Publicly Verifiable Boolean Query Over Outsourced Encrypted Data

Shunrong Jiang*, Xiaoyan Zhu*, Linke Guo† and Jianqing Liu‡
*National Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China
† Binghamton University, State University of New York, Binghamton, NY, 13902
‡ University of Florida, Gainesville, FL, 32611, USA
Email: jsywow@gmail.com, xyzhu@mail.xidian.edu.cn, lguo@binghamton.edu, ljq0203@gmail.com

***Abstract*—Outsourcing storage and computation to the cloud has become a common practice for businesses and individuals. As the cloud is semi-trusted or susceptible to attacks, many researches suggest that the outsourced data should be encrypted and then retrieved by using searchable symmetric encryption (SSE) schemes. Since the cloud is not fully trusted, we doubt whether it would always process queries correctly or not. Therefore, there is a need for users to verify their query results. Motivated by this, in this paper, we propose a publicly verifiable dynamic searchable symmetric encryption scheme based on the accumulation tree. We first construct an accumulation tree based on encrypted data and then outsource both of them to the cloud. Next, during the search operation, the cloud generates the corresponding proof according to the query result by mapping Boolean query operations to set operations while keeping privacy-preservation and achieving the verification requirements: authenticity, freshness, and completeness. The security analysis and performance evaluation show that the proposed scheme is privacy-preserving and practical.**

I. Introduction

The great flexibility and economic savings of cloud computing motivate companies and individuals to outsource their data to cloud servers. By outsourcing a dataset to the cloud, the data owner or other valid data users can then issue the cloud informational queries that are answered according to the dataset. This model captures a variety of real-world applications such as outsourced SQL queries, streaming dataset and outsourced file systems. However, the privacy and confidentiality concerns of data often make them reluctant to do that. Therefore, it is natural for a data owner to encrypt data before outsourcing them to the cloud. As a result, the cloud server cannot reveal the content of the outsourced data. However, since data has been encrypted in the cloud, it obstructs the traditional data utilization service based on plaintext keyword search. Thus, how to efficiently obtain encrypted data from the cloud server is very important for outsourcing storage applications.

Motivated by this challenge, many searchable symmetric encryption (SSE) schemes [1–5] are proposed to satisfy different search functions on encrypted data. By using these schemes, a data owner can outsource encrypted data to protect the confidentiality of data, and data users can query from the cloud. Specifically, in mobile health networks, the health center collects all users' health information and outsources them to the cloud server after encryption. Then different hospitals or research institutes use these information for disease treatment or prevention according to keywords. However, due to the nature of the delegation/outsourcing, the cloud server can fully control the outsourced data and decide the SSE query result of hospitals or research institutes, which causes issues of trust.

There are several reasons for which the data owner/users cannot trust the cloud server: Firstly, the cloud server may run buggy software or its systems may be vulnerable to security breaches, leading to the incorrect result [6, 7]. Secondly, in some cases such as early cancer detection and prevention, the applications are so critical that hospitals or research institutes want to rule out accidental errors during the computation. Finally, the hospital may have competing interest with other hospitals and it can collude with the cloud to provide an incomplete result to the competing hospitals [8]. Therefore, it is necessary for the cloud server to have the ability to provide the proof with which the data owner/users are able to verify the integrity of the search result returned by the cloud server.

The query integrity verification has been studied for structured attributed-value type database [9, 10] and streaming setting [8, 11–13]. Many verification schemes are proposed to meet verification requirements (details are introduced in related work). However, these schemes may not be suitable for SSE condition: (1) These schemes assumed that the data stored at the third party publisher is in plaintext, while for SSE, the data is encrypted so that the publisher cannot see the actual content; (2) All of them are ordered by some sequences or the data update according to the time slice which makes verification easier.

Obviously, the query integrity verification in SSE should contain three aspects [9]: (1) Authenticity: Every record returned in the search result must originate from the data owner; (2) Freshness: The record values in the answer must be up-to-date; (3) Completeness: Every record that satisfies the query condition must be in the search result. Moreover, the verification process should be secure and privacy-preserving. However, even though there are many studies about the SSE, the search verification work is limited except [14]. Moving a step forward, in this paper, we present a publicly verifiable scheme which can publicly verify whether the cloud server

This work is supported by the industrial research project of Shaanxi Province under Grant No. 2015GY008, the key project of NSFC-Guangdong Union Foundation under Grant No. U1401251, and China 111 Project under Grant No. B08038.

978-1-4799-5952-5/15/$31.00 ©2015 IEEE

has faithfully executed Boolean search operations in dynamic SSE (DSSE).

To realize query integrity verification in DSSE, a possible solution is that we can map query operations to set operations, so that the query verification can be achieved by using the accumulation tree [15]. However, there are several challenges that must be overcome before we use this method: Firstly, how to map query operations to set operations in DSSE. Secondly, how to construct the accumulation tree with limited information of DSSE so that the construction process would not affect the architecture of DSSE. Thirdly, how to achieve the secure and privacy-preserving query integrity verification by using the accumulation tree in DSSE while ensuring the verification requirements. Finally, how to ensure the efficiency and scalability of the query integrity verification process in DSSE. All these challenges make the query integrity verification of DSSE different from the existing work. Thus, the contributions of our work are listed as follows:

- We propose a publicly Boolean query integrity verification scheme over the outsourced dynamic encrypted data to check the authenticity, freshness and completeness of the query result.
- We construct a special Merkle hash tree named accumulation tree in DSSE to map Boolean operations of keywords to set operations, which is different from the traditional signature or aggregate signature.
- The security and performance analysis are carried out to show that the proposed scheme is privacy-preserving and practical.

The rest of this paper is organized as follows. Section II presents the system model and preliminaries. We describe the construction of our scheme in Section III. We give the security analysis in Section IV and provide performance analysis in Section V. The related work is given in Section VI. Finally, Section VII concludes the paper.

II. System Model and Preliminaries

A. System Model and Motivation

The system model is illustrated as Fig. 1 which contains three entities: A data owner DO, who outsources a large-scale collection of d documents to the remote cloud server S. Before searching, data uses DUs should obtain the token according to the search keywords from DO. The cloud server S which provides storage services and returns the query result with the corresponding proof according to the search token of data users DUs.

B. Attack Model

The data owners are naturally trusted. Both authorized and unauthorized data users are semi-trusted, meaning that they may try to infer some sensitive information from the query result and the corresponding proof. The cloud server is not trusted as it executes the search operations, which already implies that the cloud may manipulate the outsourced encrypted data. Moreover, we also consider potential malicious data users [8] which could collude with the cloud server

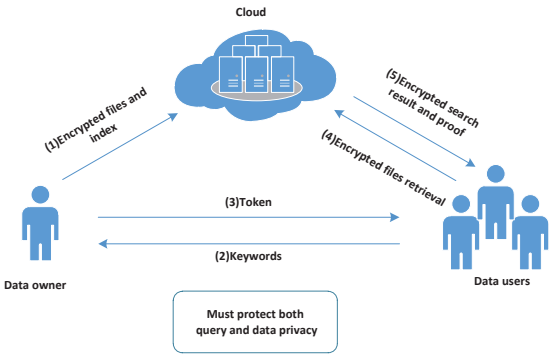

Fig. 1. The system model.

or other malicious users, or help the cloud to cheat with other users. Note that the assumption about malicious data users enables the public verifiability property of our solution. Obviously, our attack model is more general.

C. Design Objective

The objective of our work is to design a publicly Boolean query integrity verification scheme in DSSE. First, we present a leakage function $\mathcal{L}$, which covers all the information leakage in our scheme. For instance, the privacy leakage introduced by a Boolean query Q on a database DB is denoted as $\mathcal{L}(\mathsf{DB}, \mathsf{Q})$. Informally, the privacy leakage function for the query integrity verification in DSSE includes the following aspects:

- Size Pattern: The cloud can learn the total number of data records in the dataset and the total number of search queries submitted by each data user [16];
- Access Pattern: The cloud reveals the identifier of each encrypted data record that is returned for each query [16];
- Search Pattern: The cloud can learn if the same encrypted data record is retrieved by two different queries [16];
- Path Pattern: The cloud server learns the set of nodes of the accumulation tree in each query integrity verification.

To sum up, the design objective is to achieve secure and privacy-preserving Boolean query integrity verification in DSSE under the leakage function $\mathcal{L}$.

D. Preliminaries

1) Bilinear Pairing: Let $\mathbb{G}_1$ be a multiplicative cyclic group of prime order p, generated by element $g \in \mathbb{G}_1$. Let $\mathbb{G}_T$ be a multiplicative cyclic group of the same order p, such that there exists a pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ with the following properties [17]:

- Bilinearity: $e(P^a, Q^b) = e(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_n$;
- Nondegeneracy: $e(g, g) \neq 1$;
- Computability: For all $P, Q \in \mathbb{G}_1$, $e(P, Q)$ is efficiently computable.

2) Accumulation Trees: An accumulation tree [15, 18, 19] AT is a tree with $\lceil 1/\epsilon \rceil$ levels and m leaves, where $0 < \epsilon < 1$ is a parameter chosen upon setup. Each internal node of AT with degree m^ϵ and level i in the tree contains $m^{1-i\epsilon}$ nodes

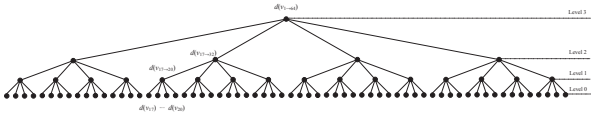

Fig. 2. The accumulation tree consisting of 64 sets where ϵ=1/3. $d(v_{17})$ is the digest of the set S_{17} and $d(v_{17\to 20})$ means the digest of the values of its all children nodes ($d(v_{17})$, $d(v_{18})$, $d(v_{19})$ and $d(v_{20})$).

(the root node of the tree lies in the maximum level). AT has a constant height for a fixed ϵ. Intuitively, it can be seen as a "flat" version of Merkel trees as shown in Fig. 2. Each leaf node contains the digest $d(v_i)$ of the accumulation value $acc(S_i) = g^{\prod \mathsf{e}_i}$ for the set S_i where e_i is the element of S_i and each internal node contains the digest of the values of its children [15].

For the Boolean query, we can map the keyword Boolean operation to the set operation. At the same time, the set operation verification can be expressed by using accumulation tree [15].

III. Publicly Verifiable Boolean Query over Encrypted Data

In this section, we construct our publicly verifiable scheme in DSSE. In order to introduce our scheme, we first describe the construction process of the query verification with the SSE scheme, i.e., OSPIR-OXT [1]. Before describing the detail of our scheme, an overview will be presented.

A. Overview

The basic idea underlying the construction is that: to achieve the public Boolean query verification for outsourced encryption documents, an accumulation tree is associated with encryption documents as illustrated in Fig. 2. Then the data owner outsources both of them to the cloud server. During the search process, after executing the privacy-preserving Boolean query over encrypted data, the cloud should compute its corresponding proof according to the query result by using the accumulation tree. To compute the proof, the cloud maps relations of Boolean queryies to the similar set operations of the accumulation tree. Finally, with the search result and the proof, the data user can verify the authenticity, completeness, and freshness of search results even without decrypting them.

B. The Proposed Scheme

We take the conjunction keyword operation as an example to introduce our scheme and the detail is described as follows:

1) Initialization: This phase mainly contains two parts: the SSE construction and the corresponding accumulation tree construction. For the SSE construction, the data owner encrypts outsourced documents and generates the document index for SSE as described in OSPIR-OXT. After that, to verify the Boolean query result, an accumulation tree is built according to encrypted indices e_i and encrypted keywords $\mathsf{stag}(w_i)$. Given the bilinear parameters $(q, \mathbb{G}_1, \mathbb{G}_T, e, p)$, DO chooses a random number $s \in \mathbb{Z}_p^*$ as the secret key, a hash function $h(\cdot) : \mathbb{G}_1 \to \mathbb{Z}_p^*$, and computes g=$\{g^s, g^{s^2}, \cdots, g^{s^q}\}$

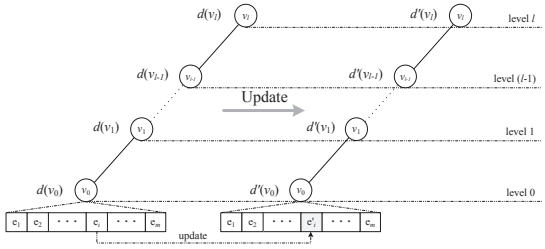

Fig. 3. The update process of the accumulation tree when e_i is changed to e_i' where $l = \lceil 1/\epsilon \rceil$. v_0, v_1, $\cdots$, $v_{\lceil 1/\epsilon \rceil}$ be the path in AT from node v_0 to the root of the tree.

where $q \geq max\{m, max_{i=1,\cdots,m}|\mathsf{stag}(w_i)|\}$. $|\mathsf{stag}(w_i)|$ means the number of e_i in $\mathsf{stag}(w_i)$ and m is the number of keywords. Finally, DO publishes the system parameter param=$\{h(\cdot), p, \mathbb{G}_1, \mathbb{G}_T, e, g, \mathsf{g}\}$. The construction of an accumulation tree is shown in **Algorithm** 1.

Algorithm 1 : The construction of an accumulation tree AT.

Require: all e_i, s, $h(\cdot)$.
1: **for** Each keyword $w_i \in \mathsf{W}$ **do**
2: **for** All encryption document indices $\mathsf{e}_i \in w_i$ **do**
3: Set $acc(w_i)$=$g^{\Pi(\mathsf{e}_i+s)}$.
4: **end for**
5: **end for**
6: Data owner picks a constant ϵ, where $0 < \epsilon < 1$.
7: DO constructs tree AT according to $\mathsf{stag}(w_i)$ that has $l = \lceil 1/\epsilon \rceil$ levels and m leaves, where m is the number of W.
8: **for** Each node v of AT **do**
9: **if** v is a leaf corresponding to keyword w_i **then**
10: DO sets $d(v) = acc(w_i)^{(i+s)}$.
11: **else**
12: Compute $d(v) = g^{\prod_{\overline{v} \in N(v)}(h(d(\overline{v})+s))}$ where $N(v)$ denotes the set consisted by children nodes of v.
13: **end if**
14: **end for**
15: DO sets d_0=$d(r)$ where r is the root of AT and keeps it.
16: Finally, DO outsources AT with encrypted DB to the cloud.

2) Update: In a DSSE scheme, we need to support functions of adding, modifying, and deleting documents. To update ind_i with ind_i', the data owner should compute e_i' to replace e_i. Then, DO updates the corresponding AT to AT′ as shown in Fig. 3. We introduce the update process of the accumulation tree as shown in **Algorithm** 2. It is clear that in the case where document ind_i' is deleted from w_i, $d'(v_0)$=$acc(w_i)^{(\mathsf{e}_i+s)^{-1}}$; for document ind_i' that is added to w_i, $d'(v_0)$=$acc(w_i)^{(\mathsf{e}_i'+s)}$.

Algorithm 2 : The update of the accumulation tree AT.

Require: e_i', s, d_0.
1: Let v_0 be the leaf node of AT corresponding to $\mathsf{stag}(w_i)$.
2: Let v_0, v_1, $\cdots$, $v_{\lceil 1/\epsilon \rceil}$ be the path in AT from node v_0 to the root of the tree in Fig. 3.
3: DO sets $d'(v_0) = acc(w_i)^{(\mathsf{e}_i'+s)(\mathsf{e}_i+s)^{-1}}$.
4: **for** $j = 1, \cdots, v_{\lceil 1/\epsilon \rceil}$ **do**
5: DO sets $d'(v_j) = d(v_j)^{(h(d'(v_j-1))+s)(h(d(v_j-1))+s)^{-1}}$.
6: **end for**
7: DO updates $d(v_j)$ by $d'(v_j)$ from v_0 to the root of AT.

3) Token Generation: This phase is almost as same as that of the OSPIR-OXT scheme, and d_0 (the root of AT) is sent to the data user.

4) Search and Proof: Different from OSPIR-OXT, in this phase, the cloud server should return the search result as well as the proof according to the query keywords $\overline{w} = (w_1, ..., w_n)$. Since the encryption search process and the accumulation tree are independent of each other, we tend to generate the proof Π after getting the query result I=$\{\mathsf{e}_1, \mathsf{e}_2, \cdots, \mathsf{e}_\hbar\}$. As proposed above, the query integrity proof should contain authenticity, freshness, and completeness. In order to accomplish these properties, the conjunction query proof Π consists of the following parts as **Algorithm** 3 shows.

Algorithm 3 : The proof generation of the query result I.

Require: $\overline{w} = (w_1, ..., w_n)$, $\mathsf{I} = \{\mathsf{e}_1, \mathsf{e}_2, \cdots, \mathsf{e}_\hbar\}$ and g.
1: **for** $i = 1, \cdots, n$ **do**
2: Let v_0 be the leaf node corresponding to $\mathsf{stag}(w_i)$ of the accumulation tree AT and v_1,$\cdots$,$v_{\lceil 1/\epsilon \rceil}$ be the node path from v_0 to the root r.
3: **for** $j = 1, \cdots, \lceil 1/\epsilon \rceil$ **do**
4: Compute $\gamma_j = g^{\prod_{\overline{v} \in N(v_j) - \{v_{j-1}\}} (h(d(\overline{v}))+s)}$.
5: **end for**
6: Cloud outputs $\pi_i = (d(v_0), \gamma_1)$,$\cdots$, $(d(v_{\lceil 1/\epsilon \rceil} - 1), \gamma_{\lceil 1/\epsilon \rceil})$ as the freshness proof of $acc(w_i)$.
7: Cloud computes $\mathsf{W}_{\mathsf{I},i} = g^{\prod_{x \in \mathsf{stag}(w_i) - \mathsf{I}} (x+s)}$ as subset proof of $\mathsf{stag}(w_i)$.
8: Cloud computes $\mathsf{F}_{\mathsf{I},i} = g^{q_j(s)}$ according to lemma III.1 as operation completeness proof.
9: Cloud outputs the proof Π=(π_i, $\mathsf{W}_{\mathsf{I},i}$, $\mathsf{F}_{\mathsf{I},i}$).
10: **end for**

Lemma III.1. *Assume set I is the intersection of sets S_1, S_2, $\cdots$, S_t if and only if there exists polynomials $q_1(s)P_1(s) + q_2(s)P_2(s) + \cdots + q_t(s)P_t(s) = 1$. Moreover, computing polynomial $q_1(s), q_2(s), \cdots, q_t(s)$ has the complexity of $O(N \log^2 N \log\log N)$.*

5) Verification: When the data user receives the search result and the corresponding proof, the query integrity verification is executed as follows:

As the conjunction query result I=$\{\mathsf{e}_1, \mathsf{e}_2, \cdots, \mathsf{e}_\hbar\}$, the verification process consists of following steps:

Verify the freshness of accumulation values by checking following equations:

1) $e(d(v_0), g) = e(acc(w_i)^{i+s}, g) \stackrel{?}{=} e(acc(w_i), g^i g^s)$.
2) $e(d(v_j), g) \stackrel{?}{=} e(\gamma_j,\ g^{h(d(v_j - 1))} g^s)$, for $j = 1, \cdots, \lceil 1/\epsilon \rceil$.

If these equations hold, we verify whether the query result I is the subset of $\mathsf{xtag}(w_i)$: Randomly choose a number $k \in \mathbb{Z}_p^*$, and then compute $\prod_{i=0}^{m}(k + \mathsf{e}_i)$ to get the coefficients $\delta_1, \cdots, \delta_\hbar$.

For $j = 1 \cdots, n$, we check:

$$e(\prod_{i=0}^{\hbar} (g^{s^i})^{\delta_i}, \mathsf{W}_{\mathsf{I},j}) = e(g^{\sum_{i=0}^{\hbar} \delta_i s^i}, \mathsf{W}_{\mathsf{I},j}) \stackrel{?}{=} e(acc(\mathsf{I}), \mathsf{W}_{\mathsf{I},j}). \quad (1)$$

At last, we execute the completeness verification if the subset verification is correct:

$$\prod_{j=1}^{n} e(\mathsf{W}_{\mathsf{I},j}, \mathsf{F}_{\mathsf{I},j}) = e(g, g)^{\sum_{j=1}^{n} q_j(s) P_j(s)} \stackrel{?}{=} e(g, g). \quad (2)$$

If the equation holds, we accept the returned query result I.

IV. Security Analysis

In this section, we present the security analysis of our scheme. We also present a game-based security definition of our scheme in the random oracle model and give the corresponding proof in the Appendix of our full version.

Security and Privacy in $\mathsf{EDBSetup(DB, RDK)}$: To generate the verification proof for the query operation, we use encrypted e_i as the set element and encrypted $\mathsf{stag}(w_i)$ as the set identity to generate accumulation values and the corresponding accumulation tree. Obviously, in this process all the information we use to generate the accumulation values and tree are based on the generation of the OSPIR-OXT which is also known by the cloud server, without any additional information about the encrypted documents.

Security and Privacy in $\mathsf{Update(EDB, AT)}$: In the update process, DO computes e_i' to replace e_i. The cloud server only needs to execute the update and the replacement operation without any computational operation. Thus, the security and privacy of our scheme can be protected during the execution of $\mathsf{Update(EDB, AT)}$.

Security and Privacy in $\mathsf{Search}(token)$: Different from OSPIR-OXT, the cloud server should compute the corresponding proof after getting the query result. To generate the proof Π, the cloud needs param, AT and the query result. As described in **Algorithm** 3, the generation of the necessary query integrity proof only uses the information known to the cloud. Thus, during the proof generation, the cloud cannot obtain any other information, and the security and privacy of $\mathsf{Search}(token)$ can be achieved.

Security and Privacy in Query Verification: To ensure the correctness of the query result, the DO sends the latest root $d(v_l)$ of the accumulation tree to DU. Based on $d(v_l)$ and the nature of the accumulation tree, the query integrity verification can be checked through the proof Π. During the verification process, all the verification proof π_i, $\mathsf{W}_{\mathsf{I},i}$, and $\mathsf{F}_{\mathsf{I},i}$ are constructed in the formula of $g^{p(s)}$. Thus, under the q-strong Diffie-Hellman assumption, the DU can securely verify the integrity of the query result without learning any additional information from the proof.

V. Performance Analysis

To demonstrate the efficiency of our scheme, we evaluate our scheme based on the benchmarks of [20] with the Intel Core i5 CPU 2.5GHz. During the implementation, the D-CLXVI [21] is used for the bilinear pairing computations, Flint [22] for the modular arithmetic, and Crypto++ [23] for SHA-256 hash operations. DCLXVI employs a 256-bit BN elliptic curve and an asymmetric optimal ate pairing, offering bit-level security of 128 bits. We represent elements of $\mathbb{G}_1$ with 768 bits using Jacobi coefficients, which yield faster operations.

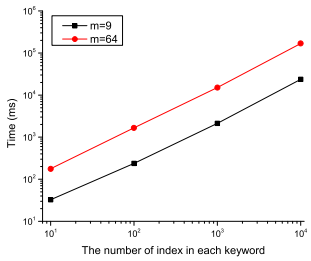

Fig. 4. Setup time vs n_w.

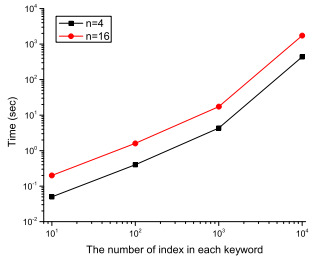

Fig. 5. Proof generation time vs n_w where $m = 64$.

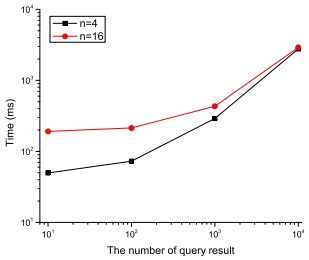

Fig. 6. Verification time vs $\hbar$ $m = 64$.

Elements in $\mathbb{G}_2$ are roughly twice as large as those of $\mathbb{G}_1$. We use the conjunction keyword query operation "w_1 AND w_2 AND $\cdots$ AND w_n" to demonstrate the efficiency of the proof generation and verification. The corresponding result is $\{\mathsf{e}_1, \mathsf{e}_2, \cdots, \mathsf{e}_\hbar\}$.

Setup: Fig. 4 indicates the computation cost versus the number of index n_w in each keyword when the number of keywords m is 9 and 64, respectively. We observe that the overhead increases almost linearly with the number of index in each keyword. What causes the result is that the overhead is dominated by the computation cost of $acc(w_i)$, which completely hides the hashing cost and the tree construction cost in the setup stage. Given $m = 64$ in Fig. 4, we can see the total setup time is about 170s when the number of n_w reaches 10,000. This is a one-time cost for the owner.

Update: For a chosen ϵ, the computation overhead of each update process increases linearly with the number of keywords contained by the update index. We do not plot the update process. The update consumption for each process is approximately 2ms and 3ms for $m = 9$ and $m = 64$, respectively.

Proof Generation: Fig. 5 plots the proof generation time at the cloud server versus the number of index n_w in each keyword. It shows that the proof generation overhead is influenced by the number of index n_w in each keyword and the number of conjunction keywords n. Given parameter m=64, ϵ=1/3, $n_w = 10,000$ and $\hbar$=0.1n_w, we can see that to generate the proof, it consumes about 430s and 1,740s for $n = 4$ and $n = 16$, respectively. The larger the number of conjunction keywords n is, the more time this proof generation phase takes. The reason leading to such result is that with the number of conjunction keywords increasing, the number of the set check process required to be executed grows correspondingly, which needs more exponentiation operations and polynomial expansion operations while the number of index n_w and $\hbar$ in each keyword are the same.

Similar to the relation between the number of conjunction keywords n and the proof generation time, as the number of index n_w in each keyword increases, the proof generation time becomes longer, and the extent of improvement gradually levels up. For scenario that the number of conjunction keywords $n = 16$, the proof generation time increases from 0.4s to 1,740s when n_w increases from 100 to $10,000$. This is because computing $\mathsf{F}_{\mathsf{l},i}$ requires running the Extended Euclidean (XGCD) algorithm, whose time increases drastically with the degree of polynomials. Thus, with the increase of the number of e_i in each $\mathsf{Stag}(w_i)$, the generation process of $\mathsf{F}_{\mathsf{l},i}$ will consume more time, which is also the major overhead of the proof generation time.

Proof verification: Fig. 6 shows the proof verification time at the DU versus the number of the query result $\hbar$ under different numbers of operation keywords n. To verify the query proof on condition that the number of query result $\hbar$=1000, it consumes about 2.7s and 3s when the number of operation keywords $n = 4$ and $n = 16$, respectively. We can observe that when there are more operation keywords, the verification overhead gets more, mainly due to the increasing of set operation which results in more pairing check operations.

It is obvious that with the increasing number of $\hbar$, the verification time also rapidly increases and the performance of schemes under different number of n converges. The reason is that the subset verification becomes the dominant factor, which effectively hides the costs of freshness and completeness verifications.

VI. Related Work

We introduce related work in three aspects: searchable encryption, query integrity verification and query integrity verification on encrypted data.

Searchable encryption: Song et al. [24] explicitly considered the problem of searchable encryption and presented a scheme with search time that was linear with the size of the data collection for the first time. Their construction supports insertions/deletions of documents in a straightforward way. Curtmola et al. [16] gave the first index-based SSE constructions to achieve sublinear search time for SSE. A similar construction was proposed by Chase and Kamara [25], but with higher space complexity. Subsequently, Kamara et al. [2] introduced a dynamic scheme which was the first one with sublinear search time. However, it did not achieve forward privacy or revealed hashes of the unique keywords contained in the document during the update. Recently, Cash et al. [3] presented a SSE scheme supporting conjunction queries over static data. Based on Cash et al. [3] work, Jarecki [1] proposed a scheme that allowed data owners to authorize third parties

and to execute private information retrieval on the outsourced database. It is obvious that, all the proposed schemes for SSE do not consider the verification problem of the search result.

Query integrity verification: Li et al. [10] introduced an efficient implementation of Merkle hash tree authenticated B^+-tree to audit the completeness of the query result, and demonstrated its superiority over signature chaining. By using signature aggregation, Pang et al. [9] proposed a scheme to verify the authenticity, completeness, and freshness of query answers from frequently updated databases that were hosted on untrusted servers. Li et al. [11] considered verifying queries on a data stream with sliding windows via Merkle trees, hence the verifier's space was proportional to the window size. Papadopoulos et al. [12] proposed a protocol to verify continuous query over streaming data, again requiring linear space on the verifier's side in the worst case. Instead of using cryptographic primitives to verify the query result, Yi et al. [13] used algebraic and probabilistic techniques to compute a small synopsis on the true query result and to store a compact authentication synopsis that helped audit the result integrity. However, unlike our solution, none of these solutions achieve public verification for encrypted data.

Query integrity verification on encrypted data: Kurosawa et al. [26] showed how to construct a (verifiable) SSE scheme that was universally composable (UC). While UC-security is a stronger notion of security, its construction requires linear search time. In addition, it did not consider the Boolean query on keywords. Zheng et al. [14] proposed a verifiable attribute-based keyword search scheme which can verify whether the cloud had faithfully executed the search operation by using signature and Bloom filter. However, to enable general multiple keywords Boolean query, the query contains a number of keywords which are not known in advance and their preceding signature may not work. Apart from that, their scheme did not consider the update process and update needed to resign documents, which is time-consuming.

VII. Conclusion

In this paper, we study the problem of verifying the authenticity, freshness, and completeness of the Boolean query result over the outsourced encrypted data. Based on OSPIR-OXT [1], we propose a publicly verifiable scheme by constructing the accumulation tree to achieve the query integrity verification while keeping privacy-preserving and efficiently practical. The security analysis shows that without protecting the access pattern, our scheme can keep the privacy-preserving of private information retrieval. The performance demonstrates our scheme is scalable.

References

[1] S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Outsourced symmetric private information retrieval," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 875–888.

[2] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 965–976.

[3] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Advances in Cryptology-CRYPTO 2013*. Springer, 2013, pp. 353–373.

[4] M. Naveed, M. Prabhakaran, and C. A. Gunter, "Dynamic searchable encryption via blind storage." in *Proceedings of the IEEE Symposium on Security and Privacy, San Jose, CA, May*, 2014.

[5] E. Stefanov, C. Papamanthou, and E. Shi, "Practical dynamic searchable encryption with small leakage." *IACR Cryptology ePrint Archive*, vol. 2013, p. 832, 2013.

[6] http://www.wired.com/2009/01/magnolia-suffer/.

[7] http://mashable.com/2011/02/27/gmail-glitch/.

[8] S. Nath and R. Venkatesan, "Publicly verifiable grouped aggregation queries on outsourced data streams," in *Proceedings of the 29th International Conference on Data Engineering (ICDE)*. IEEE, 2013, pp. 517–528.

[9] H. Pang, j. Zhang, and k. Mouratidis, "Scalable verification for outsourced dynamic databases," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 802–813, 2009.

[10] F. Li, M. Hadjieleftheriou, k. Kollios, and L. Reyzin, "Dynamic authenticated index structures for outsourced databases," in *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*. ACM, 2006, pp. 121–132.

[11] F. Li, K. Yi, M. Hadjieleftheriou, and G. Kollios, "Proof-infused streams: Enabling authentication of sliding window queries on streams," in *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment, 2007, pp. 147–158.

[12] S. Papadopoulos, Y. Yang, and D. Papadias, "Cads: Continuous authentication on data streams," in *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment, 2007, pp. 135–146.

[13] K. Yi, F. Li, G. Cormode, M. Hadjieleftheriou, G. Kollios, and D. Srivastava, "Small synopses for group-by query verification on outsourced data streams," *ACM Transactions on Database Systems (TODS)*, vol. 34, no. 3, p. 15, 2009.

[14] Q. Zheng, S. Xu, and G. Ateniese, "Vabks: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proceedings of the 2014 INFOCOM 2014*. IEEE, 2014.

[15] C. Papamanthou, R. Tamassia, and N. Triandopoulos, "Authenticated hash tables," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 437–448.

[16] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 79–88.

[17] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology-CRYPTO 2001*. Springer, 2001, pp. 213–229.

[18] C. Papamanthou, R. Tamassia, and N. Triandopoulos, "Optimal verification of operations on dynamic sets," in *Advances in Cryptology-CRYPTO 2011*. Springer, 2011, pp. 91–110.

[19] R. Canetti, O. Paneth, D. Papadopoulos, and N. Triandopoulos, "Verifiable set operations over outsourced databases," in *Public-Key Cryptography-PKC 2014*. Springer, 2014, pp. 113–130.

[20] D. Papadopoulos, S. Papadopoulos, and N. Triandopoulos, "Taking authenticated range queries to arbitrary dimensions," *in Proceedings of the 2014 ACM SIGSAC conference on Computer & communications security. ACM, 2014.*

[21] "The dclxvi library," http://cryptojedi.org/crypto/.

[22] "The flint library," http://www.flintlib.org/.

[23] "The crypto++ library," http://www.cryptopp.com/.

[24] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of the IEEE Symposium on Security and Privacy, 2000. S&P 2000*. IEEE, 2000, pp. 44–55.

[25] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *Advances in Cryptology-ASIACRYPT 2010*. Springer, 2010, pp. 577–594.

[26] K. Kurosawa and Y. Ohtaki, "Uc-secure searchable symmetric encryption," in *Financial Cryptography and Data Security*. Springer, 2012, pp. 285–298.