Searchable Encryption with Access Control on Keywords in Multi-User Setting

Lei Li[1], Chungen Xu[1, *], Xiaoling Yu[1], Bennian Dou[1] and Cong Zuo[2]

Abstract: Searchable encryption technology makes it convenient to search encrypted data with keywords for people. A data owner shared his data with other users on the cloud server. For security, it is necessary for him to build a fine-grained and flexible access control mechanism. The main idea of this paper is to let the owner classify his data and then authorizes others according to categories. The cloud server maintains a permission matrix, which will be used to verify whether a trapdoor is valid or not. In this way we can achieve access control and narrow the search range at the same time. We prove that our scheme can achieve index and trapdoor indistinguishability under chosen keywords attack security in the random oracles.

Keywords: Searchable encryption, access control, cloud computing, permission assignment.

1 Introduction

Cloud service brings great convenience to people due to its powerful computing power and rich storage resources. Nowadays more and more people are used to storing their files on the cloud server to save limited local storage. However, the cloud server cannot be fully trusted. In order to prevent personal data from leaking, users need to encrypt their data before uploading to the server. Soon people find it difficult to search over ciphertext. It seems that one solution is to let the cloud server decrypt all ciphertext and do the search work. It is equivalent to exposing all plaintext to the administrator of the server. Another solution is to download all data, decrypt them and search one by one, which needs a huge local storage space. Obviously, neither of them is feasible. Thus searchable encryption technology emerges as the times requires. Searchable encryption enables people to directly search over ciphertext with keywords, leaking little information [Li, Zhao, Jiang et al. (2017); Xiong and Shi (2018); Liu, Peng and Wang (2018)].

In the multi-user environment, for security [Xia, Xiong, Vasilakos et al. (2017)], people should have different access rights to data in the cloud server [Xia, Lu, Qiu et al. (2019)]. We take the electronic medical system as an instance. To protect a patient's privacy, we

[1] School of Science, Nanjing University of Science and Technology, Nanjing, 210094, China.

[2] Faculty of Information Technology, Monash University, Clayton, VIC 3800, Australia.

[*] Corresponding Author: Chungen Xu. Email: xuchung@njust.edu.cn

prescribe that physicians can only retrieve medical records about internal medicine and ophthalmologists can only retrieve medical records about ophthalmology. One should not be accessible to his unauthorized data. Thus we proposed a searchable encryption. The purpose is to improve search efficiency and protect privacy at the same time. Actually, in some traditional searchable encryption schemes, due to the indistinguishability of trapdoor, the server is not able to deduce any information about the keyword. So it cannot determine whether a user has access to the keyword he wants to search for, unless attaching extra information to the indexes. However, in this way, once people change, it needs to reconstruct almost all indexes.

In our scheme, to set access control for one's data, he should first classify his data into categories. For each category, he extracts a keyword as a subject heading. When a user initiates a search request to the cloud server, his trapdoor includes not only the keyword but also a subject heading. After receiving a trapdoor, the cloud server checks whether it is a valid request. In other words, the server needs to know if this user has access to the data where his trapdoor refers to. If the judgement result is "Yes", then the server will directly search within that subject. Otherwise, the server rejects this request.

In general, the contributions of this paper are listed as follows:

1. Flexible access control: Administrators can flexibly modify the access rights of other users by maintaining a permission matrix. When the user staff changes, there is no need to reconstruct indexes and change the keys.
2. Decentration: In our scheme, every user keeps the private key by himself, we do not need a third party to do the key management.
3. High efficiency: To determine the validity of a search query, the server does not need to match it with those indexes one by one. Instead, only a small amount of operations can the server to accept or refuse it.

The remaining part of this paper is arranged as follows. In the second section, we introduce some related contents of access control based on multi-user setting in searchable encryption schemes. In the third section, we introduce some preparatory work of this scheme. The fourth section will introduce our searchable encryption scheme in detail. And finally, we provide the security analysis of our scheme.

2 Related work

In 2000, Song et al. [Song, Wagner and Perrig (2000)] introduced the routing problem of untrusted server and proposed the scheme SWP, which has low efficiency but can be viewed as a primitive searchable encryption scheme. To solve this problem, in 2004, Boneh et al. [Boneh, Crescenzo and Ostrovsky (2004)] firstly introduced the public key cryptosystem into searchable encryption. In 2011, Curtmola et al. [Curtmola, Garay, Kamara et al. (2006)] proposed a searchable encryption scheme based on multi-user setting for the first time, which is much more practical than the single-user mode.

Some schemes have improved the searchable encryption scheme based on multi-user setting [Raza, Rashid and Awan (2017); Goyal, Pandey, Sahai et al. (2006); Tang (2014); Yang (2013); Li, Yu, Cao et al. (2011) etc.] everyone classifies his or her documents and then sets access control for each level of the data. Finally, all the users generate a permission matrix together and only those who have been granted are able to pass the authentication of the server. In their scheme, each user can be granted autonomously and select his search scope independently. Also their scheme weakens the role of a third party and provides a scheme for efficient key distribution.

In Bao et al. [Bao, Robert and Ding (2008)], an administrator is responsible for managing and distributing keys for all users. In the key generation algorithm, he also generates an auxiliary key, which is used for checking the validity of a search query. In this way, the administrator can achieve access control dynamically.

Wang et al. [Wang, Mu, Chen et al. (2016)] implemented an efficient searchable encryption scheme for sharing data among users in a decentralized group. Each member's public key is needed when generating an index and any user is able to generate a trapdoor by using his or her own secret key. Moreover, their scheme can adapt to the dynamic change of the group by adjusting the user's search authorities in time.

Wang et al. [Wang, Wang and Pieprzyk (2008)] put forward the concept of threshold access control based on Shamir Secret Sharing ideas [Rong (2015); Tartary and Wang (2006)]. In a group of n users, only more than t persons can collaborate to generate a valid trapdoor. Later, Zirtol et al. [Zirtol, Noroozi and Eslami (2016)] changed the scheme by supporting general access structure. Instead of the threshold limit, only the group that meets the pre-defined condition can collaborate to search the desired data.

Many other schemes set access control based on identity [Boneh, Boyen and Goh (2005); Ma, Dui and Yang (2016); Boneh and Franklin (2003); Yang (2011)] added the authorization information into each index. After receiving a trapdoor, the server starts to match it with indexes one by one. Note that at this moment, the server not only needs to determine whether an index matches the trapdoor, but also needs it to make sure whether this user is granted to search the file. The server would return the corresponding files if and only if both conditions are met. Although their scheme can achieve fine-grained access control, the complex indexes bring too much computation. Once the group changes, they need to rebuild all indexes.

3 Preliminaries

In this section we will introduce some preliminary knowledge related to our scheme.

3.1 Bilinear mapping

Definition 3.1. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups of a large prime order q. A bilinear mapping is e: $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:

1. Bilinearity. For any $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$ and $a, b \in Z_q^*$, the equation $e(u^a, v^b) = e(u,v)^{ab}$ is hold in $\mathbb{G}_T$;
2. Non-degeneracy. There is $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$ such that $e(u,v) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ is an identity element of $\mathbb{G}_T$;
3. Computability. There is an efficient algorithm such that for any $u \in \mathbb{G}_1$ and $v \in \mathbb{G}_2$,

computing $e(u,v)$ is available.

3.2 Difficulty hypothesis

Definition 3.2 Let x be a primitive root for a finite field $GF(p^n)$ and z is a non-zero element in $GF(p^n)$. The discrete logarithm problem (DLP) is to find an exponent α such that $x^\alpha \equiv z(mod\ p^n)$, here α is called the discrete logarithm of z to the base x.

Definition 3.3 (**Bilinear Diffie-Hellman Variant assumption [Lu (2017)]**) There is a negligible function *negl* such that for any PPT adversary $\mathcal{A}$ and for every sufficiently large security parameter *k*, the following equation is hold:

$$|Pr[\mathcal{A}(g_1^a, g_1^c, g_2^b, g_2^{1/a}, e(g_1,g_2)^{abc})=1] - Pr[Adv(g_1^a, g_1^c, g_2^b, g_2^{1/a}, R)=1]| = negl(k) \quad (1)$$

Definition 3.4 (**External Diffie-Hellman Variant assumption [Lu (2017)]**) There exists a negligible function *negl* such that for any PPT adversary $\mathcal{A}$ and for every sufficiently large security parameter *k*, the following equation is hold:

$$|Pr[\mathcal{A}(g_1^a, g_1^b, g_1^{ab}, g_2^{c/a}, g_2^d, g_2^{1/a})=1] - Pr[Adv(g_1^a, g_1^b, R, g_2^{c/a}, g_2^d, g_2^{1/a})=1]| = negl(k) \quad (2)$$

4 Model of scheme

4.1 Notation

The following table shows some notations used in this paper:

Table 1: Notation in the scheme

Notation	Significance
u_i	the $i-th$ user
SK / PK	the secret and public key pairs of the data owner
SK_i / PK_i	the secret and public key pairs of user u_i
C_w	the index of keyword w
$T_i(w,t)$	trapdoor for keyword w and its subject heading t of user u_i
$T_i(w)$	the new trapdoor computed by the server based on $T_i(w,t)$ of user u_i
Num_j	the number of keywords in subject heading t_j

4.2 Outline of scheme

Our scheme consists of the following seven polynomial time algorithms: ***Setup***, ***KeyGen***, ***IndexGen***, ***Grant***, ***TrapGen***, ***Test***, ***Search.***

Setup (1^k): It takes the security parameter *k* as input and outputs the public parameter *param*.

KeyGen (*param*): The data owner runs this algorithm with *param* to generate his public key *PK* and secret key *SK*, every user u_i also generates his or her key pair PK_i, SK_i, and so does the cloud server.

IndexGen (*w*, *SK*, *param*): This algorithm is run by the data owner. He takes his private key *SK* as input to encrypt each keyword *w* and generates its encrypted index C_w.

Grant (SK, PK_i): This algorithm is also run by the data owner. He takes his private key SK and every user's public key PK_i as input to generate a matrix *M* for access control.

TrapGen (*w*, *t*, SK_i): The user u_i takes his private key SK_i, a keyword *w* and its corresponding subject heading *t* as input, it outputs a trapdoor $T_i(w,t)$ for searching.

Test ($T_i(w,t)$, *M*): This algorithm is run by the cloud server. After receiving a trapdoor, the server tests it with the permission matrix *M*. If the server determines a trapdoor is valid, then it will be used for the following steps. Otherwise, the server rejects the follow-up operations and prompts "Unauthorized Access!".

Search (C_w, T_w): Once the server accepts a trapdoor, it will perform the subsequent matching work to find those files relevant to the keyword.

4.3 Security model

4.3.1 Index indistinguishability under chosen keyword attack

This requirement is aimed to protect all indexes stored on the cloud server. Neither an internal nor external adversary is able to deduce any information about any keyword even if he gets the index. In order to prove this, we firstly define a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$, then we define a game between them.

— Setup. The challenger $\mathcal{C}$ initializes the setup algorithm to generate the public parameters *param* and sends them to $\mathcal{A}$.

—Challenge. The adversary $\mathcal{A}$ selects two keywords w_0^*, w_1^* and sends them to $\mathcal{C}$, $\mathcal{C}$ tosses a coin randomly to get a bit *b* and provides the encrypted index $C_{w_b^*}$ to $\mathcal{A}$.

—Adaptive ask. $\mathcal{A}$ can ask $\mathcal{C}$ adaptively. Every time $\mathcal{C}$ provides the corresponding index C_{w_k} and trapdoor $T(w_k,t)$ to $\mathcal{A}$.

—Guess. $\mathcal{A}$ outputs a bit *b*', if *b*=*b*', he wins.

Our scheme achieves index indistinguishability under chosen keyword attack (IND-CKA) security if, for all sufficiently large *k* and for all PPT adversaries there exists a negligible function *negl* such that:

$$Pr[\mathcal{A}(k) \mid b = b'] \le \frac{1}{2} + negl(k) \tag{3}$$

4.3.2 Trapdoor indistinguishability under chosen keyword attack

This requirement is aimed to protect the trapdoor generated by a user. Even an adversary eavesdrops a trapdoor, he is not able to deduce any information about the keyword and the subject it contains. To prove it, we also define a game between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$.

—Setup. The challenger $\mathcal{C}$ initializes the system to generate the public parameters *param* and sends them to $\mathcal{A}$.

—Challenge. The adversary $\mathcal{A}$ selects w_0^*, w_1^*, t_0^*, $t_1^* \leftarrow \{0,1\}^l$ and sends them to $\mathcal{C}$, $\mathcal{C}$ tosses a coin twice randomly to get two bits $b_1, b_2 \in \{0,1\}$ and provides the trapdoor $T(w_{b_1}^*, t_{b_2}^*)$ to $\mathcal{A}$.

—Adaptive ask. $\mathcal{A}$ can ask $\mathcal{C}$ adaptively. Every time $\mathcal{C}$ provides the corresponding index C_{w_k} and trapdoor $T(w_k, t_j)$ to $\mathcal{A}$.

—Guess. $\mathcal{A}$ outputs bits b_1', b_2', if $b_1 = b_1', b_2 = b_2'$, he wins.

Our scheme satisfies trapdoor indistinguishability under chosen keyword attack if for all sufficiently large *k* and for all PPT adversaries $\mathcal{A}$, there is a negligible function *negl* such that:

$$Pr[\mathcal{A}(k) \mid b_1 = b_1', b_2 = b_2'] \leq \frac{1}{4} + negl(k) \tag{4}$$

5 Construction

Setup (1^k): The data owner takes the security paramter *k* as input to initialize the setup algorithm, it outputs public parameters $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, Z_q^*, g_1, g_2, e, H_1, H_2, H_3)$. $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are three cyclic groups of some prime order *q*, g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$. $e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an injective bilinear mapping. H_1, H_2, H_3 are three collision resistant functions: $H_1: \{0,1\}^* \rightarrow \mathbb{G}_1$; $H_2: \{0,1\}^* \rightarrow Z_q^*$; $H_3: \mathbb{G}_T \times Z_q^* \rightarrow \mathbb{G}_T$.

KeyGen (*param*): The cloud server generates a key pair $(\mu, g_1^{\frac{1}{\mu}})$, $\mu \in Z_q^*$. The data owner selects $\alpha, \beta \in Z_q^*$ randomly and sets his secret key as $SK = (SK^1, SK^2) = (\alpha, \beta)$ and the public key is $PK = (PK^1, PK^2) = (g_2^{\frac{1}{\alpha}}, g_2^{\beta})$. Every user u_i also gets his or her own secret key $SK_i = x_i$ and announces the public key $PK_i = g_2^{\frac{1}{x_i}}$.

IndexGen (*w*, *SK*, *param*): For each keyword w_i, the owner randomly selects $s_i \in Z_q^*$ and generates the index $C_{w_i} = (C_{w_{i1}}, C_{w_{i2}}) = (s_i, H_3(e(H_1(w_i), g_2^{\beta})^{\frac{1}{\alpha}}, s_i))$.

Grant (SK, PK_i): For setting access control for different kinds of files, the data owner utilizes his secret key and all users' public keys to generate a permission matrix M_{nm}, here *n* is the number of users and *m* is the number of subject headings:

$$M_{nm} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1m} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_{nm} \end{bmatrix} \quad (5)$$

We define the element $\sigma_{ij} = g_2^{\frac{\beta}{\alpha x_i r_{ij}}}$. If user u_i is granted to search data about subject t_j, then $r_{ij} = H_2(t_j)$, otherwise $r_{ij} \leftarrow_R Z_q^*$. For each subject heading $t_j (j = 1, 2, \cdots, m)$, the data owner computes $e(H_1(w_{jl}), g_2^{\beta})^{\frac{1}{\alpha}}, (l = 1, 2, \cdots, Num_j)$ and save them in the corresponding set ψ_j.

TrapGen (w_k, t_j, SK_i): User u_i utilizes his secret key to generate a trapdoor $T_i(w_k, t_j) = (g_1^{\frac{d}{\mu}}, g_1^{-d} H_1(w_k)^{x_i H_2(t_j)})$ about the keyword w_k and its subject heading t_j. d is the random number. Then he sends the trapdoor to the cloud server.

Test ($M_{nm}, T_i(w_k, t_j)$): After receiving a trapdoor, the server firstly runs the test algorithm in order to determine its validity. It goes through j and computes $e(T_i(w_k, t_j), \sigma_{ij})$, ($j = 1, 2, \ldots, m$) then it checks whether it belongs to ψ_j respectively. If there is no j such that $e(T_i(w_k, t_j), \sigma_{ij}) \in \psi_j$, the server determines its invalidity and thus halts.

Search ($C_{w_i}, T_i(w_k)$): Once accepted a trapdoor, the server views the result $e(H_1(w_k), g_2^{\beta})^{\frac{1}{\alpha}}$ in test algorithm as the new trapdoor and matches it with those indexes. It computes $H_3(T_i(w_k), s_i)$ and makes sure the equation $H_3(T_i(w_k), s_i) = C_{w_{i2}}$ holds such that the server returns the files.

6 Correctness analysis

6.1 Correctness in test algorithm

If user u_i is granted to be accessible to data about subject t_j, his privilege for searching data about t_j is $\sigma_{ij} = g_2^{\frac{\beta}{\alpha x_i r_{ij}}} = g_2^{\frac{\beta}{\alpha x_i H_2(t_j)}}$, so when the server comes to j, it computes:

$$e(T_i(w_k, t_j), \sigma_{ij}) = e(H_1(w_k)^{x_i H_2(t_j)}, g_2^{\frac{\beta}{\alpha x_i H_2(t_j)}}) = e(H_1(w_k), g_2^{\beta})^{\frac{1}{\alpha}} \quad (6)$$

As long as the keyword w_k belongs to subject t_j indeed, the value above must belong to ψ_j, which means user u_i passes the authentication.

We require that the intersection of any two classes be empty, so we know that for any $m, n, \psi_m \cap \psi_n = \varnothing$, which indicates there is no way to search the same keyword with other subjects and only by being granted can a user to search data he wants.

6.2 Correctness in test algorithm

Once the server accepts a query in last algorithm, it is going to search with the new trapdoor $e(H_1(w_k), g_2^{\beta})^{\frac{1}{\alpha}}$. If an index contains w_k, we have:

$$H_3(C_{w_{k1}}, T_i(w_k)) = H_3(s_k, e(H_1(w_k), g_2^{\beta})^{\frac{1}{\alpha}}) = C_{w_{k2}} \tag{7}$$

7 Security analysis

7.1 Index indistinguishability under chosen keyword attack

Index stored in the cloud service should not leak any information about the corresponding keywords. Even an adversary is given the most powerful ability, he still can't distinguish any two encrypted indexes with non-negligible probability.

Proof. In order to prove that our scheme can achieve IND-CKA security, we take use of several hybrid games, which starts from the one defined in the security model (4.3.1) to the last. It's easy to see that the adversary wins the last game with probability 1/2. Our proofs are based on the random oracle model and H_1 is a programmable random oracle. Game 0 is defined as follows:

Game 0:

—Setup. The challenger $\mathcal{C}_0$ initializes the system to generate public parameters *param*. Without loss of generality, adversary $\mathcal{A}_0$ can select some other users u_j who can search data about subject t_1. The challenger computes $\sigma_{j1} = g_2^{\frac{\beta}{\alpha x_j r_{j1}}} (r_{j1} = H_2(t_1))$ and sends them as well as *param* to $\mathcal{A}_0$.

—Challenge. The adversary $\mathcal{A}_0$ selects $w_0^*, w_1^* \leftarrow \{0,1\}^l$ in subject t_1 and sends them to the challenger $\mathcal{C}_0$. $\mathcal{C}_0$ randomly tosses a coin to get a bit b and makes w_b^* a keyword in subject t_1 and provides this encrypted index $[r, H_3(r, e(H_1(w_b^*), g_2^{\beta})^{\frac{1}{\alpha}})]$ to $\mathcal{A}_0$.

—Adaptive ask. $\mathcal{A}_0$ can ask $\mathcal{C}_0$ adaptively. The $k-th$ query is like:

(1). "Index of keyword w_k in subject t_1": The challenger $\mathcal{C}_0$ returns the result $[r_k, H_3(r_k, e(H_1(w_k), g_2^{\beta})^{\frac{1}{\alpha}})]$.

(2). "Trapdoor for keyword w_k and subject t_1 of user u_j": The challenger $\mathcal{C}_0$ returns $T_j(w_k, t_1) = (T_1, T_2) = (g_1^{\frac{d_k}{\mu}}, g_1^{-d_k} H_1(w_k)^{x_j H_2(t_1)})$.

—Guess. $\mathcal{A}_0$ outputs b', if $b = b'$, he wins.

Game 1:

The only difference between Game 0 and Game 1 is that in Game 1, we delete those users without privileges to search data about subject t_1. Actually, if it is secure in Game 1, then it must be secure in Game 0. Otherwise, if there exists an adversary $\mathcal{A}_0$ who wins Game 0 with non-negligible probability, we can construct another adversary $\mathcal{A}_1$ to win Game 1 with non-negligible probability as well. Actually, for those users who have access to data about subject t_1, returns $(g_1^{\frac{d}{\mu}}, g_1^{-d}H_1(w_k)^{x_j H_2(t_1)})$ to them directly, while for the others, $\mathcal{A}_1$ selects random value R and return $(g_1^{\frac{d}{\mu}}, g_1^{-d}H_1(w_k)^{R})$. The two values are actually computational indistinguishable due to the DLP assumption. Thus adversary $\mathcal{A}_1$ simulates $\mathcal{A}_0$'s inputs correctly.

Game 2:

The difference between Games 1 and 2 is that we replace $e(H_1(w_k), g_2^{\beta})^{\frac{1}{\alpha}})$ in Game 2 with random value *R*. The first step during the adaptive ask now is like "Index of keyword w_k and t_j": if $w_k = w_b^*$, challenger returns $[r_k, H_3(r_k, R)]$. Otherwise challenger returns $[r_k, H_3(r_k, R^{\theta})]$. Here θ satisfies $g_1^{\theta} = H_1(w_{1-b}^*) / H_1(w_b^*) g_1^{\theta} = H_1(w_{1-b}^*) / H_1(w_b^*)$. We prove the indistinguishability between the two games.

If there is a PPT adversary $\mathcal{A}_0$ who can distinguish the two games with non-negligible probability, then we can construct another PPT adversary $\mathcal{A}_1$ to break BDHV assumption.

The inputs $\mathcal{A}_0$ receives are "$g_1^a, g_1^c, g_2^{\frac{1}{a}}, g_2^b, T$". Here *T* is either a random value *R* or $e(g_1, g_2)^{abc}$. In order to distinguish *T*, $\mathcal{A}_0$ takes what he receives as the result when $\mathcal{A}_1$ interacts with the random oracle for w_b^*. $\mathcal{A}_1$ could ask those values before declaration during the challenge step. So $\mathcal{A}_0$ will guess which queries to H_1 are for challenge values. When $\mathcal{A}_1$ provides his challenge keywords w_0^* and w_1^* to $\mathcal{A}_0$, $\mathcal{A}_0$ firstly checks whether it has been asked before. If not, $\mathcal{A}_0$ outputs a random value and halts.

$\mathcal{A}_0$ generates *param* and sends them to $\mathcal{A}_1$, then he selects $\delta, \gamma \in Z_q^*$ and let $H_2(t_1) = \gamma$. For every query *w* of $\mathcal{A}_1$ to H_1, $\mathcal{A}_0$ does:

If this query is about keyword w_b^*, $\mathcal{A}_0$ returns g_1^c;

If this query is about keyword w_{1-b}^*, $\mathcal{A}_0$ returns $g_1^{c\delta}$;

Otherwise, chooses $\lambda \in Z_q^*$, let oracle[*w*]=λ and returns g_1^{λ}.

$\mathcal{A}_0$ receives w_0^* and w_1^* from $\mathcal{A}_1$, he firstly checks if they have appeared before. If not, $\mathcal{A}_0$ outputs a random value and halts. Otherwise $\mathcal{A}_0$ provides$[r^*, H_3(r^*, T)]$ to $\mathcal{A}_1$. For every query of $\mathcal{A}_1$ during the adaptive ask step:

For "Index of w_k": if $w_k = w_b^*$, return[$r_k, H_3(r_k, T)$]. Otherwise, return [$r_k, H_3(r_k, T^{\theta})$].

For "Trapdoor for w_k of u_j": select random numbers r_k, r_j and return $g_1^{x_j r_k r_j}$.

Actually, we see that:

$$g_2^{\beta} \leftrightarrow g_2^{a}, b \leftrightarrow 1/\alpha, g_1^{c} \leftrightarrow H_1(w_b^*), g_1^{c\delta} \leftrightarrow H_1(w_{1-b}^*) \tag{8}$$

So, if $T = e(g_1, g_2)^{abc}$, then we know that $T = e(g_1^c, g_2^a)^b = e(H_1(w_b^*), g_2^{\beta})^{\frac{1}{\alpha}}$, else T=R as in Game 2.

Therefore, we claim that the two games are computationally indistinguishable.

In Game 2, all the information about keys is useless to adversary, which means what he receives has nothing to do with the bit b at all. He wins the game just with the probability 1/2. So in Game 0 we have:

$$Pr_{\mathcal{A}_0}(k)[b = b'] \le \frac{1}{2} + negl(k) \tag{9}$$

7.2 Trapdoor indistinguishability under chosen keyword attack

This requirement is aimed to protect the trapdoor generated by a user. An adversary should learn nothing about the keyword and the subject heading from the trapdoor he eavesdropped.

Proof. We also utilize a sequence of hybrid games to complete our proof as before.

Game 0:

—Setup. The challenger $\mathcal{C}_0$ initializes the system to generate the public parameters *param*. Without loss of generality, we assume that adversary $\mathcal{A}_0$ can take control of user u_1, who has privileges to search data about all the subject headings. The challenger computes his privileges $g_2^{\frac{\beta}{\alpha x_1 r_{1j}}} (r_{1j} = H_2(t_j), j = 1, 2, \cdots, m)$ and sends them as well as *param* to $\mathcal{A}_0$.

—Challenge. Adversary $\mathcal{A}_0$ firstly selects two subject headings t_0^*, t_1^*, two searching keywords w_0^*, w_1^* and sends them together to challenger $\mathcal{C}_0$. $\mathcal{C}_0$ tosses a coin twice to get two random bits $b_1, b_2 \leftarrow \{0,1\}$, and then he returns $T(w_{b_2}^*, t_{b_1}^*) = (g_1^{\frac{d}{\mu}}, g_1^{-d} H_1(w_{b_2}^*)^{x_1 H_2(t_{b_1}^*)})$ as the trapdoor that $\mathcal{A}_0$ is going to search.

—Adaptive ask. $\mathcal{A}_0$ can ask $\mathcal{C}_0$ adaptively. The $k-th$ query is like:

(1). "Index of keyword w_k in subject t_k": The challenger $\mathcal{C}_0$ returns $[r_k, H_3(r_k, e(H_1(w_k), g_2^{\beta})^{\frac{1}{\alpha}})]$.

(2). "Trapdoor for keyword w_k in subject t_k of user u_1": The challenger $\mathcal{C}_0$ returns $T_1(w_k,t_k)=(g_1^{\frac{d_k}{\mu}},g_1^{-d_k}H_1(w_k)^{x_1H_2(t_k)})$.

Note that here the adversary has a restriction of:

$t_k \notin \{t_0^*,t_1^*\}, w_k \notin \{w_0^*,w_1^*\}$

—Guess. $\mathcal{A}_0$ outputs two bits b_1' and b_2', he wins if and only if $b_1=b_1'$ and $b_2=b_2'$.

Game 1:

The difference between Games 0 and 1 is that Game 1 deletes the first step (encrypted index asking) during the adaptive ask. Actually, the encrypted index does not leak any information helpful to the adversary.

Game 2:

The difference between Games 1 and 2 is that we replace $T(w_{b_2}^*,t_{b_1}^*)=(g_1^{\frac{d}{\mu}},g_1^{-d}H_1(w_{b_2}^*)^{x_1H_2(t_{b_1}^*)})$ with $T(w_{b_2}^*,t_{b_1}^*)=(g_1^{\frac{d}{\mu}},g_1^{-d}R)$, here R is a random value.

We prove that Game 1 and Game 2 are computational indistinguishable.

If there is a PPT adversary $\mathcal{A}_0$ who can distinguish the two games with non-negligible probability, then there is another PPT adversary $\mathcal{A}_1$ to break EDHV assumption.

The inputs $\mathcal{A}_1$ receives are "$g_1^a,g_1^b,g_1^d,g_2^{c/a},g_2^d,g_2^{\frac{1}{a}},T$". *T* is either g_1^{ab} or a random value *R*. In order to distinguish *T*, $\mathcal{A}_1$ takes what he received as the result when $\mathcal{A}_0$ interacts with the random oracle for $w_{b_2}^*$ and $t_{b_1}^*$. However, $\mathcal{A}_0$ could ask these values before declaration during the challenge step. So $\mathcal{A}_1$ will guess whether $\mathcal{A}_0$ will ask for $w_{b_1}^*$ or $w_{b_2}^*$ to the random oracle H_1.

When $\mathcal{A}_0$ provides w_0^*,w_1^*,t_0^*,t_1^* to $\mathcal{A}_1$ during the challenge step, $\mathcal{A}_1$ checks whether they have been queried before. If not, $\mathcal{A}_1$ outputs a random value and halts. Otherwise, he provides *T* to $\mathcal{A}_0$.

During the adaptive step, for "Trapdoor for keyword w_k and its subject t_k", randomly selects $R_k \in \mathbb{G}_1$ and returns $T(w_k,t_k)=(g_1^{\frac{d_k}{\mu}},g_1^{-d_k}R_k)$.

Similarly, we can see that:

$$a \leftrightarrow x_1, g_1^b \leftrightarrow H_1(w_{b_2}^*), g_2^{c/a} \leftrightarrow g_2^{\frac{\beta}{\alpha x_1}}, g_2^d \leftrightarrow g_2^{\beta} \qquad (10)$$

So, if $T=g_1^{ab}$, we know that $T^{H_2(t_{b_1}^*)}=H_1(w_{b_2}^*)^{x_1H_2(t_{b_1}^*)}$ as in Game 1, Otherwise *T*=*R* as in Game 2.

Finally, when $\mathcal{A}_0$ successfully distinguishes Game 2 from Game 1, $\mathcal{A}_1$ is able to break the EDHV assumption.

In Game 2, all the information about keys are useless, so the adversary can win the game just with the probability 1/4.

8 Comparison

We compare our scheme with some others and show the details in the following table. Here P denotes a pairing operation, E denotes an exponential operation and H denotes a hash operation. From the table, we can see that our work can achieve a balance between computation and storage cost. Also, with JPBC library in Java language, we have carried out our experiments on computer (Intel (R) Core (TM) i5-3210 M CPU 2.5 GHz).

Table 2: Computational complexity comparisons

	Qiang Tang	Zhen Li	Bao Feng	Xiaofen Wang	Ours
IndexGen	2(P+H+E)	P+H+E	P+H+2E	P+(2+k)H	P+H+E
TrapGen	H+E	H+E	H+E	P+4H	2H+E
Search	2P	P+H+E	P+H	3P+H	P+H

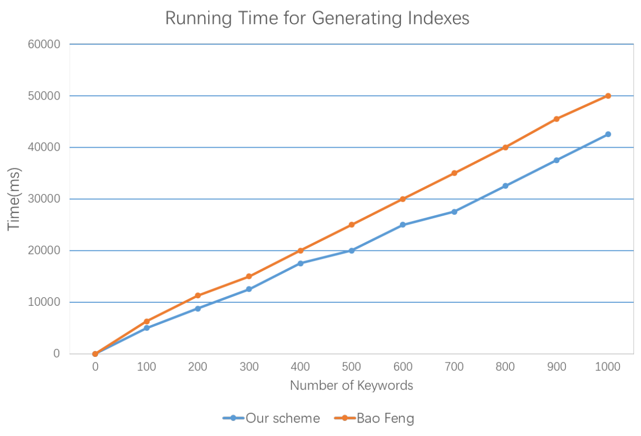

Figure 1: Comparison of running time for generating indexes

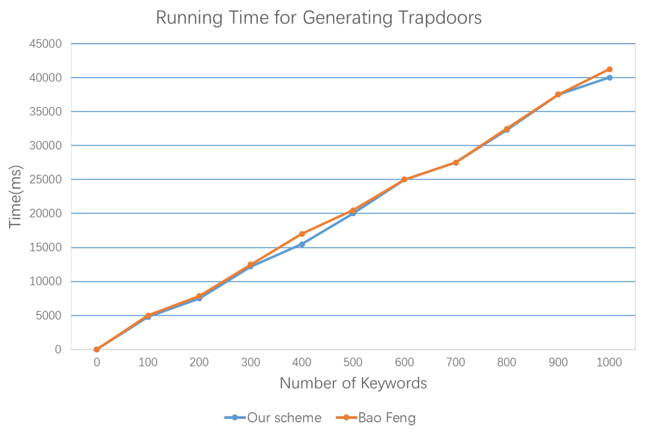

Figure 2: Comparison of running time for generating trapdoors

9 Conclusion

In this paper, we proposed a searchable encryption scheme which supports fine-grained access control. By maintaining a permission matrix, a user can manage the access rights about his data flexibly. There still remains a lot of problems to be solved in the multi-user setting, dynamic security being one of them. We are going to focus our research on it.

Acknowledgement: This work is partially supported by the Fundamental Research Funds for the Central Universities (No. 30918012204). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewer, which have improved the presentation.

References

Bao, F.; Robert, H. D.; Ding, X. H. (2008): Private query on encrypted data in multi-user settings. In *Information Security Practice and Experience, 4th International Conference, ISPEC 2008, Sydney, Australia*, *Proceedings*, pp. 71-85.

Boneh, D.; Crescenzo, G. D.; Ostrovsky, R. (2004): Public key encryption with keyword search. *Advances in Cryptology-EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, Proceedings*, pp. 506-522.

Boneh, D.; Boyen, X.; Goh, E. J. (2005): Hierarchical identity based encryption with constant size ciphertext. *Advances in Cryptology-EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, Proceedings*, pp. 440-456.

Boneh, D.; Franklin, M. K. (2003): Identity-based encryption from the weil pairing. *SIAM Journal on Scientific Computing*, vol. 32, no. 3, pp. 586-615.

Curtmola, R.; Garay, J; Kamara, S.; Ostrovsky, R. (2006): Searchable symmetric encryption: improved definitions and efficient construction. *Computer Security*, vol. 19, no. 5, pp. 895-943.

Goyal, V.; Pandey, O.; Sahai, A.; Waters, B. (2006): Attribute-based encryption for fine-grained access control of encrypted data. *ACM Conference on Computer and Communications Security*, pp. 89-98.

Hong, C. (2011): Achieving efficient dynamic cryptographic access control in cloud storage. *Journal on Communications*, vol. 32, no. 7, pp. 125-132.

Li, M.; Yu, S. C.; Cao, N.; Lou, W. J. (2011): Authorized private keyword search over encrypted data in cloud computing. *International Conference on Distributed Computing Systems, Minneapolis, Minnesota, USA*, pp. 383-392.

Li, Z.; Zhao, M. H.; Jiang, H.; Xu, Q. L. (2017): Multi-user searchable encryption with a designated server. *Annales des Telecommunications*, vol. 72, no. 9-10, pp. 617-629.

Liu, Y. L.; Peng, H.; Wang, J. (2018): Verifiable diversity ranking search over encrypted outsourced data. *Computers, Materials & Continua*, vol. 55, no. 1, pp. 37-57.

Lu, Y. (2017): Efficient designated server identity-based encryption with conjunctive keyword search. *Annals of Telecommunications*, vol. 72, no. 5-6, pp. 359-370.

Ma, B. Y.; Dui, G. S.; Yang, S. Y. (2018): ACSB: Anti-collision selective-based broadcast protocol in CR-ADHOCS. *Computers, Materials and Continua*, vol. 56, no. 1, pp. 35-46.

Ma, S. (2016): Identity-based encryption with outsourced equality test in cloud computing. *Journal of Information Science*, vol. 328, pp. 389-402.

Raza, N.; Rashid, I.; Awan, F. A. (2017): Security and management framework for an organization operating in cloud environment. *Annales des Telecommunications*, vol. 72, no. 5-6, pp. 325-333.

Rong, H. (2015): Key distribution and recovery algorithm based on shamir secret sharing. *Journal of Communication*, vol. 3, pp. 60-69.

Song, X. D.; Wagner, D.; Perrig, A. (2000): Practical techniques for searches on encrypted data. *2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA*, pp. 44-55.

Tang, Q. (2014): Nothing is for free: Security in searching shared and encrypted data. *IEEE Trans. Information Forensics and Security*, vol. 9, no. 11, pp. 1943-1952.

Tartary, C.; Wang, H. (2006): Dynamic threshold and cheater resistance for shamir secret sharing scheme. *Information Security and Cryptology, Second SKLOIS Conference, Inscrypt 2006, Beijing, China, Proceedings*, pp. 103-117.

Wang, P. S.; Wang, H. X.; Pieprzyk, J. (2008): Threshold privacy preserving keyword searches. In *SOFSEM 2008: Theory and Practice of Computer Science, 34th Conference on Current Trends in Theory and Practice of Computer Science, Novy Smokovec, Slovakia, Proceedings*, pp. 646-658.

Wang, X. F.; Mu, Y.; Chen, R. M.; Zhang, X. S. (2016): Secure channel free id-based searchable encryption for peer-to-peer group. *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 1012-1027.

Xia, Z. H.; Lu, L. H.; Qiu, T.; Shim, H. J.; Chen, X. Y. et al. (2019): A privacy-preserving image retrieval based on AC-coefficients and color histograms in cloud environment. *Computers Materials & Continua*, vol. 58, no. 1, pp. 27-43.

Xia, Z. H.; Xiong, N. N.; Vasilakos, A. V.; Sun, X. M. (2017): EPCBIR: an efficient and privacy-preserving content-based image retrieval scheme in cloud computing. *Information Sciences*, vol. 387, pp. 195-204.

Xiong, L. Z.; Shi, Y. Q. (2018): On the privacy-preserving outsourcing scheme of reversible data hiding over encrypted image data in cloud computing. *Computers, Materials and Continua*, vol. 55, no. 3, pp. 523-539.

Yang, H. B. L. (2013): Searchable encryption scheme with fine-grained access control. *Journal of Communication*, vol. 34, no. Z1, pp. 92-100.

Yang, Y. (2011): Towards multi-user private keyword search for cloud computing. *IEEE International Conference on Cloud Computing, CLOUD 2011*, pp. 758-759.

Zirtol, K. A.; Noroozi, M; Eslami, Z. (2016): Multi-user searchable encryption scheme with general access structure. *Journal of Informatics and Computer Engineering*, vol. 2, no. 3, pp. 121-126.