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

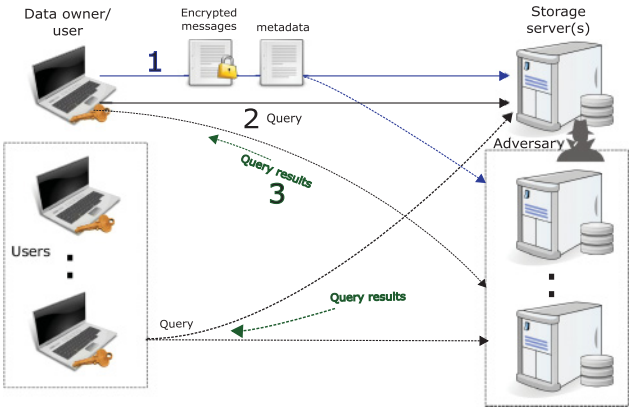


Fig. 1. General SSE architecture without intermediary server(s). 1. Data owner encrypts messages and generates metadata, sends and stores them at storage server(s). 2. Data owner or user queries the metadata at the storage server(s). 3. Storage server(s) returns query results (e.g., matching encrypted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

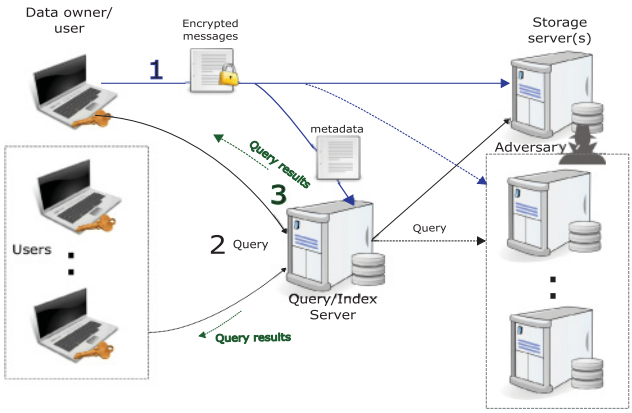


Fig. 2. General SSE architecture with helper server(s). 1. Data owner encrypts messages and generates metadata, and sends and stores encrypted messages at storage server(s) and metadata at a query (or index or helper) server. 2. Data owner or user sends queries and the query server collectively computes 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].

FHE, together with functional encryption [Boneh et al. 2012; Boneh and Waters 2007; Goldwasser et al. 2014] can be used to build searchable encryption schemes that are fully hidden. However, at the time of this research, SSE is still more practical and efficient in comparison to these other cryptographic schemes, despite having a weaker (yet controlled) security guarantee in the form of allowable leakages. In fact, SSE schemes have been shown to be capable of handling huge datasets, a typical requirement in today's data-driven society [Cash et al. 2013].

In this article, we studied 84 English language articles on SSE schemes published between 2000 and 2016. These articles were gathered from academic databases such as Springer, ACM Digital Library, IEEEXplore, and ScienceDirect, as well as IACR ePrint archive using the following keywords: searchable encryption, searchable symmetric encryption and secure cloud storage. Figure 3 shows the timeline and categorisations of the SSE articles that we have compiled. It can be observed that many SSE schemes (including attacks) were proposed beginning from 2011. Most schemes adopt the constructs and models of the early proposals of Song et al. [2000], Goh [2003], Chang and Mitzenmacher [2004], and Curtmola et al. [2006]. We note that the categorisations are loose and high level, because a scheme may have a combination of properties. A detailed categorisation is discussed in subsequent sections and summarised in Tables I, II, III, and IV.

We will now describe how the article is organised. In Section 2, we present a general framework of SSE including the involved principals, definitions and required cryptographic primitives. Section 3 discusses the different data structures applied to SSE schemes, in terms of specific properties, performance, and security assurance of a scheme. A comparative summary of the different security models is presented in Section 4. Sections 5 and 6 examine performance and the different properties of existing schemes, respectively. We discuss the challenges and future directions of SSE in Section 7 before concluding the article in Section 8.

2. GENERAL FRAMEWORK

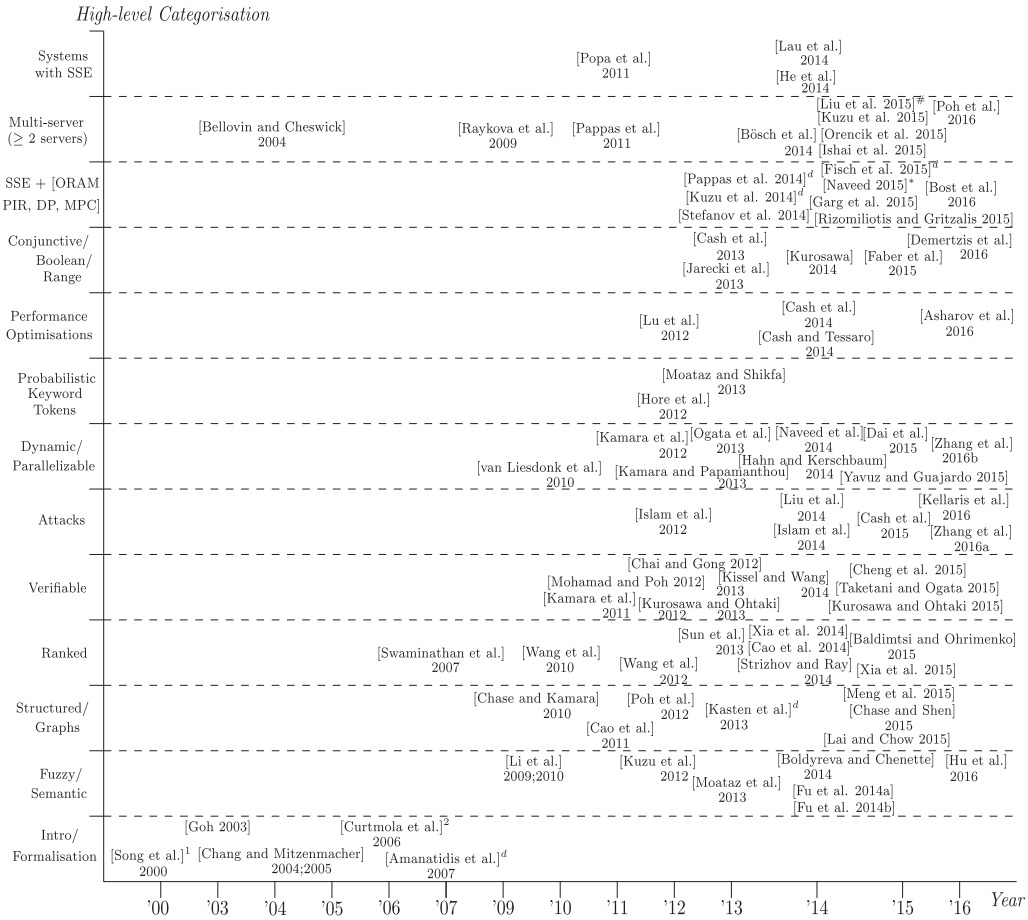
Figure 4 provides a high-level framework that lists the different properties of SSE schemes. Tables I, II, III, and IV, on the other hand, provide categorisations of SSE schemes based on this framework. These were derived based on our study of schemes proposed in the compiled articles. We also refer to the four tables whenever there is a need to state schemes that share similar properties. We note that attack and system articles (which use SSE as only part of the system) were not shown. Attack models are described in Section 4. In the following subsections, we describe the principals, the underlying cryptographic primitives, the main operations (or functions) of a SSE scheme, and the other properties in the framework.

SSE schemes generally involve two principals, and for most of the schemes, indices (or dictionaries) serve as the underlying data structure. There are mainly four approaches (stated as *structures* under Figure 4) in designing a SSE scheme, namely without index (using words/blocks), with direct indexing, with inverted indexing, or, more recently, with tree structures. Most of the SSE schemes in general assume that the keyword list and the keywords-messages pairs are pre-processed and are readily available.

2.1. Principals

The two main principals are users and servers. A user communicates with a server to outsource or query data. A server stores, searches, and/or retrieves a user's data. It is normally assumed that servers have higher computing power than users.

—*Users' roles and characteristics.* There are two types of users, namely a data owner, and a user who queries for information. A data owner is a user who outsources data



Notations: 1: Song et al. 2000 introduced SSE. 2: Curtmola et al. 2006 provided an improved formal security model, where most of the SSE schemes based on. d: Schemes that focus on database (e.g. relational database) queries. *: Naveed 2015 discussed leakages when using ORAM to build SSE. #: Liu et al. 2015 discussed multiple data sources from multiple users instead of multiple servers. PIR: Private Information Retrieval. DP: Differential Privacy. MPC: Multi-party computation.
Notes: In addition to systems that use SSE, there are many SSE prototypes and implementations, including [Kamara et al. 2011; Cao et al. 2014; Cash et al. 2014; Naveed et al. 2014; Ishai et al. 2015; Faber et al. 2015; Tan et al. 2015; Salam et al. 2015].

Fig. 3. SSE timeline and high-level categorisations.

to storage server(s) by using a SSE scheme to encrypt and store encrypted data (both messages and metadata). User(s) other than the data owner are normally only allowed to perform queries. We note that a data owner can also query for information. In most of the cases where the data owner is the sole user, the data owner is fully trusted. If there are multiple users, then the data owner and the other users may be considered malicious. The data owner might attempt to guess the search terms of others, while other users might attempt to learn more information than they are allowed to search for.

—*Servers' roles and characteristics.* There are two types of servers, namely a storage server and a query server. A storage server stores encrypted messages, and a query (or helper or index) server stores metadata, such as masked indexes. A server should

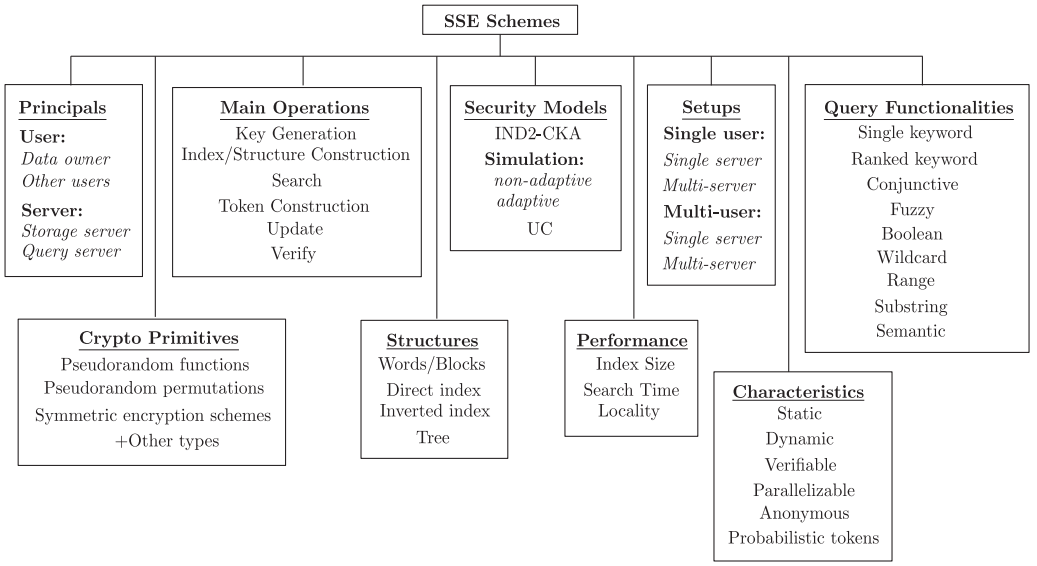


Fig. 4. A general framework.

not be able to learn any information from the stored data except for some allowable leakages. In most of the SSE schemes, a storage server provides both storage facilities and query functionalities. Both servers are considered honest-but-curious in the sense that it may try to learn as much information as possible from the stored encrypted data and/or metadata but at the same time correctly execute the SSE protocol. This is practical, since the server may wish to be able to profile the users for targeted advertisements yet may prefer to safeguard the user's information from being corrupted or modified. Certainly such a notion can be extended to malicious server who is able to also modify the stored encrypted content (as was proposed, for example, in Kamara et al. [2011] and Kurosawa and Ohtaki [2012]).

2.2. Cryptographic Primitives

The main building blocks required in the construction of SSE includes pseudo-random functions (PRFs), pseudo-random permutations (PRPs), and symmetric encryption schemes. PRFs and PRPs are polynomial-time computable functions, where a PRF, $F : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^\lambda$, for $k, \lambda \in \mathbb{N}$, is computationally indistinguishable from a random function when observed by any probabilistic polynomial-time adversary. The function F is a PRP if it is bijective. A symmetric encryption scheme is a tuple of three polynomial-time algorithms $(\mathtt{Gen}, \mathtt{E}, \mathtt{D})$ where $\mathtt{Gen}$ is a probabilistic key generation algorithm with input 1^k, produces a secret key K; $\mathtt{E}$ is a probabilistic encryption algorithm that with input K and a message m returns a ciphertext c; $\mathtt{D}$ is a deterministic decryption algorithm that takes as input K and c and returns m. For correctness, $m \leftarrow \mathtt{D}(K, \mathtt{E}(K, m))$ for every message m, where K is the key used to produce c. Formal definitions of these primitives can be found in Katz and Lindell [2007].

We remark that while these primitives represent the main building blocks for SSE, many recent schemes combine these primitives with more expensive mechanisms such as pairings, ORAM, PIR, and MPC to reduce information leakages. For instance, Cash et al. [2013] proposed schemes that use a Diffie-Hellman type of blinding to prevent a server from learning statistical information between the query results of the multiple

Table I. Development of SSE Schemes: Timeline and Properties (2000–2013)

Sch./Year/Sec.	Setups		Query Functionalities									Structures			
	u	s	1w	rnk	cnj	fz	Bl	wc	rge	ss	gr	w	di	inv	t
[Song et al. 2000]	1	1	•				•	•				•			
[Goh 2003]	1	1	•										•		
[Bellovin and Cheswick 2004]	u	2	•										•		
[Chang and Mitzenmacher 2005]	1	1	•										•		
[Curtmola et al. 2006]	u	1	•											•	
[Amanatidis et al. 2007][d]	1	1	•									•			
[Raykova et al. 2009]	u	2	•				•						•		
[Li et al. 2009]	1	1	•			•								•	
[van Liesdonk et al. 2010]	1	1	•											•	
[Chase and Kamara 2010]	1	1	•								•			•	
[Wang et al. 2010]	1	1	•	•										•	
[Pappas et al. 2011]	u	2	•				•		•				•		
[Kamara et al. 2011]	1	1	•											•	
[Cao et al. 2011]	1	1									•		•		
[Kuzu et al. 2012]	1	2	•			•								•	
[Poh et al. 2012]	1	1	•								•			•	
[Wang et al. 2012]	1	1	•	•										•	
[Kurosawa and Ohtaki 2012]	1	1	•											•	
[Chai and Gong 2012]	1	1	•												•
[Mohamad and Poh 2012]	1	1	•								•			•	
[Kamara et al. 2012]	1	1	•											•	
[Hore et al. 2012]	1	1	•										•		
[Lu et al. 2012]	1	1	•											•	•
[Ogata et al. 2013]	1	1	•											•	
[Moataz et al. 2013]	1	1	•			•								•	
[Kasten et al. 2013][d]	u	1	•								•	•			
[Sun et al. 2013]	1	1	○	•											•
[Kissel and Wang 2013]	1	1	•		•									•	•
[Kurosawa and Ohtaki 2013]	1	1	•											•	
[Kamara and Papamanthou 2013]	1	1	•												•
[Moataz and Shikfa 2013]	1	1	○		•		•						•		
[Cash et al. 2013]	1	1	○		•		•							•	
[Jarecki et al. 2013]	u	1	○		•		•							•	

Notation: u = user/multiple users, s = server/multiple servers, 1w = single keyword, rnk = ranked, cnj = conjunctive, fz = fuzzy, Bl = Boolean, wc = wildcard, rge = range, ss = substring, gr = graph, w = word/block-based, di = direct, inv = inverted, t = tree.
○: The scheme provides multi-keyword search, but can be used for single-keyword. A scheme with 1w = • but that also provides Bl (for example) is a single-keyword scheme but was extended to multi-keyword search.
x[d]: x is a scheme that primarily designed for database-type queries.

keywords and proposed an alternative that uses pairings to replace the Diffie-Hellman type of blinding.

2.3. Main Operations

The core operations of SSE are key generation, metadata (e.g., index) construction, and search. In certain schemes, there are separate operations for query token generation (such as in Chase and Kamara [2010]) or treating encryption/decryption of messages as a separate process (such as in Cash et al. [2014]). For verifiable schemes, there is an operation for verification as in Cheng et al. [2015], Kamara et al. [2011], Kurosawa and Ohtaki [2012], Mohamad and Poh [2012], while for dynamic schemes an update operation is included, for example, in Cash et al. [2014], Kamara et al. [2012], Naveed et al. [2014], and Kuzu et al. [2015].

—*Key Generation.* It is an algorithm that generates a key K with input 1^k, where k is the security parameter. For all SSE schemes, a key generation algorithm is required to generate a set of secret keys. In general, these keys consist of a secret key for the symmetric encryption scheme to encrypt/decrypt messages and key(s) for the encryption scheme and/or PRFs to generate tokens and to mask the indexes.

Table II. Development of SSE Schemes: Timeline and Properties (2000–2013)

Sch./Year/Sec.	Characteristics						Prim. (PRF, PRP, Enc.)	Security Model				
	sta	dyn	ver	an	pa	pt	+others (DH, ORAM etc.)	Ind	S1	S2	UC	sp
[Song et al. 2000]		•										•
[Goh 2003]		•						•				
[Bellovin and Cheswick 2004]		•					•* (Pohlig-Hellman)	•				
[Chang and Mitzenmacher 2005]		•							•			
[Curtmola et al. 2006]	•						• (Broadcast enc.)		•	•		
[Amanatidis et al. 2007][d]		•	•									•
[Raykova et al. 2009]		•		•			•* (Pohlig-Hellman)	•				
[Li et al. 2009]	•											•
[van Liesdonk et al. 2010]	•[1]	•[2]								•[1]		
[Chase and Kamara 2010]	•									•		
[Wang et al. 2010]	•						• (OPSE)					
[Pappas et al. 2011]		•					•* (Pohlig-Hellman)					
[Kamara et al. 2011]		•	•				• (Incremental hash)			•		
[Cao et al. 2011]	•						•* (Secure inner product)					
[Kuzu et al. 2012]	•						• (Homomorphic enc.)			•		
[Poh et al. 2012]	•									•		
[Wang et al. 2012]	•						• (OPSE)					
[Kurosawa and Ohtaki 2012]	•		•								•	
[Chai and Gong 2012]	•		•									•
[Mohamad and Poh 2012]	•		•							•		
[Kamara et al. 2012]		•								•		
[Hore et al. 2012]		•				•						•
[Lu et al. 2012]	•								•			
[Ogata et al. 2013]		•							•			
[Moataz et al. 2013]	•											
[Kasten et al. 2013][d]		•										
[Sun et al. 2013]						•						
[Kissel and Wang 2013]	•		•						•			
[Kurosawa and Ohtaki 2013]		•	•				• (RSA Accumulator)				•	
[Kamara and Papamanthou 2013]		•			•					•		
[Moataz and Shikfa 2013]	•					•	• (Gram-Schmidt process)			•		
[Cash et al. 2013]	•						• (DH, Pairings)			•		
[Jarecki et al. 2013]	•						• (DH, homomorphic sig.)			•		

Notation: sta = static, dyn = dynamic, ver = verifiable, an = anonymous, pa = parallelizable, pt = probabilistic tokens, ind = indistinguishability-based, S1 = non-adaptive simulation, S2 = adaptive simulation, UC = universal composability, sp = specific to scheme.
Primitives: Enc. = Symmetric encryption, OPSE = Order Preserving Symmetric Encryption, DH = Diffie-Hellman. All schemes use PRF, PRP, and Enc. by default. Extra primitives are denoted with • and stated primitives. Certain schemes are without PRF, PRP, and Enc (•*). Non-crypto primitives (e.g., term frequency) are not included. •[1] and •[2] denote two schemes with different security assurance.
x[d]: x is a scheme that primarily designed for database-type queries.

—*Index / Structure Construction.* It is an algorithm that takes as input K the list of keywords or keyword-message pairs, and other scheme-specific information, and outputs an encrypted data structure, which is either a set of tagged encrypted messages or masked indexes coupled with a set of ciphertexts. This is the core algorithm that differs significantly among the various SSE schemes, providing for a scheme's main properties and security assurances.
—*Search.* It is a protocol that takes as input K an encrypted data structure and a search token or a list of tokens, depending on the scheme. It outputs a set of pointers to the matched messages or the set of pointers coupled with the matching encrypted messages.
—*Token Construction.* It is an algorithm that takes as input K and a query keyword and outputs a query token. Depending on the scheme, this operation may be embedded directly in the search protocol instead of being a separate function.

Table III. Development of SSE Schemes: Timeline and Properties (2014–2016)

Sch./Year/Sec.	Setups		Query Functionalities									Structures			
	u	s	1w	rnk	cnj	fz	Bl	wc	rge	ss	gr	w	di	inv	t
[Boldyreva and Chenette 2014]	1	1	●			●									●
[Fu et al. 2014b]	1	1	●			●									●
[Fu et al. 2014a]	1	1	●			●									●
[Strizhov and Ray 2014]	1	1	○	●										●	
[Cao et al. 2014]	1	1	○	●									●		
[Xia et al. 2014]	1	1	○	●		●								●	
[Naveed et al. 2014]	1	1	●											●	
[Hahn and Kerschbaum 2014]	1	1	●										●	●	
[Cash et al. 2014]	1	1	●											●	
[Cash and Tessaro 2014]	1	1	●											●	
[Kurosawa 2014]	1	1	○		●		●						●		
[Stefanov et al. 2014]	1	1	●												●
[Kuzu et al. 2014][d]	u	1	●											●	
[Pappas et al. 2014][d]	1	3	○		●		●		●						●
[Bösch et al. 2014]	1	2	●											●	
[Dai et al. 2016]	1	1	●											●	
[Lai and Chow 2015]	1	1	●								●				●
[Chase and Shen 2015]	1	1	●							●					●
[Meng et al. 2015]	1	1									●			●	
[Xia et al. 2015]	1	1	○	●											●
[Baldimtsi and Ohrimenko 2015]	1	2	○	●										●	
[Kurosawa and Ohtaki 2015]	1	1	●											●	
[Taketani and Ogata 2015]	1	1	●											●	
[Cheng et al. 2015]	1	1	●											●	
[Yavuz and Guajardo 2015]	1	1	●										●	●	
[Faber et al. 2015]	u	1	○		●		●	●	●	●				●	●
[Rizomiliotis and Gritzalis 2015]	1	1	●											●	
[Garg et al. 2015]	1	1	●											●	
[Fisch et al. 2015][d]	1	3	○		●		●		●						●
[Ishai et al. 2015]	1	2	○						●	●					●
[Orencik et al. 2016]	1	2	○	●		●								●	
[Kuzu et al. 2015]	u	s	●											●	
[Liu et al. 2015]	u[γ]	1	●											●	
[Hu et al. 2016]	1	1	●			●		●						●	
[Asharov et al. 2016]	1	1	●											●	
[Bost et al. 2016]	1	1	●												●
[Poh et al. 2016]	1	s	●											●	
[Zhang et al. 2016b]	1	1	●											●	
[Demertzis et al. 2016]	1	1	○						●						●

u[γ]: Multi-user with data sources coming from many different users.

—*Update.* It is a protocol that takes as input K and an instruction to add or delete information from the server(s). This information normally is a new message/keyword (or a set of new messages/keywords) to be added to the server or can be an instruction to delete an existing message (or a set of existing messages).
—*Verify.* It is an algorithm that takes as input K the retrieved identifiers/encrypted messages from the query and authentication tokens and outputs `accept` if the inputs are authentic or `reject` otherwise.

For correctness, generally it is said that a SSE scheme is correct if for all K generated from the key generation algorithm, for all encrypted data structures and ciphertexts generated from the index construction algorithm, for all tokens generated from the token algorithm, and for all update operations (for dynamic schemes) output of the search results from the search protocol matches the original list of keyword-

Table IV. Development of SSE Schemes: Timeline and Properties (2014–2016)

Sch./Year/Sec.	Characteristics						Prim. (PRF, PRP, Enc.)	Security Model				
	sta	dyn	ver	an	pa	pt	+others (DH, ORAM etc.)	Ind	S1	S2	UC	sp
[Boldyreva and Chenette 2014]	•											•
[Fu et al. 2014b]	•											
[Fu et al. 2014a]	•		•									
[Strizhov and Ray 2014]	•						• (Homomorphic enc.)			•		
[Cao et al. 2014]	•											
[Xia et al. 2014]	•						• (OPE)					
[Naveed et al. 2014]		•									•	
[Hahn and Kerschbaum 2014]		•								•		
[Cash et al. 2014]	•[1]	•[2]			•				•[2]	•[1]		
[Cash and Tessaro 2014]	•								•			
[Kurosawa 2014]	•						• (Garbled cir.)		•			
[Stefanov et al. 2014]		•[α]					• (ORAM)			•		
[Kuzu et al. 2014][d]	•						• (DP)					•
[Pappas et al. 2014][d]	•						• (Garbled cir.)				•	
[Bösch et al. 2014]		•								•		
[Dai et al. 2016]	•[1]	•[2]					• (PUFs)		•[2]	•[1]		
[Lai and Chow 2015]		•			•							
[Chase and Shen 2015]	•									•		
[Meng et al. 2015]	•						• (Homomorpic enc.)			•		
[Xia et al. 2015]		•										
[Baldimtsi and Ohrimenko 2015]	•						• (Homomorpic enc.)			•		
[Kurosawa and Ohtaki 2015]	•		•								•	
[Taketani and Ogata 2015]	•		•								•	
[Cheng et al. 2015]	•		•				• (Ind. obfuscation)		•			
[Yavuz and Guajardo 2015]		•			•					•		
[Faber et al. 2015]		•					• (=[Cash et al. 2013])			•		
[Rizomiliotis and Gritzalis 2015]		•			•		• (ORAM)			•		
[Garg et al. 2015]		•					• (Garbled cir., ORAM)			•		
[Fisch et al. 2015][d]	•						• (Garbled cir.)	•			•	
[Ishai et al. 2015]	•						• (ORAM, PIR, MPC)				•	
[Orencik et al. 2016]	•					•	• (Homomorphic enc.)			•		
[Kuzu et al. 2015]		•			•					•		
[Liu et al. 2015]	•									•		
[Hu et al. 2016]	•							•				
[Asharov et al. 2016]	•									•		
[Bost et al. 2016]		•[α]	•				• (ORAM)			•		
[Poh et al. 2016]		•								•		
[Zhang et al. 2016b]		•					• (Pairings)					•
[Demertzis et al. 2016]		•[α]					• (Delegatable PRF)			•		

Primitives: OPE = Order Preserving Encryption, DP = differential privacy, PUFs = Physical Unclonable Functions, PIR = Private Information Retrieval, MPC = Multi-Party Computation. α = Dynamic schemes with forward privacy (Section 4.2).

message pairs being queried, and the verify algorithm returns `accept` (for verifiable schemes).

3. STRUCTURES

The underlying design concept of SSE is to

(1) generate cryptographic keys for message encryption and masking of tags/indices,
(2) encrypt messages or blocks of messages using a symmetric encryption scheme,
(3) *link* these encrypted messages or blocks to their matching keywords, using either tags to the blocks or index as the data structure,

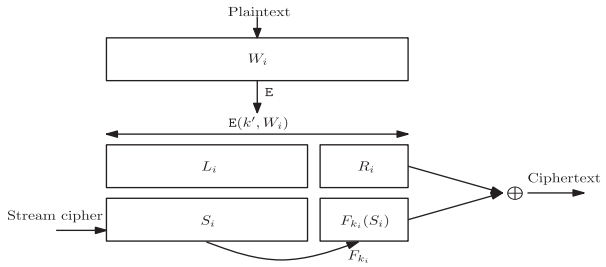


Fig. 5. Song, Wagner and Perrig's Scheme IV Song et al. [2000].

(4) mask the tags/indices using PRFs (and PRPs) in such a way that the *link* to the encrypted messages/blocks are maintained, without revealing the *link* to preserve keyword and index privacy, and
(5) allow for privacy-preserving query of messages through keyword tokens and the masked tags/indices.

For (4), and with reference to index-based data structure used in most of the schemes, the general masking technique is to run a keyed PRF on the keyword w, resulting in $F_{k_1}(w)$, and another keyed PRF F_{k_2} on the keyword-message identifier pair. The first PRF operation masks the keyword in such a way that when query, the user sends $F_{k_1}(w)$ as query instead of w. Similarly, instead of storing the plain message identifiers matching the keyword, the masked identifiers from the output of F_{k_2} are stored. For some schemes, the identifiers are permuted using PRP to obfuscate the message sequences, such as in Cash et al. [2013], Chase and Kamara [2010], Jarecki et al. [2013], and Naveed et al. [2014].

A general assumption of SSE is that there is a pre-processing stage. It prepares the set of keywords and/or the list of plain keyword-message identifier pairs, which is not considered as part of a SSE scheme.

We will now discuss the four design approaches (see Figure 4, Structures). We describe at least a scheme for every approach to give a clear idea on how these approaches work. We choose a scheme that, as far as we know, are among the first to use an approach and further discuss later scheme(s) that deploy a similar approach. Most subsequent schemes create variants or new structures based on the concepts of these early schemes. We further discuss how each approach influences the performance and properties (e.g., characteristics) of SSE schemes.

3.1. Construction without Index

The first approach is to construct an SSE without index. This approach was introduced in the seminal article of Song et al. [2000]. Messages are encrypted in such a way that they are searchable without the need for separate metadata. We briefly describe Song et al.'s Scheme IV as an example in (Figure 5).

The general idea of the scheme is to divide a message into a list of words w_i. Each word is then encrypted using a symmetric encryption scheme and a key k', resulting in $\mathrm{E}(k', w_i)$. Then, $\mathrm{E}(k', w_i)$ is divided into a left half L_i and a right half R_i. Using a stream cipher (which can be constructed from a PRF), a pseudo-random bit stream S_i, of size L_i, is generated. To create a searchable encrypted word, the scheme generates $F_{k_i}(S_i)$, where F is a keyed PRF, $k_i = F_k(L_i)$, and k is a secret key chosen by the user. The encrypted word is created as $C_i = (L_i, R_i) \oplus (S_i, F_{k_i}(S_i))$, where the purpose of XORing $(S_i, F_{k_i}(S_i))$ is to randomise the encrypted word (L_i, R_i). This is to hide the keyword distribution before query, since identical words in different locations of a message and

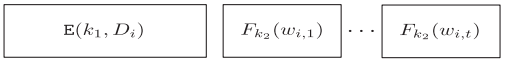


Fig. 6. Simple message-keywords approach.

other messages encrypt to identical ciphertexts. The encrypted word is divided into two halves (L_i, R_i), so it is possible for the user to decrypt a message without storing all encrypted words of a message, which would have defeated the purpose of the scheme. Specifically, if k_i is generated directly as $F_k(\mathrm{E}(k', w_i))$ instead of $F_k(L_i)$, then to decrypt, the user must know all encrypted words $\mathrm{E}(k', w_i)$ of a message.

To search for a keyword w_q, the user sends $\mathrm{E}(k', w_q)$ and $k_q = F_k(L_q)$ to the server. Given these, the server computes $\mathrm{E}(k', w_q) \oplus C_i$ for each encrypted word in a message and checks whether the result is of the form $(S_q, F_{k_q}(S_q))$. If it is, then a match has been found. This is assuming that the server has access to F and can generate S_q. Using this process, the server scans through every encrypted word of all the messages to find matching messages.

Based on the above, we may summarise that an SSE scheme without index requires (1) encrypting words in a message and (2) masking these encrypted words to avoid statistical learning of words in the message. This is because the cipher used is deterministic and similar words result in exact encrypted words if there is no masking.

Other schemes that can be loosely considered to be using word-based constructs without index (but utilising the underlying database systems) are schemes by Amanatidis et al. [2007] and Kasten et al. [2013]. Amanatidis et al. [2007]'s scheme symmetrically encrypts searchable attribute values of a record in a database. Query is performed by submitting the encrypted attribute values to the database management system. Kasten et al.'s scheme similarly encrypts searchable attribute values but based on semantic query language (SPARSQL).

Performance and Properties. In the following, we state the performance and properties of the approach:

—*Main Feature:* Encryption and search is performed directly on the words/or messages. No index is required.
—*Storage:* Schemes under this construction store searchable encrypted messages and do not require any extra storage space, thus requiring similar storage space as in storing plain messages.
—*Search:* Search time is linear to the total number of word/message, since every word/message must be checked to find the matching messages.
—*Characteristics (static or dynamic):* Schemes under this construction are inherently dynamic, since adding/removing a message is straightforward.
—*Security (leakage):* Existing schemes using this approach leak keyword distribution of the messages. For instance, query frequencies of an encrypted word can be registered, the encrypted word occurrence in messages may be counted, and locations of these words are revealed, as in Song et al. [2000]'s scheme. Access patterns, which relate to the list of messages returned through a query, are also revealed. These leakages may be used to statistically analyse to learn the actual plaintext messages.

3.2. Construction with Direct Index

In this approach, an index is built based on a (message, list of keywords) tuple. A simple mechanism is the message-keywords approach shown in Figure 6. A message D_i is encrypted using a symmetric encryption scheme, while keywords of the message

Table V. A Hypothetical Example of a Direct Index: Given a Set of *n* Messages *D* and *m* Keywords, *w*

Key	Value
D_1	$w_2, w_5, \ldots, w_m$
D_2	$w_1, w_7, \ldots, w_{30}$
$\vdots$	$\vdots$
D_n	$w_2, w_3, \ldots, w_{10}$

Table VI. An Example of an Index in Chang and Mitzenmacher [2005] for One Message. Assuming a Message D_1, a List of Eight KeyWords $\{w_1, w_2, \ldots, w_8\}$ in Total, and D_1 Contains w_2, w_3, w_4

Encrypted D_1	Masked list of keywords
$\mathrm{E}(k_1, D_1)$	$0 \oplus G_{r_1}(1)$
	$1 \oplus G_{r_2}(1)$
	$1 \oplus G_{r_3}(1)$
	$1 \oplus G_{r_4}(1)$
	$0 \oplus G_{r_5}(1)$
	$0 \oplus G_{r_6}(1)$
	$0 \oplus G_{r_7}(1)$
	$0 \oplus G_{r_8}(1)$

$(w_{i,1}, \ldots, w_{i,t})$ serve as input to a keyed PRF. The tokens generated by the PRF are then attached to the encrypted message before both tokens and ciphertext are submitted to the server. Query is straightforward and provides conjunctive search. Tokens are generated using the keyed PRF. If the tokens match the tokens of an encrypted message, then the encrypted message is returned. Recent systems that adopt this approach and its variants are M-Aegis [2014], ShadowCrypt [2014], and products by Skyhigh Networks [2016], CipherCloud [2016], and Bitglass [2016] as stated in Cash et al. [2015]. This approach is simpler compared to Song et al. [2000]'s method in the previous section, but the server is able to observe the number of keywords attached to a message, and messages that contain identical keywords, even before any query. Furthermore, the server may learn the order of the keywords unless they are randomly permuted. For multi-keyword search, a server may learn which sets of messages match the keywords by performing an intersection analysis of the keywords.

Alternatively, an index table can be created independently from the message-keywords pairs. This approach reduces leakage compared to the above simple approach, in which the index table can be masked to prevent information leakage before query. The index is usually a dictionary in the form of a $(key, value)$ structure. A *key* consists of a message identifier that links to a message. The associated *value* consists of a list of elements linking together keywords that appear in the message. The general idea of a direct index is shown in Table V. For instance, given D_1 as the key, the index returns the keywords $w_2, w_5, \ldots, w_m$, which are keywords in message D_1.

As examples we discuss two schemes from Goh [2003] and Chang and Mitzenmacher [2005], respectively. In these schemes, the direct index is masked or encrypted so a server is not able to know the exact content of the index. For instance, in Chang and Mitzenmacher [2005]'s scheme, the message identifier D_1 is encrypted while the list of keywords is masked through binary representation and XORing.

Assume we have in total eight keywords $\{w_1, w_2, \ldots, w_8\}$ and a message with D_1 as its identifier. The message contains three keywords, w_2, w_3, w_4. Chang and Mitzenmacher [2005]'s scheme works by first taking this information and creates a masked index as shown in Table VI. From the table, $\mathrm{E}(k_1, D_1)$ denotes the encryption of D_1 using k_1. It represents the token to identify D_1 during query and k_1 is a secret key held securely by the user. Encrypting the identifier hides it from being known by the server. As for the keywords, to hide them from the server, a secret key r_i, $1 \leq i \leq 8$, is generated for every keyword w_i. Every r_i is used as a key and input for a PRF $G(.)$, to generate a pseudo-random string. In this case, the string is $G_{r_i}(1)$. The input to $G(.)$ is 1, since the associated message is D_1. Given that D_1 only contains w_2, w_3, and w_4 of the eight keywords, we have the first value entry as $0 \oplus G_{r_1}(1)$, since w_1 is not in D_1,

and the second entry is $1 \oplus G_{r_2}(1)$, since w_2 is in D_1, and so on. The pseudo-random string $G_{r_i}(1)$ that is XORed to the binary value 0 or 1 effectively masks the keyword reference for a message. Certainly, in this case, to search whether, say keyword w_2 is in D_1 the user submits $G_{r_2}(1)$ to the server. The server takes $G_{r_2}(1)$ and XOR it with entry $1 \oplus G_{r_2}(1)$. Since the output is "1," the server returns $\mathrm{E}(k_1, D_1)$ as the output.

For ease of reading, we omitted a step in Chang and Mitzenmacher [2005]'s scheme, where the original keyword sequence is permuted before XORing with the PRF $G(.)$. The purpose of it is to eliminate the order of the keywords in a message.

Goh [2003] used a different construction, where the index is constructed using Bloom Filters (BFs) [Bloom 1970]. A BF consists of an array of bits. Every message is assigned a BF. All entries in the array of a BF are first initialised to "0." Given a keyword of a message, a set of tokens is generated. A pseudo-random function uses these tokens to compute a set of indices. The array entries pointed to by the set of indices are set to "1." This step is repeated for all keywords of the message. To search for a keyword, a user sends a set of keyword tokens to the server. The server uses the tokens to compute indices for the array entries of every message's BF. A match is found if the array entries in the computed indices are all "1" for a message.

In summary, a direct index approach uses message identifiers as keys and the matching keywords as values. Both the identifiers and keywords are masked to preserve index privacy. The list of schemes using direct index that we studied can be observed from Table I and Table III.

Performance and Properties. We summarise the performance and properties of direct index schemes:

—*Main Feature:* A direct index in the form of *(key, value)*, where *key* is an element representing a message (e.g., an encrypted message identifier), and *value* is a list of elements representing the set of keywords matching a message.
—*Storage:* Schemes under this construction require storage of the metadata, for example the masked index (Table VI) or list of keyword tokens attached to an encrypted messages (Figure 6), in addition to storing the encrypted messages. As compared to a construction without index, this means additional space for storing the metadata.
—*Search:* Search time is linear to the total number of messages, since to search for messages containing a keyword, all messages must be scanned through. However, search efficiency is better than schemes without index that require searching through words in a message.
—*Characteristics (static or dynamic):* There are both static and dynamic schemes under this construction, but in general static schemes can be extended to the dynamic case straightforwardly. This is because adding a message can be realised by adding a new *(key, value)* entry directly to the index. Similarly, removal of a message involves searching and removing the *(key, value)* entry in the index.
—*Security (leakage):* Schemes of this type also leak keyword distributions and access patterns. Constructions using message-words approach (Figure 6) leak this information even before any query is performed. Depending on how the masked index is created, number of keywords may be revealed as well, as in Table VI. Nevertheless, comparing with construction without index that encrypts words of a message such as Song et al. [2000]'s scheme, at least locations of the keywords are not revealed.

3.3. Construction with Inverted Index

In this approach, an index in the form of *(key, value)* similar to schemes with direct index is used. However in this case, a *key* is a keyword and a *value* consists of a list of message identifiers associated to the keyword. Many SSE schemes adopt this

Table VII. A Hypothetical Example of an Inverted Index: Given a Set of n Messages D and m Keywords, w

Key	Value
w_1	$D_1, D_4, \ldots, D_{15}$
w_2	$D_1, D_3, \ldots, D_{22}$
$\vdots$	$\vdots$
w_m	$D_2, D_3, \ldots, D_n$

approach. Curtmola et al. [2006] introduced this approach, and subsequent schemes are proposed by Wang et al. [2010], Chase and Kamara [2010], Kamara et al. [2011], Kamara et al. [2012], Kurosawa and Ohtaki [2012], Mohamad and Poh [2012], Moataz and Shikfa [2013], Cash et al. [2014], Naveed et al. [2014], and Stefanov et al. [2014]. The list of existing inverted index schemes that we studied is presented in Table I and Table III. The main reason why inverted index is used widely is because sublinear search time can be readily achieved compared to the other two approaches. However, updating is not as straightforward. Table VII illustrates a hypothetical example of an inverted index. For instance, given w_1, the table returns $D_1, D_4, \ldots, D_{15}$, the messages that contain keyword w_1.

In the following, we describe two schemes in their simplified forms. We first discuss the second scheme of Curtmola et al. [2006] due to its simple structure and adaptive security. Then, we discuss the basic scheme Π_{bas} of Cash et al. [2014] that is structurally similar to Curtmola et al.'s second scheme but is scalable to large datasets.

Curtmola et al. [2006]'s index has the form as shown in Table VIII. It is constructed as follows. For every keyword w and its matching list of messages, the user uses a keyed PRP F to generate a *key* entry to the index for every matching message. As an example, assuming we have one keyword w_1 and this keyword appears in four messages (D_1, D_3, D_5, D_n). As illustrated in Table VIII, the first *key* entry for w_1 is $F_k(w_1, 1)$ for D_1, and the last entry is $F_k(w_1, 4)$ for D_n, where 1 to 4 are the counter values representing the four messages and k a secret key held securely by the user. In this sense, the first *key* value, $F_k(w_1, 1)$, is paired with the identifier of the first message containing w_1, $\mathtt{id}(D_1)$, and so on. Certainly a simpler way is to pair $F_k(w_1)$ to all four message identifiers $\mathtt{id}(D_1), \ldots, \mathtt{id}(D_n)$ directly without using counter values. However, such an approach reveals the total number of keywords and also in the case of Curtmola et al. [2006]'s scheme, makes it difficult to prove security against adaptive adversary [Curtmola et al. 2006, 4.2]. To search the index for w_1, the user generates a list of *key* entries $(F_k(w_1, 1), \ldots, F_k(w_1, \mathrm{MAX}))$, where MAX denotes the user-defined maximum number of messages that match a keyword. Using this list, a server searches the index by iterating $F_k(w_1, i)$ for $i = 1$ to MAX to retrieve the matching message $\mathtt{id}$s.

The Π_{bas} scheme of Cash et al. [2014] also uses a similar inverted index structure of Curtmola et al. [2006]'s scheme. Table IX shows an example index, assuming the same hypothetical data as above with one keyword w_1 and four messages. A keyed PRF F is used to generate two keys for w_1, here denoted as $k_1||k_2 \leftarrow F_K(w_1)$, where K is a secret key chosen and stored securely by the user. As illustrated in Table IX, key entries (or tokens) based on counter values, $F_{k_1}(0), \ldots, F_{k_1}(3)$ are generated to link to the four message identifiers. The message identifiers $\mathtt{id}(D_1), \ldots, \mathtt{id}(D_n)$, on the other hand, are encrypted using k_2. To search for the keyword w_1, a user generates $k_1||k_2 \leftarrow F(K, w_1)$ and sends $k_1||k_2$ to the server. Since the tokens are generated based on increasing counter values starting from 0, it is straightforward for the server to generate these tokens using k_1 and the PRF F as $F_{k_1}(0)$ to $F_{k_1}(3)$. So, for example, given the token

Table VIII. An Example of an Index in Curtmola et al. [2006] for One Keyword w_1, Assuming Four Messages D_1, D_3, D_5, D_n Contain w_1

key	*value* (id of D)
$F_k(w_1, 1)$	$\texttt{id}(D_1)$
$F_k(w_1, 2)$	$\texttt{id}(D_3)$
$F_k(w_1, 3)$	$\texttt{id}(D_5)$
$F_k(w_1, 4)$	$\texttt{id}(D_n)$

Table IX. An Example of Π_{bas}'s index [Cash et al. 2014] for One Keyword w_1, Assuming Four Messages D_1, D_3, D_5, D_n Contain w_1, Where $k_1 || k_2 \leftarrow F_K(w_1)$

l	d
$F_{k_1}(0)$	$\texttt{E}(k_2, \texttt{id}(D_1))$
$F_{k_1}(1)$	$\texttt{E}(k_2, \texttt{id}(D_3))$
$F_{k_1}(2)$	$\texttt{E}(k_2, \texttt{id}(D_5))$
$F_{k_1}(3)$	$\texttt{E}(k_2, \texttt{id}(D_n))$

$F_{k_1}(0)$, the index returns $\texttt{E}(k_2, \texttt{id}(D_1))$. The server then uses k_2 to decrypt and retrieve the identifier $\texttt{id}(D_1)$. Similar steps are carried out for subsequent tokens until there is no more matching tokens. Note that keywords and message identifiers in the index are masked to preserve index privacy. This is done in the Π_{bas} scheme by using a PRF to mask the keywords and a symmetric encryption scheme to mask $\texttt{id}(D)$. In the case of Curtmola et al. [2006], $\texttt{id}(D)$ can be a random string.

Performance and Properties. In the following, we state the performance and properties of the approach:

—*Main Feature:* An inverted index in the form of *(key, value)*, where *key* is an element representing a keyword w, such as an encrypted keyword (or keyed hash value), and *value* is a list of elements representing the set of messages that contain w.
—*Storage:* Schemes under this construction require similar storage space as schemes with direct index, since encrypted messages and masked index are stored. Different schemes may have different storage efficiency due to how an index is designed.
—*Search:* Search time is sublinear (and, in fact, optimal in many cases) compared to linear search time of the two previously discussed constructs. This is because searching for a keyword immediately returns the list of message identifiers matching the keyword. Thus, keyword matching efficiency is equivalent to the $O(1)$ efficiency of a hash table (dictionary), and $O(r)$ on retrieval of the r matching messages or message identifiers. This is the main advantage of schemes under this approach.
—*Characteristics (static or dynamic):* Both static and dynamic schemes have been proposed. However, creating a dynamic scheme is generally not as straightforward compared to the previous two approaches. If one is to add/remove a message through the index, then each of the keyword tokens must be linearly scanned through to add/remove a message entry. This is the main disadvantage of this approach.
—*Security (leakage):* Schemes of this type have similar leakage profile with constructions using direct index, in which they also leak keyword distributions and access patterns. Depending on the underlying design of the index, certain schemes leak total number of keywords, while certain schemes do not. For example, the Π_{bas} scheme of Cash et al. (Table IX) reveals only the total number of index entries, concealing the number of keywords.

3.4. Tree-Based Construction

While the actual implementations of direct and inverted index may be based on a tree structure, there are SSE schemes that are designed specifically using trees (e.g., binary search tree). Example schemes include Kamara and Papamanthou [2013], Xia et al. [2015], and Demertzis et al. [2016]. Table I and Table III state tree-based schemes that we surveyed. The general idea of a tree-based index is to store the keyword tokens

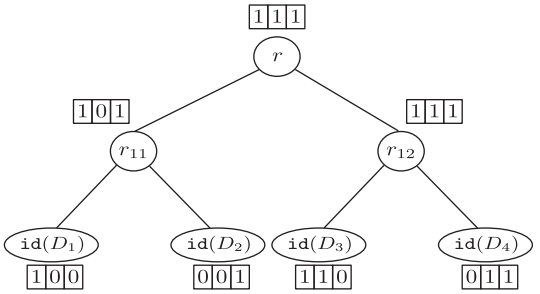


Fig. 7. A conceptual balanced Binary Search Tree (BST) in Kamara and Papamanthou [2013] with four messages D_1, D_2, D_3, D_4, and three keywords w_1, w_2, w_3. The vector of each node contains three elements, where 1 in the element at index i for the vector in the leaf nodes means keyword w_i is in the message. For example, keyword w_1 is in message D_1.

or message identifiers in the leaf nodes. Then, the internal nodes (including the root node) are generated based on the messages and matching keywords linked by these leaf nodes.

In the following, we describe Kamara and Papamanthou [2013]'s tree-based scheme in more detail as an example. Subsequent ranked-based schemes (e.g., Xia et al. [2015]) uses a similar tree structure. Every node in Kamara and Papamanthou [2013]'s tree maintains vector(s) of size equal to the total number of keywords. As an example, Figure 7 shows a tree-based index for a setting with four messages (D_1, D_2, D_3, D_4) and three keywords (w_1, w_2, w_3). The tree-based index is built in such a way that leaf nodes contain the message identifiers. For every node, a vector with three elements representing the three keywords is created. For instance, the leftmost leaf node contains the identifier of message D_1, $\mathtt{id}(D_1)$, and a vector with elements "100." The elements in the vector dictate whether a keyword appears in the message. Since the first element is "1" in the vector for $\mathtt{id}(D_1)$, keyword w_1 is in message D_1. Similarly, w_2 and w_3 are not keywords of D_1, since the second and third elements in its vector are "0." The vector elements of the parent of the leaf nodes are generated using bitwise Boolean OR operation on the two vectors. For instance, the vector of r_{11} for D_1 and D_2 leaf nodes is $100 \vee 001 = 101$.

To search for matching messages of a keyword w_i, the tree is traversed to check whether an element in the vector at position i is 1 or not. For instance, to search w_1, we first check whether the first element in the root's vector is 1. In our example, this is the case, and so we move on to the child node r_{11}. Similarly, the first element is 1 for the vector of r_{11}, and thus, we continue to the leaf node, and, since the first element is 1 as well for the vector of D_1, we say that w_1 matches D_1.

The precise tree index of Kamara and Papamanthou [2013]'s scheme before masking takes the form shown in Figure 8. The main reason that two vectors were introduced instead of one (as in Figure 7) is for the purpose of randomising the entry of 0 and 1, mimicking coin flipping for proving privacy of the vector. In the actual construct, every entry in both the vectors is masked, by either encrypting 0 or 1, or filling the non-chosen entries (grey boxes in the figure) with an equal length random string.

We may observe that such a tree-index based SSE uses similar concept of a direct index. For instance, the leaf node D_1 having a vector containing 100, which denotes w_1 appears in D_1. The main difference is due to the binary search tree structure that the search time now is logarithmic instead of linear.

Performance and Properties. The general performance and properties of a tree-based construct are as follows:

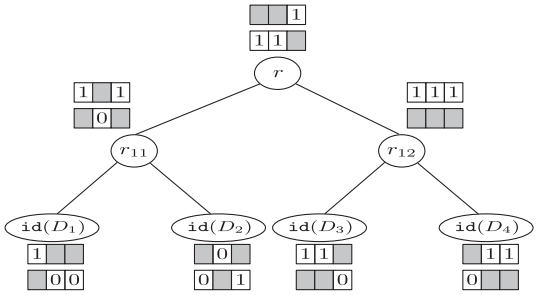


Fig. 8. A BST with two vectors for every node [Kamara and Papamanthou 2013].

Table X. Search Performance and Common Properties of the Four Approaches (Schemes Under These Approaches Can Be Observed from Table I and Table III)

Approach	Search performance	Direct update (add/delete messages)
Without Index	Linear $O(n)$, where n is the number of messages.	*Yes.* Message can be uploaded directly, and can be deleted directly from the server by querying the message identifier.
Direct Index	Linear $O(n)$ (but does not need to scan through blocks of messages).	*Yes.* Similar to approach without index.
Inverted Index	Sublinear $O(r)$, where r is the number of messages matching the keyword.	*No.* Addition/deletion of a message requires searching through every keyword (*key*) entry to insert/delete the mesage identifier stored in *value* in the *(key, value)* index.
Tree	Sublinear $O(\log n)$.	Update is based on the add/delete algorithms of the underlying search tree.

—*Main Feature:* Keywords or message identifiers are represented as leaf nodes of a tree. Internal nodes and root are constructed in a way that there are paths linking a query keyword to the leaf nodes containing the matching identifiers.
—*Storage:* Storage space requirement is similar (asymptotically) to the direct and inverted index constructs, in which in addition to the encrypted messages, metadata need to be stored. In this case, tree structure(s) instead of index table(s).
—*Search:* Search time is logarithmic due to the tree structure. It is equivalent to the efficiency of a binary search tree, $O(\log n)$, if such a tree structure is used. n denotes the number of messages.
—*Characteristics (static or dynamic):* Schemes under this construction can either be static or readily extended to a dynamic scheme based on the standard update operations provided by the underlying tree data structure (e.g., BST, B-tree).
—*Security (leakage):* The leakage profile is similar to schemes with direct and inverted index but may additionally leak the path information from root to the leaf nodes. Depending on the underlying constructions, such leakage may or may not reveal any information more than the existing leakages that will allow a server to learn the keywords or content of the messages. The construction in Figure 8, for instance, has the same leakage profile of the two previous constructions although the query paths are revealed. In contrast, as an example, a semantic keyword search construction by Fu et al. [2014b], where the tree is used to build the stem set of a keyword, the paths reveal the distribution of characters in the keywords.

Table X summarises the search performance and updatability of the four approaches.

4. SECURITY MODELS

We next discuss security of SSE schemes based on (1) the assumptions on the servers and the users, (2) the security properties, (3) the two main models, and (4) the allowable leakages and its attack models.

4.1. Assumptions

The assumption of trust on the user and the server plays an important role in determining SSE security requirements and models:

—*Honest-but-curious server (passive adversary).* In this setting, the server is able to access and to try to learn information from the stored data and metadata but not modify them. The server is known as honest-but-curious. Most of the schemes are built under this setting.
—*Malicious server (active adversary).* In this setting, the server is able to (intentionally) modify data stored by the user. This is known as a malicious server. Verifiable schemes, for example, in Kamara et al. [2011] and Kurosawa and Ohtaki [2012], are constructed based on the malicious server assumption. Other verifiable schemes that we studied can be observed in Table II and Table IV via the (ver) column.
—*Trusted Data Owner.* For single user schemes, the user (or data owner) is assumed to be fully trusted. We note, however, that Dai et al. [2016] recently discussed the potential of a user's device being compromised and keys being retrieved by attackers. This represents a new assumption on the data owner but may be addressed independently through mechanisms that provide device security. An example is the proposal by Dai et al. to use physical unclonable function (PUFs).
—*Malicious Users.* In a multi-user scenario, other users that are allowed to perform queries on the encrypted datasets may be malicious. They may collude among themselves to gain information that they are not authorised to learn, for instance, through running queries not authorised by the data owner. Some schemes that discussed in details on this issue and provide solutions are Jarecki et al. [2013], Fisch et al. [2015], and Ishai et al. [2015]. It is also possible that the data owner is malicious and may try to learn information based on the queries submitted by other users as discussed in Ishai et al. [2015].

4.2. Security Properties

Based on the discussions and definitions in Song et al. [2000], Curtmola et al. [2006], Chase and Kamara [2010], Kurosawa and Ohtaki [2012], and Stefanov et al. [2014], the security properties of a SSE scheme are summarised as follows:

—A server cannot learn anything about the messages given the corresponding ciphertexts. *This is provided through encrypting the messages or blocks of messages using a symmetric encryption scheme.*
—A server cannot learn anything more of the plaintext messages except for the search results. This is one type of information leakage that occurs in SSE and is known as *access patterns*.
—A server cannot search for an arbitrary keyword without authorisation from the user. *This is provided through a unique secret key owned by the user for the purpose of generating keyword tokens using a PRF (or PRP). Without this key, a server will not be able to generate a valid token.*
—A server cannot learn the secret query keyword from the query token provided by the user. *Similarly to the above, this is provided through generating the token using a secret key.* However, a server has the tokens, which in most cases are deterministic,

and hence the server knows when a query is repeated. This is also one type of information leakage, known as *query patterns*.
—A server knows the number of keyword-message pairs of the stored data. This is another type of information leakage, known as the *size patterns* (introduced by Stefanov et al. [2014]). Some of the SSE schemes may also leak the total number of messages and/or the total number of keywords.
—If a server is malicious, in addition to the above requirements, then the server should not be able to forge the stored encrypted data and the associated metadata. This is known as *reliability* (introduced by Kurosawa and Ohtaki [2012]).
—In a multi-user scenario, a malicious user should not be able to query using an arbitrary keyword without authorisation from the data owner. A malicious owner should not be able to learn extra information from the queries submitted by a user.

In brief, the security intuition of SSE schemes is that nothing should be leaked except for the outcome and the pattern of a sequence of queries [Curtmola et al. 2006]. The pattern is modeled by given the leakage in terms of *size patterns*, *query patterns*, and *access patterns* as stated above. A SSE scheme is said to be secure assuming the leakage patterns are allowable and provided that the underlying primitives (such as PRFs, PRPs) are secure.

Stefanov et al. [2014] also propose the notion of *forward* and *backward privacy* for update operations in dynamic schemes. A dynamic scheme is said to achieve forward privacy if a server does not learn that a newly added message contains a keyword that has been searched previously. Backward privacy means that queries cannot be performed on deleted messages. Recent schemes that provide forward privacy are in Stefanov et al. [2014], Bost et al. [2016], and Demertzis et al. [2016], and as correctly observed by one of the reviewers, forward privacy is among the most important features in dynamic scheme. Next, we discuss the two main security formalisation of SSE.

4.3. Real-Ideal Simulation Paradigm

This was first proposed for SSE by Chang and Mitzenmacher [2005] and improved by Curtmola et al. [2006] for adaptive queries. It has since been used as the main security model for many subsequent SSE schemes as can be observed from Table II and Table IV (with reference to S1 and S2 under the security model column). Intuitively, an adversary is asked to play a game to guess whether he or she is playing the **Real** game, where the actual SSE scheme is being executed, or is playing the **Ideal** game, where all the data are simulated based on the leakage patterns. The scheme is said to be secure under allowable leakage and the assumption that the underlying primitives are secure, if and only if the adversary cannot distinguish between the two games.

4.4. UC-Secure

The universal composable (UC) model for SSE was introduced by Kurosawa and Ohtaki [2012] and also used more recently by Naveed et al. [2014], Taketani and Ogata [2015], and Ishai et al. [2015]. The general idea is to formalise an ideal functionality $\mathcal{F}_{SSE}$ of the algorithms required in the scheme. The leakage of every function is also captured under ideal functionality. Security of the scheme is then proven by demonstrating that the SSE construction realises $\mathcal{F}_{SSE}$ against the adversary.

We remark that the security properties and models do not take into consideration the background knowledge of the keywords and the underlying messages an adversary might have. This leads to the inference attack in Islam et al. [2012] and subsequently more powerful attacks based on improvements on this attack. We discuss these attacks in more detail next.

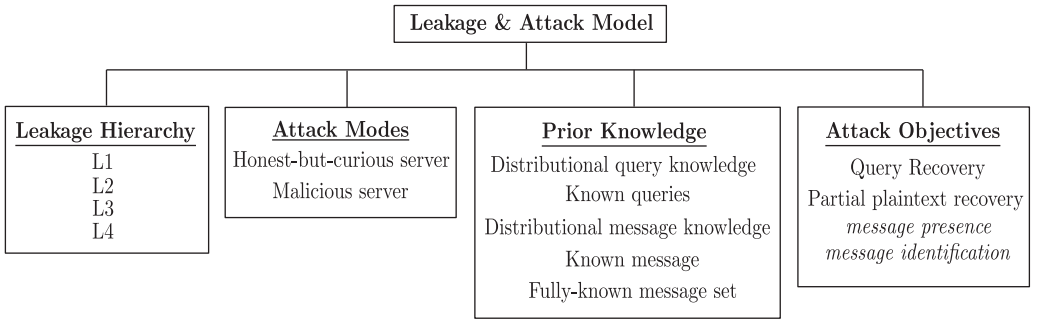


Fig. 9. A leakage and attack model in Cash et al. [2015].

Table XI. Leakage Hierarchy [Cash et al. 2015]

Level	Description
L4	*Full plaintext under deterministic word-substitution cipher*
L3	*Fully revealed occurrence pattern with keyword order*
L2	*Fully revealed occurrence pattern*
L1	*Query-revealed occurrence pattern*

4.5. Leakages and Attacks

Leakages are defined based on access and query patterns in most SSE schemes. These leakages are allowable under either the real-ideal, UC, or other specific security models. No actual analysis was performed on potential attacks based on these leakages until the inference attack conducted by Islam et al. [2012]. They provided empirical analysis on access patterns and proposed possible mitigation through injecting random noise.

The strength of the Islam et al. [2012]'s attack is that its success rate is independent of the number of queries. However, a main weakness is that it is not scalable as was experimented by Cash et al. [2015]. For a set of keywords larger than 2,500, the attack performs poorly. Furthermore, Islam et al. [2012] claimed that their attack only requires the server to know the message distributions based on keyword co-occurrence probabilities. Yet as was observed by Cash et al. [2015], in order for the attack to be successful, a server needs to know the content of all the original messages.

Cash et al. [2015] then suggested more effective leakage-abuse attacks. One of the attacks is known as count attack, which is used for query recovery. It is simpler and does not require numerical optimisation as in Islam et al. [2012]'s mechanism. The assumptions are that server knows the co-occurrence patterns of keywords and also knows the number of messages in the index that match every keyword.

A comprehensive attack model was also introduced by Cash et al. [2015]. It consists of a *leakage hierarchy*, *attack modes*, *prior knowledge*, and *attack objectives*. Figure 9 shows the category of Cash et al. [2015]'s model. This introduces a systematic way in defining and analysing attacks based on leakages. We discuss them in more detail in the following.

Leakage Hierarchy. Table XI states the four levels of leakage defined by Cash et al. [2015]. L4 defines the worst leakage, and L1 defines the least leakage. In L4, a server learns the location patterns of a keyword in the message. It also learns the occurrence frequency of the keyword. As stated by Cash et al. [2015], SSE schemes in Skyhigh Networks [2016] and CipherCloud [2016] have L4 leakage. For L3, the occurrence frequency of the keyword is not leaked, but keyword occurrence patterns are still revealed. The order of these keywords is also leaked. L2 is similar to L3 except now

keyword order is hidden. This can be realised through permuting the keywords. The word-based approach shown in Figure 6 has L3 or L2 leakage depending on whether keywords are permuted or not. L1 is similar to L2 except that the leakage only occurs for keywords that have been queried. In summary, most existing schemes follow Curtmola et al. [2006]'s model and achieve L1 leakage.

Attack modes. These refer to the assumptions on the servers, where the modes consist of either a honest-but-curious or a malicious server. We have previously discussed these assumptions in Section 4.2. However, in addition to modification of messages, Cash et al. [2015]'s malicious server can launch a chosen-message or chosen-query attack. This means an attacker can ask a user to include messages or queries selected by the attacker. Chosen-message attack has been shown to be effective by Cash et al. [2015] and Zhang et al. [2016a].

Prior knowledge. This refers to the background knowledge on the keywords and messages that an attacker possesses. It is the main ingredient (or assumption) for the attacks to be successful, as was demonstrated in Islam et al. [2012], Cash et al. [2015], and Zhang et al. [2016a]. These are further divided into five categories:

—*Distributional query knowledge.* It means the adversary has some knowledge about the queries being issued.
—*Known queries.* It means the adversary knows some actual keywords the user searches for.
—*Distributional message knowledge.* It means the adversary knows the underlying area (or domain) of the messages (e.g., health data, emails).
—*Known messages.* It means the adversary knows some actual messages (plaintexts) or significant information of the messages.
—*Fully known message set.* It means the adversary knows all the actual messages (plaintexts) and some of the actual keywords. As was stated by Cash et al. [2015], Islam et al. [2012]'s attack is based on this prior knowledge.

Attack objectives. These are the goals an attacker wishes to achieve. The two main goals are query recovery (QR) and partial plaintext recovery (PR). As the name suggests, QR means an attacker aims to learn the content of a query. PR means the attacker aims to recover as much plaintext messages as possible through analysing the leakage patterns. Two other related goals are *message presence* and *message identification*, where an attacker wishes to know whether a message is part of the set of messages or to identify the identifier that matches a known message.

The above model were used by Cash et al. [2015] to simulate attacks on publicly available datasets. They concluded that schemes allowing L2 and L3 leakages are risky to be deployed if chosen-message attack (or insertion of messages) is possible. They also concluded that L1 leakage is dangerous if we try to protect queries on known sets of messages. Zhang et al. [2016a] also examined attacks based on insertion of messages. They conclude that future research should reduce or eliminate leakage of access patterns. They further suggested that this might require exploring new approaches such as using interactive protocols or multiple servers.

Kellaris et al. [2016], on the other hand, define two sources of leakage, *access patterns* and *communication volume*, in which *communication volume* means the server learns the number of returned items of a query. They showed that any SSE scheme will leak access patterns. Based on this, they introduce reconstruction attacks for schemes with range queries. Access patterns are used on the SSE scheme, and communication volume is used on systems based on ORAM and fully homomorphic encryption. Furthermore, in contrast to attack assumptions in Islam et al. [2012], Cash et al. [2015], Zhang et al.

Table XII. Performance of Some of the SSE Schemes

Approach	Scheme	Index Size	Search Time
Without Index	[Song et al. 2000]	n/a	$O(n)$
Direct Index	[Goh 2003]	$O(ny)$	$O(n)$
	[Chang and Mitzenmacher 2005]	$O(mn)$	$O(n)$
	[Moataz and Shikfa 2013]	$O(N)$	$O(n)$
Inverted Index	[Curtmola et al. 2006]	$O(Mn)$	$O(r)$
	[Chase and Kamara 2010]	$O(Mn)$	$O(r)$
	[Kamara et al. 2012]	$O(N+m)$	$O(r)$
	[Kurosawa and Ohtaki 2012]	$O(Mn)$	$O(r)$
	[Cash et al. 2013]	$O(N)$	$O(r)$
	[Cash et al. 2014], Π_{bas}	$O(N)$	$O(r/p)$
	[Naveed et al. 2014]	$O(N)$	$O(r)$
Tree	[Kamara and Papamanthou 2013]	$O(mn)$	$O((r\log n)/p)$
	[Stefanov et al. 2014]	$O(N)$	$O(min\{\alpha+\log N, r\log^3 N\})$
	[Xia et al. 2015]	$O(mn)$	$O((r\log n)/p)$

n is the number of messages. m is the number of keywords. y is the size of a bloom filter.
M is the maximum number of identifiers matching a keyword. This means each entry to an inverted index has the same number of identifiers. For those that have less than M, padding applies. N is the total number of identifiers (matching all the keywords) in the index.
r is the number of messages matching the query keyword. p denotes the number of processors for schemes that are parallelizable.
α is the number of times the query keyword was historically added to the message-keyword pairs.

[2016a], and Kellaris et al. [2016]'s attacks can be executed without prior knowledge about the message, without knowledge of the submitted queries and their results.

5. PERFORMANCE

The performance of SSE schemes is normally measured based on the index size and search time. Both rely on the underlying general constructions (i.e., the four approaches that we discussed in Section 3).

5.1. Setup (Index Size)

The setup operation of an SSE scheme involves the encryption of messages and, in most of the schemes, the creation of a masked index. Since this operation can be performed off-line, the main concern is on the size of the index instead of the computation overhead in generating an index and encrypting the messages. As noted in the implementation by Cash et al. [2013] using readily available large datasets from Enron emails for 1.2 million keywords and ClubWeb09 datasets of 0.4TB HTML files, the size of the main index can reach 12.4GB and 144.4GB, respectively. Therefore, efficient index construct can save cost in terms of storage space.

5.2. Search (Search Time)

This is the main computational performance metric, since the efficiency of searching directly affect the practicality of a SSE scheme. Similarly to the measurement of index size, search efficiency depends on the underlying approach. As was previously summarised in Table X, SSE schemes without index and with direct index require linear time $O(n)$ in proportion to the number of messages n, while schemes with inverted index normally require sublinear time $O(r)$, where r denotes the number of messages matching the query keyword. Schemes with tree-based index, on the other hand, rely on the search efficiency of the underlying tree search operation, which is $O(\log n)$.

In Table XII, we compile performance measures on some of the schemes that we study. The schemes are categorised by the approach (e.g., direct index or inverted

index) taken by each of the scheme. We may observe that search performance largely reflects the general efficiency trend of an approach discussed in Table X.

5.3. Locality

Cash et al. [2014] first discuss the issue of *locality* in SSE schemes. It was formalised later by Cash and Tessaro [2014], where *locality* is defined as *the number of non-contigous memory accesses made by the server*. Recently, Asharov et al. [2016] provides a scheme that follows the formalisation by Cash and Tessaro [2014]. The issue on *locality* relates to the inefficiency of a theoretically efficient scheme during implementation, especially for very large datasets. The main reason behind the inefficiency is because of practical factors, such as read-write access latency, storage utilisation, and dataset distributions, which are generally not considered in theoretical analysis.

In more detail, the asymptotic search performance of an inverted index construct is known to be optimal ($O(r)$) as observed from Table X and Table XII. A query based on a keyword directly returns the r matching message identifiers or encrypted messages. However, as discussed in Cash et al. [2014] and Cash and Tessaro [2014], access latency of the r identifiers was not factored in. In practice, these identifiers are stored in pseudorandom locations on the storage hardware. This means retrieval of each identifier requires a read access. In contrast, in a plaintext search index, these identifiers are stored in contiguous sectors that can be read together.

As an example on how locality can be improved, we briefly describe the mechanism proposed in Cash et al. [2014]. Its idea is to pack (encrypt) several identifiers in the inverted index into one ciphertext. This is so the number of read access can be reduced. This idea is further extended to take into account the variance in the number of matching identifiers for each keyword-identifiers pair. For example, a keyword w_i appears in five messages and, hence, is matched to five identifiers, whereas another keyword w_j appears only in one message and, hence, is matched to one identifier. In practice, this difference can be large. To address this variance in size, the datasets are categorised into *small*, *medium*, and *large*. *Small* keyword-identifiers pairs that are packed in blocks are stored as usual in the index. For *medium* pairs, only pointers to the identifiers are packed and stored in the index, while the identifiers are stored in external memory. For *large* pair, there are two levels of indirection. This means the pointers in the index point to sets of pointers in the external memory, which then point to the actual identifiers. The reason for this categorisation is to ensure that the index only store the minimal number of blocks so the number of read access to pseudorandom locations can be reduced.

6. SETUPS, QUERY FUNCTIONALITIES AND CHARACTERISTICS

In addition to the main functionality of SSE, which is to search on encrypted data in a private manner, schemes with improved characteristics and query types have been proposed over time. In the following, we discuss the query functionalities and main characteristics listed in the general framework (Figure 4), where schemes with different properties can be observed from Tables I, II, III, and IV.

6.1. Single User or Multi-user

In certain applications, a data owner may allow other users to search on the encrypted messages that are outsourced. Two example schemes that provide this service are by Curtmola et al. [2006] and Jarecki et al. [2013]. These multi-user schemes can be seen as a generalisation of a single user scheme. It can be configured to be a single user scheme by simply setting the number of user to one. The main challenge in designing a multi-user scheme is how to ensure query and message confidentiality (and integrity) from malicious data owner and/or users. This is because, in this environment, users

may query keywords that they are not authorised to query, or a data owner may try to learn information from the queries submitted by the users. Normally, queries by users in these schemes require search authorisation by the data owner.

Some single-user schemes, such as in Li et al. [2009], Wang et al. [2010], Cao et al. [2011], Sun et al. [2013], Cao et al. [2014], Dai et al. [2016], and Orencik et al. [2016], use a multi-user architecture. These schemes describe how multiple users can be incorporated into their single-user construct.

We categorised the SSE schemes that we studied under these two settings. This can be observed under the (Setups) column of Table I and Table III. In summary, single user schemes are obviously simpler and more efficient compared to multi-user schemes but their practical usage may be limited to individual outsourcing of encrypted data.

6.2. Single Server or Multi-server

Most of the existing schemes are based on single server, but multi-server schemes have received increased interest recently. This is mainly due to their potential capability to minimise leakage. We broadly classify multi-server schemes into three types. The first type consists of schemes where encrypted data and indexing are managed separately by two different servers. Example schemes are Bösch et al. [2014], Ishai et al. [2015], and Orencik et al. [2016]. The second type of schemes provide mechanisms to divide and distribute index to many commodity hardware in a cloud storage server. Strictly speaking, if we consider a cloud storage provider as a single server (which is the case for most schemes), then the scheme is a single-server scheme. However, the index distribution mechanism may be modified to distribute the index efficiently to multiple servers. An example scheme is Kuzu et al. [2015]. The final type of schemes define an extension of a single server to multiple server schemes, which is introduced in Poh et al. [2016]. In such a case, the set of messages can be divided into many subsets and distributed among the servers. Every server holds only a subset of the messages. One potential application is that messages with different level of security clearance can be stored separately.

In summary, multi-server schemes may potentially reduce leakage that is common in most single server schemes. This is important, especially after the recent effective attacks proposed by Islam et al. [2012], Cash et al. [2015], and Zhang et al. [2016a]. However, multi-server construct induces additional performance and cost overheads. This is because extra servers must be subscribed and additional bandwidth is required for the communications between the servers and the user.

6.3. Single-Keyword or Multi-Keyword

SSE schemes with a single-keyword search allow only a single keyword to be queried at a single instance. Some example schemes include those in Goh [2003], Chang and Mitzenmacher [2005], Curtmola et al. [2006], Chase and Kamara [2010], Kurosawa and Ohtaki [2012], and Stefanov et al. [2014]. In practice, many applications may require multi-keyword search (e.g., Boolean, conjunctive, disjunctive, rank and range). Although a single-keyword scheme can be deployed for multi-keyword search, it may result in inefficiency and/or additional leakages. A simple approach to extend a single-keyword scheme to a multi-keyword one would be to send a list of keywords to the server and require the server to send back all results for the whole list of keywords. The user then finds the intersection for all these keywords from the returned results. This gives a scheme with leakage profile identical to the underlying single-keyword scheme but is inefficient and is computationally intensive for the user. Another way is to ask the server to find the intersection on the outputs and return the final results to the user. This leaks the intersection patterns to the server for all the keywords, which might not be desirable.

Many recent proposals provide multi-keyword search and focus on reducing the inefficiency and the leakage of intersection patterns of the two simple approaches described above. The index structure of a single-keyword construct is used in many such schemes but with extension or modification. These schemes can be considered as a natural extension or generalisation of a single-keyword scheme. Some example constructs are in Cash et al. [2013], Cao et al. [2014], Xia et al. [2015], and Demertzis et al. [2016]. To provide an efficient multi-keyword search comparable to a single-keyword search while minimising leakage of information, for instance, Cash et al. [2013] propose conjunctive search where the least-frequent keyword in the conjunction is first queried. Then the remaining keywords in the conjunction are checked with the returned message identifiers. The message identifiers that match with all the remaining keywords are returned as the result. The scheme improves efficiency by only searching the set of identifiers matching the least frequent keyword for the remaining keywords in the conjunctions.

Nevertheless, as long as a server is tasked to search and return results from a set of keywords (e.g., conjunctive, Boolean), additional leakage of information, such as intersection patterns in various forms (e.g., full intersection patterns as in the simple approach or the least-frequent keyword coupled with other keywords in the conjunction), could occur. Hence the following question: *Is it possible for an efficient multi-keyword scheme to achieve leakage profile identical to that of a single-keyword scheme with minimal leakage (leaking only single-keyword search and access patterns)?*

Other single-keyword and multi-keyword schemes that we have compiled can be observed under the (Query Functionalities) column in Table I and Table III. Practically, a multi-keyword search scheme is preferable for ease of use and retrieval of more accurate results. However, care must be taken in designing these schemes so provision of multi-keyword functionality does not degrade security assurance.

6.4. Static or Dynamic

In general, schemes without index such as Song et al. [2000] or with direct index such as Goh [2003] and Chang and Mitzenmacher [2005] are inherently dynamic and allow for straightforward message addition and deletion. In contrast, static schemes that used an inverted index, for example, in Curtmola et al. [2006], Chase and Kamara [2010], and Kurosawa and Ohtaki [2012], demonstrated sublinear search for SSE but are not able to provide efficient updates. To add (or delete) a new message, an entire index must be re-constructed. This means a static scheme, either with direct or inverted index, can be readily deployed as a dynamic one, where messages can be added or removed. However, such a straightforward adoption leaks substantial information on the messages (in addition to the inefficiency related to schemes with inverted index).

This is because addition and removal of messages leak information of the newly added or deleted messages and the keyword tokens attached to them. Furthermore, if there is a new keyword in the newly added message, this can be observed by the server as well. This is because for direct index, adding a new message inserts a new (message, list of keywords) index entry in the index, which is easily observable by the server. Also, a new keyword token in a newly added message is detectable when this keyword is queried. If the newly added keyword token is deterministic (e.g., as a hash value, or an encrypted item using similar key), then this is also observable as it is distinct from all other tokens. The inverted index has similar issues, since inserting a (keyword, list of messages) index entry requires the server to scan through the index to append the list of messages to an existing keyword entry (if the keyword already exists in the index) or insert a new index entry if it is a new keyword.

Thus, to allow for efficient updates and to potentially avoid the above leakages, dynamic schemes such as in Kamara et al. [2012], Ogata et al. [2013], Kamara and Papamanthou [2013], Cash et al. [2014], Stefanov et al. [2014], and Kuzu et al. [2015]

used inverted indexes that cater to dynamic usage (e.g., a static index and an update index), trees, and ORAM. In other words, dynamic schemes leak more information than static schemes, since more information is provided to the server to add or remove messages. As examples, as stated by Stefanov et al. [2014], the schemes by Kamara et al. [2012] and Kamara and Papamanthou [2013] do not achieve forward and backward privacy (Section 4.2), while Stefanov et al. [2014] do not provide backward privacy, whereas, for static schemes, forward and backward privacy is not an issue.

More static and dynamic schemes can be observed from the (sta) and (dyn) columns in Table II and Table IV. In practice, dynamic schemes are preferred, since it is common for new messages to be continuously outsourced or existing messages to be removed from storage. The main tradeoff is that dynamic schemes generally leak more information compared to static schemes, as discussed above.

6.5. Verifiability

This refers to the ability to verify integrity of stored data (and index) under the malicious server assumption (Section 4.1). Its main aim is to enable a user to detect modification by the server on the store data/index. Nevertheless, it generally does not take into account message insertion (or chosen-message) attack. This is because message insertion usually happens prior to the construction of the index and encryption of the message. In fact, the existing real-ideal security paradigm allows a server to generate the set of messages, and the leakage patterns from these SSE-encrypted index and messages are allowed. Schemes with verifiability are noted under the (ver) column in Table II and Table IV. In summary, verifiability is introduced in the SSE scheme to partially address the issue of malicious server.

6.6. Anonymity

Anonymity relates to a multi-user setting, in which each user is allowed to query other users' data. In such a case, a user may wish to search without revealing his or her real identity. This may be applicable for organisations such as intelligence agencies or police or financial institutions that query each others' databases. As far as we know, Raykova et al. [2009]'s scheme (noted under the (an) column in Table II) is the only scheme with such a feature.

6.7. Parallelization

Parallelizable schemes were proposed to take advantage of multi-core processors that are now common in many computing platforms. It helps in improving performance, especially during the query phase. Instead of the need to search sequentially through a SSE-encrypted message or index, this process is parallelized. The list of schemes can be observed under the (pa) column in Table II and Table IV.

6.8. Deterministic or Probabilistic Tokens

Deterministic keyword tokens leak at least the number of times a specific keyword is queried. Probabilistic tokens address this concern by generating a distinct token every time an identical keyword is provided. The difficulty in designing this type of scheme lies in the fact that there is no suitable existing symmetric primitives that can provide such a feature. Hence, there has yet to be many SSE proposals for probabilistic tokens. The scheme proposed by Moataz and Shikfa [2013] uses the Gram-Schmidt process together with PRFs but results in linear search time, as compared to sublinear search time for most recent schemes with deterministic tokens. In summary, for better security assurance, one would design a scheme that generates probabilistic tokens. However, this might not be achievable using purely symmetric primitives.

Remarks. There will always be tradeoff when deciding the properties to be included in designing an SSE scheme. The better the scheme in terms of setup and characteristics (e.g., multi-user, dynamic, probabilistic tokens), and query functionalities (multi-keywords), the higher the cost on performance and leakages.

7. CHALLENGES AND RESEARCH DIRECTIONS

In this section, we outline the potential challenges and research directions for SSE.

7.1. Combining Different Primitives

Many SSE schemes deployed only symmetric primitives (i.e., PRF, PRP, symmetric encryption schemes). These schemes may be limited in their characteristics and functionalities as compared to schemes that combine tools such as Diffie-Hellman blinding and OPE. This can be observed from Table II and Table IV. Normally, a scheme combines other tools to achieve more properties such as conjunctive search, ranked search, and less leakage. For example, Cash et al. [2013] use the Diffie-Hellman exchange in a group of prime order to mask the keyword for multi-keywords (conjunctive) query. As another example, Xia et al. [2014] uses OPE to provide ranked search while Ishai et al. [2015] uses ORAM and PIR to achieve less leakage. We believe that such a hybrid construct may allow for new constructions with improved security and schemes with more functionalities. An area of research would be to use this approach to create schemes for multi-users and/or with improved query expressiveness (e.g., ranked, range queries) while achieving minimal leakage through ORAM or PIR.

7.2. Using Other Index Structures

As one may have observed from the discussions in Section 3, most SSE schemes use well-established data structures, ranging from the early schemes using dictionaries to the more recent schemes using balanced binary search trees. It seems that inverted index may still serve as the core component, but other structures, such as trees and graphs, can be combined or further explored to build schemes with different properties, more efficient and/or secure. For example, Lu et al. [2012] combines inverted index and tree structures to reduce storage requirement of an index. Other possible areas of study include improving locality based on different tree structures, extending the works in Cash and Tessaro [2014] and Asharov et al. [2016].

7.3. Security and Leakages

One security issue in SSE is the tradeoff between allowable leakage of information and practicality of a scheme. Ideally, an SSE scheme should minimise leakage. It has been said that fully hidden searchable encryption can be achieved using ORAMs [Goldreich and Ostrovsky 1996; Shi et al. 2011; Stefanov and Shi 2013], asymmetric-based searchable encryption schemes [Boneh et al. 2004], and general cryptographic primitives such as fully homomorphic encryption schemes [Gentry 2009]. Unfortunately, these primitives are expensive for deployment in SSE, particularly on resource-constrained client devices. More recent SSE schemes used these primitives in certain parts of their constructs, especially ORAMs, to minimise leakage. An example scheme is presented in Stefanov et al. [2014]. However, the question remains whether it is possible to further improve SSE such that it is practical yet fully hidden (minimal leakage). Naveed [2015] argued that integrating ORAM does not necessarily provide a fully hidden SSE, which contradicts the widely accepted view that ORAM allows for fully hidden SSE constructs. It would be interesting to further investigate this contradiction.

An understudied topic is *"What can one learn from the leakage patterns?"* Most existing schemes define and allow certain leakage patterns without investigating the nature and extent of information that an attacker may learn. There had been renewed

interest in adversarial and attack models. For example, successful attacks have been demonstrated against SSE schemes by taking advantage of the leakage patterns (see Section 4.5). Given that an attacker has some background knowledge of the messages, most of the proposed schemes could be susceptible to the attacks (e.g., inferences, message insertion attack) described in Islam et al. [2012], Cash et al. [2015], and Zhang et al. [2016a]. The successful attacks by Zhang et al. [2016a] reiterate the importance of ensuring resilience against file insertion attacks using interactive protocols or multiple servers. Interestingly, in recent years, there have been a number of schemes based on multiple servers (e.g., Bösch et al. [2014], Ishai et al. [2015], and Poh et al. [2016]) presented in the literature that do not take into account the attack scenarios discussed above. An area of potential research is, thus, to investigate whether the multi-server setting in these schemes can be adopted to prevent these attacks.

Kellaris et al. [2016] presented successful reconstruction attacks to recover keywords in schemes with range queries by exploiting access patterns. An attacker, in this case, does not need to have background knowledge of the messages or any of the submitted queries or their results. One may explore further such attack mechanisms on schemes with other characteristics and functionalities, and the comparison or combination of reconstruction attacks and other attacks such as inferences and/or message insertion attacks.

We also observed that many ranked keyword search schemes (e.g., Wang et al. [2010] and Sun et al. [2013]) did not provide a formal study on leakages and some do not have proofs of security. *The challenge would be to define a security model for these schemes and how the above attacks affect these schemes.*

One of the mechanisms to eliminate query patterns is to use probabilistic tokens. However, not many such schemes have been proposed in the literature, and it remains a challenge to construct such a scheme that achieves sublinear query efficiency.

7.4. Improving Performance

The issue of locality was recently raised due to performance overhead in actual implementation of SSE (Section 5.3). It relates to read-write access latency due to the pseudorandom locations of the index entries. Most existing schemes provide only theoretical performance and locality is not considered. The challenge would be to ensure locality to improve performance without sacrificing other properties (such as security assurance) of a scheme. As discussed in Section 7.2, we may further explore different tree structures that address the locality issues and are suited for practical implementation.

More recent schemes have also been designed such that they are parallelizable, for example, in Cash et al. [2014], Kamara and Papamanthou [2013], and Xia et al. [2015]. For very large datasets, this feature is desirable in terms of efficiency. Further study can be performed on creating SSE schemes based on data structures that can take advantage of optimised parallel processes and on specific hardware such as general-purpose graphics processing unit to enhance efficiency.

7.5. Improving Setups and Characteristics

Existing schemes that cater to queries from multiple users require active involvement of the data owner. In such a case, the data owner may need to be online often to generate or check query tokens of the users. This can be unrealistic (e.g., due to time zone differences between the data owner and the other users) or even result in a bottleneck. A preferable approach is to allow users to query a database independently without involving the data owner. The challenge, however, is to create a mechanism to provide this service efficiently and securely. For instance, a data owner must still be able to ascertain that a malicious user cannot learn any information from performing the queries. A potential direction would be to combine different primitives such as PIR or use

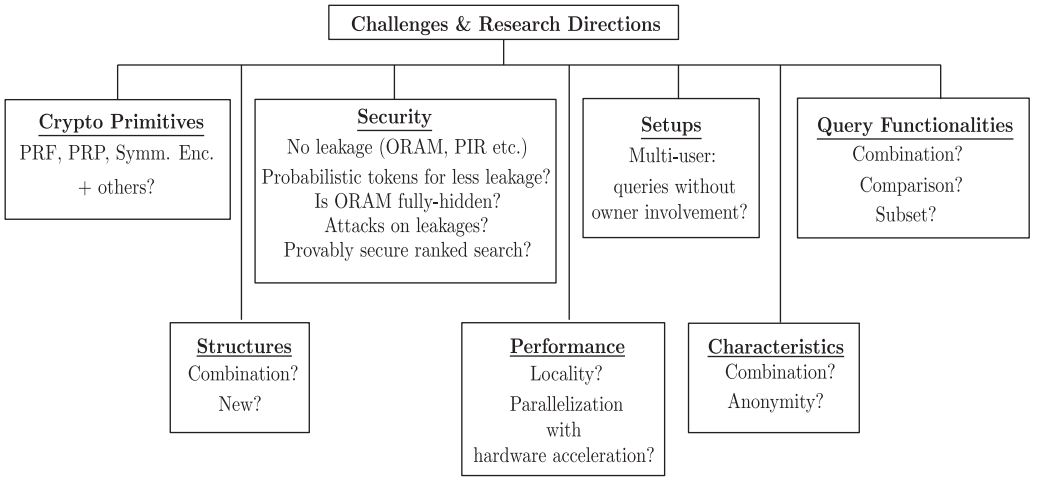


Fig. 10. Challenges and research directions of SSE.

interactive protocols or multiple servers coupled with access control mechanism to cater for multiple users. For instance, one may study multi-user schemes proposed in Jarecki et al. [2013] and Faber et al. [2015] that require users to request search tokens from owner and explore potential extension to eliminate or minimise this requirement.

Another area involving multi-user that is less studied is anonymity. As mentioned in Section 6.6, only Raykova et al. [2009] investigated this issue.

7.6. Improving Query Functionalities

Many SSE proposals focused on providing single-keyword search, and recent proposals appear to focus on multi-keyword searches such as conjunctive and Boolean queries. However, there are only a few, if any, proposals that provide comparison or subset queries as compared to an asymmetric searchable encryption scheme, where such features are provided in Boneh and Waters [2007]. Therefore, providing richer query types may be an area worth further investigation. A potential area of research is to explore graph structures, for example, a recent scheme by Demertzis et al. [2016] provides range queries based on Directed Acyclic Graph (DAG), and to borrow the mechanism of asymmetric searchable encryption schemes, if possible, for the creation of a scheme that allows for comparison or subset queries.

Remarks. Ideally, one may prefer to construct a comprehensive SSE scheme that has all the positive properties under the general framework presented in Figure 4. Nevertheless, such an ideal scheme may not be possible due to the fact that there will always be a tradeoff between performance (or usability) and security. It can, however, serve as a reference point on which practical schemes can be built. Instead, we may consider the requirements under a practical scenario, for example, an enterprise that outsources data to third-party cloud systems. In this particular instance, an SSE scheme should be dynamic, cater to multi-users, and allow multi-keyword search (e.g., conjunctive, range) with efficient implementation based on locality, yet without compromising security (achieving the least leakage with the L1 leakage profile and at least provably secure against adaptive chosen keyword attacks).

Figure 10 summarizes the potential challenges and research directions for SSE. The word "Combination?" in the figure denotes combination of different elements (e.g., structures, query functionalities, or characteristics).

8. CONCLUSION

In this survey, we examined and categorised existing SSE schemes in the literature. Specifically, we summarised the development of SSE and provided detailed description on the constructions of SSE based on our proposed general framework. These include the principals involved and, most crucially, the different structures that are deployed, on which both security and efficiency rely. We compared these structures and outlined the general search performance metrics. We then discussed the security models adopted for SSE and performance criteria, as well as presenting the various characteristics and functionalities. Based on these studies, we outlined the challenges and suggested future research directions. Our general observation on performance is that an inverted index allows for sublinear search but updating would be less direct compared to a direct index. However, a direct index normally results in a linear search. Furthermore, hybrid approaches using a combination of primitives can provide more properties securely. In addition, we remark that a new security model may need to be devised to cater for emerging attacks.

ACKNOWLEDGMENTS

The authors thank Professor Pierangela Samarati, the Associate Editor, and the three anonymous reviewers for their insightful feedback.

REFERENCES

Georgios Amanatidis, Alexandra Boldyreva, and Adam O'Neill. 2007. Provably-Secure Schemes for Basic Query Support in Outsourced Databases. In *Proceedings of the 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security (LNCS)*, Steve Barker and Gail-Joon Ahn (Eds.), Vol. 4602. Springer, 14–30.

Alexandr Andoni and Piotr Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM* 51, 1 (2008), 117–122.

Apache. 2016. Hbase. (2016). Retrieved from https://hbase.apache.org/.

Dave Archer, Dan Bogdanov, Sasha Boldyreva, Seny Kamara, Florian Kerschbaum, Yehuda Lindell, Steve Lu, Jesper Buss Nielsen, Rafail Ostrovsky, Jakob I. Pagter, Ahmad-Reza Sadeghi, and Adrian Waller. 2015. Future Directions in Computing on Encrypted Data. (2015). Retrieved from https://www.cs.bris.ac.uk/~nigel/ECRYPT-MPC/Draft.pdf.

Gilad Asharov, Moni Naor, Gil Segev, and Ido Shahaf. 2016. Searchable symmetric encryption: Optimal locality in linear space via two-dimensional balanced allocations. *IACR Cryptology ePrint Archive (and STOC 2016),* 251 (2016), 1–34.

Foteini Baldimtsi and Olga Ohrimenko. 2015. Sorting and Searching Behind the Curtain. In *Proceedings of the Financial Cryptography 2015 International Conference (FC'15) (LNCS)*, Rainer Böhme and Tatsuaki Okamoto (Eds.), Vol. 8975. Springer, 127–146.

Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. 2001. On the (Im)possibility of obfuscating programs. In *Proceedings of the International Cryptology Conference (CRYPTO'01) (LNCS)*, Joe Kilian (Ed.), Vol. 2139. Springer, 1–18.

Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. 2012. On the (im)possibility of obfuscating programs. *J. ACM* 59, 2 (2012), 6.

Steven M. Bellovin and William R. Cheswick. 2004. Privacy-enhanced searches using encrypted bloom filters. *IACR Cryptology ePrint Archive* 2004/22 (2004).

Tiziano Bianchi, Alessandro Piva, and Mauro Barni. 2009. Encrypted Domain DCT Based on Homomorphic Cryptosystems. *EURASIP J. Inf. Secur.* 2009 (Jan. 2009), Article 1, 12 pages.

bitglass. 2016. bitglass: Cloud Access Security Broker. (2016). Retrieved from http://www.bitglass.com/.

Burton H. Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* 13, 7 (1970), 422–426.

Alexandra Boldyreva and Nathan Chenette. 2014. Efficient Fuzzy Search on Encrypted Data. In *Proceedings of the Fast Sofware Encryption Workshop (FSE'14) (LNCS)*, Carlos Cid and Christian Rechberger (Eds.), Vol. 8540. Springer, 613–633.

Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. 2004. Public Key Encryption with Keyword Search. In *Proceedings of the International Conference on the Theory and Applications*

of Cryptographic Techniques (EUROCRYPT'04) (LNCS), Christian Cachin and Jan Camenisch (Eds.), Vol. 3027. Springer Berlin, 506–522.

Dan Boneh, Amit Sahai, and Brent Waters. 2012. Functional encryption: A new vision for public-key cryptography. *Commun. ACM* 55, 11 (2012), 56–64.

Dan Boneh and Brent Waters. 2007. Conjunctive, Subset, and Range Queries on Encrypted Data. In *Proceedings of the Theory of Cryptography Conference (TCC'07) (LNCS)*, Salil P. Vadhan (Ed.), Vol. 4392. Springer, 535–554.

Christoph Bösch, Pieter Hartel, Willem Jonker, and Andreas Peter. 2014. A Survey of Provably Secure Searchable Encryption. *ACM Comput. Surv.* 47, 2 (2014), Article 18, 51 pages.

Christoph Bösch, Andreas Peter, Bram Leenders, Hoon Wei Lim, Qiang Tang, Huaxiong Wang, Pieter H. Hartel, and Willem Jonker. 2014. Distributed searchable symmetric encryption. In *Proceedings of the IEEE Internationa Conference on Privacy, Security, and Trust (PST'14)*, Ali Miri, Urs Hengartner, Nen-Fu Huang, Audun Jøsang, and Joaquin García-Alfaro (Eds.). 330–337.

Raphael Bost, Pierre-Alain Fouque, and David Pointcheval. 2016. Verifiable Dynamic Symmetric Searchable Encryption: Optimality and Forward Security. *IACR Cryptology ePrint Archive* 2016/62 (2016).

Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou. 2014. Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data. *IEEE Trans. Parallel Distrib. Syst.* 25, 1 (2014), 222–233.

Ning Cao, Zhenyu Yang, Cong Wang, Kui Ren, and Wenjing Lou. 2011. Privacy-Preserving Query over Encrypted Graph-Structured Data in Cloud Computing. In *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS'11)*. 393–402.

David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. 2015. Leakage-Abuse Attacks Against Searchable Encryption. In *Proceedings of the ACM International Conference on Computer and Communications Security (CCS'15)*, Indrajit Ray, Ninghui Li, and Christopher Kruegel (Eds.). ACM, 668–679.

David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. 2014. Dynamic Searchable Encryption in Very Large Databases: Data Structures and Implementation. In *Proceedings of the Network and Distributed Systems Symposium (NDSS'14)*, Vol. 2014. Internet Society.

David Cash, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. 2013. Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries. In *Proceedings of the International Cryptology Conference (CRYPTO'13) (LNCS)*, Ran Canetti and Juan A. Garay (Eds.), Vol. 8042. Springer, 353–373.

David Cash and Stefano Tessaro. 2014. The Locality of Searchable Symmetric Encryption. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'14) (LNCS)*, Phong Q. Nguyen and Elisabeth Oswald (Eds.), Vol. 8441. Springer, 351–368.

Qi Chai and Guang Gong. 2012. Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers. In *Proceedings of the IEEE International Conference on Communications (ICC'12)*. IEEE, 917–922.

Yan-Cheng Chang and Michael Mitzenmacher. 2004. Privacy Preserving Keyword Searches on Remote Encrypted Data. *IACR Cryptology ePrint Archive* 2004/051 (2004), 51.

Yan-Cheng Chang and Michael Mitzenmacher. 2005. Privacy Preserving Keyword Searches on Remote Encrypted Data. In *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS'05) (LNCS)*, John Ioannidis, Angelos D. Keromytis, and Moti Yung (Eds.), Vol. 3531. Springer, 442–455.

Melissa Chase and Seny Kamara. 2010. Structured Encryption and Controlled Disclosure. In *Proceedings of the Internaional Conference on the Theory and Applications of Cryptology and Information Security (ASIACRYPT'10) (LNCS)*, Masayuki Abe (Ed.), Vol. 6477. Springer, 577–594.

Melissa Chase and Emily Shen. 2015. Substring-Searchable Symmetric Encryption. *Proc. Priv. Enhanc. Technol.* 2 (2015), 263–281.

Rong Cheng, Jingbo Yan, Chaowen Guan, Fangguo Zhang, and Kui Ren. 2015. Verifiable Searchable Symmetric Encryption from Indistinguishability Obfuscation. In *Proceedings of the ACM Symposium on Information, Computer and Commucations Security (ASIA CCS'15*), Feng Bao, Steven Miller, Jianying Zhou, and Gail-Joon Ahn (Eds.). ACM, 621–626.

CipherCloud. 2016. CipherCloud: Enterprise Cloud Security. (2016). Retrieved from https://www.ciphercloud.com/.

CryptDB. 2016. CryptDB. (2016). Retrieved from https://css.csail.mit.edu/cryptdb/.

Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. 2006. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In *Proceedings of the ACM Symposium on Information, Computer and Commucations Security (CCS'06*), Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati (Eds.). ACM, 79–88.

Shuguang Dai, Huige Li, and Fangguo Zhang. 2016. Memory leakage-resilient searchable symmetric encryption. *Future Gener. Comp. Syst.* 62 (Sept. 2016), 76–84.

Ioannis Demertzis, Stavros Papadopoulos, Odysseas Papapetrou, Antonios Deligiannakis, and Minos N. Garofalakis. 2016. Practical Private Range Search Revisited. In *Proceedings of the 2016 ACM International Conference on Management of Data (SIGMOD'16)*, Fatma Özcan, Georgia Koutrika, and Sam Madden (Eds.). ACM, 185–198.

Sky Faber, Stanislaw Jarecki, Hugo Krawczyk, Quan Nguyen, Marcel-Catalin Rosu, and Michael Steiner. 2015. Rich Queries on Encrypted Data: Beyond Exact Matches. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS'15) Part II (LNCS)*, Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl (Eds.), Vol. 9327. Springer, 123–145.

Ben A. Fisch, Binh Vo, Fernando Krell, Abishek Kumarasubramanian, Vladimir Kolesnikov, Tal Malkin, and Steven M. Bellovin. 2015. Malicious-Client Security in Blind Seer: A Scalable Private DBMS. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'15)*. IEEE Computer Society, 395–410.

Franco Frattolillo. 2015. Watermarking protocols: Problems, challenges and a possible solution. *Comput. J.* 58, 4 (2015), 944–960.

Zhangjie Fu, Jiangang Shu, Xingming Sun, and Nigel Linge. 2014a. Smart cloud search services: Verifiable keyword-based semantic search over encrypted cloud data. *IEEE Trans. Consum. Electron.* 60, 4 (2014), 762–770.

Zhangjie Fu, Jiangang Shu, Xingming Sun, and Daxing Zhang. 2014b. Semantic Keyword Search Based on Trie over Encrypted Cloud Data. In *Proceedings of the ACM Symposium on Information, Computer and Commucations Security (SCC@ASIACCS'14)*, Robert H. Deng, Elaine Shi, and Kui Ren (Eds.). ACM, 59–62.

Sanjam Garg, Payman Mohassel, and Charalampos Papamanthou. 2015. TWORAM: Round-Optimal Oblivious RAM with Applications to Searchable Encryption. *IACR Cryptology ePrint Archive* 2015/1010 (2015).

Craig Gentry. 2009. *A Fully Homomorphic Encryption Scheme*. Ph.D. Dissertation. Stanford University.

Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity Search in High Dimensions via Hashing. In *Proceedings of the International Conference on Very Large Data Bases (VLDB'99)*, Malcolm P. Atkinson, Maria E. Orlowska, Patrick Valduriez, Stanley B. Zdonik, and Michael L. Brodie (Eds.). Morgan Kaufmann, 518–529.

Eu-Jin Goh. 2003. Secure Indexes. IACR Cryptology ePrint Archive, Report 2003/216. (2003). Retrieved from http://eprint.iacr.org/2003/216/.

Oded Goldreich and Rafail Ostrovsky. 1996. Software Protection and Simulation on Oblivious RAMs. *J. ACM* 43, 3 (1996), 431–473.

Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. 2014. Multi-input Functional Encryption. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'14) (LNCS)*, Phong Q. Nguyen and Elisabeth Oswald (Eds.), Vol. 8441. Springer, 578–602.

Michael T. Goodrich and Michael Mitzenmacher. 2011. Privacy-Preserving Access of Outsourced Data via Oblivious RAM Simulation. In *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP'11), Part II (LNCS)*, Luca Aceto, Monika Henzinger, and Jiri Sgall (Eds.), Vol. 6756. Springer, 576–587.

Florian Hahn and Florian Kerschbaum. 2014. Searchable Encryption with Secure and Efficient Updates. In *Proceedings of the ACM Symposium on Information, Computer and Commucations Security (CCS'14)*, Gail-Joon Ahn, Moti Yung, and Ninghui Li (Eds.). ACM, 310–320.

Fei Han, Jing Qin, and Jiankun Hu. 2016. Secure searches in the cloud: A survey. (unpublished)

Warren He, Devdatta Akhawe, Sumeet Jain, Elaine Shi, and Dawn Xiaodong Song. 2014. ShadowCrypt: Encrypted Web Applications for Everyone. In *Proceedings of the ACM Symposium on Information, Computer and Commucations Security (CCS'14)*, Gail-Joon Ahn, Moti Yung, and Ninghui Li (Eds.), 1028–1039.

Bijit Hore, Ee-Chien Chang, Mamadou H. Diallo, and Sharad Mehrotra. 2012. Indexing Encrypted Documents for Supporting Efficient Keyword Search. In *Proceedings of the 9th International Conference on Very Large Data Bases Workshop and SIAM Activity Group on Data Mining and Analytics (SDM'12) (LNCS)*, Willem Jonker and Milan Petkovic (Eds.), Vol. 7482. Springer, 93–110.

Changhui Hu, Lidong Han, and Siu Ming Yiu. 2016. Efficient and secure multi-functional searchable symmetric encryption schemes. *Secur. Commun. Netw.* 9, 1 (2016), 34–42.

Piotr Indyk and Rajeev Motwani. 1998. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the ACM Symposium on Theory of Computing (STOC'98)*, Jeffrey Scott Vitter (Ed.). ACM, 604–613.

Yuval Ishai, Eyal Kushilevitz, Steve Lu, and Rafail Ostrovsky. 2015. Private Large-Scale Databases with Distributed Searchable Symmetric Encryption. *IACR Cryptology ePrint Archive (and CT-RSA 2016)*, 1190 (2015).

Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. 2012. Access Pattern Disclosure on Searchable Encryption: Ramification, Attack and Mitigation. In *Proceedings of the Network and Distributed Systems Symposium (NDSS'12)*. The Internet Society.

Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. 2014. Inference attack against encrypted range queries on outsourced databases. In *Proceedings of the ACM Conference on Data and Application Security and Privacy (CODASPY'14)*, Elisa Bertino, Ravi S. Sandhu, and Jaehong Park (Eds.). ACM, 235–246.

Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. 2013. Outsourced symmetric private information retrieval. In *Proceedings of the ACM Symposium on Information, Computer and Commucations Security (CCS'13)*, Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung (Eds.). ACM, 875–888.

Seny Kamara. 2013a. Encrypted Search. (2013). Retrieved from https://cs.brown.edu/∼seny/pubs/esearch.pdf.

Seny Kamara. 2013b. How to Search on Encrypted Data: Introduction. (2013). Retrieved from https://outsourcedbits.org/2013/10/06/how-to-search-on-encrypted-data-part-1/.

Seny Kamara and Charalampos Papamanthou. 2013. Parallel and Dynamic Searchable Symmetric Encryption. In *Proceedings of the 2013 Financial Cryptography* International Conference (FC'13) *(LNCS)*, Ahmad-Reza Sadeghi (Ed.), Vol. 7859. Springer, 258–274.

Seny Kamara, Charalampos Papamanthou, and Tom Roeder. 2011. *CS2: A Searchable Cryptographic Cloud Storage System*. Technical Report MSR-TR-2011-58. Microsoft Technical Report. Retrieved from http://research.microsoft.com/pubs/148632/CS2.pdf.

Seny Kamara, Charalampos Papamanthou, and Tom Roeder. 2012. Dynamic searchable symmetric encryption. In *Proceedings of the ACM Symposium on Information, Computer and Commucations Security (CCS'12)*, Ting Yu, George Danezis, and Virgil D. Gligor (Eds.). ACM, 965–976.

Andreas Kasten, Ansgar Scherp, Frederik Armknecht, and Matthias Krause. 2013. Towards Search on Encrypted Graph Data. In *Proceedings of the 2013 International Conference on Society, Privacy and the Semantic Web - Policy and Technology (PrivOn'13), co-located with the 12th International Semantic Web Conference (ISWC'13)*, Stefan Decker, Jim Hendler, and Sabrina Kirrane (Eds.), Vol. 1121. CEUR-WS.org.

Jonathan Katz and Yehuda Lindell. 2007. *Introduction to Modern Cryptography*. Chapman & Hall/CRC.

Stefan Katzenbeisser (ed.). 2007. List of potential applications interested by s.p.e.d. *D3.1, Philips Research (Philips), for Signal Processing in the Encrypted Domain (SPEED) Project, IST-2006-034238, Information Society Technologies* (2007). Retrieved from http://www.speedproject.eu.

Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. 2016. Generic Attacks on Secure Outsourced Databases. In *Proceedings of the ACM Symposium on Information, Computer and Commucations Security (CCS'16)*, Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi (Eds.). ACM, 1329–1340.

Zachary A. Kissel and Jie Wang. 2013. Verifiable Phrase Search over Encrypted Data Secure against a Semi-Honest-but-Curious Adversary. In *Proceedings of the IEEE International Conference on Distributed Computing (ICDCS'13)*. IEEE Computer Society, 126–131.

Zachary A. Kissel and Jie Wang. 2014. A Note on Verifiable Privacy-Preserving Tries. In *Proceedings of the IEEE International Conference on Cloud Computing (CLOUD'14)*. IEEE, 942–943.

Kaoru Kurosawa. 2014. Garbled Searchable Symmetric Encryption. In *Proceedings of the 2014 Financial Cryptography* International Conference (FC'14) *(LNCS)*, Nicolas Christin and Reihaneh Safavi-Naini (Eds.), Vol. 8437. Springer, 234–251.

Kaoru Kurosawa and Yasuhiro Ohtaki. 2012. UC-Secure Searchable Symmetric Encryption. In *Proceedings of the 2012 Financial Cryptography* International Conference (FC'12) *(LNCS)*, Angelos D. Keromytis (Ed.), Vol. 7397. Springer, 285–298.

Kaoru Kurosawa and Yasuhiro Ohtaki. 2013. How to Update Documents Verifiably in Searchable Symmetric Encryption. In *Proceedings of the 12th International Conference on Cryptology and Network Security (CANS'13) (LNCS)*, Michel Abdalla, Cristina Nita-Rotaru, and Ricardo Dahab (Eds.), Vol. 8257. Springer, 309–328.

Kaoru Kurosawa and Yasuhiro Ohtaki. 2015. How to Construct UC-Secure Searchable Symmetric Encryption Scheme. *IACR Cryptology ePrint Archive* 2015/251 (2015).

Mehmet Kuzu, Mohammad Saiful Islam, and Murat Kantarcioglu. 2012. Efficient Similarity Search over Encrypted Data. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE'12)*. IEEE Computer Society, 1156–1167.

Mehmet Kuzu, Mohammad Saiful Islam, and Murat Kantarcioglu. 2014. Efficient privacy-aware search over encrypted databases. In *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy (CODASPY'14)*. ACM, 249–256.

Mehmet Kuzu, Mohammad Saiful Islam, and Murat Kantarcioglu. 2015. Distributed Search over Encrypted Big Data. In *Proceedings of the ACM Conference on Data and Application Security and Privacy (CODASPY'15)*, Jaehong Park and Anna Cinzia Squicciarini (Eds.). ACM, 271–278.

Russell W. F. Lai and Sherman S. M. Chow. 2015. Structured Encryption with Non-interactive Updates and Parallel Traversal. In *Proceedings of the IEEE International Conference on Distributed Computing (ICDCS'15)*. IEEE, 776–777.

Billy Lau, Simon P. Chung, Chengyu Song, Yeongjin Jang, Wenke Lee, and Alexandra Boldyreva. 2014. Mimesis Aegis: A Mimicry Privacy Shield-A System's Approach to Data Privacy on Public Cloud. In *Proceedings of the 23rd USENIX Security Symposium*, Kevin Fu and Jaeyeon Jung (Eds.). USENIX Association, 33–48.

Jin Li, Qian Wang, Cong Wang, Ning Cao, Kui Ren, and Wenjing Lou. 2009. Enabling Efficient Fuzzy Keyword Search over Encrypted Data in Cloud Computing. *IACR ePrint Archive* 2009/593 (2009).

Jin Li, Qian Wang, Cong Wang, Ning Cao, Kui Ren, and Wenjing Lou. 2010. Fuzzy Keyword Search over Encrypted Data in Cloud Computing. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'10)*. IEEE, 441–445.

Shuyu Li and Jerry Gao. 2016. *Security and Privacy for Big Data*. Springer, 281–313.

Chang Liu, Liehuang Zhu, and Jinjun Chen. 2015. Efficient Searchable Symmetric Encryption for Storing Multiple Source Data on Cloud. In *Proceedings of the 2015 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom/BigDataSE/ISPA)*, Vol. 1. IEEE, 451–458.

Chang Liu, Liehuang Zhu, Mingzhong Wang, and Yu-an Tan. 2014. Search pattern leakage in searchable encryption: Attacks and new construction. *Inf. Sci.* 265 (2014), 176–188.

Ximeng Liu, Rongxing Lu, Jianfeng Ma, Le Chen, and Haiyong Bao. 2016b. Efficient and privacy-preserving skyline computation framework across domains. *Future Gener. Comp. Syst.* 62 (2016), 161–174.

Zhe Liu, Raymond Choo, and Minghao Zhao. 2016a. Practical-oriented protocols for privacy-preserving outsourced big data analysis: Challenges and future research directions. (unpublished)

Haining Lu, Dawu Gu, Chongying Jin, and Yinqi Tang. 2012. Reducing extra storage in searchable symmetric encryption scheme. In *Proceedings of the IEEE International Conference on Cloud Computing (CLOUD'12)*. IEEE Computer Society, 255–262.

Xianrui Meng, Seny Kamara, Kobbi Nissim, and George Kollios. 2015. GRECS: Graph Encryption for Approximate Shortest Distance Queries. In *Proceedings of the ACM Symposium on Information, Computer and Commucations Security (CCS'15)*, Indrajit Ray, Ninghui Li, and Christopher Kruegel (Eds.). ACM, 504–517.

Tarik Moataz and Abdullatif Shikfa. 2013. Boolean symmetric searchable encryption. In *Proceedings of the ACM Symposium on Information, Computer and Commucations Security (ASIACCS'13)*, Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng (Eds.). ACM, 265–276.

Tarik Moataz, Abdullatif Shikfa, Nora Cuppens-Boulahia, and Frédéric Cuppens. 2013. Semantic search over encrypted data. In *Proceedings of the 2013 IEEE International Conference on Telecommunications (ICT'13)*. IEEE, 1–5.

Moesfa Soeheila Mohamad and Geong Sen Poh. 2012. Verifiable Structured Encryption. In *Proceedings of the 8th International Conference on Information Security and Cryptology (Inscrypt'12) (LNCS)*, Miroslaw Kutylowski and Moti Yung (Eds.), Vol. 7763. Springer, 137–156.

Muhammad Naveed. 2015. The Fallacy of Composition of Oblivious RAM and Searchable Encryption. *IACR Cryptology ePrint Archive* 2015/668 (2015).

Muhammad Naveed, Manoj Prabhakaran, and Carl A. Gunter. 2014. Dynamic Searchable Encryption via Blind Storage. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'14)*. IEEE Computer Society, 639–654.

Skyhigh Networks. 2016. Skyhigh Networks: Cloud Security Software. (2016). Retrieved from https://www.skyhighnetworks.com/.

Wakaha Ogata, Keita Koiwa, Akira Kanaoka, and Shin'ichiro Matsuo. 2013. Toward Practical Searchable Symmetric Encryption. In *Proceedings of the International Workshop on Security (IWSEC'13) (LNCS)*, Kazuo Sakiyama and Masayuki Terada (Eds.), Vol. 8231. Springer, 151–167.

Cengiz Orencik, Ayse Selcuk, Erkay Savas, and Murat Kantarcioglu. 2016. Multi-Keyword search over encrypted data with scoring and search pattern obfuscation. *Int. J. Inf. Sec.* 15, 3 (June 2016), 251–269.

Vasilis Pappas, Fernando Krell, Binh Vo, Vladimir Kolesnikov, Tal Malkin, Seung Geol Choi, Wesley George, Angelos D. Keromytis, and Steven Bellovin. 2014. Blind Seer: A Scalable Private DBMS. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'14)*. IEEE Computer Society, 359–374.

Vasilis Pappas, Mariana Raykova, Binh Vo, Steven M. Bellovin, and Tal Malkin. 2011. Private search in the real world. In *Proceedings of the ACM Annual Computer Security Applications Conference (ACSAC'11)*, Robert H'obbes' Zakon, John P. McDermott, and Michael E. Locasto (Eds.). ACM, 83–92.

Geong Sen Poh, Moesfa Soeheila Mohamad, and Ji-Jian Chin. 2016. Searchable Symmetric Encryption Over Multiple Servers. In *Proceedings of the (ArcticCrypt'16)*.

Geong Sen Poh, Moefa Soeheila Mohamad, and Muhammad Reza Z'aba. 2012. Structured Encryption for Conceptual Graphs. In *Proceedings of the International Workshop on Security (IWSEC'12) (LNCS)*, Goichiro Hanaoka and Toshihiro Yamauchi (Eds.), Vol. 7631. Springer, 105–122.

Raluca A. Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. 2011. CryptDB: Protecting confidentiality with encrypted query processing. In *Proceedings of the Symposium on Operating Systems Principles (SOSP'11)*, Ted Wobber and Peter Druschel (Eds.). ACM, 85–100.

Yogachandran Rahulamathavan, Raphael C.-W. Phan, Jonathan A. Chambers, and David J. Parish. 2013. Facial Expression Recognition in the Encrypted Domain Based on Local Fisher Discriminant Analysis. *T. Affect. Comput.* 4, 1 (2013), 83–92.

Mariana Raykova, Binh Vo, Steven M. Bellovin, and Tal Malkin. 2009. Secure anonymous database search. In *Proceedings of the ACM Conference on Computer and Communications Security Workshop (CCSW'09)*, Radu Sion and Dawn Song (Eds.). ACM, 115–126.

Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. 1978. On data banks and privacy homomorphisms. *Foundations of Secure Computation.* Academia Press, 169–179.

Panagiotis Rizomiliotis and Stefanos Gritzalis. 2015. ORAM Based Forward Privacy Preserving Dynamic Searchable Symmetric Encryption Schemes. In *Proceedings of the ACM Conference on Computer and Communications Security Workshop (CCSW'15)*, Indrajit Ray, Xiaofeng Wang, Kui Ren, Florian Kerschbaum, and Cristina Nita-Rotaru (Eds.). ACM, 65–76.

Md Iftekhar Salam, Wei-Chuen Yau, Ji-Jian Chin, Swee-Huay Heng, Huo-Chong Ling, Raphael C.-W. Phan, Geong Sen Poh, Syh-Yuan Tan, and Wun-She Yap. 2015. Implementation of searchable symmetric encryption for privacy-preserving keyword search on cloud storage. *Hum.-Centric Comput. Inf. Sci.* 5, 1 (2015), 1–16.

Elaine Shi, T.-H. Hubert Chan, Emil Stefanov, and Mingfei Li. 2011. Oblivious RAM with O($(logN)^3$) Worst-Case Cost. In *Proceedings of the Internaional Conference on the Theory and Applications of Cryptology and Information Security (ASIACRYPT'11) (LNCS)*, Dong Hoon Lee and Xiaoyun Wang (Eds.), Vol. 7073. Springer, 197–214.

Dawn Xiaodong Song, David Wagner, and Adrian Perrig. 2000. Practical Techniques for Searches on Encrypted Data. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'00)*. IEEE Computer Society, 44.

Emil Stefanov, Charalampos Papamanthou, and Elaine Shi. 2014. Practical Dynamic Searchable Encryption with Small Leakage. In *Proceedings of the Network and Distributed Systems Symposium (NDSS'14)*. The Internet Society. http://www.internetsociety.org/events/ndss-symposium-2014.

Emil Stefanov and Elaine Shi. 2013. ObliviStore: High Performance Oblivious Cloud Storage. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'13)*. IEEE Computer Society, 253–267.

Mikhail Strizhov and Indrajit Ray. 2014. Multi-keyword Similarity Search over Encrypted Cloud Data. In *Proceedings of the IFIP Advances in Information and Communication Technology (IFIP TC 11SEC'14)*, Nora Cuppens-Boulahia, Frédéric Cuppens, Sushil Jajodia, Anas Abou El Kalam, and Thierry Sans (Eds.), Vol. 428. Springer, 52–65.

Wenhai Sun, Bing Wang, Ning Cao, Ming Li, Wenjing Lou, Y. Thomas Hou, and Hui Li. 2013. Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. In *Proceedings of the ACM Symposium on Information, Computer and Commucations Security (ASIA CCS'13)*, Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng (Eds.). ACM, 71–82.

Ashwin Swaminathan, Yinian Mao, Guan-Ming Su, Hongmei Gou, Avinash L. Varna, Shan He, Min Wu, and Douglas W. Oard. 2007. Confidentiality-preserving rank-ordered search. In *Proceedings of the 2007 ACM Workshop on Storage Security and Survivability (StorageSS'07)*, Valerie Henson (Ed.). ACM, 7–12.

Shunsuke Taketani and Wakaha Ogata. 2015. Improvement of UC Secure Searchable Symmetric Encryption Scheme. In *Proceedings of the International Workshop on Security (IWSEC'15) (LNCS)*, Keisuke Tanaka and Yuji Suga (Eds.), Vol. 9241. Springer, 135–152.

Syh-Yuan Tan, Ji-Jian Chin, Geong Sen Poh, Yvonne H. S. Kam, and Wei-Chuen Yau. 2015. A Client-Server Prototype of a Symmetric Key Searchable Encryption Scheme Using Open-Source Applications. In *Proceedings of the International Conference of IT Convergence and Security (ICITCS'15)*. 1–5.

Jun Tang, Yong Cui, Qi Li, Kui Ren, Jiangchuan Liu, and Rajkumar Buyya. 2016. Ensuring Security and Privacy Preservation for Cloud Data Services. *ACM Comput. Surv.* 49, 1 (June 2016), Article 13, 39 pages.

Qiang Tang. 2012. Search in Encrypted Data: Theoretical Models and Practical Applications. *IACR Cryptology ePrint Archive* 2012/648 (2012), 1–22.

Peter van Liesdonk, Saeed Sedghi, Jeroen Doumen, Pieter H. Hartel, and Willem Jonker. 2010. Computationally Efficient Searchable Symmetric Encryption. In *Proceedings of the International Conference on Very Large Data Bases Workshop and SIAM Activity Group on Data Mining and Analytics (SDM'10) (LNCS)*, Willem Jonker and Milan Petkovic (Eds.), Vol. 6358. Springer, 87–100.

Cong Wang, Ning Cao, Jin Li, Kui Ren, and Wenjing Lou. 2010. Secure Ranked Keyword Search over Encrypted Cloud Data. In *Proceedings of the IEEE International Conference on Distributed Computing (ICDCS'10)*. IEEE, 59–62.

Cong Wang, Ning Cao, Kui Ren, and Wenjing Lou. 2012. Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data. *IEEE Trans. Parallel Distrib. Syst.* 23, 8 (2012), 1467–1479.

Wai Kit Wong, David Wai-Lok Cheung, Ben Kao, and Nikos Mamoulis. 2009. Secure kNN computation on encrypted databases. In *Proceedings of the 2009 ACM International Conference on Management of Data (SIGMOD'09)*. ACM, 139–152.

Zhihua Xia, Xinhui Wang, Xingming Sun, and Qian Wang. 2015. A Secure and Dynamic Multi-keyword Ranked Search Scheme over Encrypted Cloud Data. *IEEE Trans. Parallel Distrib. Syst., preprints* (2015). DOI: http://dx.doi.org/doi:10.1109/TPDS.2015.2401003

Zhihua Xia, Yanling Zhu, Xingming Sun, and Lihong Chen. 2014. Secure semantic expansion based search over encrypted cloud data supporting similarity ranking. *J. Cloud Comput.: Adv. Syst. Appl.* 3, 8 (2014), 1–11.

Attila Altay Yavuz and Jorge Guajardo. 2015. Dynamic Searchable Symmetric Encryption with Minimal Leakage and Efficient Updates on Commodity Hardware. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC'15) (LNCS)*, Orr Dunkelman and Liam Keliher (Eds.), Vol. 9566. Springer, 241–259.

Rui Zhang, Rui Xue, Ting Yu, and Ling Liu. 2016b. Dynamic and Efficient Private Keyword Search over Inverted Index-Based Encrypted Data. (unpublished)

Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. 2016a. All Your Queries Are Belong to Us: The Power of File-Injection Attacks on Searchable Encryption. *IACR Cryptology ePrint Archive* 2016/172 (2016).

Received August 2016; revised January 2017; accepted March 2017