)^m q_{H_3})$, 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 $\delta_1, \delta_2, \cdots, \delta_{q_T} \in \mathbb{Z}_l^*$ 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 $1 \le i \le q_T$, and stores the pairs $(\delta_i, \frac{1}{x+\delta_i}U)$'s.

4. $\mathcal{B}$ selects random numbers $\alpha_0, \alpha_1, \alpha_2, \cdots, \alpha_m, \beta_1, \beta_2, \cdots, \beta_m \in \mathbb{Z}_p$, lets $P_1 = U$, $P_2 = \alpha_0 U$ and compute $Y_i = \alpha_i x V - \beta_i U$ for $1 \leq i \leq m$, $Y_{m+1} = \alpha_{m+1} U$, $Y_{m+2} = \alpha_{m+2} V$, $g = \hat{e}(P_1, P_1)$ and $h = \hat{e}(P_1, P_2)$. $\mathcal{B}$ gives $\mathcal{A}$ the public key $A_{pub} = [P_1, P_2, Y_1, \cdots, Y_m, g, h]$. The corresponding private key A_{priv} is $[s_1 = \alpha_1 x - \beta_1, \cdots, s_m = \alpha_m x - \beta_m, s_{m+1} = \alpha_{m+1}, s_{m+2} = \alpha_{m+2} x]$ 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 $< W_{i,j}, h_{i,j}, c_{i,j} >$ called the H_1-list. The list is initially empty. When $\mathcal{A}$ queries the random oracle H_1 at a point $W_{i,j} \in \{0,1\}^*$, Adversary $\mathcal{B}$ responds as follows:

1. If the query $W_{i,j}$ already appears in the H_1-list in a tuple $< W_{i,j}, h_{i,j}, c_{i,j} >$, then adversary $\mathcal{B}$ responds with $h_{i,j} = H_1(W_{i,j})$.
2. Else if there are no $< W_{i,j}, h_{i,j}, c_{i,j} >$ tuples such that $c_{i',j} = 0$ for $i \neq i'$ in H_1-list, $\mathcal{B}$ generates a random coin $c_{i,j} \in \{0,1\}$ so that $\Pr[c_{i,j} = 0] = m/q_{H_1}$. If there is such a tuple, set $c_{i,j} = 1$.
3. If $c_{i,j} = 0$, $\mathcal{B}$ sets $h_{i,j} = \beta_j$ and otherwise, selects a random $h_{i,j} \in \{0,1\}^{\log_2 p}$. $\mathcal{B}$ adds the tuple $< W_{i,j}, h_{i,j}, c_{i,j} >$ to the H_1-list, and responds with $h_{i,j} = H_1(W_{i,j})$.

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 $g_i \in \{0,1\}^*$, algorithm $\mathcal{B}$ 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 $\gamma_i = H_2(g_i)$.
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 $\gamma_i = H_2(g_i)$.

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 $< M_i, sk_i, r_i, X_i, Y_i >$ called the H_3-list. The list is initially empty. When $\mathcal{A}$ queries the random oracle H_3 at a point $M_i \in \{0,1\}^*$ and $sk_i \in \mathcal{K}$, algorithm $\mathcal{B}$ responds as follows:

1. If the query (M_i, sk_i) already appears in the H_3-list in a tuple $< M_i, sk_i, r_i, X_i, Y_i >$, then $\mathcal{B}$ responds with $r_i = H_3(M_i, sk_i)$.
2. Otherwise, $\mathcal{B}$ selects a random $r_i \in \mathbb{Z}_p^*$ and computes $X_i = H_2(g^{r_i})$ and $Y_i = H_2(h^{r_i}) \oplus sk_i$. $\mathcal{B}$ adds the tuple $< M_i, sk_i, r_i, X_i, Y_i >$ to the H_3-list and gives r_i 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 algorithm for responding to H_1 queries to obtain an $h_{i,j}$'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 all $c_{i,j}$'s are 0, 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 an 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 $F_i = \frac{1}{v_i(x+\delta_i)}U = \frac{1}{\Gamma_i x+\Delta_i}P_1$ is same as $(1/(\alpha_{I_{i,1}} + \cdots + \alpha_{I_{i,t_i}})x - (\beta_{I_{i,1}} + \cdots + \beta_{I_{i,t_i}}) + (h_{i,1} + \cdots + h_{i,t_i}))P_1$. Thus, $(F_i, \frac{1}{\alpha_{m+1}}F_i, u_i)$ is the correct *search trapdoor* T_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 algorithm for responding to H_1 queries to obtain an $h_{i,j}$'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 all $c_{i,j}$'s are 0, 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 an 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 $G_i = \frac{\alpha_0}{v_i(x+\delta_i)}U = \frac{1}{\Gamma_i x+\Delta_i}P_2$ is same as $(1/(\alpha_{I_{i,1}} + \cdots + \alpha_{I_{i,t_i}})x - (\beta_{I_{i,1}} + \cdots + \beta_{I_{i,t_i}}) + (h_{i,1} + \cdots + h_{i,t_i}))P_2$. Thus, $(G_i, \frac{1}{\alpha_{m+1}}G_i, u_i)$ is the correct *decrypt trapdoor* T_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: When $\mathcal{A}$ asks to decrypt an encrypted message $[U_i, A_{i,1}, A_{i,2}, \cdots, A_{i,m}, S_i, R_i]$. $\mathcal{B}$ searches H_3-list to find a tuple containing (S_i, R_i) as X_i, Y_i. $\mathcal{B}$ checks whether $U_i = \mathcal{E}_{sk_i}(M_i)$. 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 all $c_{i,j}$'s are 0 for at least one of $i \in \{0,1\}$ (i.e. $[c_{0,1} = 0, c_{0,2} = 0, \cdots, c_{0,m} = 0]$ or $[c_{1,1} = 0, c_{1,2} = 0, \cdots, c_{1,m} = 0]$), then $\mathcal{B}$ fails.
2. $\mathcal{B}$ picks a random $i \in \{0,1\}$ such that all $c_{i,j}$'s are 0.
3. $\mathcal{B}$ selects random $\rho, r_1, ..., r_m \in \mathbb{Z}_p$, $sk \in \mathcal{K}$ and computes $U = \mathcal{E}_{sk}(M_b)$, $A_i = \frac{\rho}{x}(\beta_i P_1 + (\alpha_1 x - \beta_i)P_1) + r_i P_1 = \rho\alpha_i P_1 + r_i P_1$ and $B_i = r_i Y_{m+1}$ for $1 \leq i \leq m$ and $C = \rho\alpha_{m+2}P_1$. $\mathcal{B}$ selects random S, $R \in \{0,1\}^{\log_2 p}$ and responds to $\mathcal{A}$ with the challenge $[U, A_1, \cdots, A_m, B_1, \cdots, B_m, C, S, R]$.

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 h^r in the H_3-list and computes $(h^r)^{1/(\alpha_0\rho)}$. $\mathcal{B}$ 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 $\mathcal{A}$. Look at the way $\mathcal{B}$ responds to Decrypt oracle queries $< C_i >$'s. If the Decrypt oracle is asked to decrypt a ciphertext C_i that $\mathcal{A}$ creates by encrypting a message M_i then $\mathcal{B}$ responds with M_i found in the H_3-list. For a valid encryption of M_i, $\mathcal{A}$ must have asked r_i to the H_3 oracle, therefore $\mathcal{B}$ can find M_i in its H_3-list. 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. The reason that such a simulation of the Decrypt oracle is possible is that there exists a $\lambda(k)$-knowledge extractor where $1-\lambda(k)$ is negligible. Note that the proposed scheme is a hybrid encryption with the decrypt algorithm checking the validity of the decryption value. In such a hybrid encryption scheme, an adversary can not create a ciphertext C without "knowing" its underlying plaintext M [4, 12].

Now, we calculte the probability 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. The probability of event $\mathcal{E}_1$, $\Pr[\mathcal{E}_1]$ is $(1-\frac{m}{q_{H_1}})^{q_T}$.
- $\mathcal{E}_2$: $\mathcal{B}$ does not fail preparing the challenge. The probability of event $\mathcal{E}_2$, $\Pr[\mathcal{E}_2]$ is $\left(\frac{m}{q_{H_1}}\right)^m$.
- $\mathcal{E}_3$: $\mathcal{B}$ selects the right value as $\hat{e}(P,P)^{1/x}$ in the H_3-list during the Guess step. The probability of event $\mathcal{E}_3$, $\Pr[\mathcal{E}_3]$ is $\frac{1}{q_{H_3}}$.

We can assume that q_T is sufficiently large, thus, $\Pr[\mathcal{E}_1] = (1-\frac{m}{q_{H_1}})^{q_T} > (1-\frac{m}{q_T})^{q_T} = \frac{1}{e}$. The probability that $\mathcal{B}$ wins the game is computed by the multiplication of the probabilities of the above events. Thus, $\mathcal{B}$ breaks the q_T-aBCA problem with the advantage $\epsilon' > \epsilon \times (1-\frac{m}{q_{H_1}})^{q_T} \times \left(\frac{m}{q_{H_1}}\right)^m \times \frac{1}{q_{H_3}}$ $= \frac{\epsilon}{e(q_{H_1}/m)^m q_{H_3}}$. □

B Single Keyword Search-SKBE

When we formalized the model for SKBE, we designed the SKBE to support a secure conjunctive search. In this section, we give a variant of SKBE, *Single Keyword Search-SKBE* (S-SKBE), which meets requirements of SKBE except that the S-SKBE does not support conjunctive keyword search. The S-SKBE can be simply constructed from modification of some previous works [3, 5, 19, 24].

First, by using a pair of an IBE scheme and a PEKS scheme, (IBE, PEKS), or a pair of an IBE scheme and a PECK scheme, (IBE, PECK), in a similar way by Waters *et al.* [24], the following S-SKBE schemes are simply obtained.

S-SKBE 1 : The ciphertext has the following form:

$$[E_{sk}(M), PEKS(A_{pub}, W_1), \cdots, PEKS(A_{pub}, W_m), IBE_{W_1}(flag||sk), \cdots, IBE_{W_m}(flag||sk)]$$

In S-SKBE 1, the search trapdoor becomes the trapdoor of PEKS and the decrypt trapdoor becomes the private key d_{W_i} corresponding to W_i in IBE.

S-SKBE 2 : The ciphertext has the following form:

$$[E_{sk}(M), PECK(A_{pub}, \{W_1, \cdots, W_m\}), IBE_{W_1}(flag||sk), \cdots, IBE_{W_m}(flag||sk)]$$

In S-SKBE 2, the search trapdoor becomes the trapdoor of PECK and the decrypt trapdoor becomes the private key d_{W_i} corresponding to W_i in IBE.

Second, S-SKBE can be easily constructed by converting from PEKS schemes using bilinear maps as in Section 4.2. The ciphertext for the S-SKBE has the following form:

$$[E_{sk}(flag||M), SSKBE(A_{pub}, W_1), \cdots, SSKBE(A_{pub}, W_m)]$$

For simplicity, we consider only SSKBE(A_{pub}, W).

S-SKBE 3 : We use the groups, the bilinear map and the hash functions in Section 4.1.

1. $\mathsf{KeyGen}(1^k)$: The input security parameter 1^k determines the size, p, of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$. The algorithm chooses a random $s_1 \in \mathbb{Z}_p^*$ and two different generators P_1, P_2 of $\mathbb{G}_1$. It outputs $A_{pub} = [P_1, P_2, Y_1 = s_1 P_1, g = \hat{e}(P_1, P_1), h = \hat{e}(P_1, P_2)]$ and $A_{priv} = [s_1]$.
2. $\mathsf{SSKBE}(A_{pub}, W, sk)$: Select a random $r \in \mathbb{Z}_p^*$. Obtain the hash value of keyword W and compute $A = r(Y_1 + H_1(W)P_1)$, $S = H_2(g^r)$ and $R = H_2(h^r) \oplus sk$. Output $C = [A, S, R]$.
3. $\mathsf{STrapdoor}(A_{priv}, \Omega)$: Output $T_\Omega^S = \frac{1}{s_1 + H_1(\Omega)} P_1$.
4. $\mathsf{DTrapdoor}(A_{priv}, \Omega)$: Output $T_\Omega^D = \frac{1}{s_1 + H_1(\Omega)} P_2$.
5. $\mathsf{Test}(A_{pub}, C, T_\Omega^S)$: Check the equality,

$$H_2(\hat{e}(A, T_\Omega^S)) = S$$

   If so, output 'yes'; otherwise, output 'no'.
6. $\mathsf{Decrypt}(A_{pub}, C, E_{sk}(flag||M), T_\Omega^D)$: Compute $\widetilde{sk} = H_2(\hat{e}(A, T_\Omega^D)) \oplus R$ and $\widetilde{flag}||\widetilde{M} = D_{\widetilde{sk}}(E_{sk}(flag||M))$. If $\widetilde{flag} = flag$, output $\widetilde{M}$; otherwise, output $\perp$.

S-SKBE 4 : We use the groups, the bilinear map and the hash functions in [3].

1. $\mathsf{KeyGen}(1^k)$: The input security parameter 1^k determines the size, p, of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$. The algorithm chooses a random $\alpha, \beta \in \mathbb{Z}_p^*$ and a generator g of $\mathbb{G}_1$. It outputs $A_{pub} = [g, h_1 = g^\alpha, h_2 = g^\beta]$ and $A_{priv} = [\alpha, \beta]$

2. $\mathsf{SSKBE}(A_{pub}, W, sk)$: Select a random $r \in \mathbb{Z}_p^*$. Obtain the hash value of keyword W and compute $S = H_2(\hat{e}(H_1(W), h_1^r))$ and $R = H_2(\hat{e}(H_1(W), h_2^r)) \oplus sk$. Output $C = [A, S, R]$.
3. $\mathsf{STrapdoor}(A_{priv}, \Omega)$: Output $T_\Omega^S = H_1(\Omega)^\alpha \in \mathbb{G}_1$.
4. $\mathsf{DTrapdoor}(A_{priv}, \Omega)$: Output $T_\Omega^D = H_1(\Omega)^\beta \in \mathbb{G}_1$.
5. $\mathsf{Test}(A_{pub}, C, T_\Omega^S)$: Check the equality,

$$H_2(\hat{e}(A, T_\Omega^S)) = S$$

   If so, output 'yes'; otherwise, output 'no'.
6. $\mathsf{Decrypt}(A_{pub}, C, E_{sk}(flag||M), T_\Omega^D)$: Compute $\widetilde{sk} = H_2(\hat{e}(A, T_\Omega^D)) \oplus R$ and $\widetilde{flag}||\widetilde{M} = D_{\widetilde{sk}}(E_{sk}(flag||M))$. If $\widetilde{flag} = flag$, output $\widetilde{M}$; otherwise, output $\perp$.