International Journal of Security and Its Applications
Vol. 11, No. 7 (2017), pp.43-58
http://dx.doi.org/10.14257/ijsia.2017.11.7.04

A Searchable Encryption Scheme in a Multi-user Setting for Cloud Storage

Qingqing Gan[1] and Xiaoming Wang[2*]

[1,2]*Department of Computer Science, Jinan University, Guangzhou, China, 510632*
[1]*gan_qingqing@foxmail.com*, [2]*twxm@jnu.edu.cn*

Abstract

To support encrypted keyword queries on encrypted data and to access data in a multi-user setting for cloud storage, a searchable encryption scheme is proposed by combining Chinese Remainder Theorem with Public Encryption with Keyword Search. The proposed scheme enables data owners to outsource their encrypted data to the storage of cloud servers for a number of users, who are given the ability to generate valid queries and to access data without leaking any information. More importantly, the proposed scheme does not rely on shared keys to realize multi-user searchable encryption and user dynamics. Each authorized user in the proposed scheme has a unique key and can perform encrypted keyword queries on encrypted data as well as access data without knowledge of the keys of other users, such that when some users are added or removed, other users are unaffected. We prove that the proposed scheme is secure under Bilinear Diffie-Hellman and Hash Diffie-Hellman assumptions. The performance analysis shows that the proposed scheme has lower communication and storage overheads for both cloud servers and users than other existing schemes. In addition, the proposed scheme possesses features similar to that of most existing schemes in a single-user setting and does not downgrade because of supporting multiple users.

Keywords*: outsourced data; authenticated data structure; data integrity; constant complexity*

1. Introduction

With the development of cloud computing, the demand for data storage outsourcing has increased dramatically in recent years. However, a significant barrier to the adoption of outsourcing data is that data owners are concerned about confidential data leakage and loss of privacy because cloud service providers are not to be fully trusted.

A basic solution to this problem is that sensitive data have to be encrypted before outsourcing. This solution can protect the outsourcing data from outside attackers and cloud service providers, but searching encrypted data becomes difficult.

Numerous studies have been conducted to address this problem. However, existing solutions (*e.g.*,[1]-[9],[10]-[18]) are mostly limited to a single-user setting, where a single user is allowed to perform encrypted keyword queries on encrypted data. In cloud storage, data owners upload their data to commercial cloud servers and wish to share the data with multiple users, rather than a single user. In this case, searchable encryption works in a multi-user setting.

Schemes used in a single-user setting cannot be used in a multi-user setting directly and effectively because of the increasing requirements of the latter. For example, user dynamics is not considered in a single-user setting but it is important in a multi-user setting. We consider a situation where a data owner stores his data in cloud servers and authorizes a group of subscribed users to access the data for a particular period of time. If

* Corresponding Author

ISSN: 1738-9976 IJSIA

any of these subscribed users becomes compromised or is no longer qualified to access the data, the data owner would like to revoke their access to the data. If a scheme in a single-user setting is directly used in this case, a key renewal is required among non-revoked users, and re-encryption of the outsourced data is needed. This requirement causes a great deal of performance overhead and is practically almost impossible.

In this paper, we propose a searchable encryption scheme called SEMU in a multi-user setting for cloud storage. We consider an application scenario where an encrypted data sharing system is stored in cloud servers for a number of authorized users. Our contributions are as follows:

1)We first define the SEMU and provide an efficient construction by combining Chinese Remainder Theorem with Public Encryption with Keyword Search(PEKS) [7]. We then prove that the SEMU is secure under Bilinear Diffie-Hellman and Hash Diffie-Hellman assumptions. In the SEMU, data owners, who generate encrypted data and outsource the encrypted data to cloud servers, authorize a group of users to share the data. The authorized users can perform encrypted keyword queries on the encrypted data as well as access the data, but cloud servers have no idea of any information about both the specified keywords and the data. Therefore, the SEMU realizes data privacy and privacy-preserving queries in a multi-user setting for cloud storage.

2)The SEMU supports not only multi-user searchable encryption, but also dynamic changes in authorized users. More importantly, the SEMU does not rely on a shared key to realize multi-user searchable encryption and user dynamics. Each authorized user has a unique key and can perform encrypted keyword queries on encrypted data as well as access data without knowledge of the keys of other users, such that when some users are added or removed, other users are unaffected. In addition, the SEMU performs scalable encryption on cloud server side without shipping the outsourced data back to data owner when some users are added or removed, thus reducing communication overhead.

3)The SEMU can provide privacy-preserving queries, controlling queries, unforgeable queries, and a designated tester, which means that the SEMU possesses features similar to that of most existing schemes in a single-user setting(e.g., [9, 10]) and does not downgrade because of supporting multiple users. Moreover, the SEMU outperforms some schemes in a multi-user setting (e.g., [6, 20, 21]) in terms of the ciphertext size of the encrypted keyword and the storage overhead for both the cloud server and the user.

The remainder of this paper is organized as follows: Section 2 presents related work, and Section 3 describes some definitions and other useful concepts that are used in our approach. We construct the SEMU in Section 4 and give the security proof and the performance analysis in Section 5. Finally, the concluding remarks are given in Section 6.

2. Related Work

Numerous works have been conducted on searchable encrypted data. Song *et al*. [1] first proposed a practical scheme for searching on encrypted data. Their scheme considers that a user uploads private data to an untrusted server and keeps the data private from the server administrator, and the user can retrieve the data containing a particular keyword from the server. Subsequently, several works [2, 3, 4, 5] were introduced to improve the efficiency and security of the system. However, the previous works only support a single keyword search in symmetric key setting and cannot be used for some practical applications, such as an email routing system [6]. In 2005, Boneh *et al*. [7] first proposed PEKS. Their scheme provides a mechanism that enables a message sender to store an encrypted message with some encrypted keywords at a server, where encryption is based on the public key of a receiver. The receiver may then send a trapdoor corresponding to the keywords and to his private key to the server, such that the server can test whether the encryption data and the trapdoor are made with the same key-words, but the server does not learn anything about both the query and the data. The PEKS uses asymmetric instead

of symmetric encryption and can be used for an email routing system. However, the PEKS has utilized a secure channel between the receiver and the server, which is certainly not suitable for some applications as building a secure channel is usually expensive [8]. Then Baek *et al*. [8] proposed a searchable public-key encryption for a designated tester (dPEKS) to remove the secure-channel assumption of the PEKS. In the dPEKS, only the server can test whether or not a given dPEKS ciphertext is related with a trapdoor by using its private key. Later, several variants (e.g., [9]-[15]) of the PEKS have been proposed with various improvements. Recently, Liu *et al*. [16, 17] proposed privacy preserving keyword search schemes and applied them to cloud storage. Their schemes support searchable encrypted files, and the server participates in partial decipherment to reduce a client's computational overhead. Cheng *et al*. [18] also proposed a privacy preserving keyword search for cloud storage. However, the aforementioned schemes support only a designated receiver, and do not support multiple designated receivers. In other words, these schemes are constructed in a single-user setting.

To support multi-user searches on encrypted data, Curtmola *et al*. [19] transferred their searchable encryption scheme for a single-user setting to one for a multi-user setting. However, their scheme shares the same secret key among all users to search over encrypted database. This approach is evidently unscalable for large and dynamic systems where user addition and revocation may occur frequently. Moreover, user revocation in their scheme is based on broadcast encryption, where a revocation affects all non-revoked users. Hwang and Lee [6] proposed searchable public key encryption in a multi-user setting. Their scheme enables multiple users to search over encrypted data and does not share the same secret key among all users. However, in their scheme, the ciphertext size is proportional to the number of receivers. Moreover, user revocation is not considered. Bao *et al*. [20, 21] proposed multi-user private keyword search schemes for cloud computing. Their schemes enable each user to possess a distinct query key to construct search queries, but all authorized users share the same key to generate the index for a keyword w. This requirement results in a weakness that a revoked user can write records to the dataset. In addition, their schemes need to store a U-Hkey (containing user identities and helper keys). To revoke a user u, the data owner instructs the cloud server to delete the help key hk_u of the user u from the U-Hkey list maintained by the cloud server. In other words, user revocation mainly depends on whether the cloud server removes the user help key hk_u from the U-Hkey list. However, the cloud service provider is not to be fully-trusted and may not follow the instruction of the data owner to remove the help key hk_u from the U-Hkey list to make a benefit. For example, to continue to access the data, a revoked user may give some benefits to the cloud service provider, such that the cloud service provider may not remove his help key from the U-Hkey list. This practice negatively affects the data owner, but the data owner cannot prevent this situation from occurring. In addition, all authorized users in their schemes share the same decrypted key.

3. Preliminaries and Definition

In this section we describe some definitions and other useful concepts that are used in our approach.

3.1. Preliminaries

Bilinear Pairings. Let G and G_T be two multiplicative cyclic groups of prime order p, g be a generator of G. The bilinear map $\hat{e}: G \times G \to G_T$ should be satisfied the following properties:

1) Bilinearity: $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$ for $u, v \in G$.

2) Non-degeneracy: $\hat{e}(g,g) \neq 1$.

3) Computability: There is an efficient algorithm for computing the map $\hat{e}$.

Decisional Diffie-Hellman (DDH) problem)[22]. Decisional Diffie-Hellan (DDH) problem in G is as follows: given (g, g^a, g^b, g^c) as input, output yes if $c = ab$ and no otherwise. We say that DDH is intractable if all polynomial time algorithms have a negligible advantage in solving DH problem.

Bilinear Diffie-Hellman(BDH) Problem[23]. Given $(g, g^a, g^b, g^c) \in G$ as input, com-pute $\hat{e}(g,g)^{abc} \in G_T$. We say that BDH problem is intractable if all polynomial time algorithms have a negligible advantage in solving BDH problem.

Chinese remainder theorem[24]. Let $p_1, p_2, ..., p_n$ be positive integers that are relatively prime in pairs. Then, for any given integers $b_1, b_2, ..., b_n$, the system of congruences

$$y \equiv b_i \bmod p_i (1 \leq i \leq n)$$

has a unique solution modulo $N = p_1 p_2 ... p_n$. The solution is given by

$$y = \sum_{i=1}^{n} b_i N_i N_i^{'} \bmod N$$

where $N_i = N / p_i$ and $N_i N_i^{'} \equiv 1 \bmod p_i$.

3.2. System Model

Three types of entities are used in the proposed system: data owner, cloud server, and multiple users. The data owner outsources data in an encrypted form to the cloud server and authorizes a group of users to access the data. The SEMU is a mechanism that enables only authorized users to generate valid encrypted keyword queries, and only the cloud server can verify whether encrypted data contain the encrypted keywords while learning nothing else. In this work, we assume that the cloud service provider is honest but curious, and should be trusted to perform searchable encryption and re-encryption operations correctly when the cloud server provider is not profitable.

Let m be the number of authorized users. The SEMU is consisted of nine polynomial-time algorithms as follows.

1) Global Setup(λ): Takes a security parameter λ as input, this algorithm generates a global parameter gp.

2) KeyGen(gp): Takes the global parameter gp as input, this algorithm outputs the private/public key pairs (sk_s, pk_s) of the cloud server, and (sk_d, pk_d) of the data owner as well as (sk_i, pk_i) of authorized user u_i, where $i = 1,2,...,m$.

3) EnKeyword($gp, pk_1, ..., pk_m, w$): This algorithm is executed by the data owner to encrypt a keyword w. Takes as input the global parameter gp, the public keys $(pk_1, ..., pk_m)$ of the authorized users and a keyword w, it outputs a searchable keyword ciphertext $C_1 = [A, B]$ with the public keys $(pk_1, ..., pk_m)$.

4) EnData($gp, pk_1, ..., pk_m, M$): This algorithm is executed by the data owner to encrypt data M. Takes as input the global parameter gp, the public keys $(pk_1, ..., pk_m)$ of the authorized users and data M, this algorithm outputs a ciphertext $C_2 = [D, E]$ with the public keys $(pk_1, ..., pk_m)$.

5) Trapdoor(gp, sk_i, pk_s, w): This algorithm is executed by an authorized user to generate a trapdoor. Takes as input the global parameter gp, the private key sk_i of the

authorized user, the public key pk_s of the cloud server and a keyword w, it outputs a trapdoor T_w for the keyword search over encrypted data.

6) $\mathrm{Test}(gp, C_1, T_w)$: This algorithm is executed by the cloud server to search the encrypted data with a keyword w. It takes as input the global parameter gp, a searchable keyword ciphertext C_1 and a trapdoor T_w, it outputs '1' if the ciphertext C_1 includes the keyword w, otherwise, this algorithm outputs '0'.

7) $\mathrm{DeData}(gp, A, C_2, sk_i)$: This algorithm is executed by a authorized user. Takes as input the global parameter gp, the ciphertexts (A, C_2) and the private key sk_i of the authorized user. This algorithm outputs the data M.

8) $\mathrm{Addition}(gp, u_l)$: This algorithm is executed by the data owner to add a new authorized user u_l to the system.

9) $\mathrm{Revocation}(gp, u_\gamma, sk_d, sk_s, pk_1, ..., pk_m)$: This algorithm is executed together by both the data owner and the cloud server to revoke a authorized user u_γ from the system. Takes as input the global parameter gp, the private key sk_d of the data owner, the private key sk_s of the cloud server and the public keys $(pk_1, ..., pk_m)$ of the authorized users. This algorithm revokes the authorized user u_γ from the system.

3.3 Security Definition

In this section, we define the security for the SEMU in the sense of semantic security. According to the definition of the SEMU, it consists of two encryption algorithms, i.e., $\mathrm{EnKeyword}(.)$ and $\mathrm{EnData}(.)$. Therefore, we first define the security for the $\mathrm{EnKeyword}(.)$ and the $\mathrm{EnData}(.)$, and then give the security definition of the SEMU.

Semantic Security of EnKeyword(.): The algorithm $\mathrm{EnKeyword}(.)$ is semantically secure against an adaptive chosen keyword attack if for any polynomial-time adversary, A_1, the advantage $Adv_{A_1}^{IND-CPA}(\lambda)$ is negligible in following Game 1:

1) Setup: The adversary A_1 generates his private/public key pairs (sk_{A_1}, pk_{A_1}), and gives pk_{A_1} to an algorithm B_1. The algorithm B_1 generates the private/public key pairs $(sk_1, pk_1), ..., (sk_m, pk_m)$ of authorized users and then gives $(pk_1, ..., pk_m)$ to the adversary A_1.

2) Phase 1 (Trapdoor queries): The adversary A_1 can adaptively asks B_1 for the trapdoor T_w for any keyword $w \in \{0,1\}^*$ of his choice.

3) Challenge: The adversary A_1 gives B_1 two keywords, w_0 and w_1 on which it wishes to be challenged. The restriction is that A_1 did not previously ask for the trapdoors T_{w_0} and T_{w_1}. B_1 picks a random $\beta_1 \in \{0,1\}$ and computes a ciphertext $C_1^* = [A^*, B^*]$, and sends C_1^* to A_1.

4) Phase 2 (Trapdoor queries): The adversary A_1 can adaptively asks B_1 for the trapdoor T_w for any keyword $w \in \{0,1\}^*$ of his choice as long as $w \neq w_0, w_1$.

5) Guess: The adversary A_1 outputs $\beta_1' \in \{0,1\}$ and wins the Game 1 if $\beta_1 = \beta_1'$.

We define the adversary A_1's advantage in breaking the $\mathrm{EnKeyword}(.)$ as

$$Adv_{A_1}^{IND-CPA}(\lambda) = |\Pr[\beta_1 = \beta_1'] - \frac{1}{2}| \quad (1)$$

Semantic Security of EnData(.): The algorithm EnData(.) is semantically secure against an adaptive chosen plaintext attack if for any polynomial time adversary, A_2, the advantage $Adv_{A_2}^{IND-CPA}(\lambda)$ is negligible in following Game 2:

1) Setup: Algorithm B_2 is given the public key pk_{A_2} of the adversary A_2. The algorithm B_2 generates the private/public key pairs $(sk_1, pk_1),...,(sk_m, pk_m)$ of authorized users and then gives $(pk_1,..., pk_m)$ to the adversary A_2.

2) Challenge: The adversary A_2 gives B_2 two plaintexts, M_0 and M_1 on which it wishes to be challenged. The restriction is that A_2 did not previously ask for the plaintexts M_0 and M_1. B_2 picks a bit $\beta_2 \in \{0,1\}$ and computes $(A^*, C_2^* = [D^*, E^*])$, and sends the ciphertexts (A^*, C_2^*) to A_2.

3) Guess: The adversary A_2 outputs $\beta_2^{'} \in \{0,1\}$ and wins the Game 2 if $\beta_2 = \beta_2^{'}$.

We define the adversary A_2's advantage in breaking the EnData(.) as

$$Adv_{A_2}^{IND-CPA}(\lambda) = |\Pr[\beta_2 = \beta_2^{'}] - \frac{1}{2}| \quad (2)$$

Semantic Security of the SEMU: Given the polynomial time adversary A consisting of the adversary A_1 and the adversary A_2, where the adversary A_1 initiates attacks on the EnKeyword() and the adversary A_2 initiates attacks on the EnData(.). We say that the SEMU scheme is semantically secure if the adversary A's advantage

$$Adv_{A}^{IND-CPA}(\lambda) = Adv_{A_1}^{IND-CPA}(\lambda) + Adv_{A_2}^{IND-CPA}(\lambda) \quad (3)$$

is negligible.

4. Construction

In this section, we present a concrete construction of the SEMU. The construction is based on both the Chinese Remainder Theorem and the PEKS [7].

Let $U = (u_1, u_2,...,u_m)$ be a set of authorized users, the SEMU includes following several sections.

1) GlobalSetup(λ): Choose a set of positive integers $(p_1, p_2,..., p_n)$ that are relatively prime in pairs and $p_1 < p_2 < ... < p_n (n > m > 2)$. Let G and G_T be two multiplicative cyclic groups of prime order p, and $g \in G$ be a generator. Where p is larger enough to guarantee that the discrete logarithm problem in G is intractable and $p > p_n$. Further let $\hat{e}$ be the bilinear map $\hat{e}: G \times G \to G_T$. Also assuming $H: G \to Z_{p_1}^*$, $H_1: \{0,1\}^* \to G$, and $H_2: G_T \to Z_{p_1}^*$ be hash functions that are modeled as a random.

Given a security parameter λ, this algorithm returns a global parameter $gp = (g, G, G_T, \hat{e}, p, p_1,..., p_m, H_1(.), H_2(.), H(.))$.

2) KeyGen(gp): Takes as input gp, this algorithm chooses randomly integers $(\zeta, \tau, k_i) \in Z_p^*$, and computes $pk_s = g^{\zeta} \bmod p$, $pk_d = g^{\tau} \bmod p$, $pk_i = g^{k_i} \bmod p$, where $i = 1,2,...,m$. Finally this algorithm outputs $sk_d = \tau$ to the data owner secretly, $sk_s = \zeta$ to the cloud server secretly, and $sk_i = k_i$ to authorized user u_i secretly. This algorithm publishes pk_s, pk_d, $pk_1 = (pk_1, p_1),..., pk_m = (pk_m, p_m)$ as public information.

3) EnKeyword($gp, pk_1,..., pk_m, w$): This algorithm is executed by the data owner to encrypt a keyword w. Takes as input gp, the public keys $(pk_1,..., pk_m)$ of the

authorized users and a keyword w. This algorithm chooses a random integer $r \in Z_p^*$, and computes

$$N = \prod_{i=1}^{m} p_i = \prod_{i \in U} p_i, \quad N_i = N / p_i, \quad N_i N_i^{'} = 1 \bmod p_i, \tag{4}$$

$$B_i = H_2(\hat{e}(pk_i^r, H_1(w))), \tag{5}$$

$$A = g^r \bmod \phi, \quad B = \sum_{i \in U} B_i N_i N_i^{'} \bmod N. \tag{6}$$

This algorithm outputs $C_1 = [A, B]$ to the cloud server.

4) EnData$(gp, pk_1, ..., pk_m, M)$: This algorithm is executed by the data owner to encrypt data M. Takes as input gp, the public keys $(pk_1, ..., pk_m)$ of the authorized users and data M, this algorithm chooses a random integer $x \in Z_{p_1}^*$, and encrypts data M as following

$$N = \prod_{i=1}^{m} p_i = \prod_{i \in U} p_i, N_i = N / p_i, \quad N_i N_i^{'} = 1 \bmod p_i, \tag{7}$$

$$\sigma_i = H(pk_i^r), \quad D = x \sum_{i \in U} \sigma_i N_i N_i^{'} \bmod N, \quad E = M \oplus H(x). \tag{8}$$

This algorithm outputs $C_2 = [D, E]$ to the cloud server. Where r is the value selected in the EnKeyword() phase.

5) Trapdoor(gp, sk_i, pk_s, w): This algorithm is executed by an authorized user to generate a trapdoor for the keyword w. Takes as input gp, the private key sk_i of an authorized user, the public key pk_s of the cloud server and a keyword w. This algorithm chooses a random integer $\alpha \in Z_p^*$, and computes

$$T_1 = g^{\alpha} \bmod \phi, \quad T_2 = H_1(w)^{sk_i} pk_s^{\alpha}, \quad T_3 = p_i \tag{9}$$

This algorithm outputs a trapdoor $T_w = [T_1, T_2, T_3]$ to the cloud server.

6) Test(gp, C_1, T_w): This algorithm is executed by the cloud server to search encrypted data with a keyword. It takes as input gp, the private key sk_s of the cloud server, the searchable keyword ciphertext C_1 and the trapdoor T_w. This algorithm computes

$$T = T_2 / T_1^{sk_s} = H_1(w)^{sk_i}, \tag{10}$$

$$B_i = B \bmod p_i = H_2(\hat{e}(pk_i^r, H_1(w))), \tag{11}$$

and checks if $B_i = H_2(\hat{e}(T, A))$. If the equality is satisfied, then this algorithm outputs '1'; otherwise, outputs '0'. If the algorithm outputs '1', then the cloud server sends (A, C_2) to the query user.

Correctness:

When assuming $(B_i = H_2(\hat{e}(pk_i^r, H_1(w))), A = g^r)$ are valid for the keyword w and the trapdoor $(T_1 = g^{\alpha} \bmod p, T_2 = H_1(w)^{sk_i} pk_s^{\alpha})$, the correctness of the Test(.) algorithm is verified as

$$T = T_2 / T_1^{sk_s} = H_1(w)^{sk_i}, \quad H_2(\hat{e}(T, A)) = H_2(\hat{e}(H_1(w)^{sk_i}, g^r) = B_i.$$

7) DeData(gp, A, C_2, sk_i): This algorithm is executed by an authorized user. Takes as input gp, the ciphertexts (A, C_2) and the private key sk_i of the authorized user. This algorithm computes

$$x = DH(A^{sk_i})^{-1} \bmod p_i, \quad M = E \oplus H(x). \tag{12}$$

Correctness:

When assuming $(D = x\sum_{i\in U}\sigma_i N_i N_i^{'} \bmod N, A = g^r, E = M \oplus H(x))$ are valid for the data M, the correctness of the DeData(.) algorithm is verified as

$$\sigma_i = H(pk_i^r), \quad DH(A^{sk_i})^{-1} \bmod p_i = xH(pk_i^r)H(pk_i^r)^{-1} = x$$

8) Addition(gp, u_l): This algorithm is executed by the data owner to add a new authorized user u_l to the system. Taking as input gp, this algorithm runs the KeyGen(.) algorithm to generate the public key $pk_l = (pk_l, p_l)$ of the new authorized user. This algorithm computes

$$B_l = H_2(\hat{e}(pk_l^r, H_1(w))), \tag{13}$$

$$\overline{N} = \prod_{i\in(U\cup u_l)} p_i, \quad \overline{N}_i = \overline{N}/p_i, \quad \overline{N}_i\overline{N}_i^{'} = 1 \bmod p_i, \tag{14}$$

$$\overline{B} = \sum_{i\in(U\cup u_l)} B_i\overline{N}_i\overline{N}_i^{'} \bmod \overline{N}, \tag{15}$$

and then outputs $\overline{C}_1 = [A, \overline{B}]$ to the cloud server.

9) Revocation$(gp, u_\gamma, sk_d, sk_s, pk_1, ..., pk_m)$: This algorithm is executed by both the data owner and the cloud server to revoke an authorized user u_γ from the system. The data owner chooses randomly integers $\tilde{r} \in Z_p^*$ and $\tilde{x} \in Z_{p_1}^*$, then computes

$$\tilde{N} = \prod_{i\in(U\setminus u_\gamma)} p_i \quad , \quad \tilde{N}_i = \tilde{N}/p_i, \quad \tilde{N}_i\tilde{N}_i^{'} = 1 \bmod p_i, \tag{16}$$

$$\tilde{\sigma}_i = H(pk_i^{\tilde{r}}), \quad \tilde{D} = \tilde{x}\sum_{i\in(U\setminus u_\gamma)} \tilde{\sigma}_i\tilde{N}_i\tilde{N}_i^{'} \bmod \tilde{N}, \tag{17}$$

$$\tilde{B}_i = H_2(\hat{e}(pk_i^{\tilde{r}}, H_1(w))), \quad \tilde{B} = \sum_{i\in(U\setminus u_\gamma)} \tilde{B}_i\tilde{N}_i\tilde{N}_i^{'} \bmod \tilde{N}, \tag{18}$$

$$\tilde{A} = g^{\tilde{r}} \bmod p, \quad R = H(x) \oplus H(\tilde{x}), \quad \mu = pk_s^{sk_d} \bmod p, \quad \nu = E_\mu(R). \tag{19}$$

then sends $(\tilde{C}_1 = [\tilde{A}, \tilde{B}], \tilde{C}_2 = [\tilde{D}, \tilde{E}], \nu)$ to the cloud server. Where $E_\mu(.)$ and $D_\mu(.)$ denote a symmetrical encryption and decryption operations with a key μ such as Advanced Encryption Standard(AES) algorithm.

Receiving $(\tilde{C}_1 = [\tilde{A}, \tilde{B}], \tilde{C}_2 = [\tilde{D}, \tilde{E}], \nu)$, the cloud server computes

$$\mu = pk_d^{sk_s} \bmod p, \quad R = D_\mu(\nu), \quad \tilde{F} = F \oplus R = M \oplus H(\tilde{x}). \tag{20}$$

and replaces the old (C_1, C_2) with the new $(\tilde{C}_1, \tilde{C}_2)$.

5. Security and Performance

5.1. Security

We now prove the security of the SEMU scheme under the BDH and the DDH assumption described above in the same way as [7].

Theorem 1: EnKeyword() is semantically secure against a chosen keyword attack in the Game 1 under random oracle model assuming the BDH is intractable.

Proof: Assume that A_1 is an adversary with advantage ε_1 in breaking the EnKeyword() against *IND-CPA*, and H_1 and H_2 are modeled as random oracles. Suppose A_1 makes at most $q_{H_2} > 0$ hash function queries to H_2 and at most $q_T > 0$ trapdoor queries. Then there is an algorithm B_1 that solves the BDH problem with

probability at least $\varepsilon_1^{'} = \varepsilon_1 / eq_T q_{H_2}$ and a running time $O(time(A_1))$, where e is the base of the natural logarithm. Notice that if the BDH assumption holds in G then $\varepsilon_1^{'}$ is a negligible function and consequently ε_1 must be a negligible function in the security parameter.

The algorithm B_1 is given $u_1 = g^a \in G, u_2 = g^b \in G, u_3 = g^c \in G$. Its goal is to output $v = \hat{e}(g,g)^{abc} \in G_T$. The algorithm B_1 simulates a challenger and interacts with the adversary A_1.

1) Setup: The adversary A_1 generates his public key such that $pk_{A_1} = u_1 = g^a$ and gives pk_{A_1} to the algorithm B_1. To simulate the public/private key pairs $(pk_1, sk_1),...,(pk_m, sk_m)$ of authorized users, the algorithm B_1 randomly chooses $(\delta_1,...,\delta_m) \in Z_p^*$, and then lets $pk_1 = (u_1^{\delta_1}, p_1),..., pk_m = (u_m^{\delta_m}, p_m)$. Finally, the B_1 gives $(pk_1,..., pk_m)$ to the adversary A_1.

H_1-Queries: To respond to H_1 queries, the algorithm B_1 maintains a H_1-list$=< w_i, h_i, a_i, c_i >$, which is initially empty. When the adversary A_1 queries H_1 at a point $w \in \{0,1\}^*$, B_1 checks if the query w_i already appears in the H_1-list. If it appears, B_1 responds with $H_1(w_i) = h_i \in G$. Otherwise, B_1 generates a random coin $c_i \in \{0,1\}$, so that $\Pr[c_i = 0] = 1/(q_T + 1)$. If $c_i = 0$, B_1 computes $h_i = u_2^{a_i} = g^{ba_i}$, where $a_i \in Z_p^*$ is a random value. If $c_i = 1$, B_1 computes $h_i = g^{a_i}$. B_1 adds $< w_i, h_i, a_i, c_i >$ to the H_1-list and responds to A_1 by setting $H_1(w_i) = h_i$.

H_2-Queries: To respond to H_2 queries, B_1 maintains a H_2-list$=< t_i, v_i >$, which is initially empty. When A_1 issues a query $t_j \in G_T$ to H_2, B_1 checks if t_j is already on the H_2-list. If so, B_1 responds to A_1 with $H_2(t_j) = v_j$. Otherwise, B_1 responds to a query for $H_2(t_j)$ by picking a random value $v_j \in Z_{p_1}^*$ for the t_j and setting $H_2(t_j) = v_j$, and adds $< t_j, v_j >$ to the H_2-list.

2) Trapdoor queries. When the adversary A_1 issues a query for the trapdoor corresponding to a keyword w_i, B_1 obtains $< w_i, h_i, a_i, c_i >$ from the H_1-list. If $c_i = 0$, B_1 reports failure and terminates. Otherwise, B_1 computes $T_w = u_1^{a_i \delta_i}$ and responds T_w to A_1. By the setting of $pk_i = g^{a\delta_i}$ in the Setup phase, and $H_1(w_i)^{sk_i} = g^{aa_i\delta_i}$ is satisfied, therefore T_w is the correct trapdoor for the keyword w under the public key $pk_i = g^{a\delta_i}$.

3) Challenge: The adversary A_1 gives B_1 two keywords w_0 and w_1, which A_1 wishes to be challenged on. The restriction is that A_1 did not previously ask for the trapdoors corresponding to the keyword w_0 and w_1. B_1 obtains $< w_0, h_0, a_0, c_0 >$ and $< w_1, h_1, a_1, c_1 >$ from the H_1-list. If $c_0 = 1$ and $c_1 = 1$, B_1 reports failure and terminates. Otherwise, B_1 chooses $\beta_1 \in \{0,1\}$ such that $c_{\beta_1} = 0$ and responds $C_1^* = [A^*, B^*]$ such that $A^* = u_3 = g^c \bmod p$, $B_i^* = z_i$, $B^* = \sum_{i \in U} B_i^* N_i N_i^{'} \bmod N$, where $N = \prod_{i \in U} p_i$, $N_i = N / p_i$, $N_i N_i^{'} = 1 \bmod p_i$, $z_i \in Z_{p_1}^*$ is a random value. Notice that $B_i^* = B \bmod p_i = z_i$ and $H_2(\hat{e}(T_{w_{\beta_1}}, A)) = H_2(\hat{e}((g^{ba_i})^{sk_i}, u_3)) = H_2(\hat{e}(g,g)^{abca_i\delta_i})$.

4) Guess: The adversary A_1 outputs $\beta_1^{'}$, B_1 picks a $<t_i, v_i>$ from the H_2-list, gives $t_i^{1/a_i\delta_i}$ as its guess for $\hat{e}(g,g)^{abc}$.

To show that B_1 correctly outputs $\hat{e}(g,g)^{abc}$ with a probability of at least $\varepsilon_1^{'}$, we analyze the probability that B_1 does not abort during the simulation.

We assume that the adversary A_1 does not ask for the trapdoor of the same keyword twice. Since no matter whether $c_i = 0$ or $c_i = 1$, the distribution on $H_1(w_i)$ is the same, and the value of c_i is independent of A_1's view, therefore, one trapdoor causes B_1 to report "failure" is with probability $1/(q_T+1)$. Since A_1 makes at most q_T trapdoor queries, the probability that B_1 does not abort in all trapdoor queries is $(1\text{-}1/(q_T+1)^{q_T}) \geqslant 1/e$.

The algorithm B_1 will abort during the challenge phase if A_1 issues the trapdoor queries about w_0 and w_1 with $c_0 = c_1 = 1$. Since A_1 has not queried for the trapdoor for w_0 and w_1, both c_0 and c_1 are independent of A_1's current view. In addition, the two values are independent of one another. Therefore, the probability that B_1 does not abort in Challenge is at least $1/q_T$. Since A_1 can never issue a trapdoor query for the keywords w_0 and w_1, the probability that B_1 does not abort is at least $1/eq_T$.

Following we show that B_1 outputs the solution to the given the BDH instance with probability at least ε_1 / q_{H_2}.

Let ε be the event that A_1 does not issue a query for either one of $H_2(\hat{e}(pk_i^r, H_1(w_0)))$ or $H_2(\hat{e}(pk_i^r, H_1(w_1)))$. Because C_1^* is an encryption of w_0 or w_1 is independent of A_1's view, A_1's output $\beta_1^{'}$ will satisfy $\beta_1 = \beta_1^{'}$ with probability at most $1/2$. By definition of the security game, we know that $|\Pr[\beta_1 = \beta_1^{'}] - (1/2)| \geq \varepsilon_1$, and $\Pr[\beta_1 = \beta_1^{'}] \leq (1/2) + (1/2)\Pr[\neg\varepsilon]$, $\Pr[\beta_1 = \beta_1^{'}] \geq (1/2) - (1/2)\Pr[\neg\varepsilon]$. Therefore, $\Pr[\neg\varepsilon] \geq 2\varepsilon_1$. Since B_1 will choose the correct pair in H_2-list with the probability at least $1/q_{H_2}$, then produces the correct answer with the probability at least ε_1 / q_{H_2}.

Because B_1 does not abort with probability at least $1/eq_T$, B_1's success probability overall is at least $\varepsilon_1 / eq_T q_{H_2}$ as required.

Theorem 2: EnData is semantically secure against a chosen plaintext attack in the Game 2 under assuming the Decisional Diffie-Hellman (DDH) is intractable.

Proof: Assume that A_2 is an adversary with advantage ε_2 in breaking the EnData(.) against *IND-CPA*. We construct a algorithm B_2 which has an advantage $\varepsilon_2^{'} = \varepsilon_2$ in solving the DDH problem in G.

The algorithm B_2 is given a random DDH challenge (g, g^a, g^b, g^c) and $H: G \to Z_{p_1}^*$, its goal is to guess $c = ab$ when $\beta_2 = 0$ and c is a random values from Z_p^* otherwise. Where β_2 is a bit $\in \{0,1\}$, and (a,b) are randomly and independently chosen from Z_p^*.

1) Setup: The algorithm B_2 is given the adversary A_2's public key $pk_{A_2} = g^a$. To simulate the public/private key pairs $(pk_1, sk_1),...,(pk_m, sk_m)$ of authorized users, the algorithm B_2 randomly chooses $(\xi_1,...,\xi_m) \in Z_p^*$, and lets $pk_1 = g^{a\xi_1},..., pk_m = g^{a\xi_m}$.

Finally, B_2 gives $(pk_1,...,pk_m)$ to the adversary A_2. Notice that $pk_i = g^{a\xi_i} = g^{sk_i}$, where $i = 1,2,...,m$.

2) Challenge: The adversary A_2 gives the algorithm B_2 two plaintexts, M_0 and M_1 which it wishes to be challenged on. B_2 picks a random $\beta_2 \in \{0,1\}$ and responds with the challenge ciphertext $(A^*, C_2^* = [D^*, E^*])$ such that

$$N = \prod_{i=1}^{m} p_i = \prod_{i \in U} p_i \quad , N_i = N / p_i , \quad N_i N_i^{'} = 1 \bmod p_i , \tag{21}$$

$$A^* = g^b , D = x \sum_{i \in U} H(g^{c\xi_i}) N_i N_i^{'} \bmod N , \quad E^* = M_{\beta_2} \oplus H(x) \tag{22}$$

and sends $(A^*, C_2^* = [D^*, E^*])$ to A_2. Where $c \in Z_p^*$ and $x \in Z_{p_1}^*$ are random values.

3) Guess: The adversary A_2 outputs $\beta_2^{'}$. B_2 guesses that $\beta_2 = 0$ if and only if $\beta_2 = \beta_2^{'}$.

If $\beta_2 = 0$, then $c = ab$ and the ciphertext (A^*, C_2^*) is a valid ciphertext. Notice that

$$D^* H(A^{*sk_i})^{-1} \bmod p_i = xH(g^{c\xi_i}) H((g^b)^{a\xi_i})^{-1} = xH(g^{c\xi_i}) H(g^{ab\xi_i})^{-1} .$$

Also when $\beta_2 = 0$, B_2 succeeds if and only if A_2 succeeds, which occurs with probability ε_2. Because β_2 is chosen uniformly at random and independent from A_2's view, and A_2 has no information about the value of β_2 from (A^*, C_2^*), so the probability that B_2 succeeds (A_2 succeeds when $\beta_2 = \beta_2^{'}$) is

$$\Pr[\beta_2 = \beta_2^{'}] = (1/2) + (1/2)\varepsilon_2 \tag{23}$$

Here $\Pr[\beta_2 = 0] = \Pr[\beta_2 = 1] = 1/2$. Because ε_2 is non-negligible, this shows that B_2 violates the assumption that the DDH is hard.

Theorem 3: The SEMU is semantically secure against a chosen keyword attack in the Game1 under the BDH assumption and a chosen plaintext attack in the Game 2 under the DDH assumption.

Proof: The theorem 3 is proved directly by theorem 1 and theorem 2. It shows that an *IND-CPA* adversary A on the SEMU with the advantage $\varepsilon = \varepsilon_1^{'} + \varepsilon_2^{'}$ as required.

Claim 1. If the EnKeyword() and the EnData(.) are secure encryption algorithms, then the SEMU can prevent revoked users from searching over encrypted data and accessing the data, thus achieving revocability.

Proof: A revoked user u_j will no longer be able to search over encrypted data because the revoked user u_j is removed from the ciphertext B. In the SEMU, when the cloud server processes the search queries of a user, it first obtains B_i from $B(B = \sum_{i \in U} B_i N_i N_i^{'} \bmod N)$, and verified whether $B_i = H_2(\hat{e}(T, A))$. Where

$$B_i = H_2(\hat{e}(pk_i^r, H_1(w))), \quad T = H_1(w)^{sk_i} , \quad A = g^r \tag{24}$$

If the user u_j is revoked from the system, his corresponding B_j should be removed from B. Without B_j, performing a search is impossible. Therefore, the revoked user u_j cannot search over encrypted data and access the data.

Claim 2. The SEMU can control queries.

Proof: In the SEMU, the private key of an authorized user is needed to generate a valid trapdoor, and anyone cannot obtain such private key except for the authorized user himself. Without the private key of the authorized user, no one can generate a valid trapdoor, and the cloud server or outside attackers cannot perform trapdoor test processing. Therefore, the SEMU can control queries.

Claim 3. The SEMU can provide query privacy.

Proof: The notion of query privacy in this paper is that secret information or data of authorized users may be leaked to an unauthorized party in the query process. In the SEMU, the cloud server or outside attackers cannot obtain any information about the keyword according to the trapdoor T_w because of the secure one-way hash function and the difficulty in computing discrete logarithms. Moreover, the cloud server or outside attackers cannot obtain any information from the ciphertexts C_1 and C_2. Although the anonymous information $T_3 = p_i$ of an authorized user is inevitably leaked to the cloud server, the cloud server cannot determine who corresponds to p_i, such that this information leakage to the cloud server is tolerable because no secret information is disclosed to the cloud server or to other users in the query phase. Therefore, the SEMU can provide query privacy.

Claim 4. The SEMU can achieve designated tester.

Proof: In the trapdoor test phase, the private key of the cloud server is needed to compute for test parameter T. Thus, outside attackers cannot perform trapdoor test processing without this private key. Therefore, only the designated cloud server can perform trapdoor test processing.

Claim 5. The SEMU can provide data confidentiality.

Proof: In the SEMU, data are encrypted by the data owner and then uploaded to the cloud server. Authorized users can recover the data using their private key. Both unauthorized users and the cloud server cannot recover the data because they do not possess the private keys of authorized users. Therefore, the SEMU can protect the data from both unauthorized users and the cloud server and provide data confidentiality.

Claim 6. The SEMU can provide query unforgeability.

Proof: The proof is straightforward. In the SEMU, the private key of an authorized user is used to generate valid keyword queries, such that neither other authorized users nor the cloud server can generate valid keyword queries on behalf of the authorized user unless the private key of this authorized user is disclosed. Therefore, the SEMU can provide query unforgeability.

5.2. Performance

In this section, we compare the performance of the SEMU with that of previous works.

We first view the ciphertext size of an encrypted keyword for m users. In scheme [6], the ciphertext size of a encrypted keyword is $(m+3)L$, where L is an element in G, and m is the number of users. In schemes [20, 21], the process of encrypting a keyword includes two parts: Enroll and GenIndex. The total ciphertext size in the two parts is $(m+1)L+\overline{L}$, where $\overline{L}$ denotes an element in G_T. However the total ciphertext size of the SEMU is $L+|N|$, where $|N|$ denotes N size, and $N = p_1 p_2 ... p_m$. We use the fact that the prime order p of G is sufficiently large to guarantee that the discrete logarithm problem in G is intractable. However, in the Chinese Remainder Theorem, $(p_1, p_2, ..., p_m)$ are positive integers that are relatively prime in pairs, so that $p > p_m > ... > p_1$. In other words, $(m+3)L > (L+|N|)$. Therefore, the ciphertext size of an encrypted keyword in the SEMU is shorter than that of the Schemes [6, 20, 21] in a multi-user setting. Notably, short ciphertext size is efficient for the storage of the cloud server because the server should store a large amount of ciphertexts for a number of users.

If we view the number of transmitting components between the data owner and the authorized users or the data owner and the cloud server as communication cost, no additional traffic is generated in the SEMU compared with the previous schemes [7, 9, 10,

11] in a single-user setting because the same number of ciphertexts are transmitted, although the content of packets are changed.

From the perspective of computational cost at an authorized user side, the SEMU mainly involves three exponentiation operations in generating the trapdoor phase. This computational cost is the same as that of schemes [7, 9, 10, 11] in a single-user setting and that of scheme [6] in a multi-user setting. In the decryption phase, the SEMU needs to perform one inverse computation, one exponentiation, and a multiplication operation, after which it performs one Exclusive or (*XOR*) operation. This computational cost is almost the same as that of previous schemes [7, 9, 10, 11] in a single-user setting.

In scheme [6], when a user wants to send the same message with keyword w to m users, the sender should encrypt the keyword w for m users by additionally computing B_j for $1 \le j \le m$ and the encryption of the message. When the cloud server searches the encryption with the keyword w, which a user u_i wants to search, using the trapdoor of u_i, the cloud server has to take $O(m)$ time to test trapdoor matching with $(B_1, B_2, ...B_m)$. In schemes [20, 21], upon receipt of a query trapdoor for a keyword w from the user u_j, the server first looks for the helper key hk_{u_j} of user u_j in $(hk_{u_1}, hk_{u_2}, ..., hk_{u_m})$ and then searches for the corresponding encryption with the keyword w by the hk_{u_j}. This process should take $O(m)$ time. However, in the case of the SEMU, the sender only needs to send B, which is computed by the Chinese Remainder Theorem, to the cloud server, rather than sending $(B_1, B_2, ...B_m)$. Thus, the cloud server only needs to take $O(1)$ time to test the trapdoor matching with $B_i = B \bmod p_i$, which is the same as that in a single-user setting [7, 9, 10, 11]. Therefore the SEMU is more efficient than schemes [6, 20, 21] in terms of the query time at the server side. Notably, less query time is important for both user and server.

The SEMU has very efficient overhead for the storage of both the server and users compared with schemes [6, 20, 21] in a multi-user setting. The SEMU requires only $n|C| + n(L + |N|)$ storage of a server for n data of m users, whereas scheme [6] and schemes [13, 14] respectively need $n|C| + n(m+3)L$ and $n|C| + n(mL + 2|Z_p^*|)$ storage, where $C = \text{EnData}(pk_1, pk_2, ..., pk_m, M)$. At the user side, the SEMU only needs to keep one private key sk_i, which is the same as that of schemes [7, 9, 10, 11] in a single-user setting and the scheme [6] in a multi-user setting, but fewer than that of schemes [20, 21] in a multi-user setting.

More importantly, the SEMU does not rely on shared keys to generate valid encrypted keyword queries. In the SEMU, each authorized user has only one private key, which is different from that of other users. Using their own private key, authorized users can generate valid encrypted keyword queries and decrypt the encrypted data without knowing the keys of other users. This condition significantly simplifies user revocation, and is more efficient and practical for a multi-user setting. Although each authorized user in schemes [20, 21] has a distinct query key to construct search queries, all authorized users share the same key to generate the index for the keyword w. which causes a weakness in that a revoked user can write records to the dataset. In addition, their schemes need to store a U-Hkey (containing user identities and their helper keys) on the cloud server, thereby increasing storage overhead on the cloud server. Furthermore, schemes [20, 21] use the fact that user revocation mainly depends on whether the cloud server removes the help key of the user from the U-Hkey list. Although we assume that the cloud service provider is honest-but-curious, the cloud service provider may not remove the help key hk_u of the user from the U-Hkey list when he can obtain some benefits.

Table 1. Comparison of Previous Schemes with the Proposed SEMU

	[10]	[6]	[20,21]	SEMU
Setting	single-user	multi-users	multi-users	multi-users
Designated Test	yes	no	no	yes
Controlling queries	yes	yes	yes	yes
query privacy	yes	yes	yes	yes
Revocation	no	no	incompletion	yes
Addition	no	no	yes	yes
Ciphertext size	$L+\lambda+\lvert\overline{C}\rvert$	$(m+3)L+\lvert C\rvert$	$(m+1)L+\lvert C\rvert+\overline{L}$	$L+\lvert N\rvert+\lvert C\rvert$
Trapdoor cost	3exp	3exp	exp	3exp
Test cost	exp+ BM	$(m+2)$BM	h_k + BM	exp+ BM+ Mod
User storage cost	$\lvert Z_p^*\rvert$	$\lvert Z_p^*\rvert$	$2\lvert Z_p^*\rvert$	$\lvert Z_p^*\rvert$
Server storage cost	$G+\lambda+\lvert\overline{C}\rvert$	$(m+3)L+\lvert C\rvert$	$(m+1)L+\lvert C\rvert+\overline{L}$	$L+\lvert N\rvert+\lvert C\rvert$

L: element in G; $\overline{L}$: element in G_T; exp: exponentials;

h_k: a keyed hash fuction; BM : bilinear pairings;

Mod: modulo p_i; m: the number of the authorized users;

$|N|$: N 's size, where $N = p_1 p_2 ... p_m$;

$|C|$: C 's size, where $C = \text{EnData}(pk_1, pk_2, ..., pk_m, M)$;

$|\overline{C}|$: $\overline{C}$'s size, where $\overline{C} = \text{EnData}(pk, M)$.

Moreover, when some users are added or removed, the SEMU performs scalable encryption at the cloud server side without shipping the outsourced data back to the data owner,thus reducing communication overhead. When the decrypted key is set up and updated, the SEMU scheme does not require any interaction between the data owner and the authorized users.

The aforementioned analysis shows that the SEMU minimizes the communication and storage overhead for both the server and users as well as has efficient computation overhead compared with previous schemes in a multi-user setting. The SEMU achieves features similar to that of most existing schemes in a single-user setting and does not downgrade because of supporting multiple users. Table 1 shows a comparison between other schemes and the proposed SEMU in terms of performance and features. In this work, we omit small computation operations such as multiplication in G or G_T.

6. Conclusion

In this paper, we have presented a searchable encryption scheme in a multi-user setting for cloud storage. When a user wants to share the same data with particular users, the proposed SEMU can be efficiently used for this application. The proposed SEMU avoids repeatedly encrypting the same data and the same keywords with the public keys of particular users using a general PEKS, thus reducing communication and storage overheads. The proposed SEMU performs scalable encryption on the cloud server without shipping the outsourced data back to the data owner when group membership is changed. We also have analyzed security and showed that the proposed SEMU is secure. For future work, we plan to extend our ideas for more types of queries.

Acknowledgments

This work was partially supported by National Natural Science Foundation of China under Grants 61272415, 61070164; Natural Science Foundation of Guangdong Province, China, under Grant S2012010008767; Science and Technology Planning Project of Guangdong Province, China, under Grant 2013B010401015. This work was also supported by the Zhuhai Top Discipline -Information Security.

References

[1] D.Song, and D. Wagner, and A.Perrig, "Practical Techniques for Searching on Encrypted Data", IEEE Symposium on Research in Security and Privacy, Berkeley, CA, **(2000)**, pp. 44-55.
[2] Z. Yang, S. Zhong, and R.N.Wright,"Privacy-preserving queries on encrypted data", Proc. of the 11th European conference on Research in Computer Security, **(2006)**; Hamburg, Germany.
[3] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, J.M. Lee, G. Neven, and H. Shi, "Searhable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions", The 25th Annual International Cryptology Conference, Santa Barbara, **(2005)**; California, USA.
[4] Y.C. Chang, and M. Mitzenmacher, "Privacy Preserving Keyword Searches on Remote Encrypted Data, "Applied Cryptography and Network Security, Third International Conference, ACNS, **(2005)**; New York, USA.
[5] R. Curtmola, J. Garay, S. Kamara and R. Ostrovsky, "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions", Proceedings of the 13th ACM Conference on Computer and Communications Security, **(2006)**; Alexandria, USA.
[6] Y. H. Hwang, and P. J. Lee, "Public Key Encryption with Conjunctive Keyword Search and Its Extension to a Multiuser System". Lecture Notes in Computer Science, vol. 4575, **(2007)**, pp. 2-22.
[7] D. Boneh, G. di Crescenzo, R. Ostrovsky and G.Persiano, "Public key encryption with keyword search", Eurocrypt'04, **(2004);** Interlaken, Switzerland.
[8] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited". Proceedings of Applied Cryptography and Information Security, **(2006);** Glasgow, UK.
[9] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption", Lecture Notes in Computer Science, vol. 4622, **(2007)**, pp. 535-552.
[10] 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", The Journal of Systems and Software, vol.837, **(2010)**, pp.63-771.
[11] B. Joonsang, S. N. Reihaneh, S. Willy, "Public key encryption with keyword search revisited", Lecture Notes in Computer Science, vol. 5072, **(2008)**, pp. 1249--1259.
[12] J. Ni, Y. Yu, Q. Xia, and L. Niu, "Cryptanalysis of Two Searchable Public Key Encryption Schemes with a Designated Tester", Journal of Information and Computational Science, vol.16, no.9, **(2012)**, pp. 4819-4825.
[13] M. Nishioka, "Perfect keyword privacy in PEKS systems", Lecture Notes in Computer Science, Vol.7496, **(2012)**, pp.175-192.
[14] C.Hu, and P.Liu, "An enhanced searchable public key encryption scheme with a designated tester and its extensions", Journal of Computers, vol.7, no.3, **(2012)**, pp.716-723.
[15] P. Xu, H. Jin, Q.H. Wu, and W. Wang, "Public-Key Encryption with Fuzzy Keyword Search: A Provably Secure Scheme under Keyword Guessing Attack", IEEE Transactions on Computers, vol.61, no.11, **(2013)**, pp. 2266-2277.
[16] Q. Liu, G. Wang, and J. Wu, "Efficient Privacy Preserving Keyword Search Scheme in Cloud Computing", The 12th IEEE International Conference on Computational Science and Engineering, **(2009)**; Vancouver, Canada.
[17] Q. Liu, G. Wang, and J. Wu, "Secure and privacy preserving keyword searching for cloud storage services", Journal of Networks and Computer Applications, vol.35, no.3, **(2012)**, pp. 927-933.
[18] L.Chen, and Q.Y.Wen,"A novel privacy preserving keyword searching for cloud storage", 2013 Eleventh Annual Conference on Privacy, Security and Trust (PST), Tarragona, **(2013)**.
[19] R. Curtmola, J.Garay, S. Kamara, and R.Ostrovskey,"Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions", Proceedings of ACM Conference on Computer and Communications Security, **(2006)**; Alexandria, USA.
[20] F.Bao, R.H.Deng, X.Ding, and Y.Yang, "Private Query on Encrypted Data in Multi-user Settings", Information Security Practice and Experience, 4th International Conference, **(2008)**; Sydney, Australia.
[21] Y.Yang, X.Ding, R.H.Deng, and F.Bao, "Multi-User Private Queries over Encrypted Databases", International Journal of Applied Cryptography Archive, vol.1, no.4, **(2009)**, pp. 309-319.
[22] M.Abdalla, M.Bellare, and P.Rogaway, "DHIES: an encryption scheme based on the Diffie-Hellman problem", Lecture Notes in Computer Science, vol. 2020, **(2001)**, pp. 143-158.
[23] D.Boneh, and M.Franklin, "Identity Based Encryption from the Weil Pairing", SIAM Journal of Computing, vol.32, no.3, **(2003)**, pp.586-615.

[24] Y.D. Luuu, M.L. Wu, "A fully public key traitor tracing scheme. WSEA Transaction on Circuits", vol.1, **(2002)**, pp.88-93.

