A Secure Multi-keyword Ranked Search over Encrypted Cloud Data against Memory Leakage Attack

Lanxiang Chen[1,2], Linbing Qiu[1,2], Kuan-Ching Li[3,4], Shuming Zhou[1,2]

[1] College of Mathematics and Informatics, Fujian Normal University, China
[2] Key Laboratory of Network Security and Cryptology, Fujian Normal University, China
[3] Department of Computer Science and Information Engineering (CSIE), Providence University, Taiwan
[4] Hubei University of Education, China
lxiangchen@fjnu.edu.cn, 975201835@qq.com, kuancli@gm.pu.edu.tw, zhoushuming@fjnu.edu.cn

Abstract

To obtain greater flexibility and cost savings, outsourcing private data to public cloud servers while enabling users to search the data becomes the first choice for more and more users. In view of security, the private data must be encrypted before outsourcing which makes the method of traditional keyword search infeasible. Therefore, searchable encryption is extensively explored in recent years. Taking the practicality into account, multi-keyword ranked search over encrypted data is essential. However, almost all of existing multi-keyword ranked search schemes are suffering the security threats of non-volatile memory leakage attack. To solve this problem, a secure multi-keyword ranked search scheme which resists memory leakage attack (MRSS-ML) is proposed. The proposed scheme utilizes physically unclonable functions (PUFs) to randomize the keywords and document identifiers. Owing to the noisy properties of PUFs, the fuzzy extractor (FE) is used to recover the secret keys. To further enhance the security of the proposed scheme, an order-preserving function is selected to encode the similarity scores. MRSS-ML can resist the memory leakage attack from inner or external attackers. Security analysis and experimental results show that the MRSS-ML scheme is efficient whilst achieve higher security requirements against memory leakage attack.

Keywords: Searchable symmetric encryption, Multi-keyword, Ranked search, Memory leakage attack

1 Introduction

Cloud storage provides unlimited storage and computing capability and it enables users to save initial investment and facilitates data management. It has been widely accepted and used in various areas. However, when storage is moved from the local physical control of a company or individual to reside in a shared environment controlled by others, it brings security issues such as privacy protection. Usually, the user encrypts the data before it is outsourced to the server, so searching for the encrypted data becomes an urgent problem.

Searchable encryption refers to the ability of users to search the encrypted data stored in the cloud server, and simultaneously ensure the security of data and the privacy of queries. As well known, searchable encryption is divided into searchable symmetric encryption (SSE) and searchable asymmetric encryption based on the cryptographic algorithm used. Boneh et al. [1] presented the problem of searching on data that is encrypted using a public key algorithm. Due to there are a large amount of data in cloud storage and public key encryption is very expensive, data is usually encrypted by symmetric cryptography algorithm. The first SSE scheme was proposed by Song et al. [2] in 2000. Since then, searchable symmetric encryption has been extensively exploited. Accordingly, we will mainly focus on the construction of an efficient and secure SSE scheme.

Although there are several secure and efficient multi-keyword ranked search schemes, barely all existing schemes do not take into account the security problem of memory leakage attack. To tackle this problem, Physically Unclonable Functions (PUFs) [3] are utilized to randomize the keywords and document identifiers. Besides, owing to the noisy properties of PUFs, the fuzzy extractor [4] is used to recover the decryption keys of the index, trapdoor and encrypted documents. The proposed searchable scheme improves the memory leakage-resilient SSE (MLR-SSE) scheme [5] to enable multi-keyword ranked search. Like the MLR-SSE scheme, the lookup table is built to retrieve the candidate document identifiers containing the interested keywords. Different from MLR-SSE, the similarity score table is constructed to achieve multi-keyword ranked search, and a PUF is used to randomize the addresses of the score table. To further

*Corresponding Author: Lanxiang Chen; E-mail: lxiangchen@fjnu.edu.cn
DOI: 10.3966/160792642018011901016

enhance the security on the proposed scheme, we select an order-preserving function to encode the similarity score. With experimental results showing that the proposed scheme is efficient whilst ensuring stronger security, the contributions of this paper are summarized as:

- A novel and secure multi-keyword ranked search scheme improving MLR-SSE scheme [5] is proposed,
- The proposed scheme is able to resist non-volatile memory attack from any possible attackers.

The remainder of this paper is organized as follows. Section 2 presents related work and Section 3 describes the system model and security model, notations and preliminaries. Section 4 presents the proposed secure multi-keyword ranked search scheme which resists memory leakage attack (MRSS-ML). The security and performance analysis followed by Section 5, and finally, section 6 concludes the paper with future work.

2 Related Work

Song *et al.* [2] proposed the first SSE scheme, which only supports single keyword search. In order to enrich the search function and improve the search accuracy, there are a variety of methods and techniques, including multi keyword search [6-7], fuzzy and similarity search [8-11], dynamic search [12-13], multi-user search [14], ranked search [15-16] and verifiable search [17-18].

Cao *et al.* [6] proposed a secure multi-keyword ranked search (MRSE) scheme, where similarity measure of "coordinate matching" and "inner product similarity" were incorporated to quantitatively evaluate such similarity measure. This approach can return the ranked results of search according to the number of matching keywords. However, MRSE does not take the access frequencies of keywords into account. In addition, two dense matrices are used to encrypt secure index and trapdoor for document vector and query vector in MRSE. The efficiency of inner product will be significantly reduced when the number of keyword in the dictionary increases gradually. Besides, since the location of keywords is fixed, the vector structure cannot be modified after the keyword dictionary is generated. To solve these issue, Chen *et al.* [7] proposed an efficient dynamic multi-keyword ranked search (DMRS) scheme in which the sparse block matrices are utilized to replace the original dense matrices in the process of index construction and trapdoor generation. It will greatly save the user's computing resources. In DMRS, a reverse data structure is utilized to achieve the dynamic update operation of the document.

Gajek [12] presented a dynamic symmetric searchable encryption from constrained functional encryption scheme, where the proposed scheme could realize logarithmic search efficiency in the size of keyword set and is proved to be secure based on the subgroup decision problem in bilinear groups. Yang et al. [13] proposed a dedicated and dynamic SSE scheme for e-healthcare applications where personal health information is generated and stored in the cloud periodically. Strizhov and Ray [14] proposed a secure and efficient multi-keyword similarity searchable encryption (MKSim), which extends the searchable encryption to multi-user setting. It is provably secure against adaptive chosen-keyword attacks (CKA2-secure) in the random oracle model. Zhang et al. [16] presented a secure ranked multi-keyword search scheme in a multi-owner model (PRMSM). It utilizes an additive order and privacy preserving function family to rank the search results, as also applies a novel dynamic secret key generation protocol and a new data user authentication protocol to prevent the attackers from eavesdropping secret keys and pretending to be legal data users submitting searches. Furthermore, PRMSM supports efficient data user revocation.

Wu *et al.* [19] worked on a survey that contains most searchable encryption schemes and analyzed individual contributions respectively. Poh *et al.* [20] gave a comprehensive survey on almost all existing SSE schemes. They summarized the development of SSE and provided detailed description on the constructions of SSE based on the proposed general framework. They compared these structures from the general search performance metrics, security models and the various characteristics and functionalities. Based on these studies, they outlined the challenges and suggested future research directions.

However, the security of all existing SSE schemes is based on the assumption that the data owner holds a secret key that is unknown to the adversary. Unfortunately, in practice, attackers are often able to obtain some or even all of data owner's secret keys by a great variety of inexpensive and fast side channel attacks. Aimed by such attacks, all existing SSE schemes are no longer secure. In MRSE scheme [6], the index and trapdoor are respectively generated from a binary vector that indicates whether the keyword appears in the corresponding document or query. To guarantee the privacy of the proposed scheme, document vector and query vector are encrypted by multiplying two dense matrices. Moreover, the similarity score of query keywords and the document is derived by the inner product of trapdoor with each sub-index. It is worth noting that the secret keys, namely the two dense matrices, should be kept well in owner's non-volatile memory. However, the owner's non-volatile memory maybe attacked by possible attackers in the cloud storage environment.

To solve this problem, Dai *et al.* [5] proposed a secure memory leakage-resilient SSE scheme (MLR-SSE) for the first time, which could meet the higher security requirements. Regretfully, MLR-SSE only supports single keyword search.

To resist non-volatile memory leakage attack and achieve multi-keyword ranked search, we improve the MLR-SSE scheme to incorporate the multi-keyword similarity search and propose a secure multi-keyword ranked search scheme (MRSS-ML). The index construction of MRSS-ML is improved from the scheme proposed in [14] which is based on SSE-2 inverted index data construction previously introduced in [21]. In order to realize the higher security requirements, keywords or document identifiers are randomized by a physically unclonable function (PUF). Due to the noisy influence of PUFs, the fuzzy extractor is chosen to recover the secret keys. Moreover, a similarity score table is constructed to achieve the functionality of multi-keyword ranked search. Besides, the order-preserving function [16] is improved to protect the privacy of similarity between keywords and documents. With these steps, a secure multi-keyword ranked search scheme against memory leakage attack (MRSS-ML) is achieved.

3 MRSS-ML Definition

3.1 Problem Formulation

3.1.1 System Model

There are three different entities in MRSS-ML as illustrated in Figure 1: the data owner, the data user and cloud server. The data owner intends to outsource a document collection *D* to cloud server. He will first extract keywords collection *W'* from *D* and construct a keyword dictionary *W* with some inserted dummy keywords. Next, he constructs an encrypted searchable index *I* from collection *D*, and each document in *D* is encrypted to generate one encrypted document collection *C*. Finally, he sends the encrypted index *I* and the collection *C* to cloud server. Data users are entities that are authorized to access the documents of the data owner. Cloud server stores the encrypted searchable index and the collection of encrypted documents. It has the responsibility to search over the encrypted index and return the top-*k* most relevant encrypted documents to users.

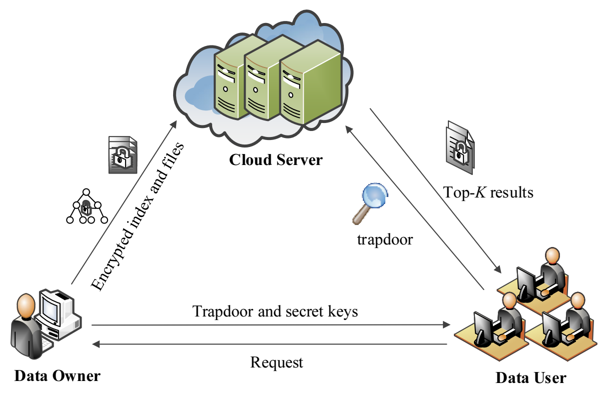


Figure 1. System model of the MRSS-ML scheme

3.1.2 Security Model

In this sub-section, we introduce the stronger attack model, called as full non-volatile memory attack model.

Definition 1. (*Non-volatile Memory Attacker* [22]): Let S be secret information stored in data owner's non-volatile memory. Assume α as a function that: $\alpha(x) \leq x$, $x \in N$, where N is the set of natural numbers. A α-non-volatile memory attacker A can access the attack oracle O that takes as adaptively chosen a polynomial-size attack function $g(\bullet)$ and outputs $g(S)$ under the condition that the whole number of bits of $g(S)$ shall not exceed $\alpha(|S|)$.

Definition 2. (*Full Non-volatile Memory Attack*): An attack model is called full non-volatile memory attack if an attacker A meet the requirements of Definition 1 and $\alpha = id$ defined in [22].

In the traditional attack model, an attacker is assumed that he could not access any information about the secret keys stored in non-volatile memory. Obviously, this assumption is unreasonable in the actual environment. For considerations of security, the long-term secret keys should not be stored in non-volatile memory any longer. Fortunately, the physically unclonable function presented in [3] can be used to generate the secret keys in real time, with which the keys need not be stored. Accordingly, the attacker would not be able to obtain any information about the secret keys stored in non-volatile memory.

3.2 Notations and Preliminaries

3.2.1 Notations

In this paper, notations presented in Table 1 are used.

Table 1. Notation

D	The plaintext collection, denoted as a set of n documents $D = (D_1, D_2, \ldots, D_n)$.
C	The encrypted document collection, stored in cloud server and denoted as $C = (C_1, C_2, \ldots, C_n)$.
I	The encrypted searchable index.
Ω	The all possible keyword set extracted from D.
W'	The distinct keyword set extracted from D.
W	The keyword dictionary with dummy keywords $W = (w_1, w_2, \ldots, w_m)$.
Q	The interested search keywords which is the subset of *W*.
T_w	The trapdoor for the search request Q.

3.2.2 Preliminaries

In this sub-section, background for the proposed scheme follows next.

Physically unclonable functions. Willers et al. [23] think that a key requirement for most security solutions is to provide secure cryptographic key storage in a way that will easily scale in the age of the Internet of Things. They propose a microelectromechanical systems (MEMS)-based PUF. Park et al. [24] propose

PUFSec, a new device fingerprint-based security architecture which aims to design a computationally lightweight security software system architecture to trade-off dilemma between security enhancement and computation overhead. PUFs will be widely used gradually.

A Physically Unclonable Function (PUF) firstly presented in [3] is a noisy function. As described in [25], a PUF family *P* is composed of the pair of algorithms *Sample* and *Eval*. The *Sample* takes the security parameter as input and output the index identifier *idp* of the PUF family, and the *Eval* takes a stimulus *s* as input and output the response *r*.

Definition 3. (*Physically Unclonable Functions, PUFs*): A pair *P* = (*Sample*, *Eval*) is a family of the tri-tuple (l, d, δ) PUFs if it meets the properties as follows:

- *Unpredictability*: For a set Φ of the stimulus-response pair, it is difficult to predict the response r' for a new random stimulus s' in a small error range, where $\Phi = \{s_i, r_i, 1 \le i \le q\}$ and $s', r' \notin \Phi$. With the property, a PUF is called an (l, d, δ) PUF.
- *Evaluation*: The *Eval* takes security parameter 1^λ, *idp* and *s* as input, and it efficiently outputs the corresponding *r*.
- *Bounded Noise*: For all stimulus $s \in \{0, 1\}^l$, the Hamming distance $d_{1,2}$ of any two responses r_1 and r_2 should satisfy $d_{1,2} < d$ when the algorithm $Eval(1^\lambda, idp, s)$ is run twice, where *d* is a noise bound.
- *Unclonability*: Given a PUF, there is no efficient technique, which is able to clone another PUF′ and PUF′ = PUF.
- *One-wayness*: Given a PUF and a response *r*, the corresponding stimulus *s* cannot be founded and simultaneously satisfies the equation: $Eval(1^\lambda, idp, s) = r$.

Fuzzy extractor. From the above introduction, the PUF can evaluate a physical stimulus and output possibly different responses, which relies on a physical architecture. To overcome the noisy drawback of PUF, the properties of fuzzy extractor can be used to recover the necessary secret information. The fuzzy extractor is introduced in [4] as follows:

Definition 4. (*Fuzzy Extractor, FE*): An (l, d, δ) fuzzy extractor is composed of two efficient algorithms (*Gen*, *Rep*) as follows:

Gen: The algorithm takes an *l*-bit string *w* as input, it outputs a random string $st \in \{0, 1\}^\delta$ and an auxiliary data $ad \in \{0, 1\}^*$.

Rep: The reproduction algorithm takes an *l*-bit noisy string w' and the auxiliary data *ad* as inputs, it outputs a δ-bit random string *st*.

Fuzzy Extractor has the following two properties.

- Correctness: Let *dis* be the Hamming distance of two variables *w* and w'. The reproduction algorithm $FE.Rep(w', ad) = st$ is realized if and only if $dis \le d$.
- Security: Let *U* is a uniform distribution with min-entropy δ, the noisy string *w* as input is selected from *U*, the output *st* would be distributed as *U*, even if the auxiliary data string *ad* is revealed.

Order-preserving function. In order to preserve the similarity score from a potential attacker, the order-preserving function (OPF) is utilized to encode the similarity score. The formalization of OPF is improved from [16] as follows.

The order-preserving function $f(x) = \sum_{1 \le i \le \tau} a_i \cdot h(x, i) + r$, where τ is the degree of the function, a_i is a positive coefficient, $h(x, i)$ is a recursive computation, and *r* is a random number that preserves the function from the cloud server. $h(x, i)$ is further defined as:

$$h(x,i) = \begin{cases} 1 & \text{if } i = 0; \\ x & \text{if } i = 1; \\ (1+\varepsilon)\cdot(h(x,i-1)+\beta\cdot x) & \text{if } i > 1. \end{cases}$$

Here, both of β, ε are two constant numbers. In order to guarantee the ranked results, the number *r* should satisfy: $r \in (0, 2^{\gamma-1})$, where γ is an integer. The detailed definition, theorem and proof of OPF are presented in [16].

4 The MRSS-ML Scheme

In this section, we present the formal definition of MRSS-ML scheme and the detailed construction next.

4.1 Formal Definition

Definition 5. (Multi-keyword Ranked Search Scheme against Memory Leakage, MRSS-ML): An MRSS-ML scheme consists of five polynomial-time algorithms *MRSS-ML* = (*KeyGen*, *BuildIndex*, *TrapdoorGen*, *Search*, *Decrypt*), such that:

K←Keygen(1^λ): is a probabilistic algorithm that is run by data owner to setup the scheme. It takes a security parameter λ, and returns a secret key K = (PUF_1, PUF_2, PUF_3).

(I, C)←BuildIndex(K, D): is run by data owner to generate indexes and encrypted document set. It takes a secret key K and a document collection D as inputs, and returns an index I and an encrypted document set C.

T_w←TrapdoorGen(K, Q): is run by user to generate a trapdoor to search keywords that is a subset of dictionary. It takes a secret key K and interested keywords Q as inputs, and returns a trapdoor T_w.

ID_w←Search(I, T_w): is run by cloud server to search for documents in D that contains words Q. It takes an index I and a trapdoor T_w as inputs, and returns ID_w, the set of top-k most relevant ranked document identifiers.

D_w←Decrypt(K, ID_w): is run by user to decrypt the set of the top-k most relevant ranked encrypted documents that contain words Q. It takes a secret key K and the top-k most relevant ranked document identifier set ID_w as inputs, and returns the top-k most relevant ranked document set D_w that contains search keywords Q.

4.2 The MRSS-ML Construction

The MRSS-ML scheme improves the MLR-SSE scheme [5] to achieve multi-keyword ranked search. In this sub-section, we detail the concrete construction of MRSS-ML scheme which is illustrated in Figure 2.

```
Keygen($1^{\lambda}$):
1. Choose a ($p+\log_2 n, d_1, \delta_1$) $PUF_1$, a ($t, d_2, \delta_2$) $PUF_2$ and a ($t, d_3, \delta_3$) $PUF_3$.
2. Output a secret key $K = (PUF_1, PUF_2, PUF_3)$.
BuildIndex($K, D$):
1. Initialization:
  (a) scan $D$ and build $W'$, the set of distinct words in $D$.
  (b) construct a new dictionary $W$ containing $W'$ and several inserted dummy keywords.
  (c) for each word $w_i \in W$, build $D(w_i)$.
2. Build look-up table $T$:
  (a) for $w_i \in W$ and $j \in [1, n]$, compute $ut_{i,j} = PUF_1(w_i||j)$ and $(rt_{i,j}, adt_{i,j}) \leftarrow FE_1.Gen(ut_{i,j})$.
  (b) for $w_i \in W$ and $D_j \in D$, compute $us_{i,j} = PUF_3(id(D_{i,j}))$, compute $idc_{i,j} = Enc(us_{i,j}, id_{i,j})$, where $idc_{i,j}$
      is the $j^{th}$ encrypted identifier in $D(w_i)$.
  (c) for $w_i \in W$ and $j \in [1, |D(w_i)|]$, set $T[rt_{i,j}] = idc_{i,j}$. If $v < \tilde{v}$, $\tilde{v}-v$ random strings with $m_1$ bits are
      assigned in $T$ for exactly $m$ times, and the addresses are set to random values.
3. Generate an auxiliary query table $T'$:
  (a) for $i \in [1, m]$ and $j \in [1, n]$, $T'[w_i] = adt_{i,j}$.
  (b) for $i \in [m+1, |\Omega|]$ and $j \in [1, n]$, $T'[w_i] = adrt_{i,j}$, where $adrt_{i,j}$ is a randomly generated string with
      the same length as $adt_{i,j}$.
4. Build score table $\Delta$:
  (a) for $D_j \in D$ and $w_i \in W$, compute the similarity score $S_{j,i} = Score(D_j, w_i)$, then compute $CS_{j,i} = f(S_{j,i})$,
      where $f(\bullet)$ is an order-preserving function.
  (b) for $D_j \in D$ and $w_i \in W$, set $\Delta[idc_{j,i}] = CS_{j,i}$.
5. Generate encrypted document $C_j$:
   for $D_j \in D$, compute $uc_j = PUF_2(id(D_j))$ and $(rc_j, adc_j) \leftarrow FE_2.Gen(uc_j)$, then $C_j = (Enc(rc_j, D_j), adc_j)$.
6. Output index $I = (T, \Delta)$ and encrypted document collection $C = (C_1, C_2, \ldots, C_n)$.
TrapdoorGen($K, Q$):
1. For $w_i \in Q$ and $j \in [1, n]$, compute $\hat{ut}_{i,j} = PUF_1(w_i||j)$ and $rt_{i,j} = FE_1.Rep(\hat{ut}_{i,j}, adt_{i,j})$, where $adt_{i,j}$ is an
   auxiliary data stored in $T'$.
2. Output $T_w = \{T_{w_i}, 1 \le i \le q\}$, where $T_{w_i} = (rt_{i,1}, rt_{i,2}, \ldots, rt_{i,n})$.
Search($I, T_w$):
1. Traverse the look-up table $T$ according to $T_w$: for $1 \le i \le q$ and $j \in [1, n]$, if $T[rt_{i,j}] \neq \perp$, the value $idc_{i,j}$ is
   inserted into the encrypted identifier set $IDC$ matching the searched keywords.
2. Traverse the score table $\Delta$ according to $IDC$: for $idc_{j,i} \in IDC$, if $\Delta[idc_{j,i}] \neq \perp$, compute $VS_j = \sum_{1 \le i \le q} CS_{j,i}$.
3. Output the top-$k$ most relevant document identifier set $IDC_w = \{id(D_j), 1 \le j \le k\}$.
Decrypt($K, ID_w$):
1. Server returns the corresponding encrypted document set $C_w$ according to $IDC_w$.
2. For each $D_j, j \in [1, k]$, user computes $uc_j = PUF_2(id(D_j))$ and $rc_j = FE_2.Rep(uc_j, adc_j)$.
3. User computes $D_{w_j} = Dec(rc_j, C_{w_j}), j \in [1, k]$.
4. Output the top-$k$ most relevant document set $D_w = (D_{w_1}, D_{w_2}, \ldots, D_{w_k})$ containing the search keywords.
```

Figure 2. The construction of MRSS-ML scheme

In MRSS-ML, a document collection D is associated with an encrypted index I. The index I consists of two tables. The first one is the look-up table T. Before construction, a keyword dictionary W with inserted dummy keywords and $D(w_i)$ for each word $w_i \in W$ are built. For $i \in [1, n]$ and $w_i \in W$, let $v = \sum|D(w_i)|$ and $u = max(|D_i|)$, then $\tilde{v} = n \cdot u$, where $|D_i|$ denote the number of keywords extracted from the document D_i. For $w_i \in W$ and $j \in [1, n]$, we use a physically unclonable function (PUF_1) to randomize the keywords. Next, two groups of data are generated by the generation algorithm of fuzzy extractor ($FE_1.Gen$). The first one $rt_{i,j}$ is a key used to randomize the address of tuples in T. The other $adt_{i,j}$ is an auxiliary data. For $i \in [1, m]$ and $j \in [1, n]$, the $adt_{i,j}$ is stored in another table T' stored in data owner's non-volatile memory.

Let Ω be the set of all possible keywords extracted from D, and let $|\Omega|$ be the number of all possible keywords. For $i \in [m+1, |\Omega|]$ and $j \in [1, n]$, the randomly generated strings $adrt_{i,j}$ are inserted in T'. The auxiliary data $adt_{i,j}$ is used to recover the key $rt_{i,j}$ in the process of trapdoor generation. Next, we use another PUF_3 to calculate the random string $us_{i,j}$ of the document identifiers containing the keyword w_i. Then, the symmetric encryption is employed to encrypt the document identifiers, and the symmetric key is $us_{i,j}$. The encrypted identifiers containing the keyword w_i are inserted in the entries of the randomized address in T, and the remaining entries of T are set to random strings.

The other table is the similarity score table Δ. We utilize the TF-IDF method to compute the similarity score of the document D_j and the keyword w_i. The score is encoded by an order-preserving function improved from [16]. The sum of similarity scores is used as the metric to rank the queried results. Next, the addresses of Δ are randomized by the encrypted document identifiers, and the encoded scores are inserted in these entries of Δ. Another PUF_2 is used to generate the random string of the identifier of the document D_j. The generation algorithm $FE_2.Gen$ of fuzzy extractor is called to generate two groups of data. The first one $rc_{i,j}$ is used as a key to encrypt D_j, and $adc_{i,j}$ is used as an auxiliary data to recover the key $rc_{i,j}$.

The data owner stores the index I on the cloud server with the encrypted document collection C. As one user is authorized to retrieve the documents that contain the keyword w_i ($w_i \in Q$, where Q is a set of the interested searched keywords), he computes the trapdoor encryption key by calling the function PUF_1 and the reproduction algorithm $FE_1.Rep$ of fuzzy extractor. Once receiving the trapdoor from user, the server traverses the look-up table T to obtain a candidate document identifier set IDC. Then, the server traverses Δ and computes the sum of the encoded score. Finally, the user uses PUF_2 and $FE_2.Rep$ to recover the secret keys which are used to decrypt the encrypted top-k most relevant documents.

5 Security and Performance Analysis

The security and performance analysis of MRSS-ML scheme are given in this section.

5.1 The Security Analysis

From the previous description, several dummy keywords are inserted into the index, which is used to prevent keyword frequency analysis from the potential attackers. The document collection is encrypted by symmetric encryption algorithm, since the attackers cannot learn anything about documents if the encryption algorithm is secure. We mainly focus on analyzing the security of index, trapdoor and search pattern as follows.

Theorem 1. The scheme MRSS-ML is semantically secure against full non-volatile memory attack if the PUFs used in the proposed scheme satisfy the definition 3 (described in section 3.2.2A).

Proof of index security. In the proposed scheme, the index consists of a look-up table T and a score table Δ. In order to hide the number of keywords in dictionary and document identifiers containing the keyword w_i, the entries of T are equal to the number of keywords extracted from the document D_i. Additionally, the addresses of these entries are set to random strings. Besides, the entries of Δ are processed in the similar way. Thanks to the unpredictability of PUFs and the security of fuzzy extractor (FE), the addresses of T and Δ cannot be computed by a full non-volatile memory attacker. Furthermore, the similarity scores are encoded with an order-preserving function. Assume that an attacker has selected n encoded scores for the same input $S_{j,i}$ according to the equation $f(x)=\sum_{1\leq i\leq\tau} a_i \cdot h(x, i)+r$, after that, the attacker constructs n equations. However, the attacker cannot break the order-preserving function, since n equations have $n+1$ variables. Therefore, the confidentiality of similarity score is protected.

Proof of trapdoor unlinkability. Assume an attacker independently chooses a PUF$'$ and a FE$'$, and obtains the auxiliary data stored in a non-volatile memory. The attacker can compute $ut'_{i,j} = PUF'(w_i||j)$ and $rt'_{i,j} = FE'.Rep(ut'_{i,j}, adt_{i,j})$. However, because of the unpredictability of PUFs and the security of FE, the trapdoor $T'_w = \{T'_{w_i}, 1\leq i\leq q\}$, where $T'_{w_i} = (rt'_{i,1}, rt'_{i,2},\ldots, rt'_{i,n})$ is different from the real one T_w described in Figure 2. Thus, the keyword privacy and the trapdoor unlinkability are reinforced.

Proof of search security. Suppose that an attacker monitors the user's search process in real time, and he can get the input keywords. Then, the attacker chooses two PUF (PUF$''$ and PUF$'''$), an FE$''$, and a symmetric encryption algorithm Enc, and he is able to get the auxiliary data stored in a non-volatile memory. Regardless of whether the keyword w_i appears or not, the attacker can generate a new trapdoor as the following process: compute $ut''_{i,j} = PUF''(w_i||j)$ and $rt''_{i,j} = FE''.Rep(ut''_{i,j}, adt_{i,j})$, then $T''_w = \{T''_{w_i}, 1\leq i\leq q\}$, where $T''_{w_i} = (rt''_{i,1}, rt''_{i,2}, \ldots, rt''_{i,n})$. For $1\leq i\leq q$ and $j \in [1, n]$, the attacker also can set $T[rt''_{i,j}]$, and for $id(D_j) \in ID$ and $w_i \in Q$, compute $us'''_j = PUF'''(id(D_j))$ and $idc'''_{i,j} = Enc(us'''_{i,j}, id_{i,j})$, then set $\Delta[idc'''_{i,j}]$. Obviously, if the unpredictability of PUFs and the security of FE can be guaranteed, the new trapdoor T''_w will not be equal to the real one T_w described in Figure 2 and the $T[rt''_{i,j}]$ and $\Delta[idc'''_{i,j}]$ will always be $\perp$. Therefore, the attacker cannot obtain any information about the documents and keywords even if the secret keys stored in non-volatile memory are given.

5.2 Performance Analysis

In this sub-section, we analyze the performance of the proposed MRSS-ML scheme. From the introduction in [3], the efficiency of PUFs is almost equal to hash function. Accordingly, we use a hash function instead of PUF to evaluate the performance of MRSS-ML scheme. Experiments are coded using Java on PC with Intel core i5-M480 2.67GHz Processor, 6 GB memory, 1GB AMD Radeon HD 6400M graphics card, running Win7 (64 bit) system. The performance of the proposed scheme is compared with the MRSE scheme [6], meanwhile the overhead of index construction, trapdoor generation and search will be compared and analyzed next.

5.2.1 Storage Overhead

From the above detailed description of the MRSS-ML scheme in section 4, we can see that the size of the index is mainly determined by the size of document collection and the keyword dictionary. As shown in Table 2, the index sizes of two schemes grow linearly with the number of keywords. Because the look-up table and similarity score table are built in index construction, the MRSS-ML scheme takes up more storage space.

Table 2. Size of index with n=1000

size of dictionary	1000	2000	3000	4000	5000
MRSE(MB)	15.25	30.5	45.75	61	76.25
MRSS-ML(MB)	34.33	68.66	102.99	137.33	171.66

In MRSE scheme, the trapdoor of queried keywords is generated according to the keywords dictionary. As illustrated in Table 3, the size of trapdoor in MRSE scheme grows linearly with the number of keywords. However, the size of trapdoor in the MRSS-ML scheme is not affected. The reason lies in that the trapdoor is calculated only for queried keywords.

Table 3. Size of trapdoor with n=1000, q=20

size of dictionary	2000	4000	6000	8000	10000
MRSE(KB)	15.63	31.25	46.88	62.5	78.13
MRSS-ML(KB)	0.20	0.20	0.20	0.20	0.20

5.2.2 Computing Overhead

Index construction. In the MRSS-ML scheme, index construction consists of two steps. The first one is to build a look-up table. Another one is the construction of similarity score table. Likewise, in MRSE scheme, the index construction includes two processes. The former one is generating a data vector according to the keyword dictionary, while the latter is encrypting a set of data vectors. Obviously, the index construction relies on the keyword dictionary extracted from the document collection. Therefore, the time of building whole index is related to the number of documents and keywords. Figure 3(a) shows that given the same dictionary (m=2000), the time overhead of index construction for the two schemes increases linearly with the increasing number of documents. Figure 3(b) demonstrates that, given the same number of files (n=1000), the time overhead of index construction of MRSE increases exponentially. It can be seen that the time overhead of index construction of MRSS-ML scheme is better in both situations.

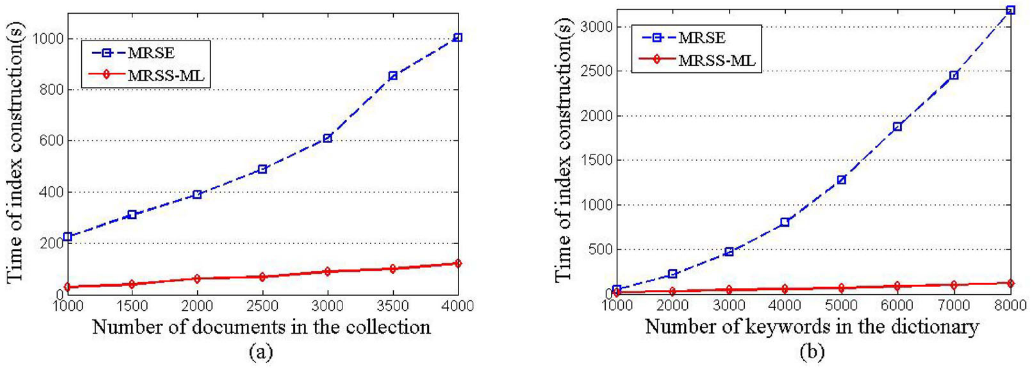


Figure 3. The time overhead of index construction. (a) For different size of document collection with the same dictionary, m=2000. (b) For different size of dictionary with the same document collection, n=1000

Trapdoor generation. Through theoretic analysis, the efficiency of trapdoor generation will be better than index construction. Figure 4(a) shows that the time of generating a trapdoor in MRSE increases dramatically as the number of keywords increases. Figure 4(b) demonstrates that the efficiency of trapdoor generation in MRSS-ML scheme is about 60 times better than that of MRSE scheme. In MRSE, the efficiency of generating a trapdoor is mainly affected by the large scale of matrices. In MRSS-ML, the time of trapdoor generation is determined by the number of queried keywords. Besides, the efficiency of PUFs and fuzzy extractor determines that the proposed scheme is better than MRSE.

Search. In search process, the cloud server is responsible for computing and ranking similarity

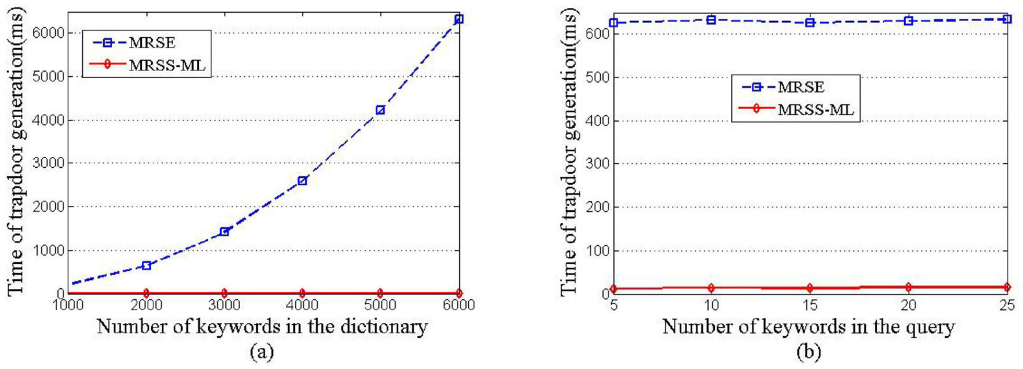


Figure 4. The overhead of trapdoor generation. (a) For different size of dictionary within the same query keywords and document collection, q=10, n=1000. (b) For different numbers of query keywords within the same dictionary and document collection, m=2000, n=1000

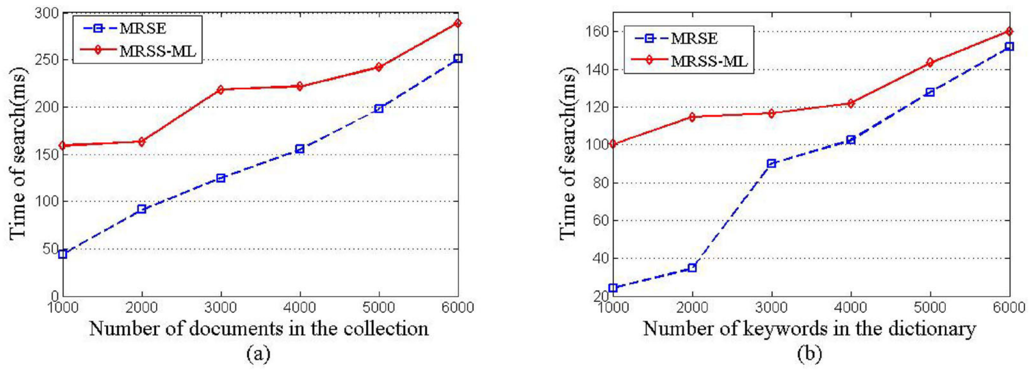


Figure 5. The search time. (a) For different size of document collection within the same query keywords and dictionary, q=10, m=2000. (b) For different size of dictionary within the same query keywords and document collection, q=10, n=1000

scores. Figure 5 shows that the search overhead is mainly determined by the size of documents collected and the number of keywords in the dictionary. As demonstrated in the figures, the growth of search time of MRSE is obviously higher than that of MRSS-ML. It can be predicted that the search time of MRSE will be more than MRSS-ML when the size of documents and keyword dictionary is large.

6 Conclusion and Future Work

In this paper, we present a novel multi-keyword ranked search scheme that resists to memory leakage attack (MRSS-ML). Like the MLR-SSE scheme, the look-up table is built to retrieve the candidate document identifiers containing the interested keywords. Different from the MLR-SSE, the score table is constructed to achieve multi-keyword ranked search, and a PUF is used to randomize the addresses of the score table. To further enhance the security of the proposed scheme, we select an order-preserving function to encode the similarity scores. Considering the noisy defects of PUFs, the fuzzy extractor is chosen to recover the secret keys. From the above performance analysis, the proposed scheme is efficient. At the same time, it can achieve higher security requirements against memory leakage attack.

In fact, there are many other challenging problems in SSE schemes. Most of SSE schemes mainly consider the security threats from the cloud server. Actually, there are many security challenges in a multi-user scheme. Fine-grained access authorization and the revocation of the user are largest ones. As future work, we will suggest and improve the SSE scheme to handle these issues, as well to explore supporting other multi-keyword functionalities over encrypted data, e.g., multi-owner setting, as also integrity check of rank order in search results.

Acknowledgment

This work was supported by the Natural Science Foundation of China (No. 61602118, No. 61572010 and No. 61472074), Fujian Normal University Innovative Research Team (No.IRTL1207), Natural Science Foundation of Fujian Province (No. 2015J01240, No. 2017J01738), Science and Technology Projects of Educational Office of Fujian Province (No. JK2014009), and Fuzhou Science and Technology Plan Project (No. 2014-G-80).

References

[1] D. Boneh, G. D. Crescenzo, R. Ostrovsky, G. Persiano, Public Key Encryption with Keyword Search, *International Conference on the Theory and Applications of Cryptographic Techniques*, Interlaken, Switzerland, 2004, pp. 506-522.

[2] D. X. Song, D. Wagner, A. Perrig, Practical Techniques for Searches on Encrypted Data, *IEEE Symposium on Security and Privacy*, Berkeley, California, 2000, pp. 44-55.

[3] S. R. Pappu, B. Recht, J. Taylor, N. Gershenfeld, Physical One-way Functions, *Science*, Vol. 297, No. 5589, pp. 2026-2030, September, 2002.

[4] Y. Dodis, R. Ostrovsky, L. Reyzin, A. Smith, Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data, *SIAM Journal on Computing*, Vol. 38, No. 1, pp. 97-139, March, 2008.

[5] S. Dai, H. Li, F. Zhang, Memory Leakage-resilient Searchable Symmetric Encryption, *Future Generation Computer Systems*, Vol. 62, pp. 76-84, September, 2016.

[6] N. Cao, C. Wang, M. Li, K. Ren, W. Lou, Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, No. 1, pp. 222-233, January, 2014.

[7] L. Chen, L. Qiu, K. C. Li, W. Shi, N. Zhang, DMRS: An efficient Dynamic Multi-keyword Ranked Search Over Encrypted Cloud Data, *Soft Computing*, Vol. 21, No. 16, pp. 4829-4841, August, 2017.

[8] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, W. Lou, Fuzzy Keyword Search over Encrypted Data in Cloud Computing, *The 29th Conference on Computer Communications*, San Diego, CA, 2010, pp. 1-5.

[9] B. Wang, S. Yu, W. Lou, Y. T. Hou, Privacy-preserving Multi-keyword Fuzzy Search Over Encrypted Data in the Cloud, *The 33rd Conference on Computer Communications*, Toronto, Canada, 2014, pp. 2112-2120.

[10] C. Wang, K. Ren, S. Yu, K. M. R. Urs, Achieving Usable and Privacy-assured Similarity Search Over Outsourced Cloud Data, *The 31st Conference on Computer Communications*, Orlando, FL, 2012, pp. 451-459.

[11] Z. Fu, J. Shu, J. Wang, Y. Liu, S. Lee, Privacy-preserving Smart Similarity Search Based on Simhash over Encrypted Data in Cloud Computing, *Journal of Internet Technology*, Vol. 16, No. 3, pp. 453-460, May, 2015.

[12] S. Gajek, Dynamic Symmetric Searchable Encryption from Constrained Functional Encryption, *The Cryptographers' Track at the RSA Conference*, San Diego, CA, 2016, pp. 75-89.

[13] L. Yang, Q. Zheng, X. Fan, RSPP: A Reliable, Searchable and Privacy-preserving E-Healthcare System for Cloud-assisted Body Area Networks, *Proceedings of the 36th Conference on Computer Communications*, Atlanta, GA, 2017.

[14] M. Strizhov, I. Ray, Secure Multi-keyword Similarity Search over Encrypted Cloud Data Supporting Efficient Multi-user Setup, *Transactions on Data Privacy*, Vol. 9, No. 2, pp. 131-159, August, 2016.

[15] R. Li, Z. Xu, W. Kang, K. Yow, C. Xu, Efficient Multi-keyword Ranked Query over Encrypted Data in Cloud Computing, *Future Generation Computer Systems*, Vol. 30, No. 1, pp. 179-190, January, 2014.

[16] W. Zhang, Y. Lin, S. Xiao, J. Wu, S. Zhou, Privacy Preserving Ranked Multi-keyword Search for Multiple Data Owners in Cloud Computing, *IEEE Transactions on Computers*, Vol. 65, No. 5, pp. 1566-1577, May, 2016.

[17] J. Wang, X. Chen, J. Li, J. Zhao, J. Shen, Towards Achieving Flexible and Verifiable Search for Outsourced Database in Cloud Computing, *Future Generation Computer Systems*, Vol. 67, pp. 266-275, February, 2017.

[18] W. Sun, X. Liu, W. Lou, Y. T. Hou, H. Li, Catch You If You Lie to Me: Efficient Verifiable Conjunctive Keyword Search Over Large Dynamic Encrypted Cloud Data, *Proceedings of the 34th Conference on Computer Communications*, Hong Kong, China, 2015, pp. 2110-2118.

[19] X. Wu, Z. Fu, X. Sun, Text-Based Searchable Encryption in Cloud: A Survey, *Journal of Internet Technology*, Vol. 18, No. 1, pp. 207-213, January, 2017.

[20] G. S. Poh, J. J. Chin, W. C. Yau, K. K. R. Choo, M. S. Mohamad, Searchable Symmetric Encryption: Designs and Challenges, *ACM Computing Surveys*, Vol. 50, No. 3, Article 40, August, 2017.

[21] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky, Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions, *13th ACM Conference on Computer and Communications Security*, Alexandria, VA, 2006, pp. 79-88.

[22] F. Armknecht, R. Maes, A. R. Sadeghi, B. Sunar, P. Tuyls, Memory Leakage Resilient Encryption Based on Physically Unclonable Functions, *Proceedings of the 15th Conference on the Theory and Application of Cryptology and Information Security*, Tokyo, Japan, 2009, pp. 685-702.

[23] O. Willers, C. Huth, J. Guajardo, H. Seidel, MEMS Gyroscopes As Physical Unclonable Functions, *23rd ACM Conference on Computer and Communications Security*, New York, NY, 2016, pp. 591-602.

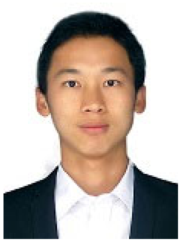
[24] S. Y. Park, S. Lim, D. Jeong, J. Lee, J. S. Yang, H. Lee, PUFSec: Device Fingerprint-based Security Architecture for Internet of Things, *Proceedings of the 36th Conference on Computer Communications*, Athlanta, GA, 2017.

[25] C. Brzuska, M. Fischlin, H. Schröder, S. Katzenbeisser, Physically Uncloneable Functions in the Universal Composition Framework, *The 31st International Cryptology Conference*, Santa Barbara, CA, 2011, pp. 51-70.

Biographies

Lanxiang Chen received the M.S. and Ph.D. Degrees in Computer Architecture from Huazhong University of Science and Technology in China. She is currently an associate professor in Fujian Normal University. She is a member of the Computer

Society of China. Her research interests include big data security, cloud computing and cloud storage security etc.



Linbing Qiu is M.S. candidate at Fujian Normal University. His research interests include cloud storage and information security.

Kuan-Ching Li is a Professor at the Providence University, Taiwan, and a Chair Professor at Hubei University of Education, China. Dr. Li is a recipient of awards and funding support from a number of agencies and industrial companies, as also received guest and distinguished chair professorships from universities in China and other countries. He has been actively involved in many major conferences and workshops in program/general/steering conference chairman positions and as a program committee member, and has organized numerous conferences related to high-performance computing and computational science and engineering. He is editor of several technical professional books published by CRC Press, McGraw-Hill, IGI Global and Springer. His topics of interest include Cloud and GPU computing and Big Data. He is a senior member of the IEEE and a Fellow of the IET.

Shuming Zhou graduated in 1996 from the Department of Computer and Applied Mathematics, Hubei Institute for Nationalities. He received his M.S. and Ph.D. degrees from Jiangxi Normal University in 2002 and Xiamen University in 2005, respectively. He is currently a Professor and Doctoral Supervisor in Fujian Normal University of China. His research interests include Group and Graph theory, Complex networks, Network computing and Fault diagnosis. He has published over 50 papers in refereed international journals on these topics since 2005.