CLKS: Certificateless Keyword Search on Encrypted Data

Qingji Zheng [†], Xiangxue Li [‡], and Aytac Azgin [†]

[†] Huawei Research Center, Santa Clara, CA, USA, 95050
[†] {qingji.zheng, aytac.azgin}@huawei.com
[‡] Dept. of Computer Science & Technology, East China Normal University, China
[‡]xxli@cs.ecnu.edu.cn

Abstract. Keyword search on encrypted data enables one to search keyword ciphertexts without compromising keyword security. We further investigate this problem and propose a novel variant, dubbed *certificateless keyword search on encrypted data* (CLKS). CLKS not only supports keyword search on encrypted data, but also brings promising features due to the certificateless cryptography. In contrast to the certificated-based keyword search, CLKS requires no validation on the trustworthy of the public key before encrypting keywords; in contrast to the identity-based keyword search, CLKS prevents the key issuer (e.g., key generator center) from penetrating any information on keyword ciphertexts by leveraging the capability of accessing all data users' (partial) private keys. Specifically, we rigorously define the syntax and security definitions for CLKS, and present the construction that is provably secure in the standard model under the Decisional Linear assumption. We implemented the proposed CLKS scheme and evaluated its performance. To the best of our knowledge, this is the *first* attempt to integrate certificateless cryptography with keyword search on encrypted data.

Keywords: keyword search, certificateless cryptography

1 Introduction

Cloud computing enables data owners to outsource their data to the cloud at affordable price and access/share the outsourced data with other users. The outsourcing, however, separates the data ownership and its (physical) storage ownership and brings the security concern [18, 20, 33] such as data privacy. Though it is natural for data owners to encrypt their own data before outsourcing, the encryption operation makes some useful functions, such as keyword search, become infeasible. Fortunately, the subject of keyword search on encrypted data has been extensively studied and a large number of solutions have been proposed. Roughly speaking, keyword search on encrypted data can be divided into three categories according to the key distribution/generation setting: symmetric key based keyword search, certificate-based keyword search[1] and identity-based (or attribute-based) keyword search[2]. Comparing with symmetric key based keyword search, the last two kinds of keyword search are more flexible and promising because they do not require any key sharing between the data owner and the target user who the data will be shared with.

For certificate-based keyword search, when the data owner wants to share the data with a target user, he will encrypt the data with the target user's public key. Note that before this operation, the data owner needs to validate the certificate which binds the target user's identity and the corresponding public key in order to assure that the public key is really associated with the target user, which needs to rely on some trust/certificate management system such as PKI. In addition, the data owner might need to conduct costly certificate chain verification (until finding a certificate authority he trusts) for certificate validation. In order to mitigate this downside, identity-based (attribute-based) keyword search [31, 34, 28] is introduced where the public key is exactly the same as the target user's identity (attributes) and therefore there is no need to validate the correctness of the public key. Despite its benefits, identity-based (attribute-based) keyword search suffers from the key escrow problem that the key generation authority has full access to all data users' private keys.

Contribution. We propose *certificateless keyword search on encrypted data* (CLKS), which preserves the merits of identity-based keyword search (e.g., no certificate management and validation) without inherent key escrow problem. To be specific, we follow the basic principle underlying the certificateless cryptography: treating the user's identity as part of the public key and letting the user' private key consisting of two components, one generated by the key generation center and the other chosen by the user. Thus, CLKS allows users to put less trust on the key generation center. We summarize our contribution as follows and compare three kinds of keyword search solutions in Table 1:

- We first integrate the certificateless cryptography with keyword search on encrypted data. We formalize the notion for CLKS, and rigorously define its security properties.
- We present a CLKS scheme that is provably secure in the standard model under the Decisional Linear assumption. Similar to other certificateless primitives, the proposed certificateless keyword search scheme leverages the identity as user's partial public key and eliminates the key escrow problem. We implemented the proposed scheme and conducted the performance evaluation on real data to show its feasibility.

Organization. Section 2 describes the related work. Section 3 presents cryptographic assumptions and primitives. Section 4 presents the system and threat

[1] Certificate-based keyword search means keyword search on encrypted data in traditional public key setting, where users generate the public/private by themselves and the certificate is used to bind the user identity and the public key.

[2] Attribute-based keyword search can be treated as the generalized version of identity-based keyword search. We put them together because of the same key generation manner.

Table 1. Comparison of three kinds of keyword search primitives that are involved with the public/private keys.

Type of keyword search	Who generates user's private key	Required trust Model/Infrastructure
Certificate-based	User	Certificate binds the user identity and public key
Identity(attribute)-based	Third party authority	Fully trusted third party authority
Certificateless (our paper)	User and third party authority	Honest-but-curious third party authority

model, and Section 5 formalizes the syntax and security definitions. Section 6 presents the main construction Section 7 presents the performance.

2 Related Work

We briefly review the relevant techniques on keyword search on encrypted data, which are separated into three categories as follows:
Symmetric key based keyword search. It allows the data owner to encrypt keywords and the corresponding data, and outsource keyword ciphertexts and data ciphertexts to the remote server. Only the data owner, or someone with the symmetric key, can generate the search token in order to ask the cloud to conduct search on keyword ciphertexts. [27] proposed the first symmetric key based keyword search scheme. Many variants, e.g., [17, 12, 14, 22, 23, 11, 8] have been proposed with various features, for example, improved security [14], dynamic support [22], UC security [23], verifiability [11] and multi-user sharing [21]. The main advantage of the symmetric key based keyword search is its high efficiency since it does not involve any costly public key operations (e.g. exponentiation, pairing). In the data sharing scenario, however, the technique requires the data owner and the target user (or remote server) sharing some common secret [21].
Certificate-based keyword search. For certificate-based keyword search, the data owner encrypts keywords and data with the public key of the target user, so that the target user can use his own private key to generate the search token and then conduct the search on keyword ciphertexts. Many solutions, e.g., [5, 4, 7, 1, 9, 10, 30, 26, 29], have been proposed after [6] initiated the first study. While certificate-based keyword search is more flexible compared with symmetric key based keyword search, it requires the data owner validating the target user's public key before encrypting keywords. The trust and certificate management infrastructure, e.g., PKI, has been introduced to facilitate the validation process.
Identity-based (attribute-based) keyword search. For identity-based (attribute-based) keyword search, the data owner encrypts the keyword and data with the target user's identity (or attributes in attribute-based keyword search). The target user, after acquiring the private key from the key generation authority, can generate the search token and then conduct search on keyword ciphertexts. [31] presented the first identity-based keyword search scheme and [34, 28] independently introduced the attribute-based keyword search. In this setting, data owner does not need to validate whether the used public key is associated with the target user or not, since the public key is exactly the same as the user's identity

or attributes. While it mitigates the certificate management issue, it indeed introduces another issue – key escrow problem, since the key generation authority can access all users' private keys.

3 Preliminaries

Let $x \overset{R}{\leftarrow} X$ denote selecting an element x from the set X uniformly at random. Let $\mathcal{I}$ be the user universe where each user is associated to a unique identity id.

Let $(e, p, g, G, G_T) \leftarrow \mathsf{BMP}(1^\ell)$ be the function that generates a bilinear map $e : G \times G \rightarrow G_T$ by taking as input a security parameter ℓ, such that p is an ℓ-bit prime, G and G_T are two cyclic groups of prime order p and g is a random generator of group G. The bilinear map e should satisfy (i) $\forall a, b \in \mathbb{Z}_p$, $e(g^a, g^b) = e(g, g)^{ab}$; (ii) $e(g, g) \neq 1$ and (iii) e can be computed efficiently.

We assume that the identities are distinct n-bit numbers. Let $H_1 : \{0,1\}^n \rightarrow G$, $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$, where H_1 is a function mapping id to an element in G (defined as in section 6) and H_2 is modeled as a collision-resistant hash function.
Decisional Linear (DL) Assumption Let $(e, p, g, G, G_T) \leftarrow \mathsf{BMP}(1^\ell)$. Given $g, h, f, Q \overset{R}{\leftarrow} G$ and g^{r_2}, h^{r_1}, where $r_1, r_2 \overset{R}{\leftarrow} \mathbb{Z}_p^*$ are unknown, the DL assumption states that any probabilistic polynomial-time algorithm $\mathcal{A}$ can determine whether $Q = f^{r_1+r_2}$ or not at most with a negligible advantage with respect to the security parameter ℓ, where the "advantage" is defined as

$$\mathsf{Adv}_{DL}(\ell) = |\Pr[\mathcal{A}(g, h, f, g^{r_2}, h^{r_1}, f^{r_1+r_2}) = 1] - \Pr[\mathcal{A}(g, h, f, g^{r_2}, h^{r_1}, Q) = 1]|.$$

4 System and Threat Model

System model We consider the system model as shown in Figure 1, consisting of three entities: the key generation center KGC issuing partial private keys to data users, the cloud server providing storage and search services, and data users (i.e., data owner and target users). The data owner encrypts his data (data files and keywords that index the data files) with the target user's public key (note that the target user can be either himself or the user who the data will be shared with). The data owner outsources to the cloud keyword ciphertexts, data file ciphertexts and the mappings between the keyword ciphertexts and data file ciphertexts (given a keyword ciphertext, the mapping can be used to find the relevant data file ciphertexts). With his own private key, the target user can generate a search token, which is sent to the cloud, so that the cloud can conduct keyword search on the keyword ciphertexts and return the corresponding data file ciphertexts. Since the data files can be encrypted by with hybrid encryption with various public key encryption, e.g., certificateless encryption [2] and certificateless proxy re-encryption [32], for simplicity, in the rest of paper we only consider how to encrypt keywords in the certificateless setting. We also assume that there exists a proper authentication protocol allowing KGC to authenticate the users' identities when issuing the partial private keys.

Application example. For illustrating the model visually, we consider the application of sharing Electronic Health Record (EHRs): The health service vendor (e.g., software infrastructure) acts as key generation center, and patients can be the data owners of EHRs, and will share EHRs with professionals for treatment purpose. When the user (either patients or professionals) registers the system, he will be assigned with a partial private key with respect to his unique identity and can generate the private/public key. To share EHRs with a professional, the patient can use that professional's public key (requiring no validation of the public key) to encrypt the indexed keywords (e.g. age, name, DOB etc.) and store the encrypted keywords and the associated encrypted EHRs in the storage system. The professional then generates the tokens based on keywords together with his own private key, and asks the storage system to return the encrypted EHRs having matched keywords. We can see that the benefit is that the vendor cannot leverage its knowledge on partial private keys to learn extra information from the encrypted index, and the patients require no validation on public key.

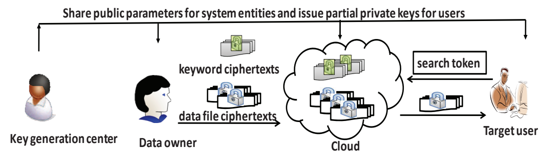


Fig. 1. The system model where the certificateless keyword search scheme can operate.

Threat model Similar to the threat model in the certificateless setting, we assume that the KGC cannot be fully trusted. That is, the KGC follows the protocol specification honestly but might attempt to use the information he obtained to penetrate more information. We assume that the users might be malicious, meaning that the users might pretend to be some target user and distribute the fake public key (i.e., the public key is different from that published by the target user) on behalf of the target user in order to learn information about the target user's ciphertexts. We assume that the cloud server is honest-but-curious, meaning that it will honestly execute the pre-defined protocols, but attempt to learn as much private information as possible. In addition, we assume that the KGC and the users cannot collude together.

5 Definitions

Definition 1. *A* CLKS *scheme is associated with the keyword space $\mathcal{M}$ and the identity space $\mathcal{I}$ and defined by seven algorithms as follows:*

– $\mathsf{Setup}(1^\ell)$*: This algorithm, run by the* KGC*, takes as input a security parameter ℓ and returns the system parameter* param *and master key* mk*. The* param *is made publicly known and* mk *is kept private. For simplicity, we implicitly assume the following algorithms take* param *as part of inputs.*

- $\mathsf{Gen\text{-}Partial\text{-}Private\text{-}Key}(\mathsf{mk}, \mathsf{id})$*: This algorithm, run by the* KGC*, takes as input* mk *and the user* id*, and generates a partial private key* $\mathsf{psk}_{\mathsf{id}}$*, which is sent to the user via a secure channel.*
- $\mathsf{Gen\text{-}Private\text{-}Key}(\mathsf{psk}_{\mathsf{id}})$*: This algorithm, run by the user, takes as input* $\mathsf{psk}_{\mathsf{id}}$*, and returns a complete private key* $\mathsf{sk}_{\mathsf{id}}$*, which is kept private.*
- $\mathsf{Gen\text{-}Public\text{-}Key}(\mathsf{sk}_{\mathsf{id}}, \mathsf{psk}_{\mathsf{id}})$*: This algorithm, run by the user, takes as input the* $\mathsf{sk}_{\mathsf{id}}$ *and* $\mathsf{psk}_{\mathsf{id}}$*, and returns a public key* $\mathsf{pk}_{\mathsf{id}}$*.*
- $\mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{id}, \mathsf{kw})$*: This algorithm, run by any user, takes as input the the target user's public key* $\mathsf{pk}_{\mathsf{id}}$*, identity* id *and the keyword* kw*, and returns a ciphertext* cph*.*
- $\mathsf{Gen\text{-}Token}(\mathsf{sk}_{\mathsf{id}}, \mathsf{kw})$*: This algorithm, run by the user* id*, takes as input the* $\mathsf{sk}_{\mathsf{id}}$ *and the keyword* kw*, and returns a search token* token*.*
- $\mathsf{Match}(\mathsf{token}, \mathsf{cph})$*: This algorithm, run by the cloud server (or entities holding keyword ciphertexts), takes as input the search token* token *and a ciphertext* cph*, and returns 1 if* token *and* cph *correspond to the same identity* id *and the same keyword. Otherwise, it returns 0.*

Correctness. We say that a CLKS scheme is correct if, for all $\mathsf{id} \in \mathcal{I}$, mk output by the Setup, $\mathsf{sk}_{\mathsf{id}}$ output by the $\mathsf{Gen\text{-}Private\text{-}Key}$, and $\mathsf{pk}_{\mathsf{id}}$ output by the $\mathsf{Gen\text{-}Public\text{-}Key}$, the following always holds: $\forall\ \mathsf{kw} \in \mathcal{M}, \mathsf{cph} \leftarrow \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{id}, \mathsf{kw}), \mathsf{token} \leftarrow \mathsf{Gen\text{-}Token}(\mathsf{sk}_{\mathsf{id}}, \mathsf{kw}) : 1 \leftarrow \mathsf{Match}(\mathsf{token}, \mathsf{cph})$
Security. Intuitively, the adversary model against CLKS consists of two kinds of adversaries similar to that of certificateless encryptions [13]: Type 1 adversary models the outsider, who is allowed to manipulate users' public key without accessing the KGC's master key mk; and Type 2 adversary models the insider, who can acquire the KGC's master key mk without manipulating users' public keys.

To be specific, we capture the two security requirements against the two adversaries via the following games. Let $\mathcal{A}$ be Type 1 adversary (Game A) or Type 2 adversary (Game B). The security games are played between $\mathcal{A}$ and the challenger, who maintains two lists:

- $\mathsf{TokenList}$: It stores the tuple $[\mathsf{id}, \mathsf{kw}]$, meaning that the search token with respect to the keyword kw for the user id has been queried by $\mathcal{A}$.
- $\mathsf{UserInfoList}$: It stores the tuple $[\mathsf{id}$, $\mathsf{psk}_{\mathsf{id}}$, $\mathsf{sk}_{\mathsf{id}}$, $\mathsf{pk}_{\mathsf{id}}$, P_1, $\mathsf{P}_2, \mathsf{P}_3]$, where the boolean value $\mathsf{P}_1 = 1$ means that $\mathcal{A}$ has acquired $\mathsf{psk}_{\mathsf{id}}$ and $\mathsf{P}_1 = 0$ not, the boolean value $\mathsf{P}_2 = 1$ means that $\mathcal{A}$ has acquired $\mathsf{sk}_{\mathsf{id}}$ and $\mathsf{P}_2 = 0$ not, and the boolean value $\mathsf{P}_3 = 1$ means that $\mathcal{A}$ has replaced id's public key $\mathsf{pk}_{\mathsf{id}}$ and $\mathsf{P}_3 = 0$ not.

Game A
Setup: The challenger runs $\mathsf{Setup}(1^{\ell})$ to initialize the system parameter param and the master key mk. The challenger sends param to $\mathcal{A}$, and sets two lists $\mathsf{TokenList}$ and $\mathsf{UserInfoList}$ empty.
Phase 1: $\mathcal{A}$ is allowed to query the following oracles in polynomial many times. We use the bracket $[\cdot]$ to indicate the input to the oracle from $\mathcal{A}$.

- $\mathsf{Gen\text{-}Partial\text{-}Private\text{-}Key[id]}$: Given the user id from $\mathcal{A}$, if $\mathsf{psk_{id}}$ in $\mathsf{UserInfoList}$ is not null, then the challenger returns $\mathsf{psk_{id}}$. Otherwise, the challenger runs $\mathsf{Gen\text{-}Partial\text{-}Private\text{-}Key(mk, id)}$ to get $\mathsf{psk_{id}}$, adds $[\mathsf{id}, \mathsf{psk_{id}}, \star, \star, 1, 0, 0]$ to $\mathsf{UserInfoList}$ where $\star$ means null, and returns $\mathsf{psk_{id}}$ to $\mathcal{A}$.
- $\mathsf{Gen\text{-}Private\text{-}Key[id]}$: Given the user id from $\mathcal{A}$, the oracle works as follows:
  - If $\mathsf{sk_{id}}$ in $\mathsf{UserInfoList}$ is not null, then the challenger retrieves $\mathsf{sk_{id}}$.
  - Else if $\mathsf{psk_{id}}$ in $\mathsf{UserInfoList}$ is not null, then the challenger retrieves $\mathsf{psk_{id}}$, runs $\mathsf{sk_{id}} \leftarrow \mathsf{Gen\text{-}Private\text{-}Key(psk_{id})}$ and adds sk_{id} to the $\mathsf{UserInfoList}$.
  - Otherwise, the challenger runs $\mathsf{psk_{id}} \leftarrow \mathsf{Gen\text{-}Partial\text{-}Private\text{-}Key(mk,\ id)}$ and $\mathsf{sk_{id}} \leftarrow \mathsf{Gen\text{-}Private\text{-}Key(psk_{id})}$, and adds $(\mathsf{id}, \mathsf{psk_{id}}, \mathsf{sk_{id}})$ to $\mathsf{UserInfoList}$.

  The challenger updates $P_1 = 1$ and $P_2 = 1$ in $\mathsf{UserInfoList}$ with respect to id and returns $\mathsf{sk_{id}}$ to $\mathcal{A}$.
- $\mathsf{Gen\text{-}Public\text{-}Key[id]}$: Given the user id from $\mathcal{A}$, the oracle works as follows:
  - If $\mathsf{pk_{id}}$ in $\mathsf{UserInfoList}$ is not null, then the challenger retrieves $\mathsf{pk_{id}}$.
  - Else if $\mathsf{sk_{id}}$ in $\mathsf{UserInfoList}$ is not null, then retrieve $\mathsf{sk_{id}}$, the challenger runs $\mathsf{pk_{id}} \leftarrow \mathsf{Gen\text{-}Public\text{-}Key(sk_{id})}$ and adds $\mathsf{pk_{id}}$ to the $\mathsf{UserInfoList}$ with respect to id.
  - Else if $\mathsf{psk_{id}}$ in $\mathsf{UserInfoList}$ is not null , then the challenger retrieves $\mathsf{psk_{id}}$, runs $\mathsf{sk_{id}} \leftarrow \mathsf{Gen\text{-}Private\text{-}Key(psk_{id})}$ and $\mathsf{pk_{id}} \leftarrow \mathsf{Gen\text{-}Public\text{-}Key(sk_{id}, psk_{id})}$ and adds $\mathsf{sk}_{id}, \mathsf{pk_{id}}$ to the $\mathsf{UserInfoList}$ with respect to id.
  - Otherwise, the challenger runs $\mathsf{psk_{id}} \leftarrow \mathsf{Gen\text{-}Partial\text{-}Private\text{-}Key(mk, id)}$, $\mathsf{sk_{id}} \leftarrow \mathsf{Gen\text{-}Private\text{-}Key(psk_{id})}$ and $\mathsf{pk_{id}} \leftarrow \mathsf{Gen\text{-}Public\text{-}Key(sk_{id}, psk_{id})}$, and adds $(\mathsf{id}, \mathsf{psk_{id}}, \mathsf{sk_{id}}, \mathsf{pk_{id}}, 0, 0, 0)$ to $\mathsf{UserInfoList}$.

  The challenger returns $\mathsf{pk_{id}}$ to $\mathcal{A}$.
- $\mathsf{Replace\text{-}Public\text{-}Key[id, pk]}$: Given the user id and the replaced public key pk from $\mathcal{A}$ (assume that $\mathsf{pk_{id}}$ has been generated before), the challenger updates $\mathsf{pk_{id}}$ in $\mathsf{UserInfoList}$ with pk, and sets $\mathsf{P}_3 = 1$ with respect to id.
- $\mathsf{Gen\text{-}Token[id, kw]}$: Given the user id and the keyword kw from $\mathcal{A}$, the challenger retrieves $\mathsf{sk_{id}}$ from $\mathsf{UserInfoList}$, runs $\mathsf{Gen\text{-}Token(sk_{id}, kw)}$ to get token. The challenger adds $[\mathsf{id}, \mathsf{kw}]$ to the $\mathsf{TokenList}$ and returns token to $\mathcal{A}$.

Challenge Phase: $\mathcal{A}$ presents two keywords of the same length, kw_0 and kw_1, and the user id^*. Let $[\mathsf{id}^*, \mathsf{psk}_{\mathsf{id}^*}, \mathsf{sk}_{\mathsf{id}^*}, \mathsf{pk}_{\mathsf{id}^*}, \mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3]$ be the tuple stored in $\mathsf{UserInfoList}$, we require that

- $\mathsf{P}_1 = 0$ and $\mathsf{P}_2 = 0$, meaning that $\mathsf{psk}_{\mathsf{id}^*}$ and $\mathsf{sk}_{\mathsf{id}^*}$ are not acquired by $\mathcal{A}$.
- Both $(\mathsf{id}^*, \mathsf{kw}_0)$ and $(\mathsf{id}^*, \mathsf{kw}_1)$ are not stored in $\mathsf{TokenList}$.

Note that P_3 can be either 0 or 1, meaning that id^*'s public key can be replaced or not. The challenger picks $\lambda \overset{R}{\leftarrow} \{0, 1\}$, runs $\mathsf{cph}^* \leftarrow \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}^*}, \mathsf{id}^*, \mathsf{kw}_\lambda)$, and returns cph^* to $\mathcal{A}$.
Phase 2: $\mathcal{A}$ continues to query the oracles as in Phase 1, while following these restrictions:

- $\mathcal{A}$ cannot query $\mathsf{Gen\text{-}Partial\text{-}Private\text{-}Key[id^*]}$ or $\mathsf{Gen\text{-}Private\text{-}Key[id^*]}$.
- $\mathcal{A}$ cannot query $\mathsf{Gen\text{-}Token}[\mathsf{id}^*, \mathsf{kw}_0]$ or $\mathsf{Gen\text{-}Token}[\mathsf{id}^*,\ \mathsf{kw}_1]$.

Guess: $\mathcal{A}$ outputs a bit λ^*. We say $\mathcal{A}$ wins the game if $\lambda^* = \lambda$.

Definition 2. *A* CLKS *scheme achieves ciphertext indistinguishability against Type 1 adversary if for any probabilistic polynomial time algorithm $\mathcal{A}$, it wins the above security game with a negligible advantage at most with respect to the security parameter ℓ, where the "advantage" is defined as* $|\Pr[\lambda = \lambda^*] - \frac{1}{2}|$.

Game B

Setup: The challenger runs $\mathsf{Setup}(1^\ell)$ to initialize the system parameter param and the master key mk. The challenger sends param and mk to $\mathcal{A}$, and initializes the empty lists $\mathsf{TokenList}$ and $\mathsf{UserInfoList}$.

Phase 1: $\mathcal{A}$ is allowed to query the following oracles in polynomial many times. We use the bracket $[\cdot]$ to indicate the input to the oracle from $\mathcal{A}$.

- $\mathsf{Gen\text{-}Partial\text{-}Private\text{-}Key[id]}$: Given the user id from $\mathcal{A}$, if $\mathsf{psk_{id}}$ in $\mathsf{UserInfoList}$ is not null, then the challenger returns $\mathsf{psk_{id}}$. Otherwise, the challenger runs $\mathsf{Gen\text{-}Partial\text{-}Private\text{-}Key(mk, id)}$ to get $\mathsf{psk_{id}}$, adds $[\mathsf{id}, \mathsf{psk_{id}}, \star, \star, 1, 0, 0]$ to $\mathsf{UserInfoList}$ where $\star$ means null, and returns $\mathsf{psk_{id}}$ to $\mathcal{A}$.
- $\mathsf{Gen\text{-}Private\text{-}Key[id]}$: Given the user id from $\mathcal{A}$, the oracle works as follows:
  - If $\mathsf{sk_{id}}$ in $\mathsf{UserInfoList}$ is not null with respect to id, then retrieve $\mathsf{sk_{id}}$.
  - Otherwise, run $\mathsf{psk_{id}} \leftarrow \mathsf{Gen\text{-}Partial\text{-}Private\text{-}Key(mk, id)}$ and $\mathsf{sk_{id}} \leftarrow \mathsf{Gen\text{-}Private\text{-}Key(psk_{id})}$, and add ($\mathsf{id}$, $\mathsf{psk_{id}}$, $\mathsf{sk_{id}}$) to $\mathsf{UserInfoList}$.

  The challenger updates $\mathsf{P}_1 = 1$ and $\mathsf{P}_2 = 1$ in $\mathsf{UserInfoList}$ with respect to id and returns $\mathsf{psk_{id}}$, $\mathsf{sk_{id}}$ to $\mathcal{A}$.
- $\mathsf{Gen\text{-}Public\text{-}Key[id]}$: Given the user id from $\mathcal{A}$, the oracle works as follows:
  - If $\mathsf{pk_{id}}$ in $\mathsf{UserInfoList}$ is not null with respect to id, then retrieve $\mathsf{pk_{id}}$.
  - Else, the challenger runs $\mathsf{psk_{id}} \leftarrow \mathsf{Gen\text{-}Partial\text{-}Private\text{-}Key(mk, id)}$, $\mathsf{sk_{id}} \leftarrow \mathsf{Gen\text{-}Private\text{-}Key(psk_{id})}$ and $\mathsf{pk_{id}} \leftarrow \mathsf{Gen\text{-}Public\text{-}Key(sk_{id}, psk_{id})}$, and add ($\mathsf{id}$, $\mathsf{psk_{id}}$, $\mathsf{sk_{id}}$, $\mathsf{pk_{id}}$) to $\mathsf{UserInfoList}$.

  The challenger updates $\mathsf{P}_1 = 1$ in $\mathsf{UserInfoList}$ and returns $\mathsf{psk_{id}}, \mathsf{pk_{id}}$ to $\mathcal{A}$.
- $\mathsf{Gen\text{-}Token[id, kw]}$: Given the user id and the keyword kw from $\mathcal{A}$, the challenger retrieves $\mathsf{sk_{id}}$ from $\mathsf{UserInfoList}$, runs $\mathsf{Gen\text{-}Token(sk_{id}, kw)}$ to get token. The challenger adds $[\mathsf{id}, \mathsf{kw}]$ to the $\mathsf{TokenList}$ and returns token to $\mathcal{A}$.

Challenge Phase: $\mathcal{A}$ presents two keywords of the same length, kw_0 and kw_1, and the user id^*. Let [id^*, $\mathsf{psk_{id^*}}$, $\mathsf{sk_{id^*}}$, $\mathsf{pk_{id^*}}$, P_1, P_2, P_3] be the tuple stored in $\mathsf{UserInfoList}$, we require that

- $\mathsf{P}_2 = 0$, meaning that $\mathsf{sk_{id^*}}$ is not acquired by $\mathcal{A}$.
- Both $(\mathsf{id}^*, \mathsf{kw}_0)$ and $(\mathsf{id}^*, \mathsf{kw}_1)$ are not stored in $\mathsf{TokenList}$.

The challenger picks $\lambda \overset{R}{\leftarrow} \{0, 1\}$, runs $\mathsf{cph}^* \leftarrow \mathsf{Enc}(\mathsf{pk_{id^*}}, \mathsf{id}^*, \mathsf{kw}_\lambda)$, and returns cph^* to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ continues to query the oracles as in Phase 1, while following the below restrictions:

- $\mathcal{A}$ cannot query $\mathsf{Gen\text{-}Private\text{-}Key[id^*]}$.

– $\mathcal{A}$ cannot query $\mathsf{Gen\text{-}Token}[\mathsf{id}^*, \mathsf{kw}_0]$ or $\mathsf{Gen\text{-}Token}[\mathsf{id}^*, \mathsf{kw}_1]$.

Guess: $\mathcal{A}$ outputs a bit λ^*. We say $\mathcal{A}$ wins the game if $\lambda^* = \lambda$.

Definition 3. *A* CLKS *scheme achieves ciphertext indistinguishability against Type 2 adversary if for any probabilistic polynomial time algorithm $\mathcal{A}$, it wins the above security game with a negligible advantage at most with respect to the security parameter ℓ, where the "advantage" is defined as* $|\Pr[\lambda = \lambda^*] - \frac{1}{2}|$.

6 Main construction

High level idea Motivated by the DL assumption, we encrypt the keyword kw as follows: Let a, b, c be the private key ($a, b, c \overset{R}{\leftarrow} \mathbb{Z}_p^*$) and g^a, g^b, g^c be the public key, and set the keyword ciphertext as

$$W_1 = g^{cr_1}, \qquad W_2 = g^{a(r_1+r_2)} g^{bH_2(\mathsf{kw})r_1}, \qquad W_3 = g^{r_2},$$

where $H_2 : \{0,1\}^* \to \mathbb{Z}_p^*$ is a collision resistant hash function. According to the DL assumption, the keyword kw is perfectly hidden. Note that g^b in the term $g^{bH_2(\mathsf{kw})r_1}$ is to facilitate the security proof.

If one knows the term g^{ac}, then he can generate the search token with respect to any keyword kw': Select $s \overset{R}{\leftarrow} \mathbb{Z}_p^*$ (s is to preserve the secrecy of g^{ac}) and set

$$V_1 = g^{acs}, \qquad V_2 = g^{cs}, \qquad V_3 = g^{as} g^{bH_2(\mathsf{kw}')s}.$$

If $\mathsf{kw} = \mathsf{kw}'$, then the respective keyword ciphertext and search token should match:

$$e(W_2, V_2) = e(W_1, V_3)e(W_3, V_1).$$

This is because

$$\begin{aligned} e(W_2, V_2) &= e(g,g)^{acs(r_1+r_2)} e(g,g)^{bcsr_1H_2(\mathsf{kw})}, \\ e(W_1, V_3) &= e(g,g)^{acsr_1} e(g,g)^{bcsr_1H_2(\mathsf{kw}')} \\ e(W_3, V_1) &= e(g,g)^{acsr_2}, \end{aligned}$$

On the other hand, we observe that the adversary can only acquire either KGC's master key (w.r.t Type 1 adversary) or the user's secret values (w.r.t Type 2 adversary with replaced public key), rather than both. This observation motivates use to expand the private key setting, such that KGC has the private key a, b, c and public key g^a, g^b, g^c, and the user has the private key a', c' ($a', c' \overset{R}{\leftarrow} \mathbb{Z}_p^*$) and public key $g^{a'}, g^{c'}$. Therefore, the keyword ciphertext is constructed:

$$W_1 = g^{(c+c')r_1}, \quad W_2 = g^{(a+a')(r_1+r_2)} g^{bH_2(\mathsf{kw})r_1}, \quad W_3 = g^{r_2}, \quad W_4 = H_1(\mathsf{id})^{r_2},$$

where H_1 is a hash function converting id to an element of G. Noting that we implicitly let the user id as part of the public key (cf. W_4), so that the ciphertext is associated to the user id. To this end, only knowing either a, c or a', c' cannot infer any information about the ciphertext.

In order to generate search token, the user id, holding a', c', needs to acquire the term g^{ac}. We achieve it based on the idea how the secret key was distributed in attribute-based encryption, where user id is treated as an attribute. That is, KGC generates the partial private key for the user id as

$$\mathsf{sk}_1 = g^{t_1} \quad \text{and} \quad \mathsf{sk}_2 = g^{ac} H_1(\mathsf{id})^{t_1} \quad \text{where } t_1 \overset{R}{\leftarrow} \mathbb{Z}_p^*.$$

Our construction We now show the CLKS construction as follows.

- $\mathsf{Setup}(1^\ell)$: The KGC runs $(e, p, g, G, G_T) \leftarrow \mathsf{BMP}(1^\ell)$ to generate the bilinear map $e : G \times G \to G_T$. Let $a, b, c \overset{R}{\leftarrow} \mathbb{Z}_p^*$ so that it has g^a, g^b, g^c. Let id be an n-bit number such that $\mathsf{id} = \mathsf{id}_1 \mathsf{id}_2 \ldots \mathsf{id}_n$. It selects vectors $(u, u_1, \ldots, u_n) \overset{R}{\leftarrow} G^{n+1}$ and defines the hash function $H_1(\mathsf{id}) = u \prod_{j=1}^{n} u_j^{\mathsf{id}_j}$. Let $H_2 : \{0,1\}^* \to \mathbb{Z}_p^*$ be a collision-resistant hash function and set the system parameter $\mathsf{param} = (e, G, G_T,\, p, g, g^a,\, g^b, g^c, u, u_1,\, \ldots, u_n,\, H_1, H_2)$.
- $\mathsf{Gen\text{-}Partial\text{-}Private\text{-}Key}(\mathsf{mk}, \mathsf{id})$: Given the user id, KGC proceeds as follows:
  - It selects $t_1 \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and sets $\mathsf{sk}_1 = g^{t_1}, \mathsf{sk}_2 = g^{ac} H_1(\mathsf{id})^{t_1}$.
  - The partial private key is set to $\mathsf{psk}_{\mathsf{id}} = (\mathsf{sk}_1, \mathsf{sk}_2)$, which is sent to the user id via a secure channel.
- $\mathsf{Gen\text{-}Private\text{-}Key}(\mathsf{psk}_{\mathsf{id}})$: Given $\mathsf{psk}_{\mathsf{id}}$, the user selects $a', c' \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and sets $\mathsf{sk}_{\mathsf{id}} = (a', c', \mathsf{sk}_1, \mathsf{sk}_2)$.
- $\mathsf{Gen\text{-}Public\text{-}Key}(\mathsf{sk}_{\mathsf{id}})$: Given $\mathsf{sk}_{\mathsf{id}}$, the user sets its public key as $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2) = (g^{a'}, g^{c'})$.
- $\mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{id}, \mathsf{kw})$: The user encrypts the keyword kw and returns a ciphertext cph by selecting $r_1, r_2 \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and setting $\mathsf{cph} = (W_1, W_2, W_3, W_4)$ where
$$W_1 = g^{(c+c')r_1}, \quad W_2 = g^{(a+a')(r_1+r_2)} g^{bH_2(\mathsf{kw})r_1}, \quad W_3 = g^{r_2}, \quad W_4 = H_1(\mathsf{id})^{r_2}.$$
- $\mathsf{Gen\text{-}Token}(\mathsf{sk}_{\mathsf{id}}, \mathsf{kw})$: The user id generates a search token token by selecting $s \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and setting $\mathsf{token} = (V_1, V_2, V_3, V_4)$ where
$$\begin{aligned} V_1 &= \mathsf{sk}_1^s = g^{t_1 s}, \qquad V_2 = (g^{ac'} g^{a'c} g^{a'c'})^s \mathsf{sk}_2^s = g^{(a+a')(c+c')s} H_1(\mathsf{id})^{t_1 s}, \\ V_3 &= g^{(c+c')s}, \qquad V_4 = g^{(a+a')s} g^{bH_2(\mathsf{kw})s}. \end{aligned}$$
- $\mathsf{Match}(\mathsf{token}, \mathsf{cph})$: This algorithm returns 1 if $e(W_2, V_3) = \frac{e(W_3, V_2)}{e(W_4, V_1)} e(W_1, V_4)$. Otherwise, return 0.

Correctness. The correctness can be verified as follows:

$$\text{Because} \quad \frac{e(W_3, V_2)}{e(W_4, V_1)} = \frac{e(g^{r_2}, g^{(a+a')(c+c')s} H_1(\mathsf{id})^{t_1 s})}{e(H_1(\mathsf{id})^{r_2}, g^{t_1 s})} = e(g,g)^{(a+a')(c+c')sr_2},$$

$$\text{and} \quad e(W_1, V_4) = e(g^{(c+c')r_1}, g^{(a+a')s} g^{bH_2(\mathsf{kw})s}) = e(g,g)^{(c+c')r_1 s(a+a'+bH_2(\mathsf{kw}))},$$

$$\text{then} \quad \frac{e(W_3, V_2)}{e(W_4, V_1)} e(W_1, V_4) = e(g,g)^{(a+a')(c+c')s(r_1+r_2)} e(g,g)^{b^2 H_2(\mathsf{kw}) r_1 s}.$$

$$e(W_2, V_3) = e(g^{(a+a')(r_1+r_2)} g^{bH_2(\mathsf{kw})r_1}, g^{(c+c')s}) = e(g,g)^{(a+a')(c+c')(r_1+r_2)s} e(g,g)^{b^2 H_2(\mathsf{kw}) r_1 s}$$

Remark. Given a search token, the attacker can launch the off-line keyword guessing attack (aka. predicate privacy [25]) because he can utilize the public information (e.g., public key) to test whether the given search token corresponds to some keyword. Such attack is inherent for certificate-based keyword search, identity-based (attribute-based) keyword search and our proposed CLKS. Some research efforts has been made to prevent such attack. For example, [24, 19, 31] considered the approaches where the search token can be transmitted over the public channel (i.e., search token indistinguishability) but the user needs to share some secret with the remote server, which cannot completely solve the off-line keyword guessing attack because the remote server can still launch this attack.

The security of our proposed scheme can be assured by the following theorems, which proofs are shown in the Appendix.

Theorem 1. *Suppose $\mathcal{A}$ is the Type 1 adversary making at most ξ_{eppk} queries to oracle* Gen-Partial-Private-Key *and ξ_{prk} queries to oracle* Gen-Private-Key*. Let Adv_{H_2} be the advantage of $\mathcal{A}$ breaking the collision-resistant hash function H_2 (i.e., $\mathsf{Adv}_{H_2} = \Pr[(H_2(\mathsf{kw}_1) = H_2(\mathsf{kw}_2)) \cap (\mathsf{kw}_1 \neq \mathsf{kw}_2) | \mathsf{kw}_1, \mathsf{kw}_2 \in \{0,1\}^*]$) and $\mathsf{Adv}_{DL}(\ell)$ be the advantage of $\mathcal{A}$ breaking the DL assumption. Then the advantage of Type 1 adversary breaking the ciphertext indistinguishability is*

$$\mathsf{Adv}_{\textit{Type 1}} \leq \mathsf{Adv}_{H_2} + 2(\xi_{prk} + \xi_{eppk})(n+1)\mathsf{Adv}_{DL}(\ell)$$

Theorem 2. *Suppose $\mathcal{A}$ is the Type 2 adversary making at most ξ_{prk} queries to oracle* Gen-Private-Key *and ξ_{tk} queries to oracle* Gen-Token*. Let Adv_{H_2} be the advantage of $\mathcal{A}$ breaking the collision-resistant hash function H_2 and $\mathsf{Adv}_{DL}(\ell)$ be the advantage of $\mathcal{A}$ breaking the DL assumption. Then the advantage of Type 2 adversary breaking the ciphertext indistinguishability is*

$$\mathsf{Adv}_{\textit{Type 2}} \leq \mathsf{Adv}_{H_2} + 2\xi_{prk}\xi_{tk}(n+1)\mathsf{Adv}_{DL}(\ell)$$

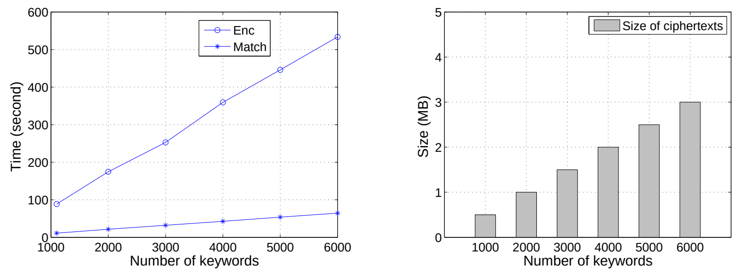
7 Performance Evaluation

We implemented the CLKS scheme in Java using the Java Pairing Based Cryptography Library [15]. In our implementation, we instantiated the bilinear map with Type A pairing ($\ell = 512$) offering a level of security equivalent to 1024-bit Discrete Logarithm security. To process the keywords and the identities (both are strings), we used the SHA-1 to hash them into the 128-bit data, which are then transformed to the elements in $\mathbb{Z}_p^*$ and G respectively.

To evaluate the feasibility of the CLKS in practice, we conducted the experiments with the real data, which is composed of 6,000 distinct keywords extracted from the ACM Digital Library. We ran the experiments on two machines: The user machine is a Windows 7 running laptop with Intel i5 2.60GHz CPU and 8GB RAM to simulate the user that runs the algorithm Enc; the server machine is a a Linux running desktop with Intel Core i7-2600 3.4GHz CPU and 4GB RAM to simulate the cloud server that runs the algorithm Match.

We ran six experiments using different number of keywords varying from 1000 to 6000 with step 1000. We repeated each experiment 5 times to determine the average execution time. Figure 2(a) compares the execution time for encrypting all keywords (on the user machine) and for finding the matched keyword ciphertexts with a given search token (on the server machine).

From Figure 2(a) we see that encrypting entire keywords are more costly than finding the matched keyword ciphertexts. Fortunately, encrypting entire keywords will be executed only once, while finding matched ciphertext needs to be executed once per search request. Figure 2(b) shows the size of keyword ciphertexts that are serialized to the disk. We can see that the size of keyword ciphertexts is linear to the number of keywords.



(a) Execution time (b) Size of keyword ciphertexts

Fig. 2. Performance of CLKS. Note that we set that it needs to run Match algorithm on half of the entire keyword ciphertexts to find a matched keyword ciphertext.

8 Conclusion

We have proposed a novel variant of keyword search – certificateless keyword search(CLKS), supporting keyword search on encrypted data while enjoying the benefits from certificateless cryptography. We presented a concrete construction, and proved its security under the DL assumption in the standard model.

References

1. M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency

properties, relation to anonymous ibe, and extensions. *J. Cryptol.*, 21(3):350–391, 2008.

2. S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In *Advances in Cryptology - ASIACRYPT 2003, Taipei, Taiwan, 2003*, pages 452–473, 2003.
3. J. Baek, R. Safavi-Naini, and W. Susilo. Certificateless public key encryption without pairing. In *Information Security, 8th International Conference, ISC 2005, Singapore, Sept. 20-23, 2005*, pages 134–148, 2005.
4. J. Baek, R. Safavi-Naini, and W. Susilo. Public key encryption with keyword search revisited. In *Computational Science and Its Applications - ICCSA 2008, International Conference, Perugia, Italy, 2008, Part I*, pages 1249–1259, 2008.
5. F. Bao, R. H. Deng, X. Ding, and Y. Yang. Private query on encrypted data in multi-user settings. In *ISPEC 2008, Sydney, Australia, 2008*, pages 71–85, 2008.
6. D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Cryptology - EUROCRYPT 2004, Interlaken, Switzerland, 2004*, pages 506–522, 2004.
7. D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC 2007, Amsterdam, The Netherlands, 2007*, pages 535–554, 2007.
8. C. Bösch, A. Peter, B. Leenders, H. W. Lim, Q. Tang, H. Wang, P. H. Hartel, and W. Jonker. Distributed searchable symmetric encryption. In *2014 Twelfth Annual International Conference on Privacy, Security and Trust, Toronto, ON, Canada, 2014*, pages 330–337, 2014.
9. J. Camenisch, M. Kohlweiss, A. Rial, and C. Sheedy. Blind and anonymous identity-based encryption and authorised private searches on public key encrypted data. In *Public Key Cryptography - PKC 2009, Irvine, CA, USA, 2009*, pages 196–214, 2009.
10. N. Cao, C. Wang, M. Li, K. Ren, and W. Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Trans. Parallel Distrib. Syst.*, 25(1):222–233, 2014.
11. Q. Chai and G. Gong. Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers. In *ICC 2012, Ottawa, ON, Canada, 2012*, pages 917–922, 2012.
12. Y. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *ACNS 2005, New York, NY, USA, 2005*, pages 442–455, 2005.
13. Y.-C. Chen, R. Tso, W. Susilo, X. Huang, and G. Horng. Certificateless signatures: Structural extensions of security models and new provably secure schemes. Cryptology ePrint Archive, Report 2013/193, 2013. `http://eprint.iacr.org/`.
14. R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *CCS 2006, Alexandria, VA, USA, 2006*, pages 79–88, 2006.
15. A. De Caro and V. Iovino. jpbc: Java pairing based cryptography. In *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*, pages 850–855, Kerkyra, Corfu, Greece, 2011.
16. A. W. Dent. A note on game-hopping proofs. Cryptology ePrint Archive, Report 2006/260, 2006. `http://eprint.iacr.org/`.
17. E. Goh. Secure indexes. *IACR Cryptology ePrint Archive*, 2003:216, 2003.
18. J. Homer, S. Zhang, X. Ou, D. Schmidt, Y. Du, S. R. Rajagopalan, and A. Singhal. Aggregating vulnerability metrics in enterprise networks using attack graphs. *Journal of Computer Security*, 21(4):561–597, 2013.
19. C. Hu and P. Liu. An enhanced searchable public key encryption scheme with a designated tester and its extensions. *JCP*, 7(3):716–723, 2012.

20. H. Huang, S. Zhang, X. Ou, A. Prakash, and K. A. Sakallah. Distilling critical attack graph surface iteratively through minimum-cost SAT solving. In *ACSAC 2011, Orlando, FL, USA, 2011*, pages 31–40, 2011.
21. S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. Outsourced symmetric private information retrieval. In *CCS'13, Berlin, Germany, 2013*, pages 875–888, 2013.
22. S. Kamara, C. Papamanthou, and T. Roeder. Dynamic searchable symmetric encryption. In *CCS'12, Raleigh, NC, USA, 2012*, pages 965–976, 2012.
23. K. Kurosawa and Y. Ohtaki. Uc-secure searchable symmetric encryption. In *Financial Cryptography and Data Security - 16th International Conference, FC 2012, Kralendijk, Bonaire, 2012, Revised Selected Papers*, pages 285–298, 2012.
24. H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee. Trapdoor security in a searchable public-key encryption scheme with a designated tester. *Journal of Systems and Software*, 83(5):763–771, 2010.
25. E. Shen, E. Shi, and B. Waters. Predicate privacy in encryption systems. In *TCC 2009, San Francisco, CA, USA, 2009*, pages 457–473, 2009.
26. J. Shi, J. Lai, Y. Li, R. H. Deng, and J. Weng. Authorized keyword search on encrypted data. In *Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security, Wroclaw, Poland, 2014. Part I*, pages 419–435, 2014.
27. D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA, 2000*, pages 44–55, 2000.
28. W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li. Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. In *2014 IEEE Conference on Computer Communikations, INFOCOM 2014, Toronto, Canada, 2014*, pages 226–234, 2014.
29. Q. Tang and X. Chen. Towards asymmetric searchable encryption with message recovery and flexible search authorization. In *8th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '13, Hangzhou, China, 2013*, pages 253–264, 2013.
30. C. Wang, N. Cao, K. Ren, and W. Lou. Enabling secure and efficient ranked keyword search over outsourced cloud data. *IEEE Trans. Parallel Distrib. Syst.*, 23(8):1467–1479, 2012.
31. T.-Y. Wu, T.-T. Tsai, and Y.-M. Tseng. Efficient searchable id-based encryption with a designated server. volume 69, pages 391–402. Springer Paris, 2014.
32. L. Xu, X. Wu, and X. Zhang. CL-PRE: a certificateless proxy re-encryption scheme for secure data sharing with public cloud. In *ASIACCS '12, Seoul, Korea, 2012*, pages 87–88, 2012.
33. S. Zhang, X. Zhang, and X. Ou. After we knew it: empirical study and modeling of cost-effectiveness of exploiting prevalent known vulnerabilities across iaas cloud. In *,ASIA CCS '14, Kyoto, Japan - June 03 - 06, 2014*, pages 317–328, 2014.
34. Q. Zheng, S. Xu, and G. Ateniese. VABKS: verifiable attribute-based keyword search over outsourced encrypted data. In *2014 IEEE Conference on Computer Communikations, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*, pages 522–530, 2014.
35. B. Zhu, B. Zhu, and K. Ren. Peksrand: Providing predicate privacy in public-key encryption with keyword search. In *Proceedings of IEEE International Conference on Communications, ICC 2011, Kyoto, Japan, 5-9 June, 2011*, pages 1–6, 2011.

9 Appendix: Theorems and Their Security Proof

9.1 Proof of Theorem 1

Proof. We prove this theorem with the game hopping technique [16]: Let S_i denote the event of $\mathcal{A}$ winning the game i and Adv_i be the advantage of $\mathcal{A}$. Suppose game $i+1$ proceeds the same as game i except for the case that game $i+1$ will halt and output a random bit when a pre-defined event Ev happens. If Ev is detectable (i.e., $\Pr[\mathsf{Ev}]$ is non-negligible) and Ev is independent from S_i, then

$$\begin{aligned}\Pr[S_{i+1}] &= \Pr[S_{i+1}|\mathsf{Ev}]\Pr[\mathsf{Ev}] + \Pr[S_{i+1}|\neg\mathsf{Ev}]\Pr[\neg\mathsf{Ev}]\\ &= \frac{1}{2}\Pr[\mathsf{Ev}] + \Pr[S_{i+1}|\neg\mathsf{Ev}]\Pr[\neg\mathsf{Ev}]\\ &= \frac{1}{2}(1-\Pr[\neg\mathsf{Ev}]) + \Pr[S_i]\Pr[\neg\mathsf{Ev}]\end{aligned}$$

$$\text{Therefore,}\quad |\Pr[S_{i+1}] - \frac{1}{2}| = \Pr[\neg\mathsf{Ev}]|\Pr[S_i] - \frac{1}{2}|$$
$$\mathsf{Adv}_{i+1} = \Pr[\neg\mathsf{Ev}]\mathsf{Adv}_i$$

<u>Game 1:</u> In this game, $\mathcal{A}$ proceeds with the steps as defined in Game A. That is, the challenger $\mathcal{B}$ generates the master key, public parameter and users' partial private key. In addition, $\mathcal{B}$ can answer the queries oracle $\mathsf{Gen\text{-}Token}$ for search tokens. Let $(\mathsf{id}^*, \mathsf{pk}_{\mathsf{id}^*})$ be the user identity and public key in the challenger phase and let $W^* = (W_1^*, W_2^*, W_3^*, W_4^*)$ be the ciphertext returned to A.

<u>Game 2:</u> In this game, $\mathcal{A}$ proceeds with the steps as defined in Game 1 except that H_2 is set to be a perfect collision-resistant hash function. Therefore, $\Pr[S_2] = \Pr[S_1] - \mathsf{Adv}_{H_2}$ and $\mathsf{Adv}_2 = \mathsf{Adv}_1 - \mathsf{Adv}_{H_2}$.

<u>Game 3:</u> In this game, $\mathcal{B}$ plays the game the same as Game 2 except for the generation of the public parameter:

- $\mathcal{B}$ selects $a, b, c \overset{R}{\leftarrow} \mathbb{Z}_p^*$. It also selects $\kappa_u \in \{0, \ldots, n\}$ (note that id is n-bit number) and $\tau_u \in \{0, \ldots, p\}$ such that $\tau_u(n+1) < p$.
- $\mathcal{B}$ selects $x'_u \overset{R}{\leftarrow} \mathbb{Z}_{\tau_u}$ and the vector $(x_{u1}, \ldots, x_{un})$ where $x_{uj} \in \mathbb{Z}_{\tau_u}, 1 \le j \le n$.
- $\mathcal{B}$ selects $y'_u \overset{R}{\leftarrow} \mathbb{Z}_p$ and the vector $(y_{u1}, \ldots, y_{un})$ where $y_{uj} \overset{R}{\leftarrow} \mathbb{Z}_p, 1 \le j \le n$.
- The public parameters are set as

$$\begin{aligned}u &= (g^a)^{x'_u - \kappa_u\tau_u} g^{y'_u}\\ u_j &= (g^a)^{x_{uj}} g^{y_{uj}} \quad \text{for } 1 \le j \le n\end{aligned}$$

We can see that the generation process did not change the distribution of the public parameter. Therefore, $\Pr[S_3] = \Pr[S_2]$ and $\mathsf{Adv}_3 = \mathsf{Adv}_2$.

<u>Game 4:</u> In this game, $\mathcal{B}$ plays the game the same as Game 3 except for the guess phase (Ev_4):

- $\mathcal{B}$ defines two functions for the input $\mathsf{id} = \mathsf{id}_1 \dots \mathsf{id}_n$:

$$J_u(\mathsf{id}) = x'_u - \kappa_u \tau_u + \sum_{j=1}^{n} \mathsf{id}_j x_{uj}$$

$$K_u(\mathsf{id}) = y'_u + \sum_{j=1}^{n} \mathsf{id}_j y_{uj}$$

- At the guess phase, $\mathcal{B}$ then checks that whether $J_u(\mathsf{id}^*) = 0$. If $J_u(\mathsf{id}^*) \neq 0$, then $\mathcal{B}$ halts and outputs $\lambda' \overset{R}{\leftarrow} \{0,1\}$ as $\mathcal{A}$'s guess. Otherwise, $\mathcal{B}$ proceeds the same as Game 3.

Since $(x'_u, x_{u1}, \dots, x_{un})$ are unknown to $\mathcal{A}$, $\Pr[J_u(\mathsf{id}^*) = 0 \mod p] = \frac{1}{\tau_u(n+1)}$. Therefore, $\mathsf{Adv}_4 = \frac{\mathsf{Adv}_3}{\tau_u(n+1)}$.

Game 5: In this game, $\mathcal{B}$ plays the game the same as Game 4 except for the guess phase (Ev_5): $\mathcal{B}$ checks the oracle queries to see whether one of the following happens,

- $J_u(\mathsf{id}) = 0$ for some id that was queried to partial private key oracle Gen-Partial-Private-Key.
- $J_u(\mathsf{id}) = 0$ for some id that was queried to private key oracle Gen-Private-Key.

If Ev_5 happens, then $\mathcal{B}$ halts and outputs $\lambda' \overset{R}{\leftarrow} \{0,1\}$ as $\mathcal{A}$'s guess. Since Ev_5 might be not independent of Game 4 (note that $J_u(\mathsf{id}^*) = 0$ in Game 4), we will estimate the lower bound of $\Pr[\neg\mathsf{Ev}_5]$.

$$\begin{aligned}
\Pr[\neg\mathsf{Ev}_5] &= \Pr[(\cap_{\mathsf{id}\in\Omega} J_u(\mathsf{id}) \neq 0) \bigwedge (\cap_{\mathsf{id}\in\Omega'} J_u(\mathsf{id}) \neq 0)] \\
&= 1 - \Pr[(\cup_{\mathsf{id}\in\Omega} J_u(\mathsf{id}) \neq 0) \bigvee (\cup_{\mathsf{id}\in\Omega'} J_u(\mathsf{id}) \neq 0)] \\
&\geq 1 - \sum_{\mathsf{id}\in\Omega} \Pr[J_u(\mathsf{id}) \neq 0] - \sum_{\mathsf{id}\in\Omega'} \Pr[J_u(\mathsf{id}) \neq 0] \\
&= 1 - \frac{\xi_{eppk} + \xi_{prk}}{\tau_u}
\end{aligned}$$

where Ω is the set of id queried to the oracle Gen-Partial-Private-Key, Ω' is the set of id queried to the oracle Gen-Private-Key, ξ_{eppk} is the number of oracle queries to Gen-Partial-Private-Key and ξ_{prk} is the number of oracle queries to Gen-Private-Key.

Therefore, if we set $\tau_u = 2(\xi_{prk} + \xi_{eppk})$, then $\Pr[\neg\mathsf{Ev}_5] \geq \frac{1}{2}$. Hence, $\mathsf{Adv}_5 \geq \frac{1}{2}\mathsf{Adv}_4$.

Game 6: In this game, $\mathcal{B}$ plays the game the same as Game 5 except for taking g^a, g^b, g^c as public key, where a, b, c are unknown to $\mathcal{B}$. $\mathcal{B}$ proceeds with the oracles according to the algorithm specification except for the oracle Gen-Partial-Private-Key[id].

– $\mathsf{Gen\text{-}Partial\text{-}Private\text{-}Key}[\mathsf{id}]$: Given id, if $J_u(\mathsf{id}) = 0$, then $\mathcal{B}$ aborts (simulated by Game 5). Otherwise, it picks $t_1' \xleftarrow{R} \mathbb{Z}_p^*$ and sets

$$\mathsf{sk}_1 = (g^b)^{\frac{-1}{J_u(\mathsf{id})}} g^{t_1'} \quad \text{and} \quad \mathsf{sk}_2 = (g^b)^{\frac{-K_u(\mathsf{id})}{J_u(\mathsf{id})}} H_1(\mathsf{id})^{t_1'}$$

Note that $\mathsf{sk}_1, \mathsf{sk}_2$ are valid components for the partial private key because:

$$\begin{aligned} H_1(\mathsf{id}) &= g^{aJ_u(\mathsf{id})} g^{K_u(\mathsf{id})} \\ \mathsf{sk}_1 &= g^{t_1' - \frac{c}{J_u(\mathsf{id})}} \\ \mathsf{sk}_2 &= (g^c)^{\frac{-K_u(\mathsf{id})}{J_u(\mathsf{id})}} H_1(\mathsf{id})^{t_1'} \\ &= g^{ac}(g^{aJ_u(\mathsf{id})} g^{K_u(\mathsf{id})})^{\frac{-c}{J_u(\mathsf{id})}} (g^{aJ_u(\mathsf{id})} g^{K_u(\mathsf{id})})^{t_1'} \\ &= g^{ac} H_1(\mathsf{id})^{t_1' - \frac{c}{J_u(\mathsf{id})}} \end{aligned}$$

by implicitly setting $t_1 = t_1' - \frac{c}{J_u(\mathsf{id})}$.

We can see that the distributions of the master key, public keys and the partial private key are identical to that of Game 5. Therefore, $\mathsf{Adv}_6 = \mathsf{Adv}_5$.

Game 7: In this game, $\mathcal{B}$ plays the game the same as Game 6 except for the challenge phase.

Given $(\mathsf{pk}_{\mathsf{id}^*} = (\mathsf{pk}_1^*, \mathsf{pk}_2^*), \mathsf{id}^*, \mathsf{kw}_0, \mathsf{kw}_1)$ from $\mathcal{A}$, if $J_u(\mathsf{id}^*) \neq 0$, then $\mathcal{B}$ aborts (simulated by Game 4). Otherwise, $\mathcal{B}$ picks $\lambda \xleftarrow{R} \{0,1\}$, selects $r_1, r_2 \xleftarrow{R} \mathbb{Z}_p^*$ and set $\mathsf{cph} = (W_1^*, W_2^*, W_3^*, W_4^*)$ where

$$\begin{aligned} W_1^* &= (g^c \mathsf{pk}_1^*)^{r_1}, \\ W_2^* &= (g^a \mathsf{pk}_2^*)^{r_1 + r_2} g^{bH_2(\mathsf{kw}_\lambda) r_1}, \\ W_3^* &= g^{r_2}, \\ W_4^* &= g^{K_u(\mathsf{id}^*) r_2} \text{because } J_u(\mathsf{id}^*) = 0 \end{aligned}$$

We can see that $\mathsf{Adv}_7 = \mathsf{Adv}_6$ because the distribution of the challenge ciphertext did not change.

Game 8: In this game, $\mathcal{B}$ plays the game the same as Game 7 except for the challenge ciphertext generation. Given a DL instance $(g, h, f, g^{r_2}, h^{r_1}, Q)$, $\mathsf{cph} = (W_1^*, W_2^*, W_3^*, W_4^*)$ is set, where

$$\begin{aligned} W_1^* &= h^{r_1}, \\ W_2^* &= Q h^{H_2(\mathsf{kw}_\lambda) r_1}, \\ W_3^* &= g^{r_2}, \\ W_4^* &= g^{K_u(\mathsf{id}^*) r_2}. \end{aligned}$$

by implicitly setting $g^b = h, g^c = h/pk_2^*$ and $g^a = f/\mathsf{pk}_1^*$. In this game, $\mathcal{B}$ does not use a, b, c, r_1, r_2 at all. The distinguishable probability between Game 7 and Game 8 is related to the DL problem. Let $\mathsf{Adv}_{DL}(\ell)$ be the advantage of A

distinguishing Decisional Linear problem. Then $|\Pr[S_7] - \Pr[S_8]| \leq \mathsf{Adv}_{DL}(\ell)$. Moreover, as in Game 8, kw_λ is hidden perfectly, so $\Pr[S_8] = \frac{1}{2}$.

This completes the simulation and has the following inequalities

$$\begin{aligned}\mathsf{Adv}_1 &= \mathsf{Adv}_2 + \mathsf{Adv}_{H_2}\\ \mathsf{Adv}_2 &= \mathsf{Adv}_3\\ \mathsf{Adv}_3 &= \tau_u(n+1)\mathsf{Adv}_4\\ \mathsf{Adv}_4 &\leq 2\mathsf{Adv}_5\\ \mathsf{Adv}_5 = \mathsf{Adv}_6 = \mathsf{Adv}_7 &\leq \mathsf{Adv}_{DL}(\ell)\end{aligned}$$

Therefore,

$$\mathsf{Adv}_1 \leq \mathsf{Adv}_{H_2} + 2(\xi_{prk} + \xi_{eppk})(n+1)\mathsf{Adv}_{DL}(\ell)$$

9.2 Proof of Theorem 2

The proof for this theorem uses similar technologies as in Theorem 1, hence we skip it here. Note that $\mathcal{A}$ will not query the oracle Gen-Partial-Private-Key because it has the master key.