7-2016

Linear encryption with keyword search

Shiwei ZHANG

Guomin YANG
Singapore Management University, gmyang@smu.edu.sg

Yi MU

Linear Encryption with Keyword Search

Shiwei Zhang(✉), Guomin Yang, and Yi Mu

Centre for Computer and Information Security Research,
School of Computing and Information Technology, University of Wollongong,
Wollongong, Australia
{sz653,gyang,ymu}@uow.edu.au

Abstract. Nowadays an increasing amount of data stored in the public cloud need to be searched remotely for fast accessing. For the sake of privacy, the remote files are usually encrypted, which makes them difficult to be searched by remote servers. It is also harder to efficiently share encrypted data in the cloud than those in plaintext. In this paper, we develop a searchable encryption framework called *Linear Encryption with Keyword Search* (LEKS) that can semi-generically convert some existing encryption schemes meeting our *Linear Encryption Template* (LET) to be searchable without re-encrypting all the data. For allowing easy data sharing, we convert a Key-Policy Attributed-Based Encryption (KP-ABE) scheme to a Key-Policy Attributed-Based Keyword Search (KP-ABKS) scheme as a concrete instance of our LEKS framework, making both the encrypted data and the search functionality under fine-grained access control. Notably, the resulting KP-ABKS is the first proven secure ABKS scheme with IND-sCKA security in the random oracle model, assuming the hardness of the ℓ-DCBDH problem derived from the (P, f)-DBDH problem family.

Keywords: Searchable encryption · Keyword search · Cloud security

1 Introduction

Cloud computing [14] provides on-demand computing resources that are accessible via the Internet, including computing power and data storage. With the convenient cloud services, users can outsource their computing resources to the cloud, and access them through terminals with low computing capabilities, such as mobile devices. Usually, those terminals also have low network connectivity due to the transmission technology, access cost, and other factors.

In terms of data storage, one important function is data search. Since all the user data are stored on the cloud server, users have to send search queries to the server to search for the data containing certain keywords. However, the normal search operation for plaintext is no longer working when data privacy is considered, since all the data are encrypted and cannot be read by the server.

To perform search on encrypted data, it is impractical for the user to do the search locally with all the data downloaded from the server, due to the high

J.K. Liu and R. Steinfeld (Eds.): ACISP 2016, Part II, LNCS 9723, pp. 187–203, 2016.
DOI: 10.1007/978-3-319-40367-0_12

demand on the bandwidth. It is also impractical to give the server the user secret key due to privacy concerns. Thus searchable encryption has been introduced such that the search operation is performed by the server, but the server cannot get any meaningful information from the search query or the encrypted data. In searchable encryption, all the data files and their associated keywords are encrypted. To search for the data with certain keyword, the user generates a trapdoor for the keyword and enquires the server with the trapdoor. The server searches the whole database to locate the data where the encrypted keyword matches the keyword embedded in the trapdoor. During the searching process, the server only knows whether an encrypted keyword matches the user trapdoor or not, and nothing else. After that, the server returns the search result to the user who can download the ciphertexts and decrypt the data.

In Public-key Encryption with Keyword Search (PEKS) [8], the data and the keywords are encrypted for only one user (i.e., the intended receiver of the data). In contrast, data can be encrypted with certain attributes in Attribute-Based Encryption (ABE) [16]. For instance, Alice can encrypt some data with attributes "full-time" and "student". Later, any user can decrypt the resulting ciphertext if the attributes in the ciphertext match the policy associated with the user. Thus Bob associated with a policy "(full time AND student) OR staff" can decrypt the above ciphertext. The corresponding searchable encryption for ABE is named Attribute-Based Keyword Search (ABKS) [19,21]. As in ABE, Alice can encrypt the data and its associated keywords using certain attributes. After uploading the ciphertexts to the server, Bob can do the search and decryption since the attributes used by Alice in the encryption matches Bob's policy. This feature is very important in the cloud environment where a user can share data with multiple users by encrypting the data only once. However, to the best of our knowledge, no ABKS scheme proposed in the literature is proven secure. Hence, one of our goals is to construct ABKS schemes with provable security.

In addition, keyword search functionality is usually associated with an encryption scheme where both the data and the keywords are encrypted for the same receiver(s). This paper also aims to provide a universal construction of searchable encryption schemes from some existing encryption schemes. This enables us to add a compatible keyword search functionality to an existing cryptosystem without re-encrypting all the data.

1.1 Related Work

Diffie and Hellman introduced the notion of Public-Key Encryption (PKE) [11] where Alice encrypts a message with Bob's public key, and Bob decrypts the ciphertext with his secret key. Based on the idea of using the user identity as the public key [17], Boneh and Franklin proposed a practical Identity-Based Encryption (IBE) scheme [9] where Alice encrypts the message with Bob's identity. In 2005, Sahai and Waters introduced Fuzzy Identity-Based Encryption which can be treated as the first Attribute-Based Encryption (ABE) [16], an instance of Function Encryption [20]. In ABE, the decryption keys of the users

and the ciphertexts are associated with access policies and attributes, respectively. If and only if the attributes match the policy, the ciphertext can be successfully decrypted. Depending on how the identity and the ciphertext are associated, Attribute-Based Encryption schemes are classified into Key-Policy ABE (KP-ABE) [3,12,15,16] and Ciphertext-Policy ABE (CP-ABE) [4]. In KP-ABE, Bob's secret key is associated with a policy. After receiving the ciphertext encrypted with some attributes from Alice, Bob can decrypt it if and only if the attributes match his policy. In CP-ABE, the ciphertexts are associated with policies, and the secret keys are associated with attributes.

To enable the search functionality for encrypted data, various searchable encryption schemes [1,8,10,13,19,21] have been proposed under different settings. Boneh et al. [8] introduced PEKS, which is used with a conventional public key encryption scheme. Later, Identity-Based Keyword Search (IBKS) schemes were also proposed [1,10]. Recently, due to the popularity of ABE, there have been some research works on ABKS [19,21]. In addition, there are also keyword search schemes for other encryption variants, such as Broadcast Encryption [2].

To the best of our knowledge, [19,21] are the only ABKS schemes proposed in the literature. However, neither of those schemes is proven secure. In particular, after analysing the ABKS scheme in [19], we found the scheme is flawed where an adversary can always distinguish keywords from a ciphertext by testing $e(\hat{D}, T_{i'}^{\frac{1}{H(w_{\mu'})}}) \stackrel{?}{=} e(g, D_{i'})$. For the KP-ABKS scheme in [21], the security proof is invalid (see Sect. 4.3) and thus the security of this scheme remains unknown. For the CP-ABKS scheme in [21], no formal security proof has been provided.

In terms of provable security, it depends on the hardness of some computational problems (e.g. Discrete Logarithm Problem (DLP), Diffie-Hellman Problem (DHP) [11], etc.). Shoup [18] introduced the generic group model which was used to obtain the complexity lower bound regarding the hardness of DLP and DHP. Later, dealing with bilinear maps, Boneh et al. [6] introduced the generic bilinear group model and the general Diffie-Hellman Exponent Problem. Besides, the generic bilinear group model is also used in [5,7] for analysing the Decisional Linear (DLIN) Problem and q-Strong Diffie-Hellman (q-SDH) Problem.

1.2 Our Contribution

In this paper, we introduce a new problem family named *Decisional Bilinear* (P, f)*-Diffie-Hellman problem* ((P, f)-DBDH problem, for short). We prove the (P, f)-DBDH problem is computationally hard in generic bilinear group model if the polynomial f is not dependent on the polynomial set P. Based on the (P, f)-DBDH problem, we derive a hard computational problem named *Decisional* ℓ*-Combined Bilinear Diffie-Hellman problem* (ℓ-DCBDH problem).

As the main contribution of this work, we introduce two new notions named *Linear Encryption Template* (LET) and *Linear Encryption with Keyword Search* (LEKS), and provide their formal definitions. LET can model different asymmetric encryption schemes, including but not limited to PKE, IBE and ABE schemes, which have the property of linearity. The linearity property requires

a sub-algorithm $e(g,g)^{\alpha s} \leftarrow \mathcal{D}(SK, C_1, \dots)$ in the decryption algorithm where SK is the secret key involved, $(C_1, \dots)$ are the ciphertext components and for all $t \in \mathbb{Z}_p$, $\mathcal{D}(SK^t, C_1, \dots) = D(SK, C_1, \dots)^t$. Given an encryption fitting LET, we provide a semi-generic conversion to a LEKS scheme where the construction is generic but we require security proofs for individual conversions. We also define two security models for LEKS schemes: Indistinguishability under Adaptive Chosen Keyword Attack (IND-CKA) and its weaker Selective-ID version (IND-sCKA). With LET and our conversion from LET to LEKS, we can construct PEKS from PKE, IBKS from IBE, ABKS from ABE, and so on.

To illustrate the feasibility of our semi-generic framework, we give an instance of LET and then apply our conversion to procude a LEKS scheme. We first show that a variant [15] of Goyal et al.'s ABE scheme [12] fits LET by proving it has the property of linearity. Then we apply our LEKS conversion to convert the KP-ABE scheme into a KP-ABKS scheme. After that, we prove the resulting KP-ABKS scheme is IND-sCKA secure in the random oracle model under ℓ-DCBDH assumption. It is worth noting that to the best of our knowledge, our converted KP-ABKS scheme is the first proven secure KP-ABKS scheme.

1.3 Paper Organisation

The rest of this paper is organised as follows. Beginning with Sect. 2, we define (P, f)-DBDH problem family and ℓ-DCBDH problem, and prove the hardness of those problems. In Sect. 3, we define LEKS and its security model, followed by the definition of LET and the LEKS conversion from LET. After that, an instance of LEKS conversion is given in Sect. 4, converting a KP-ABE scheme to a KP-ABKS scheme. The resulted KP-ABKS scheme is proven secure in Sect. 4.3 under the security model defined in Sect. 3.2. Finally, the conclusion is addressed in Sect. 5.

2 Decisional Diffie-Hellman Problem Family

In this paper, we use the same bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ as in [9] for simplicity where $\mathbb{G}_1$, $\mathbb{G}_2$ are multiplicative cyclic groups of prime order p and g is a generator of $\mathbb{G}_1$.

Definition 1. *Let $P = (p_1, \dots, p_s) \in \mathbb{F}_p[X_1, \dots, X_n]^s$ be a s-tuples of n-variate polynomial over $\mathbb{F}_p$. We define that a polynomial $f \in \mathbb{F}_p[X_1, \dots, X_n]^s$ is dependent on P if exists $s^2 + 2s$ constants $a_{i,j}$, b_k and c_l such that*

$$f = \frac{\sum_{i=1}^{s}\sum_{j=1}^{s} a_{i,j} p_i p_j}{\sum_{k=1}^{s} b_k p_k} + \sum_{l=1}^{s} c_l p_l \quad \text{or} \quad f = \sum_{l=1}^{s} c_l p_l \pm \sqrt{\sum_{i=1}^{s}\sum_{j=1}^{s} a_{i,j} p_i p_j}$$

Equivalently, f is dependent on P if exists s^2+s+1 constants $a_{i,j}$, b_k and c

$$cf^2 + \sum_{k=1}^{s} b_k p_k f + \sum_{i=1}^{s}\sum_{j=1}^{s} a_{i,j} p_i p_j = 0$$

where at least one of b_k or c is non-zero.

Let $g^{P(x_1,\ldots,x_n)} = (g^{p_1(x_1,\ldots,x_n)}, \ldots, g^{p_s(x_1,\ldots,x_n)})$, d_f denote the total degree of $f \in \mathbb{F}_p[X_1,\ldots,X_n]$, and $d_P = \max\{d_f \mid f \in P \in \mathbb{F}_p[X_1,\ldots,X_n]^s\}$. We present the family of Diffie-Hellman problems as follows.

Definition 2 (Decisional Bilinear (P,f)-Diffie-Hellman problem). *Let $P = (p_1,\ldots,p_s) \in \mathbb{F}_p[X_1,\ldots,X_n]^s$ be a s-tuples of n-variate polynomial over $\mathbb{F}_p$, $f \in \mathbb{F}_p[X_1,\ldots,X_n]$ be a n-variate polynomial over $\mathbb{F}_p$. Let $(x_1,\ldots,x_n) \in_R \mathbb{Z}_p^n$, and $Z \in_R \mathbb{G}_1$. Giving two probability distributions $\mathcal{D}_\tau = (g^{P(x_1,\ldots,x_n)}, g^{f(x_1,\ldots,x_n)})$ and $\mathcal{D}_\rho = (g^{P(x_1,\ldots,x_n)}, Z)$, there is an algorithm $\mathcal{A}$ can distinguish $\mathcal{D}_\tau$ and $\mathcal{D}_\rho$ with advantage:*

$$Adv_{\mathcal{A}}^{(P,f)-DBDH} = \frac{1}{2}\left|\Pr\left[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_\tau)\right] - \Pr\left[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_\rho)\right]\right|$$

where $D \in_R \mathcal{D}$ represents that D is uniformly and independently chosen from $\mathcal{D}$. Alternatively, the problem can be represented as

$$b \in_R \{0,1\}, \quad Z_b = g^{f(x_1,\ldots,x_n)}, \quad Z_{1-b} \in_R \mathbb{G}_1,$$
$$Adv_{\mathcal{A}}^{(P,f)-DBDH} = \left|\Pr\left[b = b' \leftarrow \mathcal{A}(g^{P(x_1,\ldots,x_n)}, Z_0, Z_1)\right] - \frac{1}{2}\right|$$

As from the definition above, Decisional Bilinear (P,f)-Diffie-Hellman ((P,f)-DBDH in short) problem family is an enhanced DDH problem on the group $\mathbb{G}_1$ where the adversary $\mathcal{A}$ is now able to do bilinear pairing operations on $\mathbb{G}_1$. The (P,f)-DBDH problem family is computational hard if and only if the advantage $Adv_{\mathcal{A}}^{(P,f)-DBDH}$ is negligible. Since there is no known proof of the hardness of this problem family, we show the complexity lower bound in the generic bilinear group model [6]. As in [6], we emphasise that a lower bound in generic groups does not imply a lower bound in any specific group.

Theorem 1. *Let $\varepsilon_1, \varepsilon_2 : \mathbb{Z}_p^+ \to \{0,1\}^m$ be two random encodings (injective maps) where $\mathbb{G}_1 = \{\varepsilon_1(x) \mid x \in \mathbb{Z}_p^+\}$, $\mathbb{G}_2 = \{\varepsilon_2(x) \mid x \in \mathbb{Z}_p^+\}$. Let $d = 2 \cdot \max(d_P, d_f)$. If f is not dependent on P, the lower bound of the advantage $Adv^{(P,f)-DBDH}$ of solving the (P,f)-DBDH problem (Definition 2) for the adversary $\mathcal{A}$ is stated as follows with at most $q_{1,\times}$, $q_{2,\times}$ queries to the group operation oracles $\mathcal{O}_\times^1$, $\mathcal{O}_\times^2$ and q_e queries to the bilinear pairing oracle $\mathcal{O}_e : \varepsilon_1 \times \varepsilon_1 \to \varepsilon_2$.*

$$Adv_{\mathcal{A}}^{(P,f)-DBDH} \leq \frac{(q_{1,\times} + q_{2,\times} + q_e + s + 2)^2 d}{2p}$$

Our schemes are based on a dynamic version of the above (P, f)-DBDH problem. To describe and show the hardness of the problem, we begin with the following lemma.

Lemma 1. *Let $P = (p_1, \ldots, p_s), Q = (q_1, \ldots, q_s) \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be two s-tuple of n-variate polynomials over $\mathbb{F}_p$, $f \in \mathbb{F}_p[X_1, \ldots, X_n]$, $O = (P, Q) = (p_1, \ldots, p_s, q_1, \ldots, q_s)$ be a $2s$-tuple of n-variate polynomial. Let T be a variate, $R = (P, QT) = (p_1, \ldots, p_s, q_1T, \ldots, q_sT) = (r_1, \ldots, r_{2s})$ be a $2s$-tuple of $(n+1)$-variate polynomial. If f is not dependent on O, f is not dependent on R.*

Lemma 2. *Let $P = (p_1, \ldots, p_s), Q = (q_1, \ldots, q_s) \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be two s-tuple of n-variate polynomials over $\mathbb{F}_p$, $f \in \mathbb{F}_p[X_1, \ldots, X_n]$, $O = (P, Q) = (p_1, \ldots, p_s, q_1, \ldots, q_s)$ be a $2s$-tuple of n-variate polynomial. Let $T_1, \ldots, T_\ell$ be ℓ variates, $R = (P, QT_1, \ldots, QT_\ell) = (p_1, \ldots, p_s, q_1T_1, \ldots, q_sT_1, \ldots, q_1T_\ell, \ldots, q_sT_\ell)$ be an $(\ell+1)s$-tuple of $(n+\ell)$-variate polynomial. If f is not dependent on O, f is not dependent on R.*

Definition 3 (Decisional ℓ-Combined Bilinear Diffie-Hellman problem). *Let $a, b, c, d, e, f_1, \ldots, f_\ell \in_R \mathbb{Z}_p$, $h = g^e$, and $Z \in_R \mathbb{G}_1$. Giving two probability distributions $\mathcal{D}_{DCBDH} = (g, g^a, g^b, h, h^c, h^d, \{(g^{f_i}, g^{af_i}, h^{f_i}, h^{af_i})\}_{i=1\ldots\ell}, g^{ab}h^{cd})$ and $\mathcal{D}_\rho = (g, g^a, g^b, h, h^c, h^d, \{(g^{f_i}, g^{af_i}, h^{f_i}, h^{af_i})\}_{i=1\ldots\ell}, Z)$, there is an algorithm $\mathcal{A}$ can distinguish $\mathcal{D}_{DCBDH}$ and $\mathcal{D}_\rho$ with advantage:*

$$Adv_{\mathcal{A}}^{\ell\text{-}DCBDH} = \frac{1}{2}\left|\Pr\left[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_{DCBDH})\right] - \Pr\left[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_\rho)\right]\right|$$

Alternatively, the problem can be represented as

$$b \in_R \{0,1\}, \quad Z_b = g^{ab}h^{cd}, \quad Z_{1-b} \in_R \mathbb{G}_1, \quad Adv_{\mathcal{A}}^{\ell\text{-}DCBDH} =$$
$$\left|\Pr\left[b = b' \leftarrow \mathcal{A}(g, g^a, g^b, h, h^c, h^d, \{(g^{f_i}, g^{af_i}, h^{f_i}, h^{af_i})\}_{i=1\ldots\ell}, Z_0, Z_1)\right] - \frac{1}{2}\right|$$

The Decisional ℓ-Combined Bilinear Diffie-Hellman (ℓ-DCBDH) problem belongs to the (P, f)-DBDH problem family. We prove that the ℓ-DCBDH problem is hard by showing the advantage $Adv_{\mathcal{A}}^{\ell\text{-DCBDH}}$ is negligible.

Theorem 2. *The lower bound of the advantage $Adv_{\mathcal{A}}^{\ell\text{-}DCBDH}$ of solving the ℓ-DCBDH problem (Definition 3) for the adversary $\mathcal{A}$ is stated as follows with at most q queries to group operations and bilinear pairing operations.*

$$Adv_{\mathcal{A}}^{\ell\text{-}DCBDH} \leq \frac{3 \cdot (q + 4\ell + 8)^2}{p}$$

Due to the space limitation, the proofs of the above Theorems and Lemmas will be provided in the full version of the paper.

3 Linear Encryption with Keyword Search

3.1 Definition

In general, a searchable encryption scheme involves three roles and consists of two encryption parts. In detail, the roles are **contributor**, **server** and **user**, and the encryption parts are the message encryption part and the keyword encryption part. A general purpose searchable encryption scheme works as follows. Alice, as a **contributor**, encrypts a file using the message encryption scheme and the related keywords using the keyword encryption part for the target **users**, including Bob. Let **header** denote the keyword ciphertext, and **payload** denote the file ciphertext. Since a file may be associated with multiple keywords, Alice may generate multiple headers for the payload. After that, Alice assembles the headers and the payload as a single ciphertext, and sends the ciphertext to the **server**. Bob, as one of the target **user**, can ask the **server** to search the ciphertext with certain keywords. To do secured search, Bob generates a **trapdoor** for each keyword to be searched, and then uploads the trapdoors to the server via a secure communication channel. Once the server receives the query with the trapdoors from Bob, the server begins to test whether the keywords in the headers match those in the trapdoors. Note that the keywords are not visible to the server, and the headers and trapdoors match only when the corresponding keywords are the same and Bob is one of the intended users that the headers are encrypted for. After searching for all related ciphertexts, the server allows Bob to download the matching payloads. Finally, Bob can download the payloads with matching headers. In addition, a **trusted authority** is required in the identity or attribute-based setting.

Formally, we define *Linear Encryption with Keyword Search* as follows, focusing on the keyword encryption part in a general searchable encryption scheme.

Definition 4 (Linear Encryption with Keyword Search). *A linear encryption with keyword search (LEKS) scheme, involving the contributors, the servers, the users and the trusted authority, consists of the following five (probabilistic) polynomial time algorithms:*

- $(MSK, PK) \leftarrow Setup(1^\lambda)$*: The system setup algorithm run by the trusted authority takes a security parameter* 1^λ*, and outputs a pair of master secret key* MSK *and public key* PK *for the trusted authority.*
- $SK \leftarrow KeyGen(MSK, I_S)$*: The user key generation algorithm run by the trusted authority takes a master secret key* MSK *and a user identity* I_S*, and generates a user secret key* SK *for the user associated with that identity.*
- $C \leftarrow LEKS(PK, I_C, W)$*: The keyword encryption algorithm run by the contributor takes a public key* PK*, a target identity* I_C *and a keyword* W*, and outputs a ciphertext* C *of the keyword* W*. To maximum the generality,* I_C *is viewed as a set that the user* I_S *can access the ciphertext only if* $I_S \in I_C$*. It is equivalent to* $F(I_S, I_C) = 1$ *with a predicate function* F*.*
- $T \leftarrow Trapdoor(SK, W)$*: The trapdoor generation algorithm run by the user takes a secret key* SK *and a keyword* W*, and generates a trapdoor* T *of the keyword* W*.*

– $1/0 \leftarrow Test(C, T)$: *The deterministic test algorithm run by the server takes a ciphertext* $C \leftarrow LEKS(PK, I_C, W)$ *and a trapdoor* $T \leftarrow Trapdoor(SK, W)$ *where* $SK \leftarrow KeyGen(MSK, I_S)$, *and outputs*

$$\begin{cases} 1 & if\, W = W' \wedge I_S \in I_C, \\ 0 & otherwise. \end{cases}$$

In the public key scenario where users are identified using public keys generated by themselves, the trusted authority is not required and the algorithm $KeyGen$ is not used. Instead, the *Setup* algorithm is run by individual users, and outputs a pair of secret key SK and public key PK for that user. In addition, the scheme is required to be correct.

Definition 5 (Correctness). *A LEKS scheme is correct if the following statement is always true:*

$$\forall (MSK, PK) \leftarrow Setup(1^\lambda), \forall I_c, \forall W \in \{0,1\}^*, \forall C \leftarrow LEKS(PK, I_c, W),$$
$$\forall I_s \in I_c, \forall SK \leftarrow KeyGen(MSK, I_s), \forall T \leftarrow Trapdoor(SK, W), Test(C, T) = 1.$$

3.2 Security Model

In LEKS, we consider that the server is *honest but curious*. In addition, we do not consider the keyword guessing attack (KGA), since the server can always generate ciphertexts with certain keywords to test with the trapdoor legitimately. However, we can prevent anyone from extracting the keyword directly from the trapdoor by applying an one-way function such as a preimage-resistant hash function.

We present two security games: *Indistinguishability under Adaptive Chosen Keyword Attack* (IND-CKA) and its weaker *Selective-ID* version (IND-sCKA). We first define the IND-sCKA game (Game 1) where an adaptive adversary $\mathcal{A}$ tries to distinguish a ciphertext generated from either keywords W_0 or W_1:

1. $\mathcal{A}$ selects a target identity set I_T and submits it to the challenger $\mathcal{S}$.
2. $\mathcal{S}$ runs $Setup(1^\lambda)$ to generate a key pair (MSK, PK) and passes PK to $\mathcal{A}$.
3. $\mathcal{A}$ can adaptively ask $\mathcal{S}$ for the secret key SK of the user with identity I by querying the key generation oracle $\mathcal{O}_{KeyGen}$. At the same point, $\mathcal{S}$ records I in the identity list $\mathcal{I}$. The restriction is that I must not be in I_T.
4. $\mathcal{A}$ can adaptively ask $\mathcal{S}$ for the trapdoor T of the user identity I with the keyword W by querying the trapdoor generation oracle $\mathcal{O}_{Trapdoor}$. If I is not in I_T, it can be resolved that $\mathcal{A}$ queries the oracle $\mathcal{O}_{KeyGen}$ to obtain the secret key SK of I and further obtains the trapdoor $T \leftarrow Trapdoor(SK, W)$. Otherwise, $\mathcal{S}$ runs the algorithm $KeyGen$ and then the algorithm $Trapdoor$ to get the trapdoor, and passes it to $\mathcal{A}$. At the same point, $\mathcal{S}$ records the queried keyword W in the keyword list $\mathcal{W}$.
5. At some point, $\mathcal{A}$ outputs two keywords W_0 and W_1 to be challenged where those two keywords must not be in the keyword list $\mathcal{W}$.

6. $\mathcal{S}$ randomly selects b to be either 0 or 1 uniformly. Then $\mathcal{S}$ generates a ciphertext $C \leftarrow LEKS(PK, I_T, W_b)$ and passes it to $\mathcal{A}$.
7. $\mathcal{A}$ can continue to query all oracles with the same restriction. In addition, $\mathcal{A}$ cannot query the target keywords W_0 and W_1 to the oracle $\mathcal{O}_{Trapdoor}$.
8. Eventually, $\mathcal{A}$ outputs a bit b'. $\mathcal{A}$ wins the game if $b = b'$.

We define the advantage of winning Game 1 as follows

$$Adv_{\mathcal{A}}^{\text{IND-sCKA}} = \left| \Pr\left[b = b' \wedge \mathcal{I} \cap I_T = \emptyset \wedge W_0, W_1 \notin \mathcal{W}\right] - \frac{1}{2} \right|$$

Definition 6 (IND-sCKA Security). *A LEKS scheme is Indistinguishable under Selective-ID Adaptive Chosen Keyword Attack if $Adv_{\mathcal{A}}^{IND\text{-}sCKA}$ is a negligible function for all adversary $\mathcal{A}$ winning the Game 1 in polynomial time.*

Next, we define the IND-CKA game (Game 2), which is similar as the IND-sCKA game. The difference is that $\mathcal{A}$ is given the public key PK in IND-CKA before submitting the target identity set I_T.

Definition 7 (IND-CKA Security). *A LEKS scheme is Indistinguishable under Adaptive Chosen Keyword Attack if $Adv_{\mathcal{A}}^{IND\text{-}CKA}$ is a negligible function for all adversary $\mathcal{A}$ winning the Game 2 in polynomial time.*

$$\begin{aligned}
&Game_{\text{IND-sCKA}}^{\lambda}: \\
&\quad \mathcal{I}, \mathcal{W} \leftarrow \emptyset, \quad I_T \leftarrow \mathcal{A}, \quad (MSK, PK) \leftarrow Setup(1^{\lambda}), \\
&\quad (W_0, W_1) \leftarrow \mathcal{A}^{\mathcal{O}_{KeyGen}, \mathcal{O}_{Trapdoor}}(PK), \quad b \in_R \{0,1\}, \\
&\quad C \leftarrow LEKS(PK, I_T, W_b), \quad b' \leftarrow \mathcal{A}^{\mathcal{O}_{KeyGen}, \mathcal{O}_{Trapdoor}}(C) \\
&\mathcal{O}_{KeyGen}: \quad \mathcal{I} \leftarrow \mathcal{I} \cup \{I\}, \quad \text{return } SK \leftarrow KeyGen(MSK, I) \\
&\mathcal{O}_{Trapdoor}: \quad \mathcal{W} \leftarrow \mathcal{W} \cup \{W\}, \quad \text{return } T \leftarrow Trapdoor(SK, W) \\
&Adv_{\mathcal{A}}^{\text{IND-sCKA}} = \left| \Pr\left[b = b' \wedge \mathcal{I} \cap I_T = \emptyset \wedge W_0, W_1 \notin \mathcal{W}\right] - \frac{1}{2} \right|
\end{aligned}$$

Game 1: IND-sCKA

$$\begin{aligned}
&Game_{\text{IND-CKA}}^{\lambda}: \\
&\quad \mathcal{I}, \mathcal{W} \leftarrow \emptyset, \quad (MSK, PK) \leftarrow Setup(1^{\lambda}), \\
&\quad (I_T, W_0, W_1) \leftarrow \mathcal{A}^{\mathcal{O}_{KeyGen}, \mathcal{O}_{Trapdoor}}(PK), \quad b \in_R \{0,1\}, \\
&\quad C \leftarrow LEKS(PK, I_T, W_b), \quad b' \leftarrow \mathcal{A}^{\mathcal{O}_{KeyGen}, \mathcal{O}_{Trapdoor}}(C) \\
&Adv_{\mathcal{A}}^{\text{IND-CKA}} = \left| \Pr\left[b = b' \wedge \mathcal{I} \cap I_T = \emptyset \wedge W_0, W_1 \notin \mathcal{W}\right] - \frac{1}{2} \right|
\end{aligned}$$

Game 2: IND-CKA

3.3 Linear Encryption Template

In this subsection, we define the *Linear Encryption Template* (LET). Informally, a LET models an asymmetric encryption scheme, consisting of the **senders**, the **recipients** and the **trusted authority**. Alice, as the **recipient**, gets her secret key from the **trusted authority** using her identity where her public key is her identity. If LET is modelling a PKE scheme, Alice's secret/public key pair is generated by herself, and the **trusted authority** is not required. To securely send a message to a set of recipients, including Alice, the **sender** Bob encrypts the message into a ciphertext, and sends it to Alice. Once Alice receives the ciphertext, she can decrypts and obtains the original message if and only if she is one of the target recipients. Furthermore, if an encryption scheme fits LET, we can use it to construct the corresponding LEKS scheme in Sect. 3.4. Formally, we describe the definition of *Linear Encryption Template* as follows.

Definition 8 (Linear Encryption Template). *A linear encryption template, involving the senders, the recipients, and the trusted authority, consists of the following four (probabilistic) polynomial algorithms:*

- $(MSK, PK) \leftarrow Setup(params, \alpha)$*: The system setup algorithm run by the trusted authority takes a set of system parameters, such as the description of groups, security parameters and randomnesses, and it reuses these parameters. The algorithm also takes a component* α*, which is used to create the ciphertext. The output of this algorithm is a pair of master secret key* MSK *and public key* PK *of the trusted authority.*
- $SK \leftarrow KeyGen(MSK, I_S)$*: The user key generation algorithm run by the trusted authority takes a master secret key* MSK *and a user identity* I_S*, and generates a user secret key* SK *for the user associated with that identity.*
- $C \leftarrow Encrypt(PK, I_C, M, s)$*: The encryption algorithm run by the sender takes a public key* PK*, a target identity set* I_C*, a message* M *and a randomness* s*, and outputs a ciphertext* C *of the message* M*. The randomness* s *is used to bind the ciphertext parts in* C *and further to bind other ciphertext parts when constructing LEKS schemes. It is required that the ciphertext must be in the form of* $C = (C_0, C_1, \dots)$ *where* $C_0 = M \cdot e(g,g)^{\alpha s}$*.*
- $M \leftarrow Decrypt(SK, C)$*: The deterministic decryption algorithm run by the recipient takes a secret key* SK *and a ciphertext* C*, and outputs the original message* M*. The decryption process is required to be two steps. The first step is to run the sub-decryption algorithm* $\mathcal{D}$ *to get* $e(g,g)^{\alpha s} \leftarrow \mathcal{D}(SK, C_1, \dots)$*. Then the second step is to extract the message* $M = \frac{C_0}{e(g,g)^{\alpha s}}$*. Importantly, the sub algorithm* $\mathcal{D}$ *is required to have* ***linearity****:*

$$\forall t \in \mathbb{Z}_p, \quad \mathcal{D}(SK^t, C_1, \dots) = D(SK, C_1, \dots)^t$$

If SK *consists multiple elements that* $SK = (SK_1, SK_2, \dots)$*, the term* SK^t *denotes* $(SK_1^t, SK_2^t, \dots)$*.*

If there is no trusted authority that users generate their key pairs by themselves, the algorithm $KeyGen$ is not used and the algorithm $Setup$ is run by the user, outputing a pair of user secret key SK and public key SK. In addition, the scheme is required to be correct.

Definition 9 (Correctness). *A LET scheme is correct if the following statement is always true:*

$$\forall (MSK, PK) \leftarrow Setup(params, \alpha), \forall I_c, \forall I_s \in I_c, \forall SK \leftarrow KeyGen(MSK, I_s),$$
$$\forall M \in \mathbb{G}_2, \quad \forall s \in \mathbb{Z}_p, \quad \forall C \leftarrow Encrypt(PK, I_c, M, s), \quad Decrypt(SK, C) = M.$$

3.4 Keyword Search from Linear Encryption Template

In this subsection, we build our LEKS scheme with from a LET scheme as the keyword encryption part. To construct a fully searchable encryption scheme, we can reuse the LET scheme as the message encryption part, and combine with the LEKS scheme. Alternatively, we also can use other encryption schemes as the message encryption part. The main idea of the construction is to use the LET part for authentication and combine it with a keyword equality test with the same randomness. Let $\Pi = (Setup, KeyGen, Encrypt, Decrypt)$ be a LET modelled encryption scheme. Our LEKS scheme works as follows.

- $(MSK, PK) \leftarrow Setup(1^\lambda)$: Given a security parameter 1^λ, the algorithm generates two groups $\mathbb{G}_1$, $\mathbb{G}_2$ of prime order p, and specifies a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. The algorithm also selects a random generator g of $\mathbb{G}_1$, and a preimage resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_1$, which may be modelled as an random oracle. After that, the algorithm chooses two randomness $x_1, x_2 \in_R \mathbb{Z}_p^+$, and calculates $g_1 = g^{x_1}$ and $g_2 = g^{x_2}$. Then the algorithm packs all above elements into $params$, sets $\alpha = x_1 x_2$, and passes to the algorithm $\Pi.Setup$ to obtain the key pair $\Pi.MSK$ and $\Pi.PK$. Finally, the algorithm keeps the master secret key $MSK = \Pi.MSK$, and publishes the public key $PK = (\mathbb{G}_1, \mathbb{G}_2, e, g, g_1, g_2, \Pi.PK)$.

$$\mathbb{G}_1 = \langle g \rangle, \quad e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2, \quad H : \{0,1\}^* \rightarrow \mathbb{G}_1, \quad x_1, x_2 \in_R \mathbb{Z}_p^+,$$
$$g_1 = g^{x_1}, \quad g_2 = g^{x_2}, \quad params = (\mathbb{G}_1, \mathbb{G}_2, e, g, H, x_1, x_2),$$
$$(\Pi.MSK, \Pi.PK) \leftarrow \Pi.Setup(params, x_1 x_1)$$

  return $(MSK, PK) = (\Pi.MSK, (\mathbb{G}_1, \mathbb{G}_2, e, g, g_1, g_2, H, \Pi.PK))$.
- $SK \leftarrow KeyGen(MSK, I_S)$: For key generation, the algorithm $\Pi.KeyGen$ is directly invoked. return $SK \leftarrow \Pi.KeyGen(MSK, I_S)$.
- $C \leftarrow LEKS(PK, I_C, W)$: To encrypt a keyword W for a target identity set I_C, the algorithm chooses two randomness $r_1, r_2 \in_R \mathbb{Z}_p^+$. Then it computes $C_1' = g_2^{r_2} H(W)^{r_1}$ and $C_2' = g_1^{r_1}$ to encrypt the keyword W. After that, the algorithm invokes $\Pi.Encrypt$ with r_2 to get the ciphertext $(C_0, C_1, \dots)$ to assure the target identity set I_C. Finally, the algorithm assembles two parts together $C = (C_1', C_2', C_1, \dots)$ as the full ciphertext bound using r_2 where

$C_0 = M \cdot e(g,g)^{x_1 x_2 r_2}$ is dropped. Since C_0 is not used in C, we can safely setting the message M to 0 when invoking $\Pi.Encrypt$.

$$r_1, r_2 \in_R \mathbb{Z}_p^+, \quad C_1' = g_2^{r_2} H(W)^{r_1}, \quad C_2' = g_1^{r_1}$$
$$(C_0, C_1, \dots) \leftarrow \Pi.Encrypt(PK, I_C, 0, r_2)$$

return $C = (C_1', C_2', C_1, \dots)$.

– $T \leftarrow Trapdoor(SK, W)$: To generate a trapdoor of the keyword W, the algorithm selects a randomness $s \in_R \mathbb{Z}_p^+$. Then it calculates $T = (T_1, T_2, T_3)$ where $T_1 = g_1^s$, $T_2 = H(W)^s$ and $T_3 = SK^s$. For SK^s, the operation works the same as in Definition 8.

$$s \in_R \mathbb{Z}_p^+, \quad T_1 = g_1^s, \quad T_2 = H(W)^s, \quad T_3 = SK^s$$

return $T = (T_1, T_2, T_3)$.

– $1/0 \leftarrow Test(C, T)$: For equality tests of both the keyword and the identity, the algorithm tests the equality of the following return statement.

return $e(C_1', T_1)/e(C_2', T_2) \stackrel{?}{=} \Pi.\mathcal{D}(T_3, C_1, \dots)$.

Theorem 3. *The proposed conversion from the LET scheme to the LEKS scheme is correct if the corresponding encryption scheme modelled by LET is correct.*

Proof. To verify, we calculate the left hand side of the test equation first.

$$E_1 = \frac{e(C_1', T_1)}{e(C_2', T_2)} = \frac{e(g_2^{r_2} H(W)^{r_1}, g_1^s)}{e(g_1^{r_1}, H(W)^s)} = \frac{e(g_2^{r_2}, g_1^s) \cdot e(H(W)^{r_1}, g_1^s)}{e(g_1^{r_1}, H(W)^s)} = e(g_1, g_2)^{r_2 s}$$

Then we calculate the right hand side of the test equation.

$$\begin{aligned} E_2 &= \Pi.\mathcal{D}(T_3, C_1, \dots) = \Pi.\mathcal{D}(SK^s, C_1, \dots) = \Pi.\mathcal{D}(SK, C_1, \dots)^s \\ &= e(g,g)^{x_1 x_2 r_2 s} = e(g_1, g_2)^{r_2 s} \end{aligned}$$

As $E_1 = E_2$, the correctness is proved.

However, we are uncertain about the security of the above construction, since some components are shared outside the encryption Π that may break the security of Π in its original model. Therefore, we require individual security proof for each conversion to ensure the security.

4 Key-Policy Attribute-Based Keyword Search

In this section, we show a useful instance of our LEKS conversion by converting a KP-ABE into a KP-ABKS scheme. We starts with an ABE scheme [15] which is a variant of Goyal et al.'s scheme [12] while the function T defined in [12] is replaced with a random oracle. Then we convert it into a LEKS scheme by the method in Sect. 3.4. Finally, we prove the resulted LEKS scheme is IND-sCKA secure in random oracle model.

4.1 Base Scheme

The ABE scheme [15] modelled by LET works as follows.

- $(MSK, PK) \leftarrow Setup(params, \alpha)$: The system setup algorithm reuses the parameters $params$, $g_1 = g^{x_1}$, and $g_2 = g^{x_2}$. The master secret key is $y = x_1$. Since the function T is replaced with a random oracle, the algorithm is required to choose a cryptographic hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_1$. Return $(MSK, PK) = (x_1, (g_1, g_2, H))$.
- $SK \leftarrow KeyGen(MSK, I_S)$: In KP-ABE, the user identity set I_S is the policy modelled as an access tree $\mathcal{T}$ (details in [12]). The algorithm chooses a random polynomial q_x for each non-leaf node $x \in \mathcal{T}$ in a top-down manner. For each non-leaf node x, the degree d_x of the polynomial q_x is $d_x = k_x - 1$ where k_x is the threshold value of that node. For the root node, the algorithm sets $q_{\text{root}}(0) = x_1$. For other nodes, the algorithm sets $q_x(0) = q_{\text{parent}(x)}(\text{index}(x))$. With polynomials for the access tree $\mathcal{T}$ is decided, the algorithm generates the secret key components for the user. For each leaf node x, the algorithm chooses a random number $r_x \in_R \mathbb{Z}_p^+$, and calculates $D_x = g_2^{q_x(0)} H(attr(x))^{r_x}, R_x = g^{r_x}$. Return $SK = (\mathcal{T}, \{(D_x, R_x)\}_{x \in \text{leaves}(\mathcal{T})})$.
- $C \leftarrow Encrypt(PK, I_C, M, t)$: In KP-ABE, the target identity set I_C is the attributes γ. To encrypt, the algorithm calculates $C_0 = M \cdot e(g_1, g_2)^t, C_1 = g^t$, $C_2 = \gamma$. For each attribute $attr_i \in \gamma$, the algorithm computes $C_i = H(attr_i)^t$. As required by LET, we note that $C_0 = M \cdot e(g_1, g_2)^t = e(g^{x_1}, g^{x_2})^t = e(g, g)^{x_1 x_2 t} = e(g, g)^{\alpha t}$. Return $C = (C_0, C_1, C_2, \{C_i\}_{attr_i \in \gamma})$.
- $M \leftarrow Decrypt(SK, C)$: At first, the algorithm checks whether $\mathcal{T}(\gamma) = 1$ or not. If the attributes do not match the policy that $\mathcal{T}(\gamma) = 0$, the algorithm returns $\perp$. Otherwise, the algorithm proceeds the sub-algorithm $\mathcal{D}$ as follows. For those matching attributes $attr_i = attr(x)$, where $attr_i \in \gamma$ and leaf node $x \in \mathcal{T}$, the algorithm can decrypt that node by calculating

$$F_x = \frac{e(D_x, C_1)}{e(R_x, C_i)} = \frac{e(g_2^{q_x(0)} H(attr(x))^{r_x}, g^t)}{e(g^{r_x}, H(attr_i)^t)} = e(g, g_2)^{t \cdot q_x(0)}$$

Then the algorithm can decrypt the non-leaf node $x \in \mathcal{T}$ by using polynomial interpolation. Let S_x be the child set of the node x.

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S_x}(0)} = \prod_{z \in S_x} \left(e(g, g_2)^{t \cdot q_z(0)} \right)^{\Delta_{i,S_x}(0)} \\ &= e(g, g_2)^{t \cdot \sum_{z \in S_x} q_z(0) \cdot \Delta_{i,S_x}(0)} = e(g, g_2)^{t \cdot q_x(0)} \end{aligned}$$

Since $\mathcal{T}(\gamma) = 1$, the algorithm can decrypt the root node that

$$F_{root} = e(g, g_2)^{t \cdot q_x(0)} = e(g, g_2)^{x_1 t} = e(g, g)^{x_1 x_2 t} = e(g, g)^{\alpha t}$$

The algorithm sets F_{root} as the output of sub-algorithm $\mathcal{D}$. Finally, the algorithm computes the message $M = C_0 / F_{root}$ and returns M.

The correctness has been shown in the description of the decryption algorithm. We also show that the above scheme has the linearity property required by LET.

Theorem 4 (Correctness). *The above KP-ABE scheme is correct.*

Theorem 5 (Linearity). *The sub-algorithm $\mathcal{D}$ has linearity that*

$$\forall s \in \mathbb{Z}_p, \mathcal{D}(SK^s, C_1, C_2, \{C_i\}_{attr_i \in \gamma}) = \mathcal{D}(SK, C_1, C_2, \{C_i\}_{attr_i \in \gamma})^s$$

Proof. For the decryption of leaf nodes, the computation becomes

$$F'_x = \frac{e(D_x^s, C_1)}{e(R_x^s, C_i)} = \left(\frac{e(D_x, C_1)}{e(R_x, C_i)}\right)^s = F_x^s.$$

For the decryption of non-leaf nodes, the computation becomes

$$F'_x = \prod_{z \in S_x} {F'_z}^{\Delta_{i,S_x}(0)} = \prod_{z \in S_x} F_z^{s\Delta_{i,S_x}(0)} = \left(\prod_{z \in S_x} F_z^{\Delta_{i,S_x}(0)}\right)^s = F_x^s$$

Thus $F'_{root} = F^s_{root}$.

4.2 Construction from the Base Scheme

In this section, we apply the LEKS conversion as follows with some key notes.

- $(MSK, PK) \leftarrow Setup(1^\lambda)$: Although the hash functions in the LEKS and the KP-ABE schemes have the same domain and codomain, they cannot be merged since they will be programmed into two different random oracles.

$$\mathbb{G}_1 = \langle g \rangle, \quad e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2, \quad H_1 : \{0,1\}^* \to \mathbb{G}_1, \quad H_2 : \{0,1\}^* \to \mathbb{G}_1,$$
$$x_1, x_2 \in_R \mathbb{Z}_p^+, \quad g_1 = g^{x_1}, \quad g_2 = g^{x_2}$$

  return $(MSK, PK) = (x_1, (\mathbb{G}_1, \mathbb{G}_2, e, g, g_1, g_2, H_1, H_2))$.
- $SK \leftarrow KeyGen(MSK, I_S)$:

$$\forall x \in \text{leaves}(\mathcal{T}), \quad r_x \in_R \mathbb{Z}_p^+, \quad D_x = g_2^{q_x(0)} H_2(attr(x))^{r_x}, \quad R_x = g^{r_x}$$

  return $SK = (\mathcal{T}, \{(D_x, R_x)\}_{x \in \text{leaves}(\mathcal{T})})$.
- $C \leftarrow LEKS(PK, I_C, W)$:

$$r_1, r_2 \in_R \mathbb{Z}_p^+, \quad C_1 = g_2^{r_2} H_1(W)^{r_1}, \quad C_2 = g_1^{r_1}, \quad C_3 = g^{r_2}$$

  return $C = (C_1, C_2, C_3, \gamma, \{C_i = H_2(attr_i)^{r_2}\}_{attr_i \in \gamma})$.
- $T \leftarrow Trapdoor(SK, W)$:

$$s \in_R \mathbb{Z}_p^+, \quad T_1 = g_1^s, \quad T_2 = H_1(W)^s, \quad \{T_{x,1} = D_x^s, T_{x,2} = R_x^s\}_{\forall x \in \text{leaves}(\mathcal{T})}$$

  return $T = (T_1, T_2, \mathcal{T}, \{(T_{x,1}, T_{x,2})\}_{x \in \text{leaves}(\mathcal{T})})$.
- $1/0 \leftarrow Test(C, T)$: The algorithm follows the decryption algorithm in the KP-ABE scheme. If $\mathcal{T}(\gamma) = 0$, the algorithm returns $\perp$. Otherwise, for leaf node $x \in \mathcal{T}$, it computes $F_x = e(T_{x,1}, C_1)/e(T_{x,2}, C_i)$. For non-leaf node, it computes exactly the same as in the decryption algorithm using polynomial interpolation. Eventually, the algorithm computes F_{root} and returns $e(C_1, T_1) \stackrel{?}{=} e(C_2, T_2) \cdot F_{root}$.

4.3 Security Proof

The above converted KP-ABKS scheme is similar to Zheng et al.'s KP-ABKS scheme [21]. The only difference between two schemes is that they use $g_2 g^{b \cdot H(W)}$ as the hash function for the attributes while we use $H_2 : \{0,1\}^* \to \mathbb{G}_1$.

However, there are some issues in the security proof given in [21]. Before the simulation provided by the challenger, the adversary selects a target set of attributes $Attr^*$. In the simulation, the adversary is allowed to query the token generation oracle $\mathcal{O}_{\mathrm{TokenGen}}(T, W)$ with any keyword W other than the target keywords w_0, w_1 and any policy T that $F(Attr^*, T) = 1$. Stepping into the oracle $\mathcal{O}_{\mathrm{TokenGen}}(T, W)$, the challenger always runs the key generation oracle $\mathcal{O}_{\mathrm{KeyGen}}(T)$ to get the secret key sk, and then uses it to generate the requested trapdoor. Since the oracle $\mathcal{O}_{\mathrm{KeyGen}}(T)$ always aborts when $F(Attr^*, T) = 1$, the oracle $\mathcal{O}_{\mathrm{TokenGen}}(T, W)$ always aborts when the adversary does the queries mentioned above. This renders the proof invalid and hence the security of Zheng et al.'s KP-ABKS scheme is unknown.

We prove our KP-ABKS is secure under the ℓ-DCBDH assumption instead of the standard Decisional Linear Assumption (DLIN).

Theorem 6. *The proposed KP-ABKS is IND-sCKA (Definition 6) secure. If an adversary $\mathcal{A}$ can win Game 1 with the advantage ε, an algorithm $\mathcal{S}$ can be constructed to solve ℓ-DCBDH problem (Definition 3) in polynomial time with the advantage $\varepsilon' \geq \frac{\varepsilon}{2e(q+2)}$, querying $\mathcal{O}_{Trapdoor}$ for at most q times where $q \leq \ell$.*

Due to the space limit, the proof will be provided to the full version of the paper.

5 Conclusion

In this paper, we introduced a (P, f)-DBDH problem family and demonstrated its hardness under the generic bilinear group model. We also derived a hard computational ℓ-DCBDH problem from the (P, f)-DBDH problem family. As the main contribution of this paper, we proposed LEKS and its security model, and defined LET which can be used to convert encryption schemes into the corresponding LEKS schemes. To show a concrete instance of our LEKS conversion framework, we converted a KP-ABE scheme into a KP-ABKS scheme and proved its security in the random oracle model under the ℓ-DCBDH assumption. Our future work will be finding more LET-compatible encryption schemes, converting them into searchable schemes and proving their security.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. J. Cryptology **21**(3), 350–391 (2008)

2. Attrapadung, N., Furukawa, J., Imai, H.: Forward-secure and searchable broadcast encryption with short ciphertexts and private keys. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 161–177. Springer, Heidelberg (2006)
3. Attrapadung, N., Libert, B., de Panafieu, E.: Expressive key-policy attribute-based encryption with constant-size ciphertexts. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 90–108. Springer, Heidelberg (2011)
4. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, SP 2007, pp. 321–334, May 2007
5. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. J. Cryptology **21**(2), 149–177 (2007). http://dx.org/10.1007/s00145-007-9005-7
6. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005). doi:10.1007/11426639_26
7. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004). doi:10.1007/978-3-540-28628-8_3
8. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
9. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
10. Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (Without Random Oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006). doi:10.1007/11818175_17
11. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976)
12. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, NY, USA, pp. 89–98. (2006). http://doi.acm.org/10.1145/1180405.1180418
13. Hwang, Y.-H., Lee, P.J.: Public key encryption with conjunctive keyword search and its extension to a multi-user system. In: Takagi, T., Okamoto, E., Okamoto, T., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 2–22. Springer, Heidelberg (2007)
14. Mell, P., Grance, T.: The nist definition of cloud computiing. Technical report, National Institue of Standards and Technology (2011)
15. Pirretti, M., Traynor, P., McDaniel, P., Waters, B.: Secure attribute-based systems. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, NY, USA, pp. 99–112 (2006). http://doi.acm.org/10.1145/1180405.1180419
16. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
17. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
18. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997). doi:10.1007/3-540-69053-0_18

19. Sun, W., Yu, S., Lou, W., Hou, T., Li, H.: Protecting your right: verifiable attribute-based keyword search with fine-grainedowner-enforced search authorization in the cloud. IEEE Trans. Parallel Distrib. Syst. **PP**(99), 1 (2014)
20. Waters, B.: Functional encryption: origins and recent developments. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 51–54. Springer, Heidelberg (2013)
21. Zheng, Q., Xu, S., Ateniese, G.: Vabks: verifiable attribute-based keyword search over outsourced encrypted data. In: 2014 Proceedings IEEE INFOCOM, pp. 522–530, April 2014