Efficient Verifiable Multi-user Searchable Symmetric Encryption for Encrypted Data in the Cloud

Lanxiang Chen(✉) and Nan Zhang

Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350117, China
lxiangchen@fjnu.edu.cn

Abstract. Encryption is the basic technology to ensure the security of the data in the cloud, while ciphertext search is the key to improve the usability of the cloud storage. Most of the searchable encryption schemes consider the honest-but-curious or semi-honest cloud server. However, cloud storage in reality may be unreliable or even malicious. In this case, the encrypted data and search results returned by the server is not completely trustable, so it is crucial to verify the integrity of search results and encrypted data. Considering the untrusted cloud server security model, this paper proposes an efficient verifiable multi-user searchable symmetric encryption (VMSSE) scheme. It is efficient both in computation and storage. In particular, the work performed by the server per returned document is constant as opposed to linear in the size of the data. The computation and storage at the user is O(1). It allows the user to verify the search was computed honestly in the presence of a dishonest-and-curious server. And it supports multi-user searching. Finally, the security analysis shows that it is an efficient and feasible scheme.

Keywords: Cloud storage · Searchable symmetric encryption · Integrity verification · Multi-user searchable encryption

1 Introduction

Searchable symmetric encryption (SSE) allows clients with either limited resources or limited expertise to outsource the storage of its data to another party at low cost, while maintaining the ability to selectively retrieve segments of their data. It addresses this issue by indexing the encrypted data in such a way as to allow a server to execute a search query over the encrypted data and return the identifiers of any file that satisfies the search query.

Most of the existing work on SSE focuses on efficiently preserving confidentiality in the presence of an honest-but-curious server. This means that the server is trusted to follow the search protocol honestly but may try to infer information about data or search queries that it is unauthorized to know. However, in reality, besides its curiosity, a cloud server may be selfish in order to save its computation and/or communication resource. In this paper, we investigate the searchable encryption problem in the

J. Wan et al. (Eds.): CloudComp 2016, SPNCE 2016, LNICST 197, pp. 173–183, 2018.
https://doi.org/10.1007/978-3-319-69605-8_16

presence of a dishonest-and-curious server, which may execute only a fraction of search operations and return a fraction of search results. To fight against this strongest adversary, the verifiable SSE scheme is proposed to offer verifiable searchability in addition to the data privacy.

The remainder of the paper is organized as follows. Section 2 presents the related works and Sect. 3 gives the preliminary and symbols definition. Section 4 details the VMSSE formal definition and the scheme construction. Section 5 gives the security and performance analysis and Sect. 6 concludes the paper.

2 Related Works

The first SSE scheme is proposed by Song in [1]. The construction is proven to be a secure encryption scheme, but it is not proven to be a secure searchable encryption scheme. Goh [2] introduces a notion of security for indexes and puts forth a construction based on Bloom filters and pseudo-random functions. An inherent problem of using Bloom filters is the possibility of false positives. Chang and Mitzenmacher [3] develop two index schemes, similar to Goh [2]. The idea is to use a prebuilt dictionary of search keywords to build an index per document. The index is an m-bit array, initially set to 0, where each bit position corresponds to a keyword in the dictionary. If the document contains a keyword, its index bit is set to 1.

Curtmola et al. [4] propose two new constructions SSE-1 and SSE-2, where the idea is to add an inverted index, which is an index per distinct word in the database instead of per document. This reduces the search time to the number of documents that contain the keyword. This is not only sublinear but also optimal. They present the first sublinear solution for single-keyword search whose complexity is linear in the number of matching documents. They also improve on previous security models, in particular by providing an adaptive security definition and solutions in this model.

Recently, dynamic SSE [5–8], ranked SSE [9], similarity search [10, 11], multi-keyword and multi-function SSE [12–15] has been studied extensively. However, security against malicious servers has been overlooked in most previous constructions and these only addressed security against honest-but-curious servers. In reality, besides its curiosity, a cloud server may malfunction or even be malicious itself. Even if the server is honest, a virus, worm, trojan horse or a software bug may delete, forge or swap some encrypted files. Therefore, users need a result verification mechanism to detect the potential misbehavior in this computation outsourcing model.

In order to cope with this problem, the first verifiable SSE is proposed by Kurosawa and Ohtaki in [16]. Based on SSE-2, they designed a verification scheme for single keyword search by using RSA accumulator. But their solution is inefficient, the searches or update overhead is linear in the number of documents in the database. The verification complexity there is linear in the problem size $O(tn)$.

Sun et al. [17] propose to build the search index based on term frequency and the vector space model with cosine similarity measure to achieve higher search result accuracy. They devise a scheme upon the proposed index tree structure to sign the root of the index tree to enable authenticity check over the returned search results. Later, they propose a UC-secure verifiable SSE based on bilinear Accumulation Tree [18].

Cheng et al. [19] propose a verifiable SSE based on secure indistinguishability obfuscation (iO). Zheng et al. [20] propose a verifiable SSE based on attribute-based encryption and bloom filter. Wang et al. propose two verifiable fuzzy keyword search schemes based on Bloom filter [21] and symbol-tree [22] respectively. Fu et al. [23] propose a smart semantic search scheme, which returns not only the result of keyword-based exact match, but also the result of keyword-based semantic match. At the same time, the proposed scheme supports the verifiability of search result. Stefanov et al. [24] gave a dynamic encrypted data search scheme with small search privacy leakage, which enables result verification for single keyword search.

In this paper, we present an efficient verifiable multi-user searchable symmetric encryption (VMSSE) scheme, which takes into account searching verification in the dishonest-and-curious server model based on SSE-1 [4]. The difference lies in that only the indispensable keys are sent to the authorized user, other than all the secrets in SSE-1. This is done to avoid collusion.

3 Symbol Definition

First, two pseudo-random functions (PRF) and three pseudo-random permutations (PRP) that will be used are given as follows. $f(\bullet)$ and $g(\bullet)$ are PRF and the others are PRP with the following parameters, here k and l are security parameters, m is the total size of the encrypted document collection in "min-units", where a min-unit is the smallest possible size for a keyword (e.g., one byte). n is the number of documents in the document collection, p is the bit length of the keyword. They are polynomial-time commutable functions that cannot be distinguished from random functions by any probabilistic polynomial-time adversary.

$$
\begin{aligned}
&f : \{0,1\}^{\kappa} \times \{0,1\}^{p} \rightarrow \{0,1\}^{l+\log_2(m)}.\\
&g : \{0,1\}^{\kappa} \times \{0,1\}^{n\times\log_2(n)} \rightarrow \{0,1\}^{l+\log_2(m)}.\\
&\varphi : \{0,1\}^{\kappa} \times \{0,1\}^{\log_2(m)} \rightarrow \{0,1\}^{\log_2(m)}.\\
&\pi : \{0,1\}^{\kappa} \times \{0,1\}^{p} \rightarrow \{0,1\}^{p}.\\
&\rho : \{0,1\}^{\kappa} \times \{0,1\}^{p+\log_2(m)+l} \rightarrow \{0,1\}^{p+\log_2(m)+l}.
\end{aligned}
$$

BE = (*Gen, Enc, Add, Revoke, Dec*) is a broadcast encryption scheme [25], which is a tuple of five polynomial-time algorithms that work as follows. Let N be BE's user space, i.e., the set of all possible user identifiers. *Gen* is a probabilistic algorithm that takes as input a security parameter k and outputs a master key mk. *Enc* is a probabilistic algorithm that takes as input a master key mk, a set of users $G \subseteq N$ and a message m, and outputs a ciphertext c. *Add* is a probabilistic algorithm that takes as input a master key mk and a user identifier $U \in N$, and outputs a user key uk_U. *Revoke* is a deterministic algorithm that takes as input a user key uk_U and a user identifier $U \in N$, and outputs $R = R \cup \{U\}$, $G = G \backslash \{U\}$. Finally, *Dec* is a deterministic algorithm that takes as input a user key uk_U and a ciphertext c and outputs either a message m or the failure symbol $\perp$. Informally, a broadcast encryption scheme is secure if its ciphertexts leak no useful information about the message to any user not in G.

$D = \{D_1, D_2, \ldots, D_n\}$ is a collection of n documents. $\tau = (\tau_{w1}, \tau_{w2}, \ldots, \tau_{wd})$ is the verifiable tags. $\Delta = \{w_1, w_2, \ldots, w_d\}$ is a dictionary of d words, and 2^{Δ} be the set of all possible documents. We assume that words in Δ can be represented using at most p bits. $\Delta' \subseteq \Delta$, be the set of distinct words that exist in the document collection D.

$D(w)$ is the set of identifiers of documents in D that contain word w ordered in lexicographic order. id(D) is the identifier of document D, where the identifier can be any string that uniquely identifies a document, such as a memory location.

4 The VMSSE Scheme

4.1 VMSSE Formal Definition

We begin by reviewing the definition of a VMSSE scheme.

Definition 4.1. A VMSSE scheme is a collection of seven polynomial-time algorithms *VMSSE = (Keygen*, *BuildIndex*, *GenToken*, *Search*, *Verify*, *AddUser*, *RevokeUser*) such that:

K $\leftarrow$ ***Keygen*****(1** ^{k}**)**: is a probabilistic key generation algorithm that is run by the user to setup the scheme. It takes a security parameter k, and returns a secret key K such that the length of K is polynomially bounded in k.

(I*, *τ) $\leftarrow$ ***BuildIndex(K, D)*****:** is a (possibly probabilistic) algorithm run by the user to generate indexes. It takes a secret key K and a polynomially bounded in k document collection D as inputs, and returns an index I and the verifiable tag τ.

$(T_{U,w}, \tau_w) \leftarrow$ ***GenToken(K, w, U)*****:** is run by the owner to generate a token for a given word and an authorized user. It takes a secret key K, a word w and the authorized user U as inputs, and returns a trapdoor $T_{U,}w$ and the verifiable tag τ_w.

D**(*w*)** $\leftarrow$ ***Search*****(*I,* $T_{U,w}$**):** is run by the server S in order to search for the documents in D that contain word w. It takes an index I for a collection D and a trapdoor $T_{U,}w$ for word w as inputs, and returns $D(w)$, the set of identifiers of documents containing w.

(1, 0) $\leftarrow$ ***Verify*****(*D*(*w*),** τ_w**):** is run by the user to check whether S returns the complete list. It takes $D(w)$ and the verifiable tag τ_w as inputs, and outputs 1, if it is true, otherwise, outputs 0.

$K_U \leftarrow$ ***AddUser*****(*K*, *U*):** is run by the owner to add a user to the authorized user set. It takes a secret key K and the authorized user U as inputs, and returns K_U to the authorized user.

(*r'*, $Enc_{N\backslash R}(r')$**)** $\leftarrow$ ***RevokeUser*****(*K*, *U*):** is run by the owner to revoke a user from the authorized user set. It takes a secret key K and the authorized user U as inputs, and returns r' and $Enc_{N\backslash R}(r')$ to the server.

4.2 The Scheme Construction

We describe the details of the construction as follows. In VMSSE, a single index I is associated with a document collection D. It consists of two data structures for each word $w \in \Delta'$. The first one is an array A, which stores the encrypted $D(w)$ and a

look-up table T. We start with a collection of linked lists L_i, where the nodes of each L_i are the identifiers of documents in $D(w_i)$ for each $w_i \in \Delta'$. Before encryption, the j-th node of L_i is augmented with information about the index in A of the $(j + 1)$-th node of L_i, together with the key used to encrypt it. We then write the nodes of all lists L_i in a random order and encrypted with randomly generated keys into the array A.

The other is the look-up table T, which contains information that enables one to locate and decrypt the appropriate elements from A. It is managed using indirect addressing. We now build a look-up table T that allows one to locate and decrypt the first element of each list L_i. Each entry in T corresponds to a word $w_i \in \Delta'$ and consists of a pair <address,value>. The field value contains the index in A and the decryption key for the first element of L_i. value is itself encrypted using the output of a pseudo-random function. The field address is simply used to locate an entry in T.

The owner computes both A and T based on the unencrypted D, and stores them on the server together with the encrypted D. When the user wants to retrieve the documents that contain word w_i, it computes the decryption key and the address for the corresponding entry in T and sends them to the server. The server locates and decrypts the given entry of T, and gets the index in A and the decryption key for the first node of L_i. Since each element of L_i contains information about the next element of L_i, the server can locate and decrypt all the nodes of L_i, which gives the identifiers in $D(w_i)$. As the server is dishonest, it may execute only a fraction of search operations honestly and return a fraction of search results. To ensure the completeness and correctness of search results, it allows the user to verify whether the server execute search operations honestly. The detail of the scheme is descried in Fig. 1.

In BE scheme, the long-lived secrets are the session keys between the users and the server. The long-lived secrets are distinct for each user. Given an encrypted message, the long-lived secrets allow a user to decrypt it only if the user was non-revoked at the time the message was encrypted. Different from SSE-1, the key r currently used for ρ is not sent to the authorized user. Each time the authorized user wants to search, he need to request a token from the file owner. Each time a user is revoked, the owner picks a new r and stores it on the server in encrypted form. As the revoked user have to request search token from the owner for each keyword he wants to search, if he is revoked, the owner will not issue token to him. So even though a revoked user which has been re-authorized to search could recover (old) values of r that were used while he was revoked, these values are no longer of interest.

5 The Security and Performance Analysis

This Section gives the security and performance analysis of the proposed scheme.

5.1 The Security Analysis

Since in practice the encrypted documents will also be stored on the server, we can assume that the document sizes and identifiers will also be leaked. If we wish not to disclose the size of the documents, this can be easily achieved by “padding” each plaintext document such that all documents have a fixed size. So the VMSSE scheme

Keygen(1^k, 1^l): Generate random keys $s, x, y, z, r \xleftarrow{R} \{0,1\}^k$ and output $K = (s, x, y, z, r, 1^l)$.

BuildIndex(K, D):

1. Initialization:
   (a) scan D and build Δ', the set of distinct words in D. For each word $w \in \Delta'$, build $D(w)$;
   (b) initialize a global counter ctr = 1;
   (c) initialize the BE scheme. Set R = $\{\Phi\}$. Send r and $E_N^{BE}(r)$ to the server.
2. Generate verifiable tags: for each $D(w)$, generate $\tau_w = g_x(D(w)) \oplus f_y(w)$, and $\tau=(\tau_{w1}, \tau_{w2}, \cdots \tau_{wd})$.
3. Build array A:
   (a) for each $w_i \in \Delta'$:
   - generate $k_{i,0} \xleftarrow{R} \{0,1\}^l$;
   - for $1 \le j \le |D(w_i)|$:
     - generate $k_{i,j} \xleftarrow{R} \{0,1\}^l$ and set node $N_{i,j}$=<id($D_{i,j}$)$||k_{i,j}||\varphi_s$(ctr+1)>, where id($D_{i,j}$) is the j^{th} identifier in $D(w_i)$;
     - compute $E_{ki,j-1}(N_{i,j})$, and store it in A[φ_s(ctr)];
     - ctr = ctr + 1.
   - for the last node of Li (i.e., $N_{i,|D(w_i)|}$), before encryption, set the address of the next node to NULL.

   (b) let $m'=\sum_{wi \in \Delta'}|D(wi)|$. If $m'<m$, then set remaining (m-m') entries of A to random values of the same size as the existing m' entries of A.
4. Build look-up table T:
   (a) for each $w_i \in \Delta'$:
   - value = <addr(A($N_{i,1}$))$||k_{i,0}$> $\oplus f_y(w_i)$;
   - set T[$\pi_z(w_i)$] = value.

   (b) if $|\Delta'|<|\Delta|$, then set the remaining ($|\Delta|$ -$|\Delta'|$) entries of T to random values.
5. Output I = (A, T).

GenToken(w): Output $T_w=(\pi_z(w), f_y(w))$ and $T_{U,w}=\rho_r(T_w)$, and send ($T_{U,w}$, τ_w) to the user.

Search(I, $T_{U,w}$):

1. User sends $T_{U,w}$ to the server;
2. The server recovers $T_w = \rho^{-1}{}_r(T_{U,w})$; let $T_w=(\gamma, \eta)$. If γ is a valid virtual address, then:
3. Retrieve θ=T[γ]. Let <$\alpha||k$>=$\theta \oplus \eta$;
4. Decrypt the list L starting with the node at address α encrypted under key k;
5. Output the list of document identifiers, denoted as $D'(w)$; Otherwise, return $\perp$.

Verify($D'(w)$, τ_w): User computes $\tau = g_x(D'(w)) \oplus f_y(w)$, and checks whether $\tau = \tau_w$, if yes, it means that S returns the complete list, otherwise, the user requires S to search again.

AddUser(K, U): Send K_U=(x, y) and the long-lived secrets needed for the BE scheme to U.

RevokeUser(K, U): R = R $\cup$ {U}. Pick a new key $r' \leftarrow \{0,1\}^l$ and send r' and $E_{N\backslash R}^{BE}(r')$ to S. S overwrites the old values of r and $E_{N\backslash R\cup\{U\}}^{BE}(r)$ with r' and $E_{N\backslash R}^{BE}(r')$, respectively.

Fig. 1. The description of the VMSSE scheme.

reveals only the access pattern, the search pattern, the total size of the encrypted document collection, and the number of documents it contains. There maybe exist some security threat as follows.

Colluding. Server may collude with the users not to revoke them. However, if the server doesn't revoke some users in time, they can only search keywords which have

been searched before. As the revoked user have to request search token from the owner for each keyword he wants to search, if he is revoked, the owner will not issue token to him. So even though a revoked user could recover (old) values of r that were used while he was revoked, these values are no longer of interest.

Revealing verifiable tags and public verification. As the user requests search token, the owner will send the verifiable tag to him. Then the verifiable tags is leaked to him. However, it will not lead to adverse consequences. As the user have to generate the tag based on the search results returned from the server each time, it doesn't help even the attacker gets the verifiable tags. The proposed verification mechanism is efficient and flexible, which can be either delegated to a public trusted authority (TA) or be executed privately by data users.

As VMSSE is extended based on SSE-1, there are several theorems and claims about the security of SSE-1.

Definition 5.1. *(Non-Adaptive Indistinguishability Security for SSE) A SSE scheme is secure in the sense of non-adaptive indistinguishability if for any two adversarially constructed histories with equal length and trace, no (probabilistic polynomial-time) adversary can distinguish the view of one from the view of the other with probability non-negligibly better than 1/2.* (see Definition 3.2 [1]).

Theorem 5.2. *Non-adaptive indistinguishability security of SSE is equivalent to non-adaptive semantic security of SSE.* (see Theorem 3.4 [1]).

Theorem 5.3. *SSE-1 is a non-adaptively semantic secure SSE scheme.* (see Theorem 4.1 [1]).

Theorem 5.4. *If SSE-1 is a non-adaptive semantic secure SSE scheme, then the VMSSE is a non-adaptive semantic secure SSE scheme.*

Proof. Compared to SSE-1, we have added the verification mechanism and issued $T_{U,w}$ as trapdoor in the VMSSE scheme. In the step of *Initialization*, the verifiable tags are generated as $\tau_w = g_x(D(w)) \oplus f_y(w)$, and $\tau = (\tau_{w1}, \tau_{w2}, \ldots \tau_{wd})$. Here $g(\bullet)$ and $f(\bullet)$ are pseudo-random functions (PRF), their pseudo-randomness guarantees that each element of the verifiable tags is indistinguishable from any random counterpart. In the step of *GenToken*, the token for authorized user U is generated as $T_w = (\pi_z(w), f_y(w))$ and $T_{U,w} = \rho_r(T_w)$, and send $(T_{U,w}, \tau_w)$ to the user. Here $f(\bullet)$ is pseudo-random functions, $\pi(\bullet)$ and $\rho(\bullet)$ are pseudo-random permutations (PRP), their pseudo-randomness guarantees that the token is indistinguishable from any random counterpart.

The security of a multi-user scheme can be defined similarly to the security of a single-user scheme, as the server should not learn anything about the documents and queries beyond what can be inferred from the access and search patterns. When the owner of a document collection D gives a user U permission to search through D, it sends to U all the secret information needed to perform searches in a single-user context in SSE-1. While in VMSSE, in the step of *AddUser*, only the indispensable keys are sent to authorized user, other than all the secret in SSE-1. □

5.2 The Performance Analysis

To make the comparison easier, we assume that each document in the collection has the same size. Compared to the other schemes, our scheme is as efficient as the others while it is more simple and easy to understand, which is also more appropriate utilized in cloud storage. The multi-user construction is very efficient on the server side during a query: when given a trapdoor, the server only needs to evaluate a PRP in order to determine if the user is revoked. If access control mechanisms were used instead for this step, a more expensive authentication protocol would be required for each search query in order to establish the identity of the querier.

The experiment is running on one PC configured with Intel Core i5 CPU 1.6 GHz and 4 GB RAM. The experimental result is an average of 10 trials. We implement the hash function with SHA-256 and the PRP with AES-256.

A. Computing Overhead

To generate array A, there are two PRP operations for each file that includes the keyword. To generate T, there are one PRP and one PRF operations for each keyword. The rest is XOR and filling operations which are very efficient. We set $|\Delta| = 10{,}000$ and evaluate the overhead of index generation with $|D(w)|$ from 10 to 100. As shown in Fig. 2(a), for $|D(w)| = 40$, the overhead of index generation is 1646 s. As the process of index generation is only performed in setup phase, it is considered to be a reasonable overhead.

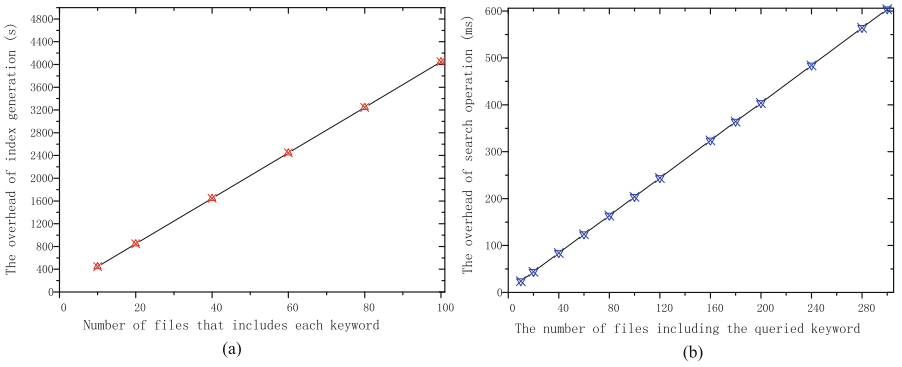

Fig. 2. (a) the overhead of index generation with $|\Delta| = 10{,}000$. (b) The search efficiency.

The search efficiency is related to the number of files including the queried keyword. There is one PRP operation for each file. We evaluate the overhead of search operation with $|D(w)|$ from 10 to 300. As shown in Fig. 2(b), for $|D(w)| = 100$, the overhead of search operation is 204 ms which is very efficient in practice.

To generate a search token, there are two PRP and one PRF operations. It is about 6.6 ms to generate a token.

B. Verification Efficiency

The verification mechanism is very efficient. It only needs two PRF operations for each verification. In [16], Kurosawa et al. designed a verification scheme using RSA accumulator. We compare the scheme of Kurosawa et al. with our scheme. As shown in Fig. 3, our scheme can be orders of magnitude faster than their scheme.

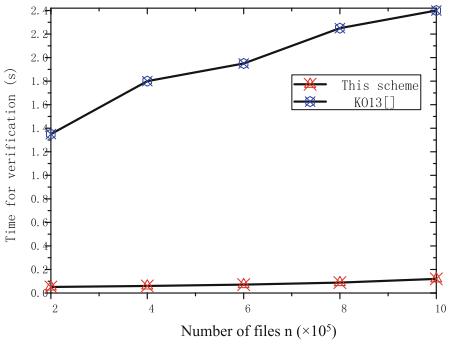

Fig. 3. The verification efficiency compared with KO13

From the above analysis, we can know that our scheme is very efficient and is very suitable for a large data set environment such as cloud storage.

Regarding the efficiency of multi-user, the main part of the overhead is the initialization phase, which requires each user to negotiate a session key with the server. The subsequent operations are highly efficient.

6 Conclusions

Searchable encryption is an important cryptographic primitive that is motivated by the popularity of cloud storage services like Google Desktop, Microsoft Skydrive etc. In the untrusted cloud storage server security model, we propose an efficient verifiable multi-user searchable symmetric encryption (VMSSE) scheme. The verification cost is efficient enough for practical use, i.e., it only depends on the corresponding search operation, regardless of the file collection size. Experimental result shows the efficiency and practicality of our scheme. The evaluation of the scheme on the real-world dataset is the future work.

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

References

1. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of SP 2000, pp. 44–55 (2000)
2. Goh, E.J.: Secure indexes. Cryptology ePrint Archive: Report 2003/216 (2003)
3. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005). https://dx.doi.org/10.1007/11496137_30
4. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of CCS 2006, pp. 79–88 (2006)
5. Kamara, S., Papamanthou, C.: Parallel and dynamic searchable symmetric encryption. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 258–274. Springer, Heidelberg (2013). https://dx.doi.org/10.1007/978-3-642-39884-1_22
6. Hahn, F., Kerschbaum, F.: Searchable encryption with secure and efficient updates. In: Proceedings of CCS 2014, pp. 310–320 (2014)
7. Naveed, M., Prabhakaran, M., Gunter, C.A.: Dynamic searchable encryption via blind storage. In: Proceedings of SP 2014, pp. 639–654 (2014)
8. Gajek, S.: Dynamic symmetric searchable encryption from constrained functional encryption. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 75–89. Springer, Cham (2016). https://dx.doi.org/10.1007/978-3-319-29485-8_5
9. Wang, C., Cao, N., Li, J., Ren, K., Lou, W.: Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Trans. Parallel Distrib. Syst. **23**(8), 1467–1479 (2012)
10. Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., Lou, W.: Fuzzy keyword search over encrypted data in cloud computing. In: Proceedings of INFOCOM 2010, pp. 441–445 (2010)
11. Wang, C., Ren, K., Yu, S., Urs, K.M.R.: Achieving usable and privacy-assured similarity search over outsourced cloud data. In: Proceedings of INFOCOM 2012, pp. 451–459 (2012)
12. Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M.-C., Steiner, M.: Highly-scalable searchable symmetric encryption with support for boolean queries. In: Canetti, R., Garay, Juan A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 353–373. Springer, Heidelberg (2013). https://dx.doi.org/10.1007/978-3-642-40041-4_20
13. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. In: Proceedings of INFOCOM 2011, pp. 829–837 (2011)
14. Fu, Z., Sun, X., Linge, N., Zhou, L.: Achieving effective cloud search services: multi-keyword ranked search over encrypted cloud data supporting synonym query. IEEE Trans. Consum. Electron. **60**(1), 164–172 (2014)
15. Wang, B., Yu, S., Lou, W., Hou, Y.T.: Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud. In: Proceedings of INFOCOM 2014, pp. 2112–2120 (2014)
16. Kurosawa, K., Ohtaki, Y.: UC-secure searchable symmetric encryption. In: Keromytis, Angelos D. (ed.) FC 2012. LNCS, vol. 7397, pp. 285–298. Springer, Heidelberg (2012). https://dx.doi.org/10.1007/978-3-642-32946-3_21
17. Sun, W., Wang, B., Cao, N., Li, M., Lou, W., Hou, Y.T., Li, H.: Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. IEEE Trans. Parallel Distrib. Syst. **25**(11), 3025–3035 (2014)
18. Sun, W., Liu, X., Lou, W., Hou, Y.T., Li, H.: Catch you if you lie to me: efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data. In: Proceedings of INFOCOM 2015, pp. 2110–2118 (2015)

19. Cheng, R., Yan, J., Guan, C., Zhang, F., Ren, K.: Verifiable searchable symmetric encryption from indistinguishability obfuscation. In: Proceedings of ASIACCS 2015, pp. 621–626 (2015)
20. Zheng, Q., Xu, S., Ateniese, G.: VABKS: verifiable attribute-based keyword search over outsourced encrypted data. In: Proc. of INFOCOM 2014, pp. 522–530 (2014)
21. Wang, J., Ma, H., Tang, Q., Li, J., Zhu, H., Ma, S., Chen, X.: A new efficient verifiable fuzzy keyword search scheme. J. Wireless Mobile Netw. Ubiquit. Comput. Dependable Appl. **3**(4), 61–71 (2012)
22. Wang, J., Ma, H., Li, J., Zhu, H., Ma, S., Chen, X.: Effcient verifiable fuzzy keyword search over encrypted data in cloud computing. Comput. Sci. Inf. Syst. **10**(2), 667–684 (2013)
23. Fu, Z., Shu, J., Sun, X., Linge, N.: Smart cloud search services: verifiable keyword-based semantic search over encrypted cloud data. IEEE Trans. Consum. Electron. **60**(4), 762–770 (2014)
24. Stefanov, E., Papamanthou, C., Shi, E.: Practical dynamic searchable symmetric encryption with small leakage. In: Proceedings of NDSS 2014 (2014)
25. Fiat, A., Naor, M.: Broadcast encryption. In: Proceedings of CRYPTO 1993, pp. 480–491 (1994)