Searchable Symmetric Encryption: Designs and Challenges

Article *in* ACM Computing Surveys · May 2017

DOI: 10.1145/3064005

CITATIONS
104

READS
1,260

5 authors, including:

Geong Sen Poh

50 PUBLICATIONS **635** CITATIONS

SEE PROFILE

Ji-Jian Chin

Multimedia University

64 PUBLICATIONS **312** CITATIONS

SEE PROFILE

Kim-Kwang Raymond Choo

University of Texas at San Antonio

1,036 PUBLICATIONS **41,030** CITATIONS

SEE PROFILE

Searchable Symmetric Encryption: Designs and Challenges

GEONG SEN POH, MIMOS Berhad
JI-JIAN CHIN, Multimedia University
WEI-CHUEN YAU, Xiamen University Malaysia
KIM-KWANG RAYMOND CHOO, The University of Texas at San Antonio
MOESFA SOEHEILA MOHAMAD, MIMOS Berhad

Searchable Symmetric Encryption (SSE) when deployed in the cloud allows one to query encrypted data without the risk of data leakage. Despite the widespread interest, existing surveys do not examine in detail how SSE's underlying structures are designed and how these result in the many properties of a SSE scheme. This is the gap we seek to address, as well as presenting recent state-of-the-art advances on SSE. Specifically, we present a general framework and believe the discussions may lead to insights for potential new designs. We draw a few observations. First, most schemes use index table, where optimal index size and sublinear search can be achieved using an inverted index. Straightforward updating can only be achieved using direct index, but search time would be linear. A recent trend is the combinations of index table, and tree, deployed for efficient updating and storage. Secondly, mechanisms from related fields such as Oblivious RAM (ORAM) have been integrated to reduce leakages. However, using these mechanisms to minimise leakages in schemes with richer functionalities (e.g., ranked, range) is relatively unexplored. Thirdly, a new approach (e.g., multiple servers) is required to mitigate new and emerging attacks on leakage. Lastly, we observe that a proposed index may not be practically efficient when implemented, where I/O access must be taken into consideration.

CCS Concepts: • **Security and privacy → Privacy-preserving protocols**; *Block and stream ciphers*; *Hash functions and message authentication codes;*

Additional Key Words and Phrases: Searchable encryption, cloud security, privacy-preserving search, computing in the encrypted domain

ACM Reference Format:
Geong Sen Poh, Ji-Jian Chin, Wei-Chuen Yau, Kim-Kwang Raymond Choo, and Moesfa Soeheila Mohamad. 2017. Searchable symmetric encryption: Designs and challenges. ACM Comput. Surv. 50, 3, Article 40 (May 2017), 37 pages.
DOI: http://dx.doi.org/10.1145/3064005

1. INTRODUCTION

Cloud services are widely used by both individual users and organisations. One of the most popular (or "mainstream") cloud services is data outsourcing, which allows an organisation to reduce operational costs and offers ease of data access (i.e., 24/7 access without geographical constraints). In such a setting, the cloud storage providers have

K.-K. R. Choo is funded by the Cloud Technology Endowed Professorship from the 80/20 Foundation, Rackspace and other industry partners.
Authors' addresses: G. S. Poh and M. S. Mohamad, Information Security Lab, MIMOS Berhad, Technology Park Malaysia, 57000, Kuala Lumpur, Malaysia; emails: {gspoh, soeheila.mohamad}@mimos.my; J.-J. Chin, Faculty of Engineering, Multimedia University, 63100 Cyberjaya, Selangor, Malaysia; email: jjchin@mmu.edu.my; W.-C. Yau, Xiamen University Malaysia, Jalan Sunsuria, Bandar Sunsuria, 43900 Sepang, Selangor; email: wcyau@xmu.edu.my; K.-K. R. Choo, Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249, USA; email: raymond.choo@fulbrightmail.org.

DOI: http://dx.doi.org/10.1145/3064005

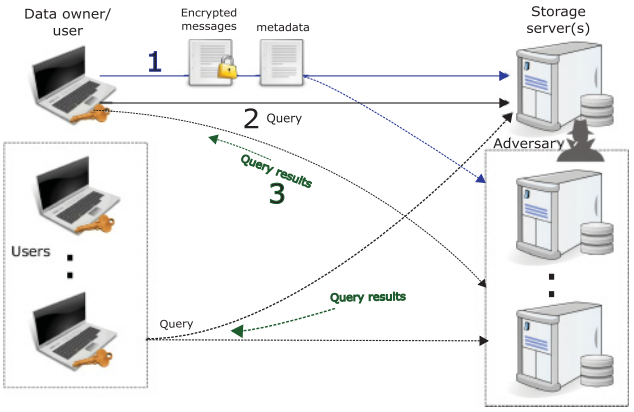

Fig. 1. General SSE architecture without intermediary server(s). 1. Data owner encrypts messages and generates metadata, sends and stores them at storage server(s). 2. Data owner or user queries the metadata at the storage server(s). 3. Storage server(s) returns query results (e.g., matching encrypted messages).

access to the stored data unless data are encrypted prior to outsourcing. Encrypted data, however, complicates user querying. A naive approach is to provide cloud storage providers the encryption key, which can be used to decrypt all data and search on the (decrypted) data to retrieve items of interest. This approach, however, defeats the purpose of encrypting the data as a corrupted insider (e.g., corrupted cloud storage provider's employee or a compromised machine within the cloud storage provider's premise) can gain unauthorised access to user data. Alternatively, we can send all encrypted data to the user to be decrypted prior to searching for the required items. This is clearly an inefficient and expensive approach due to costs incurred for the uploading and downloading of a large amount of data.

To address the above issue, searchable encryption was proposed as a mechanism to allow querying of the encrypted data without the server learning any information on the data. The first practical scheme was introduced by Song et al. [2000], using only symmetric primitives. Since their seminal article, a number of schemes based on this similar concept were proposed. These schemes are collectively known as Searchable Symmetric Encryption (SSE) schemes.

In general, most SSE schemes deploy a masked index table as metadata that facilitates encrypted search (see Goh [2003], Chang and Mitzenmacher [2005], Curtmola et al. [2006], Chase and Kamara [2010], Kamara et al. [2012], Cash et al. [2013, 2014], Naveed et al. [2014], and Kurosawa and Ohtaki [2015]). Newer schemes such as those presented in Bösch et al. [2014], Ishai et al. [2015], and Orencik et al. [2016] also introduce schemes with multiple servers. In these schemes, given a set of messages, a user first encrypts the messages using a symmetric encryption scheme with a secret key as input. The user also creates a masked index table based on pre-processed message-keyword pairs. The encrypted messages and the masked index table are submitted to the storage or query server(s). To perform a search, the user generates a search token. Using this token, the server searches through the index. If a match is found, then the matching encrypted data is returned to the user. An exception is the scheme of Song et al. [2000], which directly searches encrypted data via a specially crafted symmetric encryption construct, without the need for any masked index. Many schemes assume an honest-but-curious adversary that has access to the server(s).

Figure 1 illustrates the architecture in which users communicate directly with storage servers that store metadata (e.g., masked indexes) and encrypted messages. Figure 2, on the other hand, presents an architecture whereby a query server is deployed

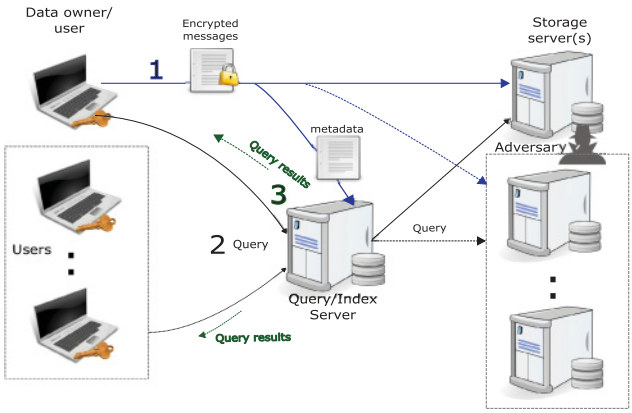

Fig. 2. General SSE architecture with helper server(s). 1. Data owner encrypts messages and generates metadata, and sends and stores encrypted messages at storage server(s) and metadata at a query (or index or helper) server. 2. Data owner or user sends queries and the query server collectively computes query results with storage server(s). 3. Query server returns query results (e.g., matching message identifiers).

to provide query services, usually for the purpose of minimising information leakages. Encryption keys are always generated and kept secret by the data owner. Depending on the scenario, other users may also submit encrypted messages or have access to keys to generate query tokens. In both Figures 1 and 2, information flow with dotted lines denote the setting for multiple users and/or multiple servers. The process of updating (i.e., addition and deletion of keywords and messages) is not shown in both figures. These activities normally involve similar processes of providing/removing encrypted messages and updating of the metadata efficiently and securely.

A comprehensive survey on searchable encryption that includes SSE was presented by Bösch et al. [2014], where an overview on key techniques was presented for a non-specialist audience. The survey does not provide in-depth technical discussion on the structures, properties, security and efficiency of a SSE scheme. Prior to this, Kamara also provided a brief overview on searchable encryption [Kamara 2013a, 2013b]. Other surveys include those of Tang et al. [2016], Han et al. [2016], Li and Gao [2016], Tang [2012], and Liu et al. [2016a]. These surveys share a similar focus with the survey of Bösch et al.'s. Thus, in this article, we seek to complement these existing surveys by presenting an in-depth study of SSE schemes. Specifically, we systematically examine the underlying designs and structures of existing schemes to obtain insights on their benefits and pitfalls, and identify research challenges and potential research topics.

We remark that the surveys in Bösch et al. [2014], Kamara [2013a], Kamara [2013b], Han et al. [2016], and Tang [2012] also investigate searchable encryption based on public key settings, which was first proposed by Boneh et al. [2004] and also known as Public Key Encryption with Keyword Search (PEKS). In our study, however, we do not consider PEKS based on asymmetric primitives, which differ substantially from SSE's main symmetric constructs. We further note that in a broader sense, the concept of processing encrypted data (privacy homomorphisms) introduced by Rivest et al. [1978] can be applied to searchable encryption. Many applications were proposed using homomorphic primitives such as in the signal processing community [Katzenbeisser 2007; Bianchi et al. 2009; Rahulamathavan et al. 2013; Frattolillo 2015] and more recently skyline computation for big data [Liu et al. 2016b]. It has become an active field in recent times, partly due to the proposal of fully homomorphic encryption (FHE) scheme by Gentry [2009].