Towards Efficient Verifiable Conjunctive Keyword Search for Large Encrypted Database

Jianfeng Wang[1], Xiaofeng Chen[1(✉)], Shi-Feng Sun[2,3], Joseph K. Liu[2], Man Ho Au[4], and Zhi-Hui Zhan[5]

[1] State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an, China
{jfwang,xfchen}@xidian.edu.cn
[2] Faculty of Information Technology, Monash University, Melbourne, Australia
{shifeng.sun,joseph.liu}@monash.edu
[3] Data 61, CSIRO, Melbourne, Australia
shifeng.sun@data61.csiro.au
[4] Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China
csallen@comp.polyu.edu.hk
[5] School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
cszhanzhh@scut.edu.cn

Abstract. Searchable Symmetric Encryption (SSE) enables a client to securely outsource large encrypted database to a server while supporting efficient keyword search. Most of the existing works are designed against the honest-but-curious server. That is, the server will be curious but execute the protocol in an honest manner. Recently, some researchers presented various verifiable SSE schemes that can resist to the malicious server, where the server may not honestly perform all the query operations. However, they either only considered single-keyword search or cannot handle very large database. To address this challenge, we propose a new verifiable conjunctive keyword search scheme by leveraging accumulator. Our proposed scheme can not only ensure verifiability of search result even if an empty set is returned but also support efficient conjunctive keyword search with sublinear overhead. Besides, the verification cost of our construction is independent of the size of search result. In addition, we introduce a sample check method for verifying the completeness of search result with a high probability, which can significantly reduce the computation cost on the client side. Security and efficiency evaluation demonstrate that the proposed scheme not only can achieve high security goals but also has a comparable performance.

Keywords: Searchable encryption · Verifiable search
Conjunctive keyword search · Accumulator

1 Introduction

Cloud computing, as a promising computing paradigm, offers seemly unbounded data storage capability and computation resource in a pay-as-you-go manner.

J. Lopez et al. (Eds.): ESORICS 2018, LNCS 11099, pp. 83–100, 2018.
https://doi.org/10.1007/978-3-319-98989-1_5

More and more resource-constrained users trend to move their own data into the cloud so that they can enjoy superior data storage services without data maintenance overheads locally. Despite its tremendous benefits, data outsourcing suffers from some security and privacy concerns [10,12,23]. One main challenge is the secrecy of outsourced data [11]. That is, the cloud server should not learn any useful information about the private data. For example, it has been reported recently that up to 87 million users' private information in Facebook is leaked to the Cambridge Analytica firm [26]. Although traditional encryption technology can guarantee the confidentiality of outsourced data, it heavily impedes the ability of searching over outsourced data [22].

A promising solution, called Searchable Symmetric Encryption (SSE), has attracted considerable interest from both academic and industrial community. SSE enables a data owner to outsource encrypted data to the cloud server while reserving searchability. Specifically, the data owner encrypts data with his private key before outsourcing and then stores the ciphertext associated with some metadata (e.g., indices) into the cloud server. Upon receiving a search token, the server performs the search operation and returns all the matched results to the user. The primitive of SSE has been widely studied [5,7,13,16,18,20,28]. Note that the above works only consider single-keyword search. To enhance the search availability, SSE supporting conjunctive keyword search has been extensively studied [3,6,17,27]. However, those schemes either suffer from low search efficiency or leak too much information on the queried keyword. Recently, Cash et al. [8] presented the first sublinear SSE scheme with support for conjunctive keyword search, named OXT. In their construction, the search complexity is linear with the matched document of the least frequent keyword, which makes it adaptable to the large database setting. There are different extensions of OXT subsequently. Sun et al. [29] extended this scheme to a multi-user setting, in which any authorized user can submit a search query and retrieve the matched documents. Following that, a fast decryption improvement has been given in [34]. Another multi-user scheme has been proposed in [19] by using a threshold approach. Zuo et al. [35] gave another extension supporting general Boolean queries. Note that all the above works are constructed in the honest-but-curious cloud model, where the cloud server is assumed to honestly perform all search operations.

In practice however, the cloud server may be malicious, since it may return an incorrect and/or incomplete search result for selfish reasons. According to Veritas [31], 28% of organizations admit to suffering from permanent data loss in the past three years. Thus, in order to resist to malicious server, verifiable SSE attracted more and more attention [2,4,9,25,30]. Azraoui et al. [2] proposed a publicly verifiable conjunctive keyword search scheme by integrating Cuckoo hashing and polynomial-based accumulators. The main idea is that the server performs search for each individual keyword and then computes the intersection of all the matched document subsets. The drawback of this method is that the search cost increases linear with the entire database size and reveals more intra-query leakage information, such as access pattern on each distinct keyword in a conjunction query. Recently, Bost et al. [4] proposed an efficient verifiable

SSE scheme by using verifiable hash table. Nevertheless, their solution can just support verifiable single keyword search. To our best knowledge, how to simultaneously achieve verifiability of the search result and efficient conjunctive keyword search on large database remains a challenge problem.

1.1 Our Contribution

In this paper, we focus on verifiable conjunctive keyword search scheme for large encrypted database. Our contribution can be summarized as follows:

- We propose a new verifiable conjunctive keyword search scheme based on accumulator, which can ensure correctness and completeness of search result even if an empty set is returned. The search cost of the proposed scheme depends on the number of documents matching with the least frequent keyword, which is independent of the entire database.
- The proposed scheme can achieve verifiability of the search result with constant size communication overhead between the client and server. That is, the server returns a constant size proof (i.e., *witness*) of the search result, and the client is able to check the correctness of the search result with a constant computation overhead. In addition, we introduce a sample check method to check the completeness of search result, which can achieve low false-positive by checking only a fraction of the search result and reduce the computation overhead on the client side.
- We present a formal security analysis of our proposed construction and also provide a thorough implementation of it on a real-world dataset. The experiment results demonstrate that our proposed construction can achieve the desired property with a comparable computation overhead.

1.2 Related Work

Song et al. [28] proposed the first SSE scheme, in which the document is encrypted word by word. As a result, the search cost grows linear with the size of database. Goh [16] constructed a search index for each document to improve the search efficiency. However, the search complexity is linear with the number of documents. Curtmola et al. [13] introduced the first sublinear SSE scheme, in which an inverted index is generated based on all distinct keyword. Kamara et al. [18] extended [13] to support dynamic search. Subsequently, a line of research focused on dynamic search [5,7,20]. In order to enrich query expressiveness, Golle et al. [17] presented the first secure conjunctive keyword search scheme. The search complexity is linear with the number of documents in the whole database. Later, some work [3,6,27] are proposed to enhance the search efficiency. However, those solutions can only support search on the structured data. In 2013, Cash et al. [8] proposed the first sublinear SSE scheme supporting conjunctive keyword search, which can be used to perform search on structured data as well as free text. The search complexity is only linear in the number of

documents which contain the least frequency keyword among the queried keywords. Thus, it is suitable to deploy in large-scale database setting. Recently, Sun et al. [29] extended this scheme to multi-user setting. That is, it not only support for the data owner but also an arbitrary authorized user to perform search.

Chai et al. [9] first considered verifiable search in malicious server model and presented a verifiable keyword search scheme based on the character tree. However, the proposed scheme can only support the exact keyword search in plaintext scenario. Kurosawa and Ohtaki [21] proposed the first verifiable SSE scheme to support the verifiability of the correctness of search result. Wang et al. [33] presented a verifiable fuzzy keyword search scheme, which can simultaneously achieve fuzzy keyword search and verifiability query. Recently, plenty of works [4,25,32] were dedicated to give a valid proof when the search result is an empty set. That is, when the search query has no matched result, the client can verify whether there is actually no matched result or it is a malicious behavior. Note that all the above-mentioned solutions are focused on single keyword search. Sun et al. [30] proposed a verifiable SSE scheme for conjunctive keyword search based on accumulator. However, their construction cannot work when the server purposely returned an empty set. Similarly, Azraoui et al. [2] presented a verifiable conjunctive keyword search scheme which support public verifiability of search result.

1.3 Organization

The rest of this paper is organized as follows. We present some preliminaries in Sect. 2. The verifiable conjunctive-keyword search scheme is proposed in Sect. 3. Section 4 presents the formal security analysis of the proposed construction. Its performance evaluation is given in Sect. 5. Finally, the conclusion is given in Sect. 6.

2 Preliminaries

In this section, we first present some notations (as shown in Table 1) and basic cryptographic primitives that are used in this work. We then present the formal security definition.

2.1 Bilinear Pairings

Let $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$ be two cyclic multiplicative groups of prime order p, and g be a generator of $\mathbb{G}$. A bilinear pairing is a mapping $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties:

1. Bilinearity: $e(u^a, v^b) = e(u, v)^{ab}$ for all $u, v \in \mathbb{G}$, and $a, b \in \mathbb{Z}_p$;
2. Non-degeneracy: $e(g, g) \neq 1$, where 1 represents the identity of group $\mathbb{G}_{\mathbb{T}}$;
3. Computability: there exists an algorithm to efficient compute $e(u, v)$ for any $u, v \in \mathbb{G}$.

Table 1. Notations.

Notations	Meaning
λ	Security parameter of the system
ind_i	Identifier of the i-th file
W_i	Keyword list of ind_i
$\mathrm{W} = \cup_{i=1}^{T} \mathrm{W}_i$	Keyword set of the whole database
$\mathsf{DB} = (\mathrm{ind}_i, \mathrm{W}_i)_{i=1}^{T}$	The representation of the whole database
$F : \{0,1\}^\lambda \times \{0,1\}^\lambda \mapsto \{0,1\}^\lambda$	A pseudo-random function
$F_p : \{0,1\}^\lambda \times \{0,1\}^\lambda \mapsto Z_p^\star$	A pseudo-random function mapped into group $Z_p^\star$
stag_w	Trapdoor of a given keyword w
Stag	Trapdoors of all the keywords in a search query
$\mathbf{e}_w$	File identifiers contain keyword w
$Acc(S)$	Accumulator value for the set S
$\mathrm{W}_{x,S}$	The proof of membership of element x for set S
$\hat{\mathrm{W}}_{x,S}$	The proof of non-membership of element x for set S
Td	The user's search token with a set of keywords
R_{w_1}	Search result contain the keyword w_1
R	Final result satisfied the search criteria
proof	The evidence of search result, i.e., Witness

2.2 Complexity Assumptions

Decisional Diffie-Hellman (DDH) Assumption. Let $a, b, c \in_R \mathbb{G}$ and g be a generator of $\mathbb{G}$. We say that the DDH assumption holds if there no probabilistic polynomial time algorithm can distinguish the tuple (g, g^a, g^b, g^{ab}) from (g, g^a, g^b, g^c) with non-negligible advantage.

q-Strong Diffie-Hellman (q-SDH) Assumption. Let $a \in_R \mathbb{Z}_p$ and g be a generator of $\mathbb{G}$. We say that the q-SDH assumption holds if given a $q+1$-tuple $(g, g^a, g^{a^2}, \ldots, g^{a^q})$, there no probabilistic polynomial time algorithm can output a pair $(g_1^{1/a+x}, x)$ with non-negligible advantage, where $x \in \mathbb{Z}_p^*$.

2.3 Security Definitions

Similar to [8,29], we present the security definition of our VSSE scheme by describing the leakage information $\mathcal{L}_{\mathrm{VSSE}}$, which refers to the maximum information that is allowed to learn by the adversary.

Let $\Pi = (\mathsf{VEDBSetup}, \mathsf{KeyGen}, \mathsf{TokenGen}, \mathsf{Search}, \mathsf{Verify})$ be a verifiable symmetric searchable encryption scheme, VEDB be the encrypted version database, $\mathcal{A}$ be an polynomial adversary and $\mathcal{S}$ be a simulator. Assuming that PK/MK be the public key/system master key, SK be the private key for a given authorized user. We define the security via the following two probabilistic experiments:

- $\mathbf{Real}_{\mathcal{A}}^{\Pi}(\lambda)$: $\mathcal{A}$ chooses a database DB, then the experiment runs $\mathsf{VEDBSetup}(\lambda, \mathsf{DB})$ and returns (VEDB, PK) to $\mathcal{A}$. Then, $\mathcal{A}$ generates the authorized

private key SK by running $\mathsf{KeyGen}(\mathrm{MK}, \mathbf{w})$ for the authorized keywords $\mathbf{w}$ of a given client, adaptively chooses a conjunctive query $\bar{\mathbf{w}}$ and obtains the search token Td by running $\mathsf{TokenGen}(\mathrm{SK}, \bar{\mathbf{w}})$. The experiment answers the query by running $\mathsf{Search}(\mathrm{Td}, \mathrm{VEDB}, \mathrm{PK})$ and $\mathsf{Verify}(\mathsf{R}_{w_1}, \mathsf{R}, \{\mathsf{proof}_i\}_{i=1,2}, \mathrm{VEDB}))$, then gives the transcript and the client output to $\mathcal{A}$. Finally, the adversary $\mathcal{A}$ outputs a bit $b \in \{0, 1\}$ as the output of the experiment.

- $\mathbf{Ideal}^{\Pi}_{\mathcal{A},\mathcal{S}}(\lambda)$: The experiment initializes an empty list $\mathbf{q}$ and sets a counter $i = 0$. Adversary $\mathcal{A}$ chooses a database DB, the experiment runs $\mathcal{S}(\mathcal{L}_{\mathrm{VSSE}}(\mathsf{DB}))$ and returns (VEDB, PK) to $\mathcal{A}$. Then, the experiment insert each query into $\mathbf{q}$ as $\mathbf{q}[i]$, and outputs all the transcript to $\mathcal{A}$ by running $\mathcal{S}(\mathcal{L}_{\mathrm{VSSE}}(\mathsf{DB}, \mathbf{q}))$. Finally, the adversary $\mathcal{A}$ outputs a bit $b \in \{0, 1\}$ as the output of the experiment.

We say the Π is $\mathcal{L}$-semantically secure verifiable symmetric searchable encryption scheme if for all probabilistic polynomial-time adversary $\mathcal{A}$ there exist an simulator $\mathcal{S}$ such that:

$$|\Pr[\mathbf{Real}^{\Pi}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Ideal}^{\Pi}_{\mathcal{A},\mathcal{S}}(\lambda)]| \leq negl(\lambda)$$

2.4 Leakage Function

The goal of searchable symmetric encryption is to achieve efficient search over encrypted data while revealing as little as possible private information. Following with [13,29], we describe the security of our VSSE scheme with leakage function $\mathcal{L}_{\mathrm{VSSE}}$.

For the sake of simplicity, given $\mathbf{q} = (\mathbf{s}, \mathbf{x})$ represent a sequence of query, where $\mathbf{s}(\mathbf{x})$ denotes *sterm* (*xterm*) array of the query $\mathbf{q}$. The i-th query is expressed as $\mathbf{q}[i] = (\mathbf{s}[i], \mathbf{x}[i])$. On input DB and $\mathbf{q}$, the leakage function $\mathcal{L}_{\mathrm{VSSE}}$ consists of the following leakage information:

- $K = \cup_{i=1}^{T} \mathrm{W}_i$ is the total number of keywords in the DB.
- $N = \sum_{i=1}^{T} |\mathrm{W}_i|$ is the total number of keyword/document identifier pairs in DB.
- $\bar{\mathbf{s}} \in \mathbb{N}^{|q|}$ is the equality pattern of the *sterm* set $\mathbf{s}$, where each distinct *sterm* is assigned by an integer according to its order of appearance in $\mathbf{s}$. i.e., if $\mathbf{s} = (a, b, c, b, a)$, then $\bar{\mathbf{s}} = (1, 2, 3, 2, 1)$.
- SP is the size pattern of the queries, which is the number of document identifiers matching the *sterm* in each query, i.e., $\mathrm{SP}[i] = |\mathsf{DB}(\mathbf{s}[i])|$. In addition, we define $\mathrm{SRP}[i] = \mathsf{DB}(\mathbf{s}[i])$ as the search result associated with the *sterm* of the i-th query, which includes the corresponding proof.
- RP is the result pattern, which consists of all the document identifiers matching the query $\mathbf{q}[i]$. That is, the intersection of the *sterm* with all *xterm* in the same query.
- IP is the conditional intersection pattern, which represents a matrix satisfies the following condition:

$$\mathrm{IP}[i, j, \alpha, \beta] = \begin{cases} \mathsf{DB}(\mathbf{s}[i]) \cap \mathsf{DB}(\mathbf{s}[j]), & \text{if } \mathbf{s}[i] \neq \mathbf{s}[j] \text{ and } \mathbf{x}[i, \alpha] = \mathbf{x}[j, \beta] \\ \emptyset, & \text{otherwise} \end{cases}$$

3 Verifiable Conjunctive Keyword Search Scheme Based on Accumulator

In this section, we firstly introduce some building blocks adopted in the proposed construction. Then, we present the proposed verifiable conjunctive keyword search scheme in detail.

3.1 Building Block

Bilinear-Map Accumulators. We briefly introduce the accumulator based on bilinear maps supporting non-membership proofs [14]. It can provide a short proof of (non)-membership for any subset that (not) belong to a given set. More specifically, given a prime p, it can accumulates a given set $S = \{x_1, x_2, \cdots, x_N\} \subset \mathbb{Z}_p$ into an element in $\mathbb{G}$. Given the public parameter of the accumulator $\mathrm{PK} = (g^s, \ldots, g^{s^t})$, the corresponding private key is $s \in \mathbb{Z}_p$. The accumulator value of S is defined as

$$Acc(S) = g^{\prod_{i=1}^{N}(x_i+s)}$$

Note that $Acc(S)$ can be reconstructed by only knowing set S and PK in polynomial interpolation manner. The proof of membership for a subset $S_1 \subseteq S$ is the witness $\mathrm{W}_{S_1,S} = g^{\prod_{x\in S-S_1}(x+s)}$, which shows that a subset S_1 belongs to the set S.

Using the witness $\mathrm{W}_{S_1,S}$, the verifier can determine the membership of subset $\mathrm{W}_{\mathrm{S}_1}$ by checking the following equation $e(\mathrm{W}_{S_1,S}, g^{\prod_{x\in S_1}(x+s)}) = e(Acc(S), g)$[1].

To verify some element $x_i \notin \mathrm{S}$, the witness consists of a tuple $\hat{\mathrm{W}}_{x_i,S} = (w_{x_i}, u_{x_i}) \in \mathbb{G} \times \mathbb{Z}_p^*$ satisfying the following requirement:

$$u_x \neq 0$$
$$(x_i + s) \mid [\prod_{x\in S}(x+s) + u_{x_i}]$$
$$w_x^{x_i+s} = Acc(\mathrm{S}) \cdot g^{u_{x_i}}$$

In particular, let $f_S(s)$ denote the product in the exponent of $Acc(S)$, i.e., $f_S(s) = \prod_{x\in S}(x+s)$, any $y \notin S$, the unique non-membership witness $\hat{\mathrm{W}}_{x_i,S} = (w_y, u_y)$ can be denoted as:

$$u_y = -f_S(-y) \mod p = -\prod_{x\in S}(x-y) \mod p$$
$$w_y = g^{[f_S(s)-f_S(-y)]/(y+s)}.$$

The verification algorithm in this case checks that

$$e(w_y, g^y \cdot g^s) = e(Acc(S) \cdot g^{u_y}, g)$$

[1] Particularly, for a given element x, the corresponding witness is $\mathrm{W}_{x,S} = g^{\prod_{\hat{x}\in S:\hat{x}\neq x}(\hat{x}+s)}$, the verifier can check that $e(\mathrm{W}_{x,S}, g^x \cdot g^s) = e(Acc(S), g)$.

3.2 The Concrete Construction

In this section, we present a concrete verifiable SSE scheme which mainly consists of five algorithms $\Pi = (\mathsf{VEDBSetup}, \mathsf{KeyGen}, \mathsf{TokenGen}, \mathsf{Search}, \mathsf{Verify})$. We remark that each keyword w should be a prime in our scheme. This can be easily facilitated by using a hash-to-prime hash function such as the one used in [29]. For simplicity, we omit this "hash-to-prime" statement in the rest of this paper and simply assume that each w is a prime number. The details of the proposed scheme are given as follows.

- $\mathsf{VEDBSetup}(1^\lambda, \mathsf{DB})$: the data owner takes as input of security parameter λ, a database $\mathsf{DB} = (\mathrm{ind}_i, \mathrm{W}_i)_{i=1}^T$, and outputs the system master key MK, public key PK and the Verifiable Encrypted Database (VEDB). As shown in the following Algorithm 1.

Algorithm 1. $\mathsf{VEDBSetup}(\lambda, \mathsf{DB})$

Input: $\mathsf{DB} = (\mathrm{ind}_i, \mathrm{W}_i)_{i=1}^T$
Output: VEDB, MK, PK

The data owner selects two big primes p, q, random keys K_X, K_I, K_Z for PRF F_P, K_s for PRF F, and computes $n = pq$.
The data owner randomly selects $g \xleftarrow{R} \mathbb{G}$, $g_1, g_2, g_3 \xleftarrow{R} \mathbb{Z}_n^*$ and secret key $sk = s$, then computes $pk = (g^s, \ldots, g^{s^t})$, where t is an upper bound on the number of the cardinality. The system public/master key pair are defined as: MK $\leftarrow \{K_X, K_I, K_Z, K_S, p, q, s\}$,PK $\leftarrow \{n, g, pk\}$.
TSet, XSet, Stag $\leftarrow \phi$;
for $w \in \mathrm{W}$ **do**
    $\mathbf{e}_w \leftarrow \phi$; $c \leftarrow 1$
    $K_e \leftarrow F(K_S, w)$; $\mathrm{stag}_w \leftarrow F(K_S, g_1^{1/w} \mod n)$
    Stag$\leftarrow$ Stag $\cup$ $\{\mathrm{H}(\mathrm{stag}_w)\}$, where $\mathrm{H}(\cdot)$ is a secure hash function, i.e., SHA-256.
    for ind $\in \mathsf{DB}(w)$ **do**
        $\mathrm{xind} \leftarrow F_p(K_I, \mathrm{ind})$; $z \leftarrow F_p(K_Z, g_2^{1/w} \mod n \parallel c)$
        $l \leftarrow F(stag_w, c)$; $e \leftarrow \mathrm{Enc}(K_e, \mathrm{ind})$; $y \leftarrow \mathrm{xind} \cdot z^{-1}$
        $\mathrm{TSet}[l] = (e, y)$; $\mathbf{e}_w \leftarrow \mathbf{e}_w \cup \{e\}$;
        $\mathrm{xtag}_w \leftarrow g^{F_p(K_X, g_3^{1/w} \mod n) \cdot \mathrm{xind}}$; XSet $\leftarrow$ XSet $\cup \{\mathrm{xtag}_w\}$
        $c \leftarrow c + 1$
    end for
    $l \leftarrow F(stag_w, 0)$
    $Acc(\mathbf{e}_w) \leftarrow g^{\prod_{\mathbf{e}_i \in \mathbf{e}_w} (\mathbf{e}_i + s)}$
    $\mathrm{TSet}[l] = (Acc(\mathbf{e}_w), H(K_e, Acc(\mathbf{e}_w)))$
end for
Compute accumulator values:
- $Acc(\mathrm{XSet}) \leftarrow g^{\prod_{\mathrm{xtag} \in \mathrm{XSet}} (\mathrm{xtag} + s)}$
- $Acc(\mathrm{Stag}) \leftarrow g^{\prod_{\mathrm{stag} \in \mathrm{Stag}} (\mathrm{stag} + s)}$

Set VEDB $\leftarrow \{\mathrm{TSet}, \mathrm{XSet}, \mathrm{Stag}, Acc(\mathrm{Stag}), Acc(\mathrm{XSet})\}$
return {VEDB, MK, PK}

- $\mathsf{KeyGen}(\mathrm{MK}, \mathbf{w})$: Assume that an authorized user is allowed to perform search over an authorized keywords $\mathbf{w} = \{w_1, w_2, \ldots, w_N\}$, the data owner computes $sk_{\mathbf{w}}^{(i)} = (g_i^{1/\prod_{j=1}^{N} w_j} \mod n)$ for $i \in \{1, 2, 3\}$ and generates search key $sk_{\mathbf{w}} = (sk_{\mathbf{w}}^{(1)}, sk_{\mathbf{w}}^{(2)}, sk_{\mathbf{w}}^{(3)})$, then sends the authorized private key $\mathrm{SK} = (K_S, K_X, K_Z, K_T, sk_{\mathbf{w}})$ to the authorized user.

Algorithm 2. $\mathsf{TokenGen}(\mathrm{SK}, \bar{\mathsf{w}})$

Input: SK, $\bar{\mathsf{w}}$
Output: Td
1: $\mathrm{Td}, \mathrm{xtoken} \leftarrow \phi$
2: $\mathrm{stag} \leftarrow F(K_S, (sk_{\bar{\mathsf{w}}}^{(1)})^{\prod_{w \in \mathrm{kw} \setminus w_1} w} \mod n) = F(K_S, g_1^{1/w_1} \mod n)$
3: **for** $c = 1, 2, \ldots$ until the server stops **do**
4: **for** $i = 2, \ldots, d$ **do**
5: $\mathrm{xtoken}[c, i] \leftarrow g^{F_p(K_Z, (sk_{\bar{\mathsf{w}}}^{(2)})^{\prod_{w \in \bar{\mathsf{w}} \setminus w_1} w} \mod n \| c) \cdot F_p(K_X, (sk_{\bar{\mathsf{w}}}^{(3)})^{\prod_{w \in \bar{\mathsf{w}} \setminus w_i} w}) \mod n} = g^{F_p(K_Z, g_2^{1/w_1} \mod n \| c) \cdot F_p(K_X, g_3^{1/w_i} \mod n)}$
6: **end for**
7: $\mathrm{xtoken}[c] = \mathrm{xtoken}[c, 2], ..., \mathrm{xtoken}[c, d]$
8: **end for**
9: $\mathrm{Td} \leftarrow (\mathrm{stag}, \mathrm{xtoken}[1], \mathrm{xtoken}[2], ...)$
10: **return** the search token Td

- $\mathsf{TokenGen}(\mathrm{SK}, \bar{\mathsf{w}})$: Suppose that an authorized user wants to perform a conjunctive query $\bar{\mathsf{w}} = (w_1, \ldots, w_d)$, where $d \leq N$. Let *sterm* be the least frequent keyword in a given search query. Without loss of generality, we assume that the *sterm* of the query $\bar{\mathsf{w}}$ is w_1, then the search token *st* of the query is generated with Algorithm 2 and sent to the cloud server.
- $\mathsf{Search}(\mathrm{Td}, \mathrm{VEDB}, \mathrm{PK})$: Upon receiving the search token Td, the cloud server firstly performs a single keyword search with stag, and then returns all the document identities that matching the search criteria along with the corresponding proof. The detail is shown in Algorithm 3.
- $\mathsf{Verify}(\mathsf{R}_{w_1}, \mathsf{R}, \{\mathsf{proof}_i\}_{i=1,2}, \mathrm{VEDB}))$: To verify the integrity of search result, the user checks the validity in terms of both correctness and completeness. At the end of the verification, it outputs *Accept* or *Reject* that represents the server is honest or malicious. Precisely, the algorithm is performed according to the following cases:

Case 1: When R_{w_1} is an empty set, it implies that there is no matched tuples in the TSet. The server returns the corresponding verification information $(\mathsf{proof}_1, Acc(\mathrm{Stag}))$. Let $f_{\mathrm{Stag}}(s)$ denote the product in the exponent of $Acc(\mathrm{Stag})$, that is $f_{\mathrm{Stag}}(s) = \prod_{x \in \mathrm{Stag}}(x + s)$. The non-membership witness $\hat{\mathrm{W}}_{stag_{w_1}, \mathrm{Stag}} = \mathsf{proof}_1 = (w_{stag_{w_1}}, u_{stag_{w_1}})$ can be denoted as:

$$u_{stag_{w_1}} = -f_{\mathrm{Stag}}(-stag_{w_1}) \mod p = -\prod_{x \in \mathrm{Stag}} (x - stag_{w_1}) \mod p$$

$$w_{stag_{w_1}} = g^{[f_{\mathrm{Stag}}(s) - f_{\mathrm{Stag}}(-stag_{w_1})]/(stag_{w_1} + s)}.$$

Algorithm 3. Search(Td, VEDB, PK)

Input: Td, VEDB, PK
Output: $\mathsf{R}_{w_1}, \mathsf{R}, \mathsf{proof}$

$\mathsf{R}_{w_1}, \mathsf{R}, \mathsf{B} \leftarrow \phi$; $\mathsf{proof} \leftarrow NULL$
Phase 1 :
$l \leftarrow F(stag_{w_1}, 0)$; $(Acc(\mathbf{e}_w), H(K_e, Acc(\mathbf{e}_w)) \leftarrow \mathrm{TSet}[l])$
for $c = 1, 2, \cdots$ **do**
  $l \leftarrow F(stag_{w_1}, c)$; $(e_c, y_c) \leftarrow \mathrm{TSet}[l]$; $\mathsf{R}_{w_1} \leftarrow \mathsf{R}_{w_1} \cup \{e_c\}$
end for
if $\mathsf{R}_{w_1} = \phi$ **then**
  $\mathsf{proof}_1 \leftarrow \hat{\mathrm{W}}_{\mathrm{stag}_{\mathrm{w}_1},\mathrm{Stag}}$ // Refer to Case 1 below
return $(\mathsf{proof}_1, Acc(\mathrm{Stag}))$ and exit
else
  $\mathsf{proof}_1 \leftarrow \mathrm{W}_{\mathsf{R}_{w_1},\mathrm{Stag}}$ // Refer to Case 2 and 3 below

end if
Phase 2 :
for $c = 1, \cdots, |\mathsf{R}_{w_1}|$ **do**
  for $i = 2, \cdots, d$ **do**
    $b[c, i] \leftarrow \mathrm{xtoken}[c, i]^{y_c}$
  end for
  if $\forall i = 2, ..., d : b[c, i] \in \mathrm{XSet}$ **then**
    $\mathsf{R} \leftarrow \mathsf{R} \cup \{e_c\}$; $\mathsf{B} \leftarrow \mathsf{B} \cup \{b[c, i]\}$
  end if
end for
if $\mathbf{R} \neq \phi$ **then**
  $\mathsf{proof}_2 \leftarrow \mathrm{W}_{\mathsf{B},\mathrm{XSet}}$ // Refer to Case 3 below

end if
$\mathsf{proof} \leftarrow \{Acc(\mathrm{Stag}), Acc(\mathrm{XSet}), \{\mathsf{proof}_i\}_{i=1,2}\}$
return $(\mathsf{R}_{w_1}, \mathsf{R}, \mathsf{proof})$

The user checks the equalities $u_{stag_{w_1}} \neq 0$ and $e(w_{stag_{w_1}}, g^{stag_{w_1}} \cdot g^s) = e(Acc(\mathrm{Stag}) \cdot g^{u_{stag_{w_1}}}, g)$. If pass, the process terminates and outputs *Accept*.

Case 2: When R_{w_1} is not an empty set and R is an empty one, the cloud claims that there is no tuple satisfied the query condition. To verify the correctness of the search result, the user firstly verifies the integrity of R_{w_1}. Then the user randomly selects some elements from the R_{w_1} and requires the cloud to feedback the corresponding non-member proof for the query condition. The detail of the process is described as follows:

Step 1: The user checks the integrity of R_{w_1} with the following equality:

$$e(\mathrm{W}_{\mathsf{R}_{w_1},\mathrm{Stag}}, g^{\prod_{x \in \mathsf{R}_{w_1}} (x+s)}) = e(Acc(\mathrm{Stag}), g)$$

If it holds, then go to Step 2.

Step 2: The user randomly selects k elements from R_{w_1} and checks the membership with the query condition. The detail is as shown in Algorithm 4.

Case 3: When both R_{w_1} and R are non-empty set, the verifications of R_{w_1} and R is very similar to that of Case 2. The difference is that the user randomly selects k elements from $\mathsf{R}_{w_1} \setminus \mathsf{R}$. For the detail of verifying process, please refer to the Case 2.

Algorithm 4. $\mathsf{Verify}(\mathsf{R}_{w_1}(\mathsf{R}), \bar{\mathsf{w}}, \mathrm{XSet})$

Input: $(\mathbf{R}_{w_1}(\mathsf{R}), \bar{\mathsf{w}}, \mathrm{XSet})$
Output: *Accept* or *Reject*

User side :
The user randomly selects $\{\mathrm{ind}_1, \cdots, \mathrm{ind}_k\} \in \mathsf{R}_{w_1}$
for $i = 1, \cdots, k$ **do**
  $\mathrm{xtag}[i] \leftarrow \phi$
  $\mathrm{xind} \leftarrow F_p(K_I, \mathrm{ind}_i)$
  for $j = 2, \cdots, d$ **do**
    $\mathrm{xtag}[i,j] \leftarrow g^{F_p(K_X, g_3^{1/w_j} \bmod n) \cdot \mathrm{xind}}$
    $\mathrm{xtag}[i] \leftarrow \mathrm{xtag}[i] \cup \mathrm{xtag}[i,j]$
  end for
end for
The user sends the $\mathrm{xtag} \leftarrow \{\mathrm{xtag}[1], \cdots, \mathrm{xtag}[k]\}$ to the cloud.

Cloud side :
for all $\mathrm{xtag}[i] \in \mathrm{xtag}$ **do**
  for $j = 2, \cdots, d$ **do**
    if $\mathrm{xtag}[i,j] \notin \mathrm{XSet}$ **then**
      $\mathsf{proof}[\mathrm{i}] \leftarrow \hat{\mathrm{W}}_{\mathrm{xtag}[i,j],\mathrm{XSet}}$ and **break**
    end if
  end for
  $\mathsf{proof} \leftarrow \mathsf{proof} \cup \{\mathsf{proof}[\mathrm{i}]\}$
end for

User side :
for all $\mathsf{proof}[\mathrm{i}] \in \mathsf{proof}$ **do**
  if $\mathsf{proof}[\mathrm{i}]$ holds **then**
    return *Accept*
  else
    return *Reject*
  end if
end for

Remark 1. Note that we accumulate all $ind \in \mathsf{DB}[w]$ into a accumulator value $Acc(\mathbf{e}_w)$, it can be used to ensure the completeness of search result. In order to associate with the corresponding $stag$, it is assigned into TSet indexed by $l = F(stag, 0)$. More precisely, a tuple consisting of $(Acc(e_w), H(K_e, Acc(e_w)))$

is stored in TSet$[l]$, where $K_e = F(K_S, w)$. The integrity of $Acc(e_w)$ can be verified by reconstructing the keyed-hash value.

Remark 2. Inspired by [1], we introduce a sample check method to verify the completeness of search result. We determine the completeness of search result by randomly choosing a fraction of the matched documents. More specifically, we just randomly choose k element in $\mathsf{R}_{w_1} \setminus \mathsf{R}$ for completeness checking. Let P_X be the probability that the user detects the misbehavior of the server. We have $P_X = 1 - \frac{n-t}{n} \cdot \frac{n-1-t}{n-1} \cdot \frac{n-2-t}{n-2} \cdot \ldots \cdot \frac{n-k+1-t}{n-k+1}$, where n is the size of $\mathsf{R}_{w_1} \setminus \mathsf{R}$, t is the missed documents of search result. Since $\frac{n-i-t}{n-i} \geq \frac{n-i-1-t}{n-i-1}$, P_X satisfies that $1 - (\frac{n-t}{n})^k \leq P_X \leq 1 - (\frac{n-k+1-t}{n-k+1})^k$. Similar to scheme [1], once the percentage of the missed documents is determined, the misbehavior can be detected with a certain high probability by checking a certain number of documents that is independent of the size of the dataset. For example, in order to achieve a 99% probability, 65, 21 and 7 documents should be checked when $t = 10\% \cdot n$, $t = 20\% \cdot n$ and $t = 50\% \cdot n$, respectively.

Note that the server can generate a non-membership proof for each document. So we need to perform multiple times non-membership verification to ensure completeness of the search result. Although the sample check method can greatly reduce the verification overhead at the expense of low false positive, it remains a drawback in our proposed scheme. Thus, one interesting question is whether there is an efficient way to achieve non-membership verification in a batch manner.

4 Security and Efficiency Analysis

4.1 Security Analysis

In this section, we present the security of our proposed VSSE scheme with simulation-based approach. Similar to [8,29], we first provide a security proof of our VSSE scheme against non-adaptive attacks and then discuss the adaptive one.

Theorem 1. *The proposed scheme is $\mathcal{L}$-semantically secure VSSE scheme where $\mathcal{L}_{\text{VSSE}}$ is the leakage function defined as before, assuming that the q-SDH assumption holds in $\mathbb{G}$, that F, F_P are two secure PRFs and that the underlying (Enc, Dec) is an IND-CPA secure symmetric encryption scheme.*

Theorem 2. *Let $\mathcal{L}_{\text{VSSE}}$ be the leakage function defined as before, the proposed scheme is $\mathcal{L}$-semantically secure VSSE scheme against adaptive attacks, assuming that the q-SDH assumption holds in $\mathbb{G}$, that F, F_P are two secure PRFs and that the underlying (Enc, Dec) is an IND-CPA secure symmetric encryption scheme.*

Due to space constraints, we will provide the detailed proofs in our full version.

4.2 Comparison

In this section, we compare our scheme with Cash et al.'s scheme [8] and Sun et al.'s scheme [29]. Firstly, all of the three schemes can support conjunctive keyword search. In particular, Our scheme can be seen as an extension from Cash et al. scheme and Sun et al. scheme, which supports verifiability of conjunctive query based on accumulator. All the three schemes have almost equal computation cost in search phase. Secondly, our scheme and Sun et al.'s scheme support the authorized search in multi-user setting. Note that our scheme can support verifiability of search result. Although it requires some extra computation overhead to construct verifiable searchable indices, we remark that the work is only done once.

To achieve the verification functionality in our scheme, the server requires to generate the corresponding proofs for the search result. Here we assume that the search result R is not empty. In this case, the proofs contain three parts. The first part is used to verify both the correctness and completeness of the search result for the sterm R_{w_1}. The second part and the third part are used to verify the correctness and completeness of the conjunctive search result R, respectively. The proofs generation are related to the size of database, but they can be done in parallel on the powerful server side. In contrast, it is efficient for the client to verify the corresponding proofs.

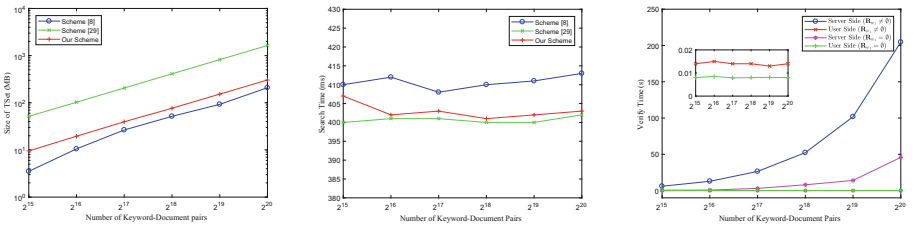
Table 2. Performance comparison

Schemes	Scheme [8]	Scheme [29]	Our scheme
Query-type	Conjunctive	Conjunctive	Conjunctive
Multi-user	No	Yes	Yes
Verification	No	No	Yes
EDBSetup	$\|\mathrm{DB}\|E$	$(\|\mathrm{DB}\|+2\|\mathrm{W}\|)E$	$(\|\mathrm{DB}\|+3\|\mathrm{W}\|+2)E$
TokenGen	$\|\mathsf{R}_{w_1}\|(d-1)E$	$(\|\mathsf{R}_{w_1}\|(d-1)+(d+1))E$	$(\|\mathsf{R}_{w_1}\|(d-1)+(d+1))E$
Search	$\|\mathsf{R}_{w_1}\|(d-1)E$	$\|\mathsf{R}_{w_1}\|(d-1)E$	$\|\mathsf{R}_{w_1}\|(d-1)E$
Verify (Server)	-	-	$(\|\mathrm{DB}\|-\|\mathrm{R}\|-1)E+k(\|\mathrm{DB}\|-2)E$
Verify (User)	-	-	$2E+4P+k(2E+2P)$

Table 2 provides the detailed comparison of all the three schemes. For the computation overhead, we mainly consider the expensive operations, like exponentiation and pairing operations. We denote by E an exponentiation, P a computation of pairing, $|\mathsf{R}_{w_1}|$ the size of search result for *sterm* w_1, $|\mathrm{DB}|$ the number of whole keyword-document pairs, $|\mathsf{R}|$ the size of conjunctive search results, d the number of queried keywords, k the number of selected identifiers which are used for completeness verification.

5 Performance Evaluation

In this section, we present an overall performance evaluation of the proposed scheme. First, we give a description of the prototype to implement the VEDB generation, the query processing and proof verification. Here the biggest challenge is how to generate the VEDB. Second, we analyze the experimental results and compare them with [8,29].



(a) (b) (c)

Fig. 1. The performance comparison among three schemes. (a) Storage cost of VEDBSetup. (b) Time cost of Search. (c) Time cost of Verify

Prototype. There are three main components in the prototype. The first and most important one is for VEDB generation. The second one is for query processing and the last one is for the verification of the search result. We leverage OpenSSL and PBC libraries to realize the cryptographic primitives. Specifically, we use Type 1 pairing function in PBC library for pairing, HMAC for PRFs, AES in CTR model to encrypt the indices in our proposed scheme and in scheme [8], ABE to encrypt the indices in Sun et al.'s scheme [29].

In order to generate the VEDB efficiently, we use four LINUX machines to establish a distributed network. Two of them are service nodes with Intel Core I3-2120 processors running at 3.30 GHz, 4G memory and 500G disk storage. The other two are compute nodes with Intel Xeon E5-1603 processors running at 2.80 GHz, 16G memory and 1T disk storage. One of the service nodes is used for storing the whole keyword-document pairs. To enhance the retrieving efficiency, the pairs are stored in a key-value database, i.e., Redis database. The other service node is used to store the VEDB which is actually in a MySQL database. In order to improve the efficiency of generating VEDB, both of the two compute nodes are used to transform the keyword-document pairs to the items in VEDB. The experiments of server side is implemented on a LINUX machine with Intel Core I3-2120 processors running at 3.30 GHz, 4G memory and 500G disk storage. In order to improve the search efficiency, the TSet is transformed to a Bloom Filter [24], which can efficiently test membership with small storage. The verification on the user side is also implemented on a LINUX machine with Intel Core I3-2120 processors running at 3.30 GHz, 4G memory and 500G disk storage.

Experimental Results. We implement the three compared schemes on the real-world dataset from Wikimedia Download [15]. The number of documents, distinct keywords and distinct keyword-document pairs are $7.8 * 10^5$, $4.0 * 10^6$ and $6.2 * 10^7$, respectively.

As shown in Fig. 1, we provide the detailed evaluation results by comparing the proposed scheme with [7] and [29]. In the phrase of system setup, we mainly focus on the size of TSet(As the XSet is almost the same in all three schemes). The size of TSet in our scheme is slightly larger than that of [8], because the TSet in our scheme contains the accumulators of each keywords, $\{Acc(\mathbf{e}_w)\}_{w \in W}$. However, the size of TSet in [29] is larger than both of our scheme and Cash et al.'s scheme as the document identities are encrypted by the public encryption scheme (i.e., ABE) in [29]. The search complexity of all the three schemes depends only on the least frequent keyword, i.e., *sterm*. It is slightly inefficient for the scheme [8] because of the different structure of TSet. In addition, we measure the verification cost for our scheme in two cases, $\mathsf{R}_{w_1} = \emptyset$ and $\mathsf{R}_{w_1} \neq \emptyset$. Obviously, it is very efficient for the client to verify the proofs given by the server. Although the proof generation on the server side is not so efficient, it can be generated in parallel for the powerful server. Besides, it is unnecessary for the server to give the search result with the proofs at the same time. Alternatively, the server can send the proofs slightly behind the search result. The experimental results show that the proposed scheme can achieve security against malicious server while maintaining a comparable performance.

6 Conclusion

In this paper, we focus on the verifiable conjunctive search of encrypted database. The main contribution is to present a new efficient verifiable conjunctive keyword search scheme based on accumulator. Our scheme can simultaneously achieve verifiability of search result even when the search result is empty and efficient conjunctive keyword search with search complexity proportional to the matched documents of the least frequent keyword. Moreover, the communication and computation cost of client is constant for verifying the search result. We provide a formal security proof and thorough experiment on a real-world dataset, which demonstrates our scheme can achieve the desired security goals with a comparable efficiency. However, our scheme needs to perform non-membership verification for each document individually. How to design a variant of accumulator that can provide a constant non-membership proof for multiple elements is an interesting problem.

Acknowledgement. This work is supported by National Key Research and Development Program of China (No. 2017YFB0802202), National Natural Science Foundation of China (Nos. 61702401, 61572382, 61602396 and U1636205), China 111 Project (No. B16037), China Postdoctoral Science Foundation (No. 2017M613083), Natural Science Basic Research Plan in Shaanxi Province of China (No. 2016JZ021), the Research Grants Council of Hong Kong (No. 25206317), and the Australian Research Council (ARC) Grant (No. DP180102199).

References

1. Ateniese, G., et. al.: Provable data possession at untrusted stores. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS 2007, pp. 598–609. ACM (2007)
2. Azraoui, M., Elkhiyaoui, K., Önen, M., Molva, R.: Publicly verifiable conjunctive keyword search in outsourced databases. In: Proceedings of 2015 IEEE Conference on Communications and Network Security, CNS 2015, pp. 619–627. IEEE (2015)
3. Ballard, L., Kamara, S., Monrose, F.: Achieving efficient conjunctive keyword searches over encrypted data. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 414–426. Springer, Heidelberg (2005). https://doi.org/10.1007/11602897_35
4. Bost, R., Fouque, P., Pointcheval, D.: Verifiable dynamic symmetric searchable encryption: optimality and forward security. IACR Cryptology ePrint Archive, p. 62 (2016). http://eprint.iacr.org/2016/062
5. Bost, R., Minaud, B., Ohrimenko, O.: Forward and backward private searchable encryption from constrained cryptographic primitives. In: Proceedings of the 24th ACM Conference on Computer and Communications Security, CCS 2017, pp. 1465–1482. ACM (2017)
6. Byun, J.W., Lee, D.H., Lim, J.: Efficient conjunctive keyword search on encrypted data storage system. In: Atzeni, A.S., Lioy, A. (eds.) EuroPKI 2006. LNCS, vol. 4043, pp. 184–196. Springer, Heidelberg (2006). https://doi.org/10.1007/11774716_15
7. Cash, D., et al.: Dynamic searchable encryption in very-large databases: data structures and implementation. In: Proceedings of the 21st Annual Network and Distributed System Security Symposium, NDSS 2014. The Internet Society (2014)
8. 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, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 353–373. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_20
9. Chai, Q., Gong, G.: Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers. In: Proceedings of IEEE International Conference on Communications, ICC 2012, pp. 917–922. IEEE (2012)
10. Chen, X., Li, J., Ma, J., Tang, Q., Lou, W.: New algorithms for secure outsourcing of modular exponentiations. IEEE Trans. Parallel Distrib. Syst. **25**(9), 2386–2396 (2014)
11. Chen, X., Li, J., Weng, J., Ma, J., Lou, W.: Verifiable computation over large database with incremental updates. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8712, pp. 148–162. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11203-9_9
12. Chu, C., Zhu, W.T., Han, J., Liu, J.K., Xu, J., Zhou, J.: Security concerns in popular cloud storage services. IEEE Pervasive Comput. **12**(4), 50–57 (2013)
13. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, pp. 79–88. ACM (2006)
14. Damgård, I., Triandopoulos, N.: Supporting non-membership proofs with bilinear-map accumulators. IACR Cryptology ePrint Archive 2008/538 (2008). http://eprint.iacr.org/2008/538

15. Wikimedia Foundation: Wikimedia downloads. https://dumps.wikimedia.org. Accessed 18 Apr 2018
16. Goh, E.: Secure indexes. IACR Cryptology ePrint Archive 2003/216 (2003). http://eprint.iacr.org/2003/216
17. Golle, P., Staddon, J., Waters, B.: Secure conjunctive keyword search over encrypted data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24852-1_3
18. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) Proceedings of the 19th ACM Conference on Computer and Communications Security, CCS 2012, pp. 965–976. ACM (2012)
19. Kasra Kermanshahi, S., Liu, J.K., Steinfeld, R.: Multi-user cloud-based secure keyword search. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017. LNCS, vol. 10342, pp. 227–247. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60055-0_12
20. Kim, K.S., Kim, M., Lee, D., Park, J.H., Kim, W.: Forward secure dynamic searchable symmetric encryption with efficient updates. In: Proceedings of the 24th ACM Conference on Computer and Communications Security, CCS 2017, pp. 1449–1463. ACM (2017)
21. Kurosawa, K., Ohtaki, Y.: UC-secure searchable symmetric encryption. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 285–298. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32946-3_21
22. Liu, J.K., Au, M.H., Susilo, W., Liang, K., Lu, R., Srinivasan, B.: Secure sharing and searching for real-time video data in mobile cloud. IEEE Netw. **29**(2), 46–50 (2015)
23. Liu, J.K., Liang, K., Susilo, W., Liu, J., Xiang, Y.: Two-factor data security protection mechanism for cloud storage system. IEEE Trans. Comput. **65**(6), 1992–2004 (2016)
24. Nikitin, A.: Bloom filter scala. https://alexandrnikitin.github.io/blog/bloom-filter-for-scala/. Accessed 10 Apr 2018
25. Ogata, W., Kurosawa, K.: Efficient no-dictionary verifiable searchable symmetric encryption. In: Kiayias, A. (ed.) FC 2017. LNCS, vol. 10322, pp. 498–516. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70972-7_28
26. Ruvic, R.D.: Facebook says up to 87M people affected in Cambridge Analytica data-mining scandal. http://www.abc.net.au/news/2018-04-05/facebook-raises-cambridge-analytica-estimates/9620652. Accessed 10 Apr 2018
27. Ryu, E., Takagi, T.: Efficient conjunctive keyword-searchable encryption. In: Proceedings of the 21st International Conference on Advanced Information Networking and Applications, AINA 2007, pp. 409–414. IEEE (2007)
28. Song, D.X., Wagner, D.A., Perrig, A.: Practical techniques for searches on encrypted data. In: 2000 IEEE Symposium on Security and Privacy, S&P 2000, pp. 44–55. IEEE (2000)
29. Sun, S.-F., Liu, J.K., Sakzad, A., Steinfeld, R., Yuen, T.H.: An efficient non-interactive multi-client searchable encryption with support for Boolean queries. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9878, pp. 154–172. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45744-4_8
30. 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 2015 IEEE Conference on Computer Communications, INFOCOM 2015, pp. 2110–2118. IEEE (2015)

31. Veritas: Accelerating digital transformation through multi-cloud data management. https://manufacturerstores.techdata.com/docs/default-source/veritas/360-data-management-suite-brochure.pdf?sfvrsn=2. Accessed 15 Apr 2018
32. Wang, J., Chen, X., Huang, X., You, I., Xiang, Y.: Verifiable auditing for outsourced database in cloud computing. IEEE Trans. Comput. **64**(11), 3293–3303 (2015)
33. Wang, J., et al.: Efficient verifiable fuzzy keyword search over encrypted data in cloud computing. Comput. Sci. Inf. Syst. **10**(2), 667–684 (2013)
34. Wang, Y., Wang, J., Sun, S.-F., Liu, J.K., Susilo, W., Chen, X.: Towards multi-user searchable encryption supporting boolean query and fast decryption. In: Okamoto, T., Yu, Y., Au, M.H., Li, Y. (eds.) ProvSec 2017. LNCS, vol. 10592, pp. 24–38. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68637-0_2
35. Zuo, C., Macindoe, J., Yang, S., Steinfeld, R., Liu, J.K.: Trusted Boolean search on cloud using searchable symmetric encryption. In: 2016 IEEE Trustcom/BigDataSE/ISPA, pp. 113–120. IEEE (2016)