Future Generation Computer Systems 62 (2016) 76–84

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Memory leakage-resilient searchable symmetric encryption

Shuguang Dai, Huige Li, Fangguo Zhang *

School of Information Science and Technology, Sun Yat-sen University, Guangzhou 510006, China
Guangdong Key Laboratory of Information Security Technology, Guangzhou 510006, China

HIGHLIGHTS

- Two symmetric searchable encryption protocols against memory leakage are proposed.
- The mechanism mainly relies on the character of physical unclonable functions.
- The attacker considered here is a non-volatile memory attacker.

ARTICLE INFO

Article history:
Received 30 April 2015
Received in revised form
20 October 2015
Accepted 4 November 2015
Available online 27 November 2015

Keywords:
Searchable symmetric encryption
Side-channel attacks
Physical unclonable functions (PUFs)
Memory leakage-resilient
Cloud-computing

ABSTRACT

Along with the popularization and rapid development of cloud-computing, more and more individuals and enterprises choose to store their data in cloud servers. However, in order to protect data privacy and deter illegal accesses, the data owner has to encrypt his data before outsourcing it to the cloud server. In this situation, searchable encryption, especially searchable symmetric encryption (SSE) has become one of the most important techniques in cloud-computing area. In the last few years, researchers have presented many secure and efficient SSE schemes. Like traditional encryption, the security of all existing SSE schemes are 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 the data owner's secret keys by a great variety of inexpensive and fast side channel attacks. Facing such attacks, all existing SSE schemes are no longer secure. In this paper, we investigate how to construct secure SSE schemes with the presence of memory attack. We firstly propose the formal definition of memory leakage-resilient searchable symmetric encryption (MLR-SSE, for short). Based on that, we present one adaptive MLR-SSE scheme and one efficient non-adaptive dynamic MLR-SSE scheme based on physical unclonable functions (PUFs), and formally prove their security in terms of our security definitions.

1. Introduction

With the rapid development of information technology, human started to enter the digitalized era, the traditional paper data is gradually replaced by electronic data. Meanwhile, with the continuous expansion of social information, the data we need to store or maintain become increasingly enormous. To reduce the data owner's data management costs, cloud-computing arises at the historic moment. Cloud-computing is a centralized and powerful computing paradigm, in which users' data are moved to the servers in the "cloud" managed by the cloud service providers in the Internet [1]. Although cloud-computing brings data owner great benefits, it also raises security and privacy concerns. The data owners worry about their data may be leaked and abused by some unsolicited visitors. Therefore, in order to protect data privacy and deter illegal accesses, the data owners have to encrypt their data before outsourcing it to the cloud servers. However, data encryption often makes the data owner losing the ability to query it efficiently.

To solve this dilemma, a new cryptographic primitive, searchable encryption, was proposed by Song et al. [2]. Searchable encryption schemes enable user to search on the encrypted data while protecting the confidentiality of data and queries. Generally, searchable encryption schemes can be divided into searchable symmetric encryption (SSE) and public key encryption with keyword search (PEKS). Although PEKS schemes, introduced by Boneh et al. [3], are often more expressive and support more flexible search queries, they have the disadvantage of low efficiency. In cloud-computing, due to the large amount of data, SSE schemes are more practical than PEKS schemes. Therefore, we will focus on how to construct secure and efficient SSE schemes.

* Corresponding author at: School of Information Science and Technology, Sun Yat-sen University, Guangzhou 510006, China.
E-mail address: isszhfg@mail.sysu.edu.cn (F. Zhang).

http://dx.doi.org/10.1016/j.future.2015.11.003
0167-739X/

In the seminal paper [2], Song et al. achieved searchable encryption by wisely constructing a particular two-tier encryption scheme. However, their construction was proven secure as an encryption scheme instead of a searchable encryption scheme and their scheme was inefficient; the search time was linear in the length of the documents collection. To overcome above shortcomings, in [4], Goh proposed the idea that data owner should build an "index" for the documents collection and the search task was operated on the index. On this basis, Goh introduced a formal security notion, semantic security against adaptive chosen keyword attack (IND-CKA), for SSE schemes with indexes, and proposed an IND-CKA secure SSE construction based on Bloom filters [5]. Unfortunately, their construction had a possibility of false positives. Based on the definition of IND-CKA, Chang et al. [6] presented the slightly stronger simulation based definition, and proposed a secure SSE construction without false positives. In CCS'06, Curtmola et al. [7] put forth the improved security definitions for both non-adaptive and adaptive SSE schemes based on ideal/real paradigm, and they introduced two efficient SSE schemes which were proved non-adaptive and adaptive respectively.

After them, how to construct adaptive secure SSE schemes supporting richer functionality became the focus of active research and researchers have got a lot of research achievements on this field, such as dynamic SSE schemes [8–10], multi-user SSE schemes [7,11,12], etc. However, above-mentioned SSE schemes all have the limitation that user can only query with a single keyword. In order to provide more practical searchable encryption technology, Golle et al. [13] proposed the notion of secure conjunctive keyword search. To further improve the practicality and efficiency of SSE schemes, Cash et al. [14] constructed an efficient SSE scheme that supported conjunctive keywords search and general Boolean queries on the encrypted data. In [12], Jarecki et al. generalized the schemes in [14] to the multi-user case. Obviously, above SSE schemes support only exact keyword search. To improve the faults tolerant reliability of SSE and tolerate some minor typos and format inconsistencies, Li et al. [15] introduced the notion of fuzzy keyword search over encrypted data. In [16], Wang et al. constructed more powerful SSE scheme that supported the similarity search. Still later, for providing a better user experience, Cao et al. [17] introduced the idea that rank the search results and return the more matching results to the cloud-computing users.

Through the introduction above, we may hasty draw the following conclusion: above mentioned SSE schemes provided enough security and privacy protection for data owners. Unfortunately, we may be overly optimistic. Just like the traditional encryption protocol, the security of all existing SSE schemes based on the assumption that the internal state of data owner is completely inaccessible by the adversary. However, in the real-world implementations of SSE schemes, the adversary may be able to obtain some information about the data owner's internal states by exploiting some inexpensive and fast physical or side-channel attacks [18–22], then he can obtain some or even all of the data owner's secret keys. Facing memory leakage or secret key leakage, the existing SSE schemes are no longer secure obviously.

A physically unclonable function (PUF), introduced by Pappu et al. [23], is a noisy function that is implemented by a physical system [24]. Its merits are lower cost and resisting to tamper. Its randomness is determined by its physical factors and it also saves the conventional nonvolatile storage steps, which can enhance the ability of resistance to detecting-attack and side-attack so as to achieve the purpose of protecting the secret key. Combining PUF with SSE can strengthen the safety of private-key. Besides, the number of documents are not fixed, namely, it needs to change continually for practicability. It is an interesting work to combine PUF, SSE and dynamic together.

Our contribution. In this work, we focus on the problem of constructing efficient SSE schemes against memory attacks [25]. We make the following contributions:

1. We present the formal definitions for memory leakage-resilient SSE (MLR-SSE, for short). Our definitions can be viewed as the improvement of the ideal/real paradigm based indistinguishability security in [7].
2. We construct two MLR-SSE schemes based on physically unclonable functions (PUFs). The first is an adaptive MLR-SSE scheme. In order to be practical, we design a non-adaptive dynamic MLR-SSE scheme, then we prove their security in the presence of non-adaptive memory attacker and adaptive memory attacker respectively. As far as we know, there is no existing secure SSE scheme against memory leakage adversary.

Organization. The rest of this article is organized as follows. In Section 2, we present some preliminaries that will be used in our construction. In Section 3, we propose the formal definitions of MLR-SSE. We present our MLR-SSE schemes and their security analysis in Section 4. In Section 5, we discuss how to apply our idea to a more sophisticated environment. Finally, Section 6 is conclusion.

2. Preliminary

In this section, we present some basic definitions and tools for this paper.

Notations. We use $a \leftarrow_R A$ to denote that a is picked uniformly at random from the set A, and $a, b, c \leftarrow_R A$ to denote that a, b, c are picked uniformly at random from the set A. Let $\mathbb{N}$ be the set of natural numbers; for $n \in \mathbb{N}$, we denote $[1, n]$ for $\{1, \ldots, n\}$ and $negl(n)$ for a negligible function of n. Let $\mathbb{Z}$ be the set of integers, $\mathbb{Z}_q (q > 0)$ denote the set of integers $\{0, 1, \ldots, q-1\}$ and $\mathbb{Z}_q^*$ be the set of integers $\{1, \ldots, q-1\}$. We write PPT for probabilistic polynomial time. The average conditional min-entropy of X given a random variable Y, denoted $\tilde{H}_\infty(X|Z)$, is defined as $\tilde{H}_\infty(X|Z) = -log_2(E_{y \leftarrow Y}[max_x Pr[X = x|Y = y]])$.

2.1. System model of SSE

In the earlier single-user SSE schemes, there are two distinct entities, the data owner and the cloud server. The architecture of the single-user SSE is illustrated in Fig. 1. The data owner has a documents collection $\mathcal{DB} = \{DB_1, \ldots, DB_n\}$, he scans $\mathcal{DB}$ and builds the set $\mathcal{W}$ which includes the distinct keywords in $\mathcal{DB}$. And then, he builds a searchable index $\mathcal{I}$ based on $\mathcal{DB}$ and $\mathcal{W}$. Let Δ be the collection of all possible keywords. Obviously, $\mathcal{W} \subseteq \Delta$. Notice that, we can consider each word as a keyword. In this situation, Δ can be considered as a dictionary and $\mathcal{W}$ is the collection of the different words in $\mathcal{DB}$. After he builds index, the data owner encrypts $\mathcal{DB}$ with his secret key K, and then outsources both encrypted $\mathcal{DB}$ and $\mathcal{I}$ to the cloud server. To search the data collection for keyword w_i, the data owner submits a trapdoor T_{w_i} to the server and obtains the corresponding collection of encrypted documents $\mathcal{DB}(w_i)$ from the server. The whole framework should be divided into the following five algorithms [7]:

- $K \leftarrow$ **KeyGen**(1^κ)
  It takes a security parameter κ as input, and outputs a secret key K.
- $\mathcal{I} \leftarrow$ **BuildIndex**$(\mathcal{DB}, \mathcal{W}, K)$
  It takes the secret key K, set of documents $\mathcal{DB}$, and keywords set $\mathcal{W}$ as input, and outputs a searchable index $\mathcal{I}$.
- $\mathcal{C} \leftarrow$ **Enc**$(\mathcal{DB}, K)$
  It takes as input the secret key K and the documents collection $\mathcal{DB}$, and outputs the encrypted documents collection $\mathcal{C}$. After that, the data owner outsources both $\mathcal{C}$ and $\mathcal{I}$ to the cloud server.