Distributed Multi-user, Multi-key Searchable Encryptions Resilient Fault Tolerance

Huafei Zhu(✉)

School of Computer and Computing Science, Zhejaing University City College, Hangzhou 310015, China
zhuhf@zucc.edu.cn

Abstract. In this paper, a construction of distributed multi-user, multi-key searchable encryptions is proposed and analyzed. Our scheme leverages a combination of the Shamir's threshold secret key sharing, the Pohlig-Hellman function and the ElGamal encryption scheme to provide high reliability with limited storage overhead. It achieves the semantic security in the context of the keyword hiding, the search token hiding and the data hiding under the joint assumptions that the decisional Diffie-Hellman problem is hard and the pseudo-random number generator deployed is cryptographically strong.

Keywords: ElGamal encryption scheme · Pohlig-Hellman function · Searchable encryptions · Shamir's threshold secret key sharing

1 Introduction

Often, different users possess data of mutual interest. The most challenging aspect of the data exchange lies in supporting of the data sharing over the encrypted database [12,18,28,32]. Searchable encryption is such a cryptographic primitive allowing for the keyword based content sharing managed and maintained by individual users. The state of the art research on searchable encryptions can be classified as the following two categories:

– Different data items (or documents, we do not distinguish the notion of data with that of the document throughout the paper as two nations are interactively used in many references cited here) outsourced are encrypted by a single key. The private information retrieval line of work [4,8,11,20] and the oblivious transfer line of work [25] fall in this category. Most of the research on searchable encryptions focused on the case when data is encrypted with same key [3,5,7,10,13,15,28,29,32] and more efficient solutions [1,9,19,21,24,26,27,30,31] have been proposed in recent years. The idea behind these constructions is that − to access a database, individually authorized user is issued a query key by the data owner and only the authorized users who have valid query keys can generate valid access queries which enable the database management server to process users' search queries without learning the keywords contained in the queries and the contents of the encrypted records.

M. Yung et al. (Eds.): INTRUST 2015, LNCS 9565, pp. 17–31, 2016.
DOI: 10.1007/978-3-319-31550-8_2

– Different data outsourced are encrypted with different keys. This problem was first studied by López-Alt et al. [17] leveraging the concept of fully homomorphic encryption schemes in which anyone can evaluate a function over data encrypted with different keys. The decryption requires all parties to come together and to run a multi-party computation protocol so that a client can retrieve all the keys under which the data was encrypted. As a result, the users need to do work proportional in the number of keys. Very recently, Popa and Zeldovich [22,23] proposed alternative design based on the bilinear map. Roughly speaking, a data owner in their model generates a set of documents $\{d_1, \ldots, d_\lambda\}$ and then an access structure is defined for all users. Each document d_j generated and the corresponding keyword set $\{w_j^{(1)}, \ldots, w_j^{(\eta_j)}\}$ extracted at session j ($sid = j$) will be encrypted by a fresh secret key k_j. The encrypted data and keyword set are then outsourced to a database server. A legitimate user is then given the corresponding encryption key k_j via a secure and authenticated channel established between the participants.

1.1 The Motivation Problem

Note that in the Popa and Zeldovich's scheme [22,23], user's primary key is assigned by the data owner while the corresponding delta keys are computed from the primary key and the specified encryption keys. The underlying access graph should be updated whenever a new document is outsourced to the server. The update procedure could be a difficult task if the frequency of data outsourcing is high since the size of stored delta keys can be proportional to the stored documents. Furthermore, when the deployed server is unreliable, as the case in modern data centers, redundancy must be introduced into the system to improve reliability against server failure (say, a complicated delta key recovery mechanism, or a Hadoop-like delta key duplication mechanism or a MapReduce-like distributed computing mechanism should be introduced). Since no countermeasure dealing with the server failure (or the delta key recovery) is known within the multi-user, multi-key searchable encryption framework, it is certainly welcome if one is able to provide such a counter-measure resilient the server failure.

1.2 This Work

This paper studies multi-user, multi-key searchable encryptions in the data owner controlled framework, where a data owner generates, manages and maintains all generated documents, documents encryption keys and keyword encryption keys (we distinguish the keys used to encrypt documents and to encrypt keywords throughout the paper). In our model, a database management system ($DBMS$), a data owner O, a set of users, a token generator (TG), a token extractor (TE) and a data extractor (DE) are introduced and formalized. All participants in our model run in the X-as-a-service model, where X = (token generation, token extraction, data extraction etc.):

- The $DBMS$ manages and maintains the system level parameters in the bulletin board model. The $DBMS$ should be able to add public information to the bulletin board so that all participants are able to obtain public information from the bulletin board. We stress that bulletin boards are used in any instance where public access to information is desired in the cryptography.
- A data owner O generates his/her data in a session and then extracts a set of keywords from the generated data accordingly (for example, by means of the inverted index program). To outsource the data generated at the ith session (the session id is denoted by $sid = i$), O first generates a secret document encryption key sk_i that will be used to encrypt the document. Then a public mask key K_i that will be used to mask the document encryption key sk_i and a keyword encryption key k_i that will be used to encrypt the generated keyword set are generated by means of a cryptographically secure random number generator.

  Let $t_i = (t_i^{(1)}, t_i^{(2)})$ be an output of cryptographically strong sustainable pseudo-random number generator at the ith session (say, the Barak-Halevi's (BH) scheme [2], or any other cryptographically strong pseudo-random number generator). $t_i^{(1)}$ is used to generate the mask encryption key $K_i \leftarrow g^{t_i^{(1)}}$ while $t_i^{(2)}$ is used to generate the keyword encryption key $k_i = H(t_i^{(2)})$, where $< g >= G \subseteq Z_p^*$, $|G| = 2q$, $p = 2q + 1$ is a large prime number and H: $\{0,1\}^* \rightarrow G$, is a cryptographically strong hash function. The auxiliary mask encryption string $t_i^{(1)}$ is shared among a set of n_D data extraction processors where any subset of m_D-out-of-the-n_D processors can be used to reconstruct K_i while the auxiliary keyword encryption string $t_i^{(2)}$ is kept secret by the data owner. The encrypted data are then outsourced to the $DBMS$.
- To support the keyword search procedure, the data owner must provide search structures for users. In many real-life situations, we don't believe that any given person can be trusted, yet it is reasonable to assume that the majority of people are trustworthy. Similarly, in on-line transactions, we may doubt that a given server can be trusted, but we hope that the majority of servers are working properly. Based on this assumption, we can create trusted entities, where the notion of token generators which manage and maintain a set of token generation processors, the notion of token extractors which manage and maintain a set of token extractor processors and the notion of data extractors which manage and maintain a set of data extraction processors are introduced. All keyword encryption keys are securely shared among token generation processors while all auxiliary mask strings are securely shared among the data extraction processors by means of the Shamir's secret sharing protocol.

An Overview of Processing. A processing of a keyword search comprises the following phases: the setup phase (including the data outsourcing); the query processing phase and the data extraction phase.

- In the setup phase, system parameters are generated for all participants; The data owner generates document encryption keys, mask keys and key-

word encryption keys for the initial data set. The auxiliary mask strings and keyword encryption keys are then securely distributed among a set of data extraction processors and a set of token generation processors respectively.
- In the query processing phase, a user first selects a keyword w, and then encrypts it by the ElGamal encryption scheme $(u = g^r, v = H(w)h^r)$, where g and H are common system strings and h is generated on-the-fly from an arbitrary subset of token extraction processors. The encrypted keyword $c = (u, v)$ is then sent to the $DBMS$ via the token generator server TG. The $DBMS$ and token extraction processors TEPs work together to extract the search token, and then retrieve data accordingly from the database server;
- In the data extraction phase, the retrieved ciphertexts such that each of which contains the specified keyword w are sent back to the user. The user then invokes m_D-out-of-n_D data extraction processors to decrypt the ciphertexts.

The Security. Intuitively, we expect the semantic security from multi-key searchable schemes:

- Keyword hiding: an adversary cannot learn the keyword one searches for;
- Token hiding: an adversary is not be able to distinguish between ciphertexts of two search tokens;
- Data hiding: if a document encryption key leaks, the contents of the other documents the user has access should not leak.

We are able to show that if the Diffie-Hellamn problem is hard and the underlying pseudo-random number generator is cryptographically strong, then the proposed multi-key searchable encryption is semantically secure.

What's New? We provide a new construction of multi-user, multi-key searchable encryptions based on the Pohlig-Hemman function and the ElGamal encryption scheme and a new method of achieving on-the-fly multi-party computation using the threshold multi-key encryptions. Our solution is different from the state-of-the-art solutions [17] leveraging the lattice based encryption scheme NTRU [14]), where a-priori bounded number of users should be defined since a decryption depends on the specified bound. Our solution is also different from the Popa and Zeldovich's methodology [22,23] which is leveraging the bilinear map based encryption scheme [6]), where a document encryption key should be distributed to all valid users. The proposed scheme leverages a combination of the Shamir's threshold secret key sharing, the Pohlig-Hellman function and the ElGamal encryption scheme to provide high reliability with limited storage overhead. It achieves the semantic security (the keyword hiding, the search token hiding and the data hiding) under the joint assumptions that the decisional Diffie-Hellman problem is hard and the pseudo-random number generator deployed is cryptographically strong.

The Road Map: The rest of this paper is organized as follows: in Sect. 2, syntax and security definition of multi-user, multi-key search protocols are presented;

An efficient construction of searchable encryptions based on the Pohlig-Hellman functions and the ElGamal encryption scheme is then proposed and analyzed; We show the proposed scheme is semantically secure in Sect. 3. We conclude this work in Sect. 4.

2 Syntax and Security Definition

This section consists of the following two parts: syntax and security definitions of multi-user, multi-key searchable encryptions.

2.1 Syntax of Multi-user, Multi-key Database Search

A multi-user, multi-key database search scheme comprises the following participants: a database management system, data owners, users, a token generator, a token extractor and a data extractor. Our scheme works in the bulletin board model where a participant can add his/her public information to it so that any other participant can use the public information available on the bulletin board. Notice that once the public information is outsourced to the bulletin board it cannot be deleted or modified by the original public information creator. The integrate of the outsourced public information is managed and maintained by a trusted certificate authority.

1. A database management system ($DBMS$) takes as input the security parameter 1^k and outputs system wide parameters `params` and a pair of public/secret keys (pk_{DB}, sk_{DB}). `params` is publicly known by all participants.
2. A set of data owners are involved in a searchable encryption scheme. Each data owner (O) takes as input the system parameters `params` and outputs a pair of public and secret keys (pk_O, sk_O);

   A procedure for outsourcing the encrypted data will be modelled as a session. In each session $sid = j$, O takes `params` and (pk_O, sk_O) as input and generates a triple of the document encryption key $sk_j^{(O)}$, the mask encryption key $K_j^{(O)}$ and the keyword encryption key $k_j^{(O)}$;
3. A token generator TG takes `params` as input and generates a pair of its own public and secret keys (pk_{TG}, sk_{TG}).

   A token generator manages and maintains a group of token generation processors (TGPs). Each TGP takes `params` as input and generates a pair of public and secret keys (pk_{TGP_i}, sk_{TGP_i}). To enable the keyword search over the outsourced encrypted data, O will distribute secret shares of a keyword encryption key $k_j^{(O)}$ generated at session j to n_G token generation processors by means of the Shamir's threshold secret key sharing scheme such that any subset of m_G (out-of-n_G) token generation processors can reconstruct the keyword encryption key $k_j^{(O)}$.
4. A token extractor (TE) takes `params` as input and generates a pair of its own public and secret keys (pk_{TE}, sk_{TE}). TE manages and maintains a group of token extraction processors (TEPs). Each TEP takes `params` as input and generates a pair of public and secret keys (pk_{TEP_i}, sk_{TEP_i}) for $i = 1, \ldots, n_E$.

5. A data extractor DE takes `params` as input to generate a pair of public and secret keys (pk_{DE}, sk_{DE}). DE manages and maintains a group of n_D data extraction processors $(DEP_1, \cdots, DEP_{n_D})$. Each data extraction processor DEP takes `params` and pk_{DE} as input to generate a pair of public and secret keys (pk_{DEP_i}, sk_{DEP_i}); To enable users to extract the retrieved encrypted documents, O will distribute shares of the auxiliary mask key string of $K_j^{(O)}$ generated at session j to n_D data extraction processors by means of the Shamir's threshold secret key sharing scheme such that any subset of m_D (out of n_D) data extraction processors can reconstruct the mask encryption key $K_j^{(O)}$.
6. A set of users are involved in the searchable encryption. Each user U (or an querier) takes `params` as input to generate a pair of public and secret keys (pk_U, sk_U). A valid user is allowed to submit a query to the $DBMS$. On input a keyword $w \in W$, U encodes w and then sends the resulting codeword $c(w)$ to TG who generates a valid search token $t(w)$ by means of the multi-party computations. The resulting search token $t(w)$ is then sent to the $DBMS$ who collaborates with TE to extract the search token and then sends back to U all retrieved encrypted data $\mathcal{D}$ such that each $D \in \mathcal{D}$ contains w.

2.2 Security of Multi-key Database Search

We formalize security requirements specified in Sect. 1 with following games: keyword hiding, token hiding and data hiding that express these goals. One holistic security definition would be a stronger guarantee, but that greatly complicate the designs and proofs. Nevertheless, the separate definitions also capture the desired security goals.

Keyword Hiding Game. The keyword hiding game is between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ on security parameter 1^k and pubic parameter `params`

- $\mathcal{C}$ invokes the $DBMS$ which takes as input 1^k to output `params` and provides `params` to $\mathcal{A}$;
- $\mathcal{C}$ invokes n token extraction processors each of which takes as input `params` to output n pairs of public and secret keys (pk_{TEP_i}, sk_{TEP_i}) $(i = 1, \cdots, n)$. $\mathcal{C}$ then provides pk_{TEP_i} $(i = 1, \cdots, n)$ to $\mathcal{A}$.
- Let Δ be an arbitrary subset of $\{1, \cdots, n\}$ containing m public key indexes. Let $pk_{TEP}^{(\Delta)}$ be a public key computed from the selected m public keys.
- Let w_0 and w_1 be two keywords selected by $\mathcal{A}$. The challenger selects a bit $b \in_R \{0, 1\}$ uniformly at random. Let $c = E_{pk_{TEP}^{(\Delta)}}(w_b)$ for $b \in_R \{0, 1\}$. The adversary is given (Δ, c) and outputs a guess $b' \in \{0, 1\}$.

Definition 1 *(Keyword Hiding). We say that the communication between users and the token extractor (and the token extraction processors) is keyword hiding if for any polynomial time adversary $\mathcal{A}$ that given (Δ, c), where $c = E_{pk_{TEP}^{(\Delta)}}(w_b)$ for $b \in_R \{0, 1\}$, outputs a guess b', the following holds: $Pr[b = b'] - 1/2|$ is at most a negligible amount:*

Token Hiding Game. The token hiding game is between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ on security parameter 1^k and pubic parameter `params`

- $\mathcal{C}$ invokes the $DBMS$ which takes as input 1^k to output `params` and provides `params` to $\mathcal{A}$;
- $\mathcal{C}$ invokes the data owner O which takes as input the system parameters `params` to output a pair of public and secret keys (pk_O, sk_O). The adversary $\mathcal{A}$ is given pk_O; In each session, say session j, O takes `params` as input and generates a keyword encryption key k_j;
- The adversary $\mathcal{A}$ invokes the database management server which takes as input `params` to output (pk_{DB}, sk_{DB}). $\mathcal{A}$ obtains pk_{DB};
- $\mathcal{C}$ invokes n token generation processors each of which takes as input `params` to output n pairs of public and secret keys (pk_{TGP_i}, sk_{TGP_i}) $(i = 1, \cdots, n)$. $\mathcal{C}$ then provides pk_{TGP_i} $(i = 1, \cdots, n)$ to $\mathcal{A}$;
- Let $k_j^{(l)}$ be a secret share of k_j shared by TGP_l for $j = 1, \cdots, \kappa$ and $l = 1, \cdots, n$, where κ is the number of k_j shared among the token processors so far. Let w be an input to the token generation processor TGP_j. The challenger then invokes TGP_j which takes $k_j^{(l)}$ as an input and then outputs $c_j^{(l)}$. The resulting ciphertext $c_j^{(l)}$ is then sent to $\mathcal{A}$ who computes the corresponding coefficient $\alpha_j^{(l)}$ of the Lagrange Interpolation Formula to output an encryption $TG(k_j, w)$ of the search token.
- The challenger then selects a bit $b \in \{0, 1\}$ uniformly at random and then given $\{c_j^{(l)}\}_{l=1}^n$ and $TG_1 = (TG(k_j, w))$ if $b = 1$ and $\{c_j^{(l)}\}_{l=1}^n$ and a random string $TG_0 \in G$ if $b = 0$. The adversary outputs a guess $b' \in \{0, 1\}$.

Definition 2 *(Token Hiding). We say that the communication between users and the token generator who manages and maintains the token generation processors is token hiding if for any polynomial time adversary $\mathcal{A}$ that given $TG_b(k, c)$ and $\{c_j^{(l)}\}_{l=1}^n$, outputs a guess b', the following holds: $Pr[b = b'] - 1/2|$ is at most a negligible amount:*

Data Hiding Game. The data hiding game is between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ on security parameter 1^k and pubic parameter `params`

- $\mathcal{C}$ invokes the $DBMS$ which takes as input 1^k to output `params` and (pk_{DB}, sk_{DB}). $\mathcal{C}$ provides `params` and pk_{DB} to $\mathcal{A}$;
- $\mathcal{C}$ invokes the data owner O which takes as input the system parameters `params` to output a pair of public and secret keys (pk_O, sk_O). The adversary $\mathcal{A}$ is given pk_O;

  In each session, say session j, O takes `params` as input and generates a mask encryption key K_j. The data m_j is then encrypted under K_j. The resulting ciphertext c_j is outsourced to the $DBMS$;
- $\mathcal{C}$ invokes n data extraction processors each of which takes as input `params` to output n pairs of public and secret keys (pk_{DEP_i}, sk_{DEP_i}) $(i = 1, \cdots, n)$. $\mathcal{C}$ then provides pk_{DEP_i} $(i = 1, \cdots, n)$ to $\mathcal{A}$;

Let $K_j^{(l)}$ be the share of auxiliary mask key string of K_j by DEP_l for $j = 1, \cdots, \kappa$ and $l = 1, \cdots, n$, where κ is the number of K_j shared among the data extraction processors so far.

To decrypt a ciphertext c_j, the data extractor first selects an arbitrary subset of $\{1, \cdots, n\}$ that contains arbitrary m public key indexes of the data extraction processors. Let Δ be the selected subset. The data extractor then invokes $DEP_j \in \Delta$ which takes $K_j^{(l)}$ as an input to output $c_j^{(l)}$. The resulting $c_j^{(l)}$ is then sent back to DE who computes the corresponding coefficient $\alpha_j^{(l)}$ of the Lagrange Interpolation Formula to output the plaintext m.

– Let m be a target document selected by $\mathcal{A}$. For the given m, $\mathcal{C}$ selects a random bit $b \in_R \{0, 1\}$. Let $c_b = E_{DEP}(m)$ for $b \in_R \{0, 1\}$ and $c_{\overline{b}} = E_{DEP}(1^{|m|})$ (an encryption of the dummy document). The adversary is given (c_0, c_1), and outputs a guess $b' \in \{0, 1\}$.

Definition 3 *(Data Hiding). We say that the communication between the owner and the data extractor is data hiding if for any polynomial time adversary $\mathcal{A}$ that given (c_0, c_1), outputs a guess b', the following holds: $Pr[b = b'] - 1/2|$ is at most a negligible amount:*

Definition 4 *(Semantic Security). We say that a multi-user, multi-key searchable encryption system is semantically secure if it achieves the keyword hiding, token hiding and data hiding properties.*

3 The Construction

We now present a construction of multi-user, multi-key searchable encryptions in the bulletin board model that realizes the functionalities described in Sect. 3.1. We analyze its security in Sect. 3.2.

3.1 The Description

Our protocol comprises the following phases: the setup phase, the outsourcing phase, the processing phase and the extraction phase. The details of each phase are depicted below

The setup phase

– On input a security parameter parameter 1^k, $DBMS$ output system parameters `params`: a large safe prime number p such that $p = 2q + 1$, p and q are prime numbers, $|p| = k$ together with a cyclic group G of order q. Let g be a random generator of G.

  $DBMS$ then takes `params` as input to generate a pair of public and secret keys (pk_{DB}, sk_{DB}), where $pk_{DB} = (g, h_{DB})$, $h_{DB} = g^{x_{DB}}$ and $sk_{DB} = x_{DB}$.

– A data owner O takes `params` as input and generates a pair of public and secret keys (pk_O, sk_O) where $pk_O = (g, h_O)$, $h_O = g^{x_O} \bmod p$ and $sk_O = x_O$ (in the following discussions, we simply assume that (pk_O, sk_O) is suitable for both the data encryption and data attestation).

– A data extractor DE takes `params` as input to generate a pair of public and secret keys (pk_{DE}, sk_{DE}), where pk_{DE} =(g, h_{DE}), $h_{DE} = g^{x_{DE}}$ and $sk_{DE} = x_{DE}$.

  DE in our model manages and maintains n_D data extraction processors (DEP_1, $\cdots$, DEP_{n_D}). Each extraction processor DEP_i generates its own public and secret key pairs $pk_{DEP_i} = (g, h_{DEP_i})$, $h_{DEP_i} = g^{x_{DEP_i}}$ and $sk_{DEP_i} = x_{DEP_i}$ independently.

  To enable users to obtain the corresponding plaintexts from the retrieved encrypted data, O delegates her decryption right to data extraction processors by invoking the Shamir's (m_D, n_D)-secret-key sharing algorithm such that any m_D combinations of shares is sufficiently to reconstruct the mask encryption key by applying the Lagrange Interpolation Formula.

  For simplicity, we assume that a secure (private and authenticated) channel has been established between O and DE and secure channels between DE and DEP_i respectively (such a secure channel assumption can be eliminated trivially under the standard PKI assumption).
– A token generator TG takes `params` as input to output a pair of public and secret keys (pk_{TG}, sk_{TG}), where pk_{TG} =(g, h_{TG}), $h_{TG} = g^{x_{TG}}$ and $sk_{TG} = x_{TG}$.

  In our model, TG manages and maintains n_G token generation processors ($TGP_1, \cdots, TGP_{n_G}$). Each token generation processor TGP_i generates its own public and secret key pairs pk_{TGP_i}= (g, h_{TGP_i}), h_{TGP_i} =$g^{x_{TGP_i}}$ and $sk_{TGP_i} = x_{TGP_i}$. Again, we assume that a secure channel has been established between TG and O (TG and TGP_i respectively).
– A token extractor TE takes `params` as input to output a pair of public and secret keys (pk_{TE}, sk_{TE}), where pk_{TE} =(g, h_{TE}), $h_{TE} = g^{x_{TE}}$ and $sk_{TE} = x_{TE}$.

  In our model, TE manages and maintains n_E token extraction processors ($TEP_1, \cdots, TEP_{n_E}$). Each token extraction processor TEP_i generates its own public and secret key pairs pk_{TEP_i}= (g, h_{TEP_i}), h_{TEP_i} =$g^{x_{TEP_i}}$ and $sk_{TEP_i} = x_{TEP_i}$. We assume that a secure channel has been established between TE and $DBMS$ (TE and TEP_i respectively).
– A user U takes `params` as input to generate a pair of public and secret keys (pk_U, sk_U), where pk_U =(g, h_U), $h_U = g^{x_U}$ and $sk_U = x_U$.

The outsourcing phase

In the outsourcing phase, the search structure of the outsourced data and keyword is defined as follows: let BH be the Barak-Helavi's (or any other cryptographically strong) pseudo-random number generator [2] and H: $\{0,1\}^* \rightarrow G$ be a cryptographically strong hash function. We view a data outsourcing activity as a session in the following depiction.

– At session $sid = i$, on input d_i, the data owner O first selects a document encryption key sk_i with suitable length (say 128-bit or 256-bit for AES) and then invokes the BH pseudo-random number generator to output a pair of

mask key K_i and keyword encryption key k_i. The document encryption key sk_i is encrypted by the mask encryption key K_i computed on the fly: let s_{i-1} be the previous internal state of the BH pseudo-random number generator at session $sid = i-1$. To generate a pair of mask key K_i and keyword encryption key k_i for d_i, O invokes the BH scheme which takes s_{i-1} as input to output (s_i, t_i), where s_i is the internal state at session $sid = i$ and t_i is the current output. O then parses t_i to two parts $(t_i^{(1)}, t_i^{(2)})$ and then enciphers the first part $t_i^{(1)}$ by computing $K_i = g^{t_i^{(1)}} \bmod p$ and the second part $t_i^{(2)}$ by computing $k_i = H(t_i^{(2)})$, where H is a cryptographically hash function (in essence, we view H as a random oracle). K_i is called the mask key that will be used to encrypt the document encryption key sk_i while k_i is called the keyword encryption key. The first part $t_i^{(1)}$ is called the auxiliary mask string while the second part $t_i^{(2)}$ is called the auxiliary keyword encryption string.

– O extracts the keyword sets $W_i = \{w_i^{(1)}, \cdots, w_i^{(\gamma_i)}\}$ from d_i by means of the inverted index.

To encrypt a keyword $w \in W_i$, O invokes the Pohlig-Hellman function to compute $c(k_i, w) \leftarrow H(w)^{k_i} \bmod p$. $c(k_i, w)$ is then outsourced the $DBMS$. Let $c(k_i, W_i)$ be an encryption of the keyword set W_i under the keyword encryption key k_i at session $sid = i$.

To enable users to search keywords, the data owner O provides a search structure by sharing k_i among n_G token generation processors managed and maintained by the token generator TG. To distribute secret shares to $TGPs$, O invokes the Shamir's threshold secret key sharing protocol below:

- O randomly selects a polynomial $f(x) = f_0 + f_1 x + \cdots + f_{m_G-1} x^{m_G-1} \pmod q$, where $f_0 = k_i$ and $k_i^{(l)} \stackrel{def}{=} f(TEP_l))$ $(i = 1, \ldots, n_G)$;
- TEP_l is given $f(TEP_l)$, $l = 1, \ldots, n_G$.

– To outsource d_i, O first invokes a cryptographically strong block cipher say, Advanced encryption standard AES which takes sk_i and d_i as input to generate the ciphertext $c(sk_i, d_i)$.

Let $c(K_i, sk_i) = (g^r, sk_i \times K_i^r)$ be an encryption of the secret key sk_i under the mask encryption key K_i. The corresponding auxiliary mask encryption string is shared among n_D data extraction processors $DEP_1, \ldots, DEP_{n_D}$ (again O applies the Shamir's threshold scheme to the auxiliary string $t_i^{(1)}$) such that a combination of m_D-out-of-n_D shares can be used to reconstruct the the auxiliary key $t_i^{(1)}$ such that $K_i = g^{t_i^{(1)}}$.

The resulting ciphertext $(c(k_i, W_i), c(K_i, sk_i), c(sk_i, d_i))$ are then sent to the $DBMS$.

The query processing phase

In the query processing phase, a computation of individual user is depicted below

– The input of a user U is a keyword w together with a description of token extraction processors whose public keys are denoted by (g, h_{TEP_1}), $\cdots$, $(g,$

$h_{TEP_{n_E}}$). U then selects m_E-out-of-n_E token extraction processors uniformly at random. Let $(g, h_{TEP_{i_1}}), \cdots, (g, h_{TEP_{i_{m_E}}})$ be m_E selected token extraction processors. Let $\Delta = (i_1, \cdots, i_{m_E})$ and $h = \Pi_{j=1}^{m_E} h_{TEP_{i_j}}$.

To hide the selected keyword w, U selects a string $r \in Z_q$ uniformly at random and computes $u = g^r \bmod p$, $v = H(w)h^r \bmod p$. Let $c = (u, v)$ and $\widetilde{\Delta}$ be an encryption of Δ under the $DBMS$' public-key pk_{DB}, i.e., $\widetilde{\Delta} = E_{pk_{DB}}(\Delta)$. Let $m_U = (\widetilde{\Delta}, c)$. Let $\sigma_U(H(m_U)$ be a signature of m_U attested by the user U. $(\sigma_U(H(m_U)), m_U)$ is then sent by the user to the token generator.

– Upon receiving $(\sigma_U(H(m_U)), m_U)$, the token generator TG who manages n_G token generation processors first checks the validity of the received message (recall that all computations are running in the X-as-a-service model). If the signature is invalid, then TG rejects the received message; otherwise, TG selects m_G-out-of-the-n_G token generation processors uniformly at random and then forwards $c = (u, v)$ to the selected m_G processors $\{TGP_{i_1}, \cdots, TGP_{i_{m_G}}\}$. Let $k_j^{(l)}$ be a secret share of k_j by TGP_l for $j = 1, \cdots, \kappa$ and $l = 1, \cdots, n_G$, where κ is the number of k_j shared so far. For each share $k_j^{(l)}$, TGP_l performs the following computations for each share $k_j^{(l)}$:
  - $u_j^{(l)} = u^{k_j^{(l)}} \bmod p$;
  - $v_j^{(l)} = v^{k_j^{(l)}} \bmod p$.

  TGP_j then sends $c_j^{(l)}$ back to TG, where $c_j^{(l)} = (u_j^{(l)}, v_j^{(l)})$;

  Upon receiving $c_j^{(l)}$, TG computes the corresponding coefficient $\alpha_j^{(l)}$ of the Lagrange Interpolation Formula and then computes $u_j = \prod_{l=i_1}^{i_{m_T}} (u_j^{(l)})^{\alpha_j^{(l)}}$ and $v_j = \prod_{l=i_1}^{i_{m_T}} (v_j^{(l)})^{\alpha_j^{(l)}}$. One can verify that $u_j = u^{k_j} = g^{rk_j}$ and $v_j = v^{k_j} = H(w)^{k_j} h^{rk_j}$. Let $c_j = (u_j, v_j)$ and $m_{TG} = (\sigma_U(H(m_U)), \widetilde{\Delta}, \{c_j\}_{j=1}^{\kappa})$. TG then generates a signature σ_{GT} on the message m_{TG} (this task is trivial under the standard PKI assumption) and then sends (m_{TG}, σ_{TG}) to the $DBMS$.

– Upon receiving (m_{TG}, σ_{TG}), $DBMS$ checks the validity of the received message. If it is invalid, then terminates the protocol; otherwise, it decrypts $\widetilde{\Delta}$ to get Δ and broadcasts $\{u_j\}_{j=1}^{\kappa}$ to all token extractors within Δ via a secure multi-cast channel (such a multi-cast channel can be efficiently implemented in the context of group communication protocol).

  Each token extraction processor TEP_l computes $\widetilde{u}_j^{(l)} = u_j^{x_l}$ and sends $(TEP_j^{(l)}, \widetilde{u}_j^{(l)})$ to TE, where $TEP_j^{(l)}$ stands for the jth input processed by the lth token extraction processor. The computing results $\{TEP_j^{(l)}, \widetilde{u}_j^{(l)}\}_{l=i_1}^{i_{m_E}}$ are then sent back to the $DBMS$.

– Upon receiving $\{TEP_j^{(l)}, \widetilde{u}_j^{(l)}\}_{l=i_1}^{i_{m_E}}$ from all token extraction processors, the $DBMS$ computes $\widehat{u}_j = \prod_{l=1}^{m_E} \widetilde{u_j}$. One can verify that $\widehat{u}_j = h^{rk_i}$. As a result, given $\widehat{u}_i$ and $\{v_j\}_{j=1}^{m_E}$, $DBMS$ is able to extract the search token $H(w)^{k_i}$. Let $\mathcal{D}$ be a set of encrypted documents so that $c(w) \in D$ for each $D \in \mathcal{D}$. $DBMS$ then sends $\mathcal{D}$ to the user U;

The data extraction phase

– Upon receiving $\mathcal{D}$, the user performs a decryption of message via the Shamir's threshold decryption protocol to obtain sk_i. Once obtains sk_i, the user U can decrypt the received ciphertexts.

This ends the description of our protocol.

3.2 The Proof of Security

The correctness of the protocol can be verified step by step and hence omitted. The rest of this section is to provide a proof of security defined in Sect. 3.

Lemma 1. *Let Δ be an arbitrary subset of $\{TEP_1, \cdots, TEP_m\}$, suppose at least one of the selected token extraction processor is honest then the proposed scheme is keyword hiding assuming that the decisional Diffie-Hellman problem is hard.*

Proof. $\mathcal{C}$ invokes n token extraction processors each of which takes as input `params` to output n pairs of public and secret keys (pk_{TEP_i}, sk_{TEP_i}) $(i = 1, \cdots, n)$. $\mathcal{C}$ then provides pk_{TEP_i} $(i = 1, \cdots, n)$ to $\mathcal{A}$. Let Δ be an arbitrary subset of $\{1, \cdots, n\}$ containing m public key indexes. Without loss of the generality, we assume that $\Delta = \{(g, pk_{TEP_1}), \ldots, (g, pk_{TEP_m})\}$, where $pk_{TEP_i} = g^{x_{TEP_i}}$. The challenger is allowed to corrupt $m-1$ token extraction processors and obtains the corresponding secret keys x_{TEP_i} for $i = 1, \ldots, m-1$. The challenger's target is to break the mth instance of the ElGamal encryption scheme.

Let h $(= pk_{TEP}^{(\Delta)}) = pk_{TEP_1} \times \cdots \times pk_{TEP_m}$. Let w_0 and w_1 be two keywords output by the adversary which is also known to the challenger. Let $c \leftarrow (g^r, H(w_b) \times h^r)$ (generated by the semantic security game of the underlying encryption scheme (g, pk_{TEP_m})). The challenger then forwards (w_0, w_1) and c to the adversary $\mathcal{A}$. The adversary outputs a guess $b' \in \{0, 1\}$. The challenger outputs what the adversary outputs.

Given $c = (g^r, H(w_b)h^r)$ and x_{TEP_i} for $i = 1, \ldots, m-1$, the challenger is able to compute $(g^r, H(w_b)h_m^r)$ where $h_m = pk_{TEP_m}$. Hence if $Pr[b = b'] - 1/2|$ is a non-negligible then the mth instance of the underlying ElGamal encryption scheme is not semantically secure which contradicts the decisional Diffie-Hellman assumption.

Lemma 2. *Let $\Delta = \{TGP_1, \cdots, TGP_m\}$ be an arbitrary subset of total token generation processors. Suppose at least one of the selected token extraction processor is honest then the proposed scheme is token hiding assuming that the decisional Diffie-Hellman problem is hard and the underlying BH pseudo-random number generator is cryptographically strong.*

Proof. Let w be a keyword selected by $\mathcal{A}$. Let k_j be the key used to generate the search token at session $sid = j$. Assuming that up to $(m-1)$ token generation processors are corrupted and the adversary obtains the corresponding shares,

say $(k_j^{(1)}, \ldots, k_j^{(m-1)})$. The mth token generation processor remains honest at the session $sid = j$. Let $k_j^{(l)}$ be a secret share of k_j by TGP_l for $l = 1, \cdots, n_G$. Notice that the only knowledge applied to $H(w)$ is $k_j^{(l)}$. For fixed set of shares $(k_j^{(1)}, \ldots, k_j^{(m-1)})$, there is a one to one mapping (the Lagrange interpolation formula) between $k_j^{(l)}$ and k_j. As a result, $H(w)^{k_j}$ is random from the point view of the adversary if the underlying pseudo-random number generator is secure. As a result, the only information leaked is the computation of $H(w)^{k_j^{(l)}}$. Since G is a cyclic group, it follows that $H(w) = g^{r_w}$ for some $r_w \in [0, q-1]$. Thus, $(g, H(w), g^{k_j^{(l)}}, H(w)^{k_j^{(l)}})$ is a Diffie-Hellman quadruple that is indistinguishable from the random quadruple. As a result, the advantage $Pr[b = b'] - 1/2|$ that the adversary outputs a correct guess is at most a negligible amount.

Lemma 3. *Let $\Delta = \{DEP_1, \cdots, DEP_m\}$ be an arbitrary subset of total data extraction processors. Suppose at least one of the selected data extraction processor is honest then the proposed scheme is data hiding assuming that the decisional Diffie-Hellman problem is hard.*

Proof. $\mathcal{C}$ invokes the data owner O which takes as input the system parameters `params` to output a pair of public and secret keys (pk_O, sk_O). The adversary $\mathcal{A}$ is given pk_O; At the session i, O takes `params` as input and generates a mask key K_i such that $K_i = t_i^{(1)}$, where the auxiliary string $t_i^{(1)}$ is the first part of the output t_i generated by the BH pseudo-random generator.

$\mathcal{C}$ invokes n data extraction processors each of which takes as input `params` to output n pairs of public and secret keys (pk_{DEP_i}, sk_{DEP_i}) $(i = 1, \cdots, n)$. $\mathcal{C}$ then provides pk_{DEP_i} to $\mathcal{A}$. Let $K_i^{(l)}$ be the share of $t_i^{(1)}$ via the Lagrange interpolation formula for $l = 1, \cdots, n$. Let Δ be an arbitrary subset of $\{DEP_1, \cdots, DEP_n\}$ and m_0 and m_1 be two documents all selected by $\mathcal{A}$. We assume that the adversary can corrupt up to $(m-1)$ data extraction processors and obtains the corresponding secret shares $K_i^{(l)}$ for $l = 1, \ldots, m-1$. The data m_b is then encrypted under K_i, i.e., $c_i = (u_i, v_i)$, where $u_i = g^r$ and $v_i = m_b K_i^r$ (here for simplicity, we assume that m_b is encrypted under K_i directly). The adversary is given c_i. The adversary obtains the $(m-1)$ secret shares each of which is holden by the corrupted parties say $DEP_1, \ldots, DEP_{m-1}$. Notice that $K_i^r = u_i^{K_i^{(1)}\alpha_1} \times \cdots \times u_i^{K_i^{(m-1)}\alpha_{m-1}} \times u_i^{K_i^{(m)}\alpha_m}$, where α_i is the ith coefficient of the Lagrange Interpolation formula. Thus, $m_b K_i^r$ is a random value from the point view of the adversary. As a result, the advantage $Pr[b = b'] - 1/2|$ that the adversary outputs a correct guess is at most a negligible amount.

Based on the lemmas above and we claim the following main result

Theorem 1. *The proposed multi-key searchable encryption is semantically secure under the joint assumptions that the decisional Diffie-Hellman problem is hard in Z_p^* and the underlying pseudo-random number generator deployed is cryptographically strong.*

4 Conclusion

In this paper, an efficient multi-user, multi-key searchable encryption scheme is presented and analyzed. Our design is simple, scalable, adaptable and sustainable. The processors are distributed to provide high reliability with limited storage, communication and computation overhead by the threshold cryptographic system.

References

1. Bao, F., Deng, R.H., Ding, X., Yang, Y.: Private query on encrypted data in multi-user settings. In: Chen, L., Mu, Y., Susilo, W. (eds.) ISPEC 2008. LNCS, vol. 4991, pp. 71–85. Springer, Heidelberg (2008)
2. Barak, B., Halevi, S.: A model and architecture for pseudo-random generation with applications to dev random. In: ACM Conference on Computer and Communications Security, pp. 203–212 (2005)
3. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007)
4. Bethencourt, J., Song, D.X., Waters, B.: New techniques for private stream searching. ACM Trans. Inf. Syst. Secur. **12**(3), 16 (2009)
5. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
6. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
7. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
8. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. J. ACM **45**(6), 965–981 (1998)
9. Cao, N., Wang, C., Li, M., et al.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Trans. Parallel Distrib. Syst. **25**(1), 222–233 (2014)
10. Curtmola, R., Garay, J.A., Kamara, S.: Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: ACM Conference on Computer and Communications Security, pp. 79–88 (2006)
11. Gertner, Y., Ishai, Y., Kushilevitz, E., Malkin, T.: Protecting data privacy in private information retrieval schemes. J. Comput. Syst. Sci. **60**(3), 592–629 (2000)
12. Gathegi, J.N.: Clouding big data: information privacy considerations. In: Gathegi, J.N., Tonta, Y., Kurbanoğlu, S., Al, U., Taşkin, Z. (eds.) Challenges of Information Management Beyond the Cloud. Springer, Heidelberg (2014)
13. Goh, E.-J.: Secure indexes. IACR Cryptology ePrint Archive, p. 216 (2003)
14. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: a ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998)
15. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In: ACM Conference on Computer and Communications Security, pp. 965–976 (2012)

16. Hahn, F., Kerschbaum, F.: Searchable encryption with secure, efficient updates. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer, Communications Security. ACM, pp. 310–320 (2014)
17. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: STOC, pp. 1219–1234 (2012)
18. Liu, J.K., Au, M.H., Huang, X., Susilo, W., Zhou, J., Yu, Y.: New insight to preserve online survey accuracy and privacy in big data era. In: Kutyłowski, M., Vaidya, J. (eds.) ICAIS 2014, Part II. LNCS, vol. 8713, pp. 182–199. Springer, Heidelberg (2014)
19. Malkin, T.: Secure computation for big data. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 355–355. Springer, Heidelberg (2013)
20. Ostrovsky, R., Skeith, W.E.: Private searching on streaming data. J. Cryptology **20**(4), 397–430 (2007)
21. Pappas, V., Raykova, M., Vo, B., Bellovin, S.M., Malkin, T.: Private search in the real world. In: ACSAC, pp. 83–92 (2011)
22. Popa, R.A., Zeldovich, N.: Multi-key searchable encryption. IACR Cryptology ePrint Archive, p. 508 (2013)
23. Popa, R., Stark, E., Helfer, J., Valdez, S., Zeldovich, N., Kaashoek, M.F., Balakrishnan, H.: Building web applications on top of encrypted data using mylar. In: NSDI (USENIX Symposium of Networked Systems Design and Implementation) (2014)
24. Orencik, C., Selcuk, A., Savas, E., et al.: Multi-Keyword search over encrypted data with scoring, search pattern obfuscation. Int. J. Inf. Secur. 1–19 (2015)
25. Rabin, M.O.: How to exchange secrets by oblivious transfer. Technical report TR-81, Aiken Computation Laboratory, Harvard University (1981)
26. Raykova, M., Cui, A., Vo, B., Liu, B., Malkin, T., Bellovin, S.M., Stolfo, S.J.: Usable, secure, private search. IEEE Secur. Privacy **10**(5), 53–60 (2012)
27. Raykova, M., Vo, B., Bellovin, S.M., Malkin, T.: Secure anonymous database search. In: CCSW, pp. 115–126 (2009)
28. Samanthula, B.K., Elmehdwi, Y., Howser, G., Madria, S.: A secure data sharing and query processing framework via federation of cloud computing. Inf. Syst. **48**, 196–212 (2015)
29. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: IEEE Symposium on Security and Privacy, pp. 44–55 (2000)
30. Tang, Y., Liu, L.: Privacy-preserving multi-keyword search in information networks (2015)
31. Yang, Y.: Towards multi-user private keyword search for cloud computing. In: IEEE CLOUD, pp. 758–759 (2011)
32. Yang, J.J., Li, J.Q., Niu, Y.: A hybrid solution for privacy preserving medical data sharing in the cloud environment. Future Gener. Comput. Syst. **43**, 74–86 (2015)

http://www.springer.com/978-3-319-31549-2

Trusted Systems
7th International Conference, INTRUST 2015, Beijing, China, December 7-8, 2015, Revised Selected Papers
Yung, M.; Zhang, J.; Yang, Z. (Eds.)
2016, XII, 235 p. 55 illus., Softcover
ISBN: 978-3-319-31549-2