FSPVDsse: A Forward Secure Publicly Verifiable Dynamic SSE scheme

First Author[1 [0000−1111−2222−3333]], Second Author[2,3 [1111−2222−3333−4444]], and Third Author[3 [2222−−3333−4444−5555]]

No Institute Given

Abstract. A symmetric searchable encryption (SSE) scheme allows a client (data owner) to search on encrypted data outsourced to an untrusted cloud server. The search may either be a single keyword search or a complex query search like conjunctive or Boolean keyword search. Information leakage is quite high for dynamic SSE, where data might be updated. It has been proven that to avoid this information leakage an SSE scheme with dynamic data must be *forward private*. A dynamic SSE scheme is said to be forward private, if adding a keyword-document pair does not reveal any information about the previous search result with that keyword.
In SSE setting, the data owner has very low computation and storage power. In this setting, though some schemes achieve forward privacy with honest-but-curious cloud, it becomes difficult to achieve forward privacy when the server is malicious, meaning that it can alter the data. Verifiable dynamic SSE requires the server to give a proof of the result of the search query. The data owner can verify this proof efficiently. In this paper, we have proposed a generic publicly verifiable dynamic SSE (DSSE) scheme that makes any forward private DSSE scheme verifiable without losing forward privacy. The proposed scheme does not require any extra storage at owner-side and requires minimal computational cost as well for the owner. Moreover, we have compared our scheme with the existing results and show that our scheme is practical.

Keywords: Searchable encryption · Forward privacy · Verifiability · BLS signature · Cloud computing.

1 Introduction

Data stored in untrusted servers is prone to attacks by the server itself. In order to protect confiential infomation, clients store encrypted data. This makes searching on data quite challenging. A searchable symmetric encryption (SSE) scheme enables a client or data owner to store its data in a cloud server without loosing the ability to search over them. When an SSE scheme supports update, it is called a dynamic SSE (DSSE) scheme.

There are plenty of works on SSE as well as DSSE. Most of them considers the cloud server to be honest-but-curious. An honest-but-curious server follows the protocol but wants to extract information about the plaintext data and the

queries. However, if the cloud itself is malicious, it does not follow the protocol correctly. In the context of search, it can return only a subset of results, intsead of all the records of the search. So, there is need to verify the results returned by the cloud to the querier. An SSE scheme for static data where the query results are verifiable is called Verifiable SSE (VSSE). Similarly, if the data is dynamic the scheme is said to be a verifiable dynamic SSE (VDSSE).

There are single keyword search VSSE schemes which are either new constructions supporting verifiability or design techniques to achieve verifiability on the existing SSE schemes by proposing generic algorithm. VSSE with single keyword search has been studied in [5], [7], [13]. In [21], [22] etc., VSSE scheme with conjunctive query has been studied. Moreover, there are also works that gives VDSSE scheme for both single keyword search ([14]) as well as complex query search including fuzzy keyword search ([27]) and Boolean query ([9]). However, Most of them are *privately verifiable*. A VSSE or VDSSE scheme is said to be privately verifiable if only querier, who receive search result, can verify it. On the other hand, a VSSE or VDSSE scheme is said to be *publicly verifiable* if any third party, including the database owner, can verify the search result without knowing the content of it.

There is also literature on public verifiability. Soleimanian and Khazaei [19] and Zhang et al. [25] have presented SSE schemes which are publicly verifiable. VSSE with Boolean range queries has been studied by Xu et al. [23]. Though, their verification method is public, since the verification is based over blockchain databases, it has extra monetary cost. Besides, Monir Azraoui [1] presented a conjunctive search scheme that is publicly verifiable. In case of dynamic database, publicly verifiable scheme by Jiang et al. [9] supports Boolean Query and that by Miao et al. [14] supports single keyword search.

However, file-injection attack [26], in which the client encrypts and stores files sent by the server, recovers keywords from future queries, has forced researchers to think about dynamic SSE schemes to be forward private where adding a keyword-document pair does not reveal any information about the previous search result with that keyword. In addition, in presence of malicious cloud server, the owner can outsource the verifiabilty to a third party auditor to reduce its computational overhead. The only forward private single keyword search VSSE scheme is proposed by Yoneyama and Kimura [24]. However, the scheme is privately verifiable and the owner requires significant amount of computation for verification.

1.1 Our Contribution

In this paper, we have contributed the followings in the literature of VSSE.

1. We have formally define a verifiable DSSE scheme. Then we have proposed a generic verifiable SSE scheme (Ψ_s) which is very efficient and easy to integrate.
2. We have proposed a generic publicly verifiable dynamic SSE scheme (Ψ_f). Our proposed scheme is forward private. This property is necessary to protect

a DSSE scheme from file injection attack. However, no previous publicly verifiable scheme is forward private. In fact, only forward private scheme [24] is privately verifiable.
3. We present formal security proofs for these schemes and shows that they are adaptively secure in random oracle model.

Both of the schemes do not uses any extra storage, at owner side, than the embedded schemes. Thus, for a resource constrained client, the schemes are very effective and efficient.

In Table 1, we have compared our proposed schemes with existing ones.

Table 1. Different verifiable SSE schemes

Data Type	static				dynamic			
Query Type	single		complex		single		complex	
Verification	private	public	private	public	private	public	private	public
Schemes	[5], [7], [18], [13], Ψ_s	[19]	[22], [11], [23]	[19]	[24], [3]	[14], Ψ_f	[27]	[9]
Forward Private	not applicable				[24], Ψ_f			

1.2 Organization

We have briefly described the works related to verifiable SSE in Section 2. We have discussed the required preliminary topics in Section 3. In Section 4, we have presented a generic approach of verifiable SSE scheme. In Section 5, we present our proposed generic construction of publicly verifiable DSSE scheme in details. We have compared its complexity with similar publicly verifiable schemes in Section 6. Finally, we summaries our work in Section 7 with possible future direction of research.

2 Related Works

The term *Searchable Symmetric Encryption* is first introduced by Curtmola et al. [8] where they have given formal definition of keyword search schemes over encrypted data. Later, Chase et al. [6] and Liesdonk et al. [12] presented single keyword search SSE for static database. Thereafter, as the importance of database updating is increased, the work has been started on dynamic SSE. Kamara et al. [10] first have introduced a dynamic single keyword search scheme based on encrypted inverted index. There are remarkable works on single keyword search on dynamic database. However, file-injection attack, by Zhang et al. [26] have forced the researchers to think about dynamic SSE schemes to be forward private. It is easy to achieve forward privacy with ORAM. However, due to large cost of communication, computation and storage, ORAM based schemes are almost impractical.

In 2016 Bost [2] has presented a non-ORAM based forward private dynamic SSE scheme. Later, few more forward private schemes have been proposed.

Though, till now there are no formal attack on non-backward private DSSE schemes, there are works [4] and [20] that provide backward privacy. Other than single keyword search, there are papers that supports complex queries including conjunctive query, Boolean query, range query etc. In most of the above mentioned schemes, the cloud service providers are considered to be honest-but-curious. However, the schemes fails to provide security in presence of malicious cloud server.

Chai and Gong [5] have introduced the first VSSE scheme. They stores the set of document identifiers in a trie like data structure where each node corresponding to some keyword stores identifiers containing it. Cheng et al. [7] have presented a VSSE scheme for static data based on the secure indistinguishability obfuscation. Their scheme also supports boolean queries and provides publicly verifiability on the return result. Ogata and Kurosawa [7] have presented a no-dictionary generic verifiable SSE scheme. Cuckoo hash table is used here for this private verifiable scheme. With multi-owner setting, Liu et al. [13] have presented a VSSE with aggregate keys. Miao et al. [16] presented a VSSE in same multi-owner setting. However, all of the above schemes were for static database and are privately verifiable where the VSSE schemes by Soleimanian and Khazaei [19] and Zhang et al. [25] are publicly verifiable.

The above works are only for static data. There are few works also that deals with complex queries when the data is static. Conjunctive query on static data has been studied by Sun et al. [21], Miao et al. [17], Wang et al. [22], Li et al. [11], Miao et al. [15] etc. These schemes have private verifiability. Boolean range queries on SSE has been studied by Xu et al. [23]. Though, their verification method is public, since the verification is based over blockchain databases it has good monetary cost. Besides, Monir Azraoui [1] presented a conjunctive search that is publicly verifiable.

Dynamic verifiable SSE with complex queries also has been studied. Zhu et al. [27] presented a dynamic fuzzy keyword search scheme which is privately verifiable and Jiang et al. [9] has studied Publicly Verifiable Boolean Query on dynamic database. Moreover, single keyword search scheme on dynamic data is described by Yoneyama and Kimura [24], Bost et al. [3] etc.

A publicly verifiable SSE scheme is recently also proposed by Miao et al. [14]. Yoneyama and Kimura [24] presented a scheme based on Algebraic PRF which is verifiable as well as forward private that performs single keyword search. However, the scheme is privately verifiable and the owner requires significant amount of computation for verification.

Our proposed scheme Ψ_f is generic forward private verifiable scheme which is compatible with any existing forward private DSSE scheme. Our scheme also do not use any extra owner-storage for verifiability and has minimal search time computation for the owner.

3 Preliminaries

3.1 Cryptographic Tools

Bilinear Map Let $\mathbb{G}$ and $\mathbb{G}_T$ be two (multiplicative) cyclic groups of prime order q. Let $\mathbb{G} =< g >$. A map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is said to be an *admissible non-degenerate bilinear map* if– a) $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$, $\forall u, v \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}$ (bilinearity) b) $\hat{e}(g, g) \neq 1$ (non-degeneracy) c) $\hat{e}$ can be computed efficiently.

Bilinear Hash Given a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ and a generator g, a bilinear hash $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{G}$ maps every random string to an element of $\mathbb{G}$. The map is defined as $\mathcal{H}(m) = g^m,\ \forall m \in \{0,1\}^*$.

Bilinear Signature (BLS) Let $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map where $|\mathbb{G}| = |\mathbb{G}_T| = q$, a prime and $\mathbb{G} =< g >$. A bilinear signature (BLS) scheme $\mathcal{S}$=(`Gen`, `Sign`, `Verify`) is a tuple of three algorithms as follows

- $(sk, pk) \leftarrow$ `Gen`: It selects $\alpha \xleftarrow{\$} [0, q-1]$. It keeps the private key $sk = \alpha$. publishes the public key $pk = g^\alpha$.
- $\sigma \leftarrow$ `Sign`(sk, m): Given $sk = \alpha$, and some message m, it outputs the signature $\sigma = (\mathcal{H}(m))^\alpha = (g^m)^\alpha$ where $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{G}$ is a bilinear hash.
- $\{0/1\} \leftarrow$ `Verify`(pk, m, σ): Return whether $\hat{e}(\sigma, g) = \hat{e}(\mathcal{H}(m), g^\alpha)$

3.2 System Model

In this section, we briefly describe the system model considered in this paper. In our model of verifiable SSE, there are three entities–Owner, Auditor and Cloud. We briefly describe them as follows.

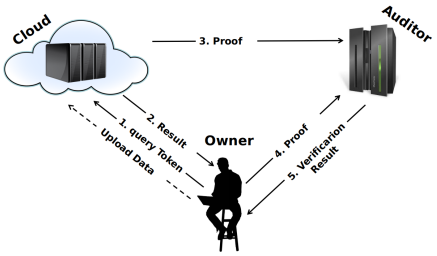

Fig. 1. The system model

1. **Owner:** Owner is the owner as well as user of the database. It is considered to be *trusted*. It builds an secure index, encrypts the data and then outsources both to the cloud. Later, it sends encrypted query to the cloud for searching. Therefore, it is the querier as well. It is the client who requires the service.
2. **Cloud:** Cloud or the cloud server is the storage and computation service provider. It stores the encrypted data sent from the owner and gives result of the query requested by it. The cloud is assumed to be *malicious*. It can deviate from protocol by not only computing on, or not storing the data but also making the querier fool by returning incorrect result.

3. **Auditor:** Auditor is an *honest-but-curious* authority which does not collude with the cloud. Its main role is to verify whether the cloud executes the protocol honestly. It tells the querier whether the returned result is correct or not.

The system model is shown in the Fig. 1.

3.3 Design Goals

Assuming the above system model, we aim to provide solution of the verifiability problem of existing forward private schemes. In our design, we take care to achieve the following objectives.

1. **Confidentiality:** The cloud servers should not get any information about the uploaded data. On the other hand, queries should not leak any information about the database. Otherwise the cloud may get knowledge about the plaintext information.
2. **Efficiency:** In our model, the cloud has a large amount of computational power as well as good storage. The owner is weak. So, in the scheme the owner should require significantly small amount of computation and storage cost while performing verifiability.
3. **Scalability:** Since, the owner have to pay for the service provided by the cloud, it is desirable to outsource as much data as possible. The owner should capable to outsource large amount of data to the cloud. On the other hand, the cloud should answer the queries fast using less computation power.
4. **Forward privacy:** It is observed previously that a DSSE scheme without forward privacy is vulnerable to even honest-but-curious adversary. So, our target is to make a publicly verifiable DSSE scheme without loosing its forward privacy property.

3.4 Definitions

Let $\mathcal{W}$ be a set of keywords. $\mathcal{D}$ be the space of document identifiers and $\mathcal{DB}$ be the set of documents to be outsourced. Thus, $\mathcal{DB} \subseteq \mathcal{D}$. For each keyword $w \in \mathcal{W}$, the set of document identifiers that includes w is denoted by $DB(w) = \{id_1^w, id_1^w, \ldots, id_{c_w}^w\}$, where $c_w = |DB(w)|$ and $id_i^w \in \mathcal{DB}$. Thus, $\bigcup_{w \in \mathcal{W}} DB(w) \subseteq \mathcal{DB}$. Let $\overline{DB} = \{c_{id} : id \in \mathcal{D}\}$ where c_{id} denotes the encrypted document that has identifier id.

We assume that there is a one-way function H' that maps each identifier id to certain random numbers. These random numbers is used as document name corresponding to the identifier. The function is can be computed by both the owner and cloud. However, from a document name, the identifier can not be recovered. Throughout, we use identifiers. However, when we say cloud returns documents to the owner, we assume the cloud performs the function on every identifiers before returning them.

Let, $H : \{0,1\}^* \to \{0,1\}^\lambda$ be a cryptographic hash function, $\mathcal{H}$ be a bilinear hash, $R : \{0,1\}^* \to \{0,1\}^*$ be a PRNG and $F : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\lambda$ be a HMAC. A *stateful algorithm* stores its previous states and use them to compute the current state.

3.5 Verifiable Dynamic Searchable Symmetric Encryption (VDSSE)

An SSE scheme allows a client to outsource a dataset it owns to a cloud service provider in encrypted form without loosing the ability to perform query over the data. The most popular query is the keyword search where the dataset is a collection of documents. The client can retrieve partial encrypted data without revealing any meaningful information to the cloud. Throughout we take query as single keyword search query.

A *dynamic SSE* (DSSE) scheme is a SSE scheme that supports updates. A *Verifiable DSSE* (VDSSE) scheme is a DSSE scheme together with verifiability. The verification can be done either by an external auditor or the owner. The primary reason to bring a auditor is to reduce computational costs of verifiability at owner-side. This allows an owner to be lightweight.

Though a VDSSE scheme supports update, we do not verify whether the cloud updates the database correctly or not. We only want to get the correct result with respect to current state of the database. If cloud updates the database incorrectly, it can not give the actual result. Due to verifiability, it will be failed in verification process to the auditor. We define a verifiable DSSE scheme formally as follows.

Definition 1 (Verifiable Dynamic SSE). *A verifiable dynamic SSE (VDSSE) scheme Ψ is a tuple $\Psi = (\texttt{VKeyGen}, \texttt{VBuild}, \texttt{VSearchToken}, \texttt{VSearch}, \texttt{VUpdateToken}, \texttt{VUpdate})$ of algorithms defined as follows.*

- $K \leftarrow \texttt{VKeyGen}(1^\lambda)$*: It is a probabilistic polynomial-time (PPT) algorithm run by the owner. Given security parameter λ it outputs a key K.*
- $(\overline{DB}, \gamma) \leftarrow \texttt{VBuild}(K, \mathcal{DB})$*: The owner run this PPT algorithm. Given a key K and a set of documents $\mathcal{DB}$, it outputs the encrypted set of documents $\overline{DB}$ and an encrypted index γ.*
- $\tau_s \leftarrow \texttt{VSearchToken}(K, w)$*: On input a keyword w and the key K, the owner runs this PPT algorithm to output a search token τ_s.*
- $(R_w, \nu_w) \leftarrow \texttt{VSearch}(t_s, \gamma)$*: It is a PPT algorithm run by the cloud and the auditor collaboratively that returns a set of document identifiers result R_w to the owner with verification bit ν_w.*
- $\tau_u \leftarrow \texttt{VUpdateToken}(K, id)$*: It is a owner-side PPT algorithm that takes the key K and a document identifier id and outputs a update token τ_u.*
- $(\overline{DB}', \gamma') \leftarrow \texttt{VUpdate}(\tau_u, op, \gamma, \overline{DB})$*: It is a PPT algorithm run by the cloud. It takes an update token τ_u, operation bit op, the encrypted document set $\overline{DB}$ and the index γ and outputs updated $(\overline{DB}', \gamma')$.*

Computational Correctness A VDSSE scheme Ψ is said to be *correct* if $\forall \lambda \in \mathbb{N}$, $\forall K$ generated using $\texttt{KeyGen}(1^\lambda)$ and all sequences of search and update

operations on γ, every search outputs the correct set of identifiers, except with a negligible probability.

Verifiability Note that, when we are saying a scheme is verifiable, it means that it verifies whether the search result is from the currently updated state of the database according to the owner. Verification does not include update of the database at cloud side. For example, let an owner added a document with some keywords and the cloud does not update the database. Later, if the owner searches with some keywords present in the document and it should get the identifier of the document in the result set. Then, the result can be taken as verified.

3.6 Security Definitions

We follow security definition of [19]. There are two parts in the definition– confidentiality and soundness. We define security in adaptive adversary model where the adversary can send query depending on the previous results. Typically, most of the dynamic SSE schemes define its security in this model.

A DSSE, that does not consider verifiability, considers honest-but-curious (HbC) cloud server. In these cases, The owner of the database allows some leakage on every query made. However, it guarantees that no meaningful information about the database are revealed other than the allowed leakages. Soundness definition ensures that the results received form the cloud server are correct.

Confidentiality Confidentiality ensures that a scheme does not give any meaningful information other than it is allowed. In our model, we have considered the cloud to be malicious. However, the auditor is HbC. Since, verifiability has some monetary cost for the owner, it wants verifiability only when it is required. Also the auditor does not have the database and search ability. Given the proof, it only verifies the result. Thus, if the scheme is secure from cloud, it is so from auditor. Again, we have assumed that the cloud and the auditor do not collude. Hence, we do not consider the auditor in our definition of confidentiality.

Definition 2 (CKA2-Confidentiality). *Let* $\Psi = (\texttt{VKeyGen}, \texttt{VBuild}, \texttt{VSearchToken}, \texttt{VSearch}, \texttt{VUpdateToken})$ *be a verifiable DSSE scheme. Let* $\mathcal{A}$, $\mathcal{C}$ *and* $\mathcal{S}$ *be a stateful adversary, a challenger and a stateful simulator respectively. Let* $\mathcal{L}=(\mathcal{L}_{bld}, \mathcal{L}_{srch}, \mathcal{L}_{updt})$ *be a stateful leakage algorithm. Let us consider the following two games.*

$\boldsymbol{Real}_{\mathcal{A}}(\lambda)$*:*

1. *The challenger* $\mathcal{C}$ *generates a key* $K \leftarrow \texttt{VKeyGen}(1^{\lambda})$.
2. $\mathcal{A}$ *generates and sends* $\mathcal{DB}$ *to* $\mathcal{C}$.
3. $\mathcal{C}$ *computes* $(\overline{DB}, \gamma) \leftarrow \texttt{VBuild}(K, \mathcal{DB})$ *and sends* $(\overline{DB}, \gamma)$ *to* $\mathcal{A}$.
4. $\mathcal{A}$ *makes a polynomial number of adaptive queries. In each of them, it sends either a search query for a keyword* w *or an update query for a keyword-document pair* (w, id) *and operation bit* op *to* $\mathcal{C}$.

5. *$\mathcal{C}$ returns either a search token $\tau_s \leftarrow \texttt{VSearchToken}(K, w)$ or an update token $\tau_u \leftarrow \texttt{VUpdateToken}(K, id)$ to $\mathcal{A}$ depending on the query.*
6. *Finally $\mathcal{A}$ returns a bit b that is output by the experiment.*

Ideal$_{\mathcal{A},\mathcal{S}}(\lambda)$*:*

1. *$\mathcal{A}$ generates a set $\mathcal{DB}$ of documents and gives it to $\mathcal{S}$ together with $\mathcal{L}_{bld}(\mathcal{DB})$.*
2. *$\mathcal{S}$ generates $(\overline{DB}, \gamma)$ and sends it to $\mathcal{A}$*
3. *$\mathcal{A}$ makes a polynomial number of adaptive queries q. For each query, $\mathcal{S}$ is given either $\mathcal{L}_{srch}(w, \mathcal{DB})$ or $\mathcal{L}_{updt}(op, w, id)$ depending on the query.*
4. *$\mathcal{S}$ returns, depending on the query q, to $\mathcal{A}$ either search token τ_s or update token τ_u.*
5. *Finally $\mathcal{A}$ returns a bit b' that is output by the experiment.*

We say Ψ is $\mathcal{L}$-secure against adaptive dynamic chosen-keyword attacks if $\forall$ PPT adversary $\mathcal{A}$, $\exists$ a simulator $\mathcal{S}$ such that

$$|Pr[\textbf{\textit{Real}}_{\mathcal{A}}(\lambda) = 1] - Pr[\textbf{\textit{Ideal}}_{\mathcal{A},\mathcal{S}}(\lambda) = 1]| \leq \mu(\lambda) \tag{1}$$

where $\mu(\lambda)$ is negligible in λ.

Soundness The soundness property ensures that if a malicious cloud tries to make the owner fool by returning incorrect result it will be caught to the auditor. We define game-based definition of soundness as follows.

Definition 3. *Let $\Psi =$ (*`VKeyGen`*,* `VBuild`*,* `VSearchToken`*,* `VSearch`*,* `VUpdateToken`*) be a verifiable DSSE scheme. Let us consider the following game.*

sound$_{\mathcal{A},\Psi}(\lambda)$*:*

1. *The challenger $\mathcal{C}$ generates a key $K \leftarrow \texttt{VKeyGen}(1^\lambda)$.*
2. *$\mathcal{A}$ generates and sends $\mathcal{DB}$ to $\mathcal{C}$.*
3. *$\mathcal{C}$ computes $(\overline{DB}, \gamma) \leftarrow \texttt{VBuild}(K, \mathcal{DB})$ and sends $(\overline{DB}, \gamma)$ to $\mathcal{A}$.*
4. *$\mathcal{A}$ makes a polynomial number of adaptive queries. In each of them, it sends either a search query for a keyword w or an update query for a keyword-document pair (w, id) and operation bit op to $\mathcal{C}$.*
5. *$\mathcal{C}$ returns either a search token $\tau_s \leftarrow \texttt{VSearchToken}(K, w)$ or an update token $\tau_u \leftarrow \texttt{VUpdateToken}(K, id)$ to $\mathcal{A}$ depending on the query.*
6. *After making polynomial number of queries, $\mathcal{A}$ chooses a target keyword w and send search query to $\mathcal{C}$.*
7. *$\mathcal{C}$ returns a search token τ_s. $\mathcal{A}$ executes and gets (R_w, ν_w) where $\nu_w = accept$ is verification bit from $\mathcal{C}$.*
8. *$\mathcal{A}$ generates pair (R_w^*) for a keyword w and gets verification bit $\nu_w^* = accept$.*
9. *If $\nu_w^* = accept$ even when $R_w^* \neq DB(w)$, $\mathcal{A}$ returns 1 as output of the game, otherwise returns 0.*

We say that Ψ is sound *if $\forall$ PPT adversaries $\mathcal{A}$, $Pr[\textbf{\textit{sound}}_{\mathcal{A},\Psi}(\lambda) = 1] \leq \mu(\lambda)$.*

4 Verifiable SSE with static data

Since, in a verifiable SSE scheme, there is no update, it does not have `VUpdateToken` or `VUpdate` operation. We present a generic scheme that will make any SSE scheme verifiable. Our target is to achieve verifiability, in presence of malicious server, without loosing any other security property with minimal communication and computational costs.

4.1 Issues with the existing verifiable SSE schemes

There are papers who considered static SSE schemes and suggested authentication tag generation using MAC to protect the integrity of the search result. They generates a tags tag_w, for each keyword w, such that $tag_w = H(id_1^w||id_2^w||\ldots||id_{c_w}^w)$ where H is a one-way hash function. Trivially, if the tags are stored at the owner side then the scheme becomes privately verifiable. In that case, when a search is required, the owner can check integrity after receiving the result from the cloud.

However, this integrity checking does not protect the SSE scheme from malicious adversary *if the tags are outsourced* to the cloud. Checking integrity provides security only from honest-but curious cloud servers. Let us consider an example. Suppose a keyword $w \in \mathcal{W}$ is searched and cloud gets the result $R_w = \{id_1^w, id_2^w, \ldots, id_{c_w}^w, tag_w\}$. Later, if some other keyword w' is searched, the cloud can return the same result and will pass the integrity checking.

4.2 A generic verifiable SSE scheme without client storage

Since, it is desirable to outsource the data as well as tags to the cloud, the above result shows that checking integrity in the above way can not be considered. It is easy to see that the scheme with checking integrity of the result identifiers are not enough because there is no binding of the keyword with the tags. Here, we present a generic idea that makes any SSE scheme verifiable.

Scheme Description Let $\Sigma_s = (\texttt{KeyGen}, \texttt{Build}, \texttt{SearchToken}, \texttt{Search})$ be a result revealing static SSE scheme. We present a VSSE scheme Ψ_s=(`VKeyGen`, `VBuild`, `VSearchToken`, `VSearch`) for static database as follows.

Let H be a one-way hash function and a key K' is chosen at random. For each keyword $w \in \mathcal{W}$, a key $k_w = H(K', w)$ is generated. k_w is then used to bind the keyword with corresponding tag $tag_w = H(k_w||id_1^w||id_2^w||\ldots||id_{c_w}^w)$. Finally, for each keyword w, $\{id_1^w, id_2^w, \ldots, id_{c_w}^w, tag_w\}$ is encrypted at build phase. Thus, while performing search with a keyword w, as search result, the owner receives $\{id_1'^w, id_2'^w, \ldots, id_{c_w}'^w, tag'_w\}$. The owner accepts it if the regenerated tag tag'_w from the received identifiers matched with the received one.

So, the main idea of the scheme is that instead of generating tags only with identifiers, they are bound with k_w which is dependent on w and can be computed by the owner only. After search, if the cloud returns incorrect set of document identifiers then the tag won't get matched. The scheme is shown in Fig. 2.

Note that, for static case, computing tag is enough to validate a result. Since, one-way hash computation is very efficient and requires small amount of resource, we do not consider any external authority like auditor for verifiability. So, the scheme is privately verifiable.

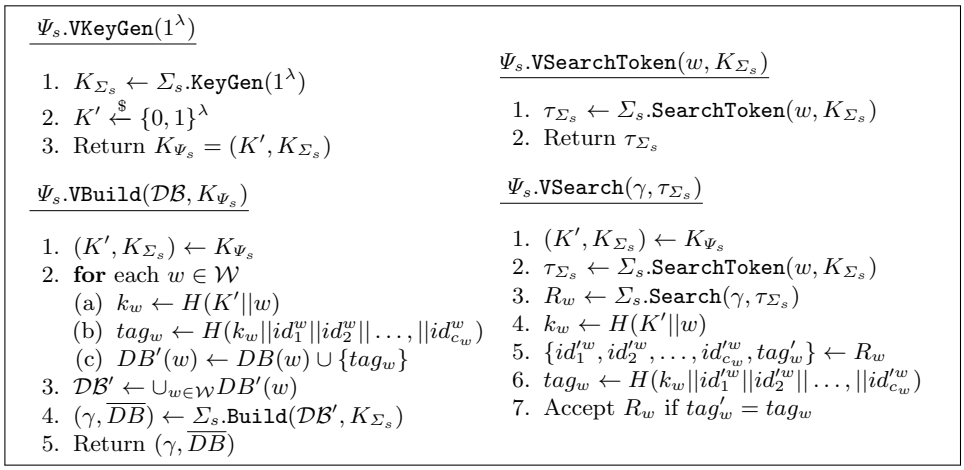

Fig. 2. Algorithm for generic verifiable SSE scheme Ψ_s

Cost for verifiability The cloud storage is increased by $|\mathcal{W}|$ tags. However, depending on the scheme the actual increment might be less than $|\mathcal{W}|$ tags but still it is asymptotically $O(|\mathcal{W}|)$. The communication cost for verifiable search is only increased by one tags from cloud the owner. If we consider computation, to verify a search result, the owner only has to compute a hash value which is very little.

Soundness In case the cloud does not want to perform search properly, then it can not get the identifiers and corresponding tag. So, it has to send either random identifiers or identifiers corresponding to other searched keyword. In both case, It cannot be passed verifiability test to the owner.

Security The security of our proposed scheme follows from the security of the embedded SSE scheme.

5 Our Proposed Forward Secure Publicly Verifiable DSSE scheme

In this section, we propose a simple generic dynamic SSE scheme which is forward secure as well as verifiable. Let Σ_f = (`KeyGen`, `Build`, `Search`, `SearchToken`, `Update`, `UpdateToken`) be a result revealing forward secure dynamic SSE scheme.

It is to be noted that any forward private SSE scheme stores the present state of the database at client side. Corresponding to each keyword, most of

them stores the number of documents containing it. Let $C = \{c_w : w \in \mathcal{W}\}$ be the list of such numbers.

Since, it considers any forward secure scheme Σ_f, it only adds an additional encrypted data structure to make the scheme verifiable. The algorithms of Our proposed scheme are given in Figure 3. They are divided into three phases– initialization, search and update.

Initialization phase: In this phase, secret and public keys are generated by the owner and thereafter the encrypted searchable structure is built. During key generation, three types of keys are generated– K_{Σ_f} for the Σ_f; (sk, pk) for the bilinear signature scheme; and two random strings K_s, K_t for seed and tag generation respectively.

Thereafter, a signature table T_{sig} is generated, before building the secure index γ and encrypted database $\overline{DB}$, to store the signature corresponding to each keyword-document pair. For each pair (w, id_i^w), the position $pos_i^w = F(tag_w, id_i^w||i)$ is generated with a HMAC F. The position is actually act as key of a key-value pair for a dictionary. The document identifier is bounded with pos_i^w together with $tag_w = F(K_t, w)$. The tag_w is fixed for a keyword and is given to the server to find pos_i^w. The signature σ_i^w for the same pair is also bounded with random number r_i^w which can only be generated with the seed s_w. Then (σ_i^w, pos_i^w) pair is added in the table T_{sig} as key-value pair. After the building process, the owner outsources γ, $\overline{DB}$ and T_{sig} to the cloud.

Search Phase: In this phase, the owner first generates a search token τ_{Σ_f} to search on Σ_f. Then, it regenerates the tag_w and the seed s_w. The owner sends τ_{Σ_f} and tag_w to the cloud whereas s_w to the auditor.

The cloud performs search operation according to Σ_f and use the result identifiers $\{id_1, id_2, \ldots id_{c'_w}\}$ to gets the position in T_{sig} corresponding to each pair. It is not able to generate the positions if it does not search for the document identifiers. It collects the signatures stored in those positions, multiplies them and sends multiplication result to the auditor as its part pf_c of the proof. It sends the search result to the owner.

The owner first generates random numbers $\{r_1, r_2, \ldots r_{c'_w}\}$ and regenerates aggregate message $m = \sum_{i=1}^{i=c'_w} r_i.id_i^w \mod q$ of the identifiers and sends m to the auditor as pf_o, owner's part of the proof.

After receiving pf_c and pf_o, the auditor only computes $\mathcal{S}.\texttt{Verify}(pk, m, \sigma')$. It outputs accept if signature verification returns *success*.

Update Phase: In our scheme, we assume that instead updating only a keyword-document pair, a document is added as a whole. To add a document with identifier id and keyword set $\{w_1, w_2, \ldots, w_{n_{id}}\}$, the owner generates the position and the corresponding signature for each containing keyword. The cloud gets them from the owner and adds them in the table T_{sig}.

Correctness For correctness it is enough to check the following.

$$\hat{e}(\mathcal{H}(m), pk) = \hat{e}(g^m, g^\alpha) = \hat{e}(g^{m\alpha}, g) = \hat{e}(g^{\alpha \sum m_i}, g) = \hat{e}(\prod g^{\alpha m_i}, g)$$
$$= \hat{e}(\prod \sigma_i, g) = \hat{e}(\sigma, g)$$

$\Psi_f.\texttt{VKeyGen}(1^\lambda)$

1. $K_{\Sigma_f} \leftarrow \Sigma_f.\texttt{KeyGen}(1^\lambda)$
2. $(sk, pk) \leftarrow \mathcal{S}.\texttt{Setup}(1^\lambda)$
3. $K_s \leftarrow \{0,1\}^\lambda$
4. $K_t \leftarrow \{0,1\}^\lambda$
5. Return $K_{\Psi_f} = (K_t, K_s, sk, pk, K_{\Sigma_f})$

$\Psi_f.\texttt{VBuild}(\mathcal{DB}, K_{\Psi_f})$

1. $T_{sig} \leftarrow$ empty list of size $|\mathcal{W}|$
2. **for** $w \in \mathcal{W}$
   (a) $s_w \leftarrow F(K_s, w)$; $tag_w \leftarrow F(K_t, w)$
   (b) **for** $i = 1$ **to** $c_w (= |DB(w)|)$
      i. $r_i^w \leftarrow R(s_w||i)$;
      ii. $m_i^w \leftarrow r_i^w.id_i^w \mod q$
      iii. $\sigma_i^w \leftarrow \mathcal{S}.\texttt{Sign}(sk, m_i^w)$
      iv. $pos_i^w \leftarrow F(tag_w, id_i^w||i)$
      v. $T_{sig}[pos_i^w] \leftarrow \sigma_i^w$
3. $(\gamma, \overline{DB}) \leftarrow \Sigma_f.Build(\mathcal{DB}, K_{\Sigma_f})$
4. Return $(\gamma, \overline{DB}, T_{sig})$ to the cloud

$\Psi_f.\texttt{VSearchToken}(w, K_{\Psi_f})$

1. $(K_t, K_s, sk, pk, K_{\Sigma_f}) \leftarrow K_{\Psi_f}$
2. $\tau_{\Sigma_f} \leftarrow \Sigma_f.\texttt{SearchToken}(w, K_{\Sigma_f})$
3. $tag_w \leftarrow F(K_t, w)$;
4. $\tau_s^{\Psi_f} \leftarrow (\tau_{\Sigma_f}, tag_w)$
5. Return $\tau_s^{\Psi_f}$ to cloud

$\Psi_f.\texttt{VSearch}(\gamma, \tau_s^{\Psi_f})$

Cloud:

1. Receive $\tau_{\Psi_f} = (\tau_{\Sigma_f}, tag_w)$ from Owner
2. $\{id_1'^w, \ldots, id_{c_w'}'^w\} \leftarrow \Sigma_f.\texttt{Search}(\gamma, \tau_{\Sigma_f})$
3. **for** $i = 1$ **to** c_w'
   (a) $pos_i^w \leftarrow F(tag_w, id_i'^w||i)$
   (b) $\sigma_i' \leftarrow T_{sig}[pos_i^w]$;
4. $\sigma' \leftarrow \prod_{i=1}^{c_w'} \sigma_i'$
5. $R_w \leftarrow \{id_1'^w, id_2'^w, \ldots, id_{c_w'}'^w\}$
6. $pf_c \leftarrow \sigma'$
7. Return pf_c to auditor and R_w to Owner

Owner:

1. Receives R_w
2. $c_w \leftarrow C[w]$
3. If $c_w \neq c_w'$ **Return** reject bit.
4. $s_w \leftarrow F(K_s, w)$
5. **for** $i = 1$ **to** c_w **do**
   (a) $r_i^w \leftarrow R(s_w||i)$
   (b) $m_i^w \leftarrow id_i'^w.r_i^w \mod q$
6. $m = \sum_{i=1}^{c_w} m_i^w \mod q$
7. Send $pf_o = m$ to the auditor

Auditor:

1. Receives $pf_o = m$ from owner and $pf_c = \sigma'$ from cloud
2. $b_v \leftarrow \mathcal{S}.\texttt{Verify}(pk, m, \sigma')$
3. If $b_v = failure$, **Return** *reject*

$\Psi_f.\texttt{VUpdateToken}(K_{\Psi_f}, w, id)$

1. $\tau_u \leftarrow \Sigma_f.\texttt{UpdateToken}(K_{\Sigma_f}, w, id)$
2. Return τ_u

$\Psi_f.\texttt{VUpdate}(T_{tag}, \gamma, \tau_u)$

Owner:

1. $\{w_1, w_2, \ldots, w_{n_{id}}\} \in id$
2. **for** $i = 1$ **to** n_{id}
   (a) $\tau_u \leftarrow \Psi_f.\texttt{VUpdateToken}(K_{\Psi_f}, w_i, id)$ $\forall i \in [c_w]$
   (b) $b_v \leftarrow \Sigma_f.\texttt{Update}(\gamma, \tau_u)$
   (c) **if** $b_v \neq succsess$ Return
3. **for** $i = 1$ **to** n_{id}
   (a) $tag_{w_i} \leftarrow F(K_t, w_i)$
   (b) $c_{w_i} \leftarrow C[w_i]$
   (c) $s_w \leftarrow F(K_s, w)$;
   (d) $r \leftarrow R(s_w||(c_{w_i} + 1))$
   (e) $m \leftarrow id.r \mod q$
   (f) $\sigma_i \leftarrow \mathcal{S}.\texttt{Sign}(sk, m)$
   (g) $pos_i \leftarrow F(tag_{w_i}, id||(c_{w_i} + 1))$
   (h) $C[w] = C[w] + 1$
4. $pos \leftarrow \{pos_1, pos_2, \ldots, pos_{n_{id}}\}$
5. $\sigma \leftarrow \{\sigma_1, \sigma_2, \ldots, \sigma_{n_{id}}\}$
6. send $\tau_u^{\Psi_f} = (pos, \sigma)$ to cloud

Cloud:

1. $\{pos_1, pos_2, \ldots, pos_{n_{id}}\} \leftarrow pos$
2. $\{\sigma_1, \sigma_2, \ldots, \sigma_{n_{id}}\} \leftarrow \sigma$
3. $T_{sig}[pos_i] \leftarrow \sigma_i, \ \forall i \in [n_{id}]$

Fig. 3. Generic verifiable dynamic SSE scheme Ψ_f without extra client storage

Cost for verifiability We achieve, forward privacy as well as public verifiability without client storage in Ψ_f. This increases the cloud-storage by $O(N)$, where N is the number of document-keyword pairs. The proof has two parts one from the client and another from the owner. For a keyword w, the sizes of them are one group element and one random λ-bit string only. Thus Auditor receives one element from both. The owner has to compute R_w integer multiplication and addition, and then has to send one element.

Forward privacy We can see that while adding a document, it only adds some keyword-document pair, in the form of key-value pairs. So, During addition, the cloud server is adding key-value pairs in the dictionary. From these pairs, it can not guess the keywords present in it. Again, when it perform searches, it gets about the key (i.e., position on the table) only when it gets the identifiers. The one possibility to get the newly added key-value pair linked with the previous is if the added document gives the identifier of it. Since, the one-way function H' gives the document-name of the adding document, the cloud server can not linked it with the previously searched keywords.

5.1 Security

The security of the scheme is shown in two parts– confidentiality and soundness.

Soundness The cloud server can cheat the owner in three ways by sending–

1. Incorrect number of identifiers– but it is not possible as the owner keeps the number of identifiers.
2. Same size result of other keywords– m is generated with a random numbers which can be generated only with the searched keyword and signatures are bound with that. So, the signature verification will be failed.
3. Result with some altered identifiers– since signatures are bounded with keywords and the random number, altering any will change m and similarly the signature verification will be failed.

Thus the owner always will get the correct set of document identifiers.

Confidentiality Let $\mathcal{L}^{\Sigma_f} = (\mathcal{L}^{\Sigma_f}_{bld}, \mathcal{L}^{\Sigma_f}_{srch}, \mathcal{L}^{\Sigma_f}_{updt})$ the leakage function of Σ_f. Let $\mathcal{L}^{\Psi_f} = (\mathcal{L}^{\Psi_f}_{bld}, \mathcal{L}^{\Psi_f}_{srch}, \mathcal{L}^{\Psi_f}_{updt})$ be the leakage function of our scheme Ψ_f, given as follows.

$$\mathcal{L}^{\Psi_f}_{bld}(\mathcal{DB}) = \{\mathcal{L}^{\Sigma_f}_{bld}(\mathcal{DB}), |T_{sig}|\}$$
$$\mathcal{L}^{\Psi_f}_{srch}(w) = \{\mathcal{L}^{\Sigma_f}_{srch}(w), \{(id^w_i, pos^w_i, \sigma^w_i) : i = 1, 2, \ldots, c_w\}\}$$
$$\mathcal{L}^{\Psi_f}_{updt}(f) = \{id, \{(\mathcal{L}^{\Sigma_f}_{updt}(w_i, id), pos^{w_i}, \sigma^{w_i}) : i = 1, 2, \ldots, n_{id}\}\}$$

We show that Ψ is $\mathcal{L}^{\Psi_f}$-secure against adaptive dynamic chosen-keyword attacks in the random oracle model, in the following theorem.

Theorem 1. *If F is a PRF, $\mathcal{S}$ is CPA2 secure bilinear signature scheme and Σ_f is $\mathcal{L}^{\Sigma_f}$-secure, then Ψ_f is $\mathcal{L}^{\Psi_f}$-secure against adaptive dynamic chosen-keyword attacks.*

Proof. To prove the above theorem, it is sufficient to show that there exists a simulator $\mathtt{Sim}_{\Sigma_f}$ such that $\forall$ PPT adversary $\mathcal{A}$, the output of $\mathbf{Real}_{\mathcal{A}}(\lambda)$ and $\mathbf{Ideal}_{\mathcal{A},\mathtt{Sim}_{\Sigma_f}}(\lambda)$ are computationally indistinguishable. We construct such a

Simulating F We simulate R with a table RO. Given (x, y), If $RO[(x, y)] = \perp$, then do $RO[(x, y)] \leftarrow \{0,1\}^\lambda$ and return $RO[(x, y)]$, else return the existing value $RO[(x, y)]$.

Simulating Build Leakage function is given by $\mathcal{L}^{\Psi_f}_{bld}(\mathcal{DB}) = \{\mathcal{L}^{\Sigma_f}_{bld}(\mathcal{DB}), |T_{sig}|\}$. Let S_{bld} be returned by the simulator $\mathtt{Sim}_{\Sigma_f}$. Let us consider a table $\widetilde{T}_{tag}$. For each keyword w it stores a random λ-bit string. On input w, it returns $\widetilde{tag}_w \leftarrow \widetilde{T}_{tag}(w)$. $\mathtt{Sim}_{\Psi_f}$ keeps an extra table $\widetilde{T}'_{sig}$ such that it indicates whether the entry is queried or not. The simulation is done as follows.

1. Take empty tables $\widetilde{T}_{sig}$ and $\widetilde{T}'_{sig}$
2. For each $i = 1$ to $i = |T_{sig}|$ do
   (a) $pos_i \xleftarrow{\$} \{0,1\}^\lambda$
   (b) $val_i \xleftarrow{\$} \{0,1\}^\lambda$
   (c) $\widetilde{T}_{sig}[pos_i] \leftarrow val_i$
   (d) $\widetilde{T}'_{sig}[pos_i] \leftarrow 0$
3. Simulate Σ_f with $S_{bld} \leftarrow \mathtt{Sim}_{\Sigma_f}(\mathcal{DB})(\mathcal{L}^{\Sigma_f}_{bld}(\mathcal{DB}))$
4. return $(S_{bld}, \widetilde{T}_{sig})$ and keeps $\widetilde{T}'_{sig}$

Simulating Search token Leakage function for a queried keyword w is given by $\mathcal{L}^{\Psi_f}_{srch}(w) = \{\mathcal{L}^{\Sigma_f}_{srch}(w), \{(id^w_i) : i = 1, 2, \ldots, c_w\}\}$.
We keep a table RO where $(\widetilde{tag}_w, id, i)$ is the key and pos is the value. Given search leakage corresponding to the keyword w, $\mathtt{Sim}_{\Psi_f}$ does the following things.

1. If $\widetilde{T}_{tag}[w]$ is null, i.e, the keyword is searched first time
   (a) $\widetilde{tag}_w \xleftarrow{\$} \{0,1\}^\lambda$
   (b) $\widetilde{T}_{tag}[w] \leftarrow \widetilde{tag}_w$
   Else
   (a) $\widetilde{tag}_w \leftarrow \widetilde{T}_{tag}[w]$
2. If $RO[(\widetilde{tag}_w, id^w_i, i)]$ is not null,
   (a) $pos_i \leftarrow RO[(\widetilde{tag}_w, id^w_i, i)]$
   Else
   (a) $pos_i \leftarrow$ a random pos_i such that $\widetilde{T}'_{sig}[pos_i] = 0$
   (b) $RO[(\widetilde{tag}_w, id^w_i, i)] \leftarrow pos_i$
   (c) $\widetilde{T}'_{sig}[pos_i] \leftarrow 1$
3. Simulate Σ_f with $\widetilde{\tau}_{\Sigma_f} \leftarrow \mathtt{Sim}_{\Sigma_f}(\mathcal{L}^{\Sigma_f}_{srch}(w))$
4. return $\widetilde{\tau}^{\Psi_f}_s = (\widetilde{\tau}_{\Sigma_f}, \widetilde{tag}_w)$

Simulating Update token Leakage function to add a document f with identifier id containing keyword set $\{w_1, w_2, \ldots, w_{n_w}\}$ is given by $\mathcal{L}^{\Psi_f}_{updt}(f) = \{H'(id), \{(\mathcal{L}^{\Sigma_f}_{updt}(w_i, id)) : i = 1, 2, \ldots, n_{id}\}\}$.

1. For each keyword $w_i \in f$
   (a) $\widetilde{\tau}^i_u \leftarrow \mathtt{Sim}_{\Sigma_f}(\mathcal{L}^{\Sigma_f}_{updt}(w, id))$
   (b) If $\widetilde{T}_{tag}[w_i]$ is null, i.e, the keyword is searched first time
      i. $\widetilde{tag}_{w_i} \xleftarrow{\$} \{0,1\}^\lambda$
      ii. $\widetilde{T}_{tag}[w_i] \leftarrow \widetilde{tag}_w$
      Else
      i. $\widetilde{tag}_{w_i} \leftarrow \widetilde{T}_{tag}[w_i]$
   (c) $c_{w_i} \leftarrow C[w_i] + 1$
   (d) If $RO[(\widetilde{tag}_{w_i}, id, (c_{w_i} + 1))]$ is not null,
      i. $\widetilde{pos}_i \leftarrow RO[(\widetilde{tag}_{w_i}, id, (c_v + 1))]$
      Else
      i. $\widetilde{pos}_i \leftarrow$ a random pos_i such that $\widetilde{T}_{sig}[pos_i]$ is null
      ii. $RO[(\widetilde{tag}_{w_i}, id, (c_{w_i} + 1))] \leftarrow \widetilde{pos}_i$
      iii. $\widetilde{T}'_{sig}[pos_i] \leftarrow 1$
   (e) $\widetilde{\sigma}_i \xleftarrow{\$} \mathbb{G}$
2. $\widetilde{pos} \leftarrow \{\widetilde{pos}_1, \widetilde{pos}_2, \ldots, \widetilde{pos}_{n_{id}}\}$
3. $\widetilde{\sigma} \leftarrow \{\widetilde{\sigma}_1, \widetilde{\sigma}_2, \ldots, \widetilde{\sigma}_{n_{id}}\}$
4. Return $\widetilde{\tau}^{\Psi_f}_u = (\widetilde{pos}, \widetilde{\sigma})$

Fig. 4. Simulation of build, search token and update token

simulator $\mathtt{Sim}_{\Sigma_f}$ which adaptively simulates the extra data structure T_{sig} and query tokens. Let $\mathtt{Sim}_{\Sigma_f}$ be the simulator of the Σ_f. We simulate the algorithms in Figure 4.

From the simulation, it can be seen that the indistinguishability of T_{sig} and $\widetilde{T}_{sig}$ follows from pseudo-randomness of F and CPA2 security of $\mathcal{S}$. Besides, the

indistinguishability of $\widetilde{\tau}_u^{\Psi_f}, \widetilde{\tau}_s^{\Psi_f}$ with respect to $\tau_s^{\Psi_f}, \tau_u^{\Psi_f}$ respectively follows from the pseudo-randomness of F.

5.2 Deletion Support

Ψ_f can be extended to deletion support by duplicating it. Together with Ψ_f for addition, a duplicate Ψ'_f can be kept for deleted files. During search, the auditor verifies both separately. The client gets result from both Ψ_f and Ψ'_f, accepts only if both are verified and gets the final result calculating the difference.

6 Comparison with existing schemes

Our generic VSSE Ψ_s requires only one hash-value computation to verify a search which is optimal. Again, during building, the owner requires $2|\mathcal{W}|$ extra hash-value computation twice of the optimal. We can take that much computation to protect the scheme from malicious server without any extra client storage.

Table 2. Comparison of verifiable dynamic SSE schemes

Scheme Name	Forward privacy	Public verifiability	Extra Storage		Extra Computation			Extra Communication	
			owner	cloud	owner	cloud	auditor	owner	auditor
Yoneyama and Kimura [24]	✓	×	$O(\|\mathcal{W}\|)$	$O(\|\mathcal{W}\|log\|\mathcal{DB}\|)$	$O(\|R_w\|)$	$O(\|R_w\|)$	–	$O(1)$	–
Bost and Fouque [3]	×	×	$O(\|\mathcal{W}\|)$	$O(\|\mathcal{W}\|)$	$O(\|R_w\|)$	$O(1)$	–	$O(1)$	–
Miao et al. [14]	×	✓	$O(\|\mathcal{W}\|)$	$O(N+\|\mathcal{W}\|)$	$O(\|R_w\|)$	$O(\|R_w\|)$	–	$O(1)$	–
Zhu et al. [27]	×	×	$O(1)$	$O(1)$	$O(\|R_w\|)$	$O(\|R_w\|+N)$	–	$O(\|R_w\|)$	–
Jiang et al. [9]	×	✓	$O(1)$	$O(\|\mathcal{W}\|)$	$O(\log\|\mathcal{W}\|)$	$O(\|R_w\|+N)$	–	$O(1)$	–
Ψ_f	✓	✓	$O(1)$	$O(N)$	$O(\|R_w\|)$	$O(\|R_w\|)$	$O(1)$	$O(1)$	$O(1)$

Where N is the #keyword-doc pairs. Here extra storage is calculated over all storage, extra communication and computation are for a single search.

We have compared our verifiable DSSE scheme Ψ_f with verifiable dynamic schemes by Yoneyama and Kimura [24], Bost and Fouque [3], Miao et al. [14], Zhu et al. [27] and Jiang et al. [9]. The comparison is shown in Table 2. From the table, it can be observed that Ψ_f is very efficient with respect to low resource owner. Extra computation needed by the owner, to verify the search, is only $|R_w|$ multiplication which very less from the others. The owner also does not require any extra storage than the built in forward secure DSSE scheme.

7 Conclusion

Throughout, we have seen that we have successfully presented a privately verifiable SSE scheme and a publicly verifiable DSSE scheme. Both of them are simple and easy to implement. Moreover, the VDSSE scheme achieves forward secrecy. In both of the scheme we have achieved our target to make efficient for low-resource owner. Due to low computational and communication cost, we do need any auditor for VSSE. However, presence of an auditor, who verifies the search result, reduces workload of the owner. Our proposed schemes are only for single keyword search queries. There are many other complex queries too. As a future work, one can design complex queried verifiable DSSE scheme. On the other hand, while designing, keeping them forward secret is also a challenging direction of research.

References

1. Azraoui, M., Elkhiyaoui, K., Önen, M., Molva, R.: Publicly verifiable conjunctive keyword search in outsourced databases. In: 2015 IEEE Conference on Communications and Network Security, CNS 2015, Florence, Italy, September 28-30, 2015. pp. 619–627 (2015)
2. Bost, R.: $\sum o\varphi o\varsigma$: Forward secure searchable encryption. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016. pp. 1143–1154 (2016)
3. Bost, R., Fouque, P., Pointcheval, D.: Verifiable dynamic symmetric searchable encryption: Optimality and forward security. IACR Cryptology ePrint Archive **2016**, 62 (2016)
4. Bost, R., Minaud, B., Ohrimenko, O.: Forward and backward private searchable encryption from constrained cryptographic primitives. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017. pp. 1465–1482 (2017)
5. 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, Ottawa, ON, Canada, June 10-15, 2012. pp. 917–922 (2012)
6. Chase, M., Kamara, S.: Structured encryption and controlled disclosure. In: Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings. pp. 577–594 (2010)
7. Cheng, R., Yan, J., Guan, C., Zhang, F., Ren, K.: Verifiable searchable symmetric encryption from indistinguishability obfuscation. In: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '15, Singapore, April 14-17, 2015. pp. 621–626 (2015)
8. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006. pp. 79–88 (2006)
9. Jiang, S., Zhu, X., Guo, L., Liu, J.: Publicly verifiable boolean query over outsourced encrypted data. In: 2015 IEEE Global Communications Conference, GLOBECOM 2015, San Diego, CA, USA, December 6-10, 2015. pp. 1–6 (2015)
10. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In: the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012. pp. 965–976 (2012)
11. Li, Y., Zhou, F., Qin, Y., Lin, M., Xu, Z.: Integrity-verifiable conjunctive keyword searchable encryption in cloud storage. Int. J. Inf. Sec. **17**(5), 549–568 (2018)
12. van Liesdonk, P., Sedghi, S., Doumen, J., Hartel, P.H., Jonker, W.: Computationally efficient searchable symmetric encryption. In: Secure Data Management, 7th VLDB Workshop, SDM 2010, Singapore, September 17, 2010. Proceedings. pp. 87–100 (2010)
13. Liu, Z., Li, T., Li, P., Jia, C., Li, J.: Verifiable searchable encryption with aggregate keys for data sharing system. Future Generation Comp. Syst. **78**, 778–788 (2018)
14. Miao, M., Wang, J., Wen, S., Ma, J.: Publicly verifiable database scheme with efficient keyword search. Inf. Sci. **475**, 18–28 (2019)
15. Miao, Y., Ma, J., Liu, X., Jiang, Q., Zhang, J., Shen, L., Liu, Z.: VCKSM: verifiable conjunctive keyword search over mobile e-health cloud in shared multi-owner settings. Pervasive and Mobile Computing **40**, 205–219 (2017)

16. Miao, Y., Ma, J., Liu, X., Zhang, J., Liu, Z.: VKSE-MO: verifiable keyword search over encrypted data in multi-owner settings. SCIENCE CHINA Information Sciences **60**(12), 122105:1–122105:15 (2017)
17. Miao, Y., Ma, J., Wei, F., Liu, Z., Wang, X.A., Lu, C.: VCSE: verifiable conjunctive keywords search over encrypted data without secure-channel. Peer-to-Peer Networking and Applications **10**(4), 995–1007 (2017)
18. Ogata, W., Kurosawa, K.: Efficient no-dictionary verifiable searchable symmetric encryption. In: Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers. pp. 498–516 (2017)
19. Soleimanian, A., Khazaei, S.: Publicly verifiable searchable symmetric encryption based on efficient cryptographic components. Des. Codes Cryptography **87**(1), 123–147 (2019)
20. Sun, S., Yuan, X., Liu, J.K., Steinfeld, R., Sakzad, A., Vo, V., Nepal, S.: Practical backward-secure searchable encryption from symmetric puncturable encryption. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018. pp. 763–780 (2018)
21. 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: 2015 IEEE Conference on Computer Communications, INFOCOM 2015, Kowloon, Hong Kong, April 26 - May 1, 2015. pp. 2110–2118 (2015)
22. Wang, J., Chen, X., Sun, S., Liu, J.K., Au, M.H., Zhan, Z.: Towards efficient verifiable conjunctive keyword search for large encrypted database. In: Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part II. pp. 83–100 (2018)
23. Xu, C., Zhang, C., Xu, J.: vchain: Enabling verifiable boolean range queries over blockchain databases. CoRR **abs/1812.02386** (2018)
24. Yoneyama, K., Kimura, S.: Verifiable and forward secure dynamic searchable symmetric encryption with storage efficiency. In: Information and Communications Security - 19th International Conference, ICICS 2017, Beijing, China, December 6-8, 2017, Proceedings. pp. 489–501 (2017)
25. Zhang, R., Xue, R., Yu, T., Liu, L.: PVSAE: A public verifiable searchable encryption service framework for outsourced encrypted data. In: IEEE International Conference on Web Services, ICWS 2016, San Francisco, CA, USA, June 27 - July 2, 2016. pp. 428–435 (2016)
26. Zhang, Y., Katz, J., Papamanthou, C.: All your queries are belong to us: The power of file-injection attacks on searchable encryption. In: 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016. pp. 707–720 (2016)
27. Zhu, X., Liu, Q., Wang, G.: A novel verifiable and dynamic fuzzy keyword search scheme over encrypted data in cloud computing. In: 2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, August 23-26, 2016. pp. 845–851 (2016)