Efficient No-dictionary Verifiable Searchable Symmetric Encryption

Wakaha Ogata[1] and Kaoru Kurosawa[2]

[1] Tokyo Institute of Technology
ogata.w.aa@m.titech.ac.jp
[2] Ibaraki University
kaoru.kurosawa.kk@vc.ibaraki.ac.jp

Abstract. In the model of *no-dictionary* verifiable searchable symmetric encryption (SSE) scheme, a client does not need to keep the set of keywords $\mathcal{W}$ in the search phase, where $\mathcal{W}$ is called a dictionary. Still a malicious server cannot cheat the client by saying that "your search word w does not exist in the dictionary $\mathcal{W}$" when it exists. In the previous such schemes, it takes $O(\log m)$ time for the server to prove that $w \notin \mathcal{W}$, where $m = |\mathcal{W}|$ is the number of keywords.
In this paper, we show a generic method to transform any SSE scheme (that is only secure against passive adversaries) to a *no-dictionary* verifiable SSE scheme. In the transformed scheme, it takes only $O(1)$ time for the server to prove that $w \notin \mathcal{W}$.

keywords. searchable symmetric encryption, verifiable, dictionary

1 Introduction

The notion of searchable symmetric encryption (SSE) schemes was introduced by Song et al. [30]. In the store phase, a client encrypts a set of files and an index table by a symmetric encryption scheme, and then stores them on an untrusted server. In the search phase, he can efficiently retrieve the matching files for a search keyword w keeping the keyword and the files secret.

Since then, single keyword search SSE schemes [18, 15, 16, 23, 25], dynamic SSE schemes [21, 20, 24, 13, 28, 26], multiple keyword search SSE schemes [19, 1, 7, 32, 12, 22] and more [14] have been studied extensively by many researchers.

Curtmola, et al. [16, 17] gave a rigorous definition of privacy against honest but curious servers. Kurosawa and Ohtaki [23, 25] showed a definition of reliability against malicious servers who may return incorrect search results to the client, or may delete some encrypted files to save her memory space. Kurosawa and Ohtaki [23, 25] also proved a weak equivalence between the UC security and the stand alone security (i.e., the privacy and the reliability), where the UC security is a very strong security notion such that if a protocol Π is UC secure, then its security is preserved under a general protocol composition operation [9].

Now in the model of *no-dictionary* verifiable SSE scheme, a client does not need to keep the set of keywords $\mathcal{W}$ in the search phase, where $\mathcal{W}$ is called a dictionary. Still a malicious server cannot cheat the client by saying that "your search word w does not exist in the dictionary $\mathcal{W}$" if it exists. This model is really practical, but it is not an easy task to prove that $w \notin \mathcal{W}$.

Recently, Taketani and Ogata [31] constructed a *no-dictionary* verifiable SSE scheme. In their scheme, it takes $O(N \log mN)$ time for the server to prove that $w \notin \mathcal{W}$, where $m = |\mathcal{W}|$ is the number of keywords and N is the number of documents.

Very recently, Bost et al. [6] proposed a generic construction of *no-dictionary* verifiable SSE schemes as an independent work of ours. The idea of their concrete scheme (Algorithm 1 in [6]) is similar to that of Taketani and Ogata [31], and it takes $O(\log m) + time(\texttt{Search}_0)$ time for the server to prove that $w \notin \mathcal{W}$, [1] where $time(\texttt{Search}_0)$ is the search time in the underlying non-verifiable scheme. They further claim that their generic construction works for dynamic SSE schemes as well. However, they do not show how to instantiate it.

In this paper, we show a generic method to transform any SSE scheme (that is only secure against passive adversaries) to a *no-dictionary* verifiable SSE scheme. In the transformed scheme, it takes only $O(1)$ time for the server to prove that $w \notin \mathcal{W}$. The search time for $w \in \mathcal{W}$ remains almost the same as that of the original SSE scheme. We also prove that the transformed scheme is UC-secure in Appendix similarly to [23, 25].

2 Verifiable Searchable Symmetric Encryption

In this section, we define a no-dictionary (verifiable) SSE scheme and its security. Basically, we follow the notation used in [23, 25, 12].

- Let $\mathcal{D} = \{D_1, \ldots, D_N\}$ be a set of documents.
- Let $\mathcal{W} \subset \{0,1\}^*$ be a set of keywords. We call $\mathcal{W}$ a dictionary.
- For $w \in \{0,1\}^*$, define

$$\mathcal{D}(w) = \begin{cases} \text{the set of documents that contain } w & \text{if } w \in \mathcal{W} \\ \emptyset & \text{otherwise} \end{cases}$$

- Let $\mathcal{C} = \{C_1, \ldots, C_N\}$, where C_i is a ciphertext of D_i.
- Let

$$\mathcal{C}(w) = \{C_i \mid C_i \text{ is a ciphertext of } D_i \in \mathcal{D}(w)\}. \qquad (1)$$

  Note that $\mathcal{C}(w) = \emptyset$ if $w \notin \mathcal{W}$.

If X is a bit string, $|X|$ denotes the bit length of X. If X is a set, $|X|$ denotes the cardinality of X. "PPT" refers to probabilistic polynomial time, and "PT" refers to polynomial time.

[1] This is because the server needs to find $i \in \{1, \ldots, m\}$ such that $key_i < PRF_k(w) < key_{i+1}$, where $PRF_k(w)$ is sent to the server by the client in the search phase, $\{key_1, \ldots, key_m\} = \{PRF_k(w_j) \mid w_j \in \mathcal{W}\}$ is stored on the server in the store phase and $key_1 < \ldots < key_m$. PRF_k denotes a pseudo-random function with key k.

2.1 Model

An SSE scheme has two phases, the store phase (which is executed only once) and the search phase (which is executed a polynomial number of times). In the store phase, the client encrypts all documents in $\mathcal{D}$ and stores them on the server. In the search phase, the client sends a ciphertext of a word w, and the server returns $\mathcal{C}(w)$. If there is a mechanism to verify the validity of $\mathcal{C}(w)$, the scheme is called a verifiable SSE (vSSE).

Formally, a vSSE scheme consists of the following six polynomial-time algorithms $\texttt{vSSE} = (\texttt{Gen}, \texttt{Enc}, \texttt{Trpdr}, \texttt{Search}, \texttt{Dec}, \texttt{Verify})$ such that

- $K \leftarrow \texttt{Gen}(1^\lambda)$: a PPT algorithm that generates a key K, where λ is a security parameter. This algorithm is run by the client in the store phase.
- $(\mathcal{I}, \mathcal{C}) \leftarrow \texttt{Enc}(K, \mathcal{D}, \mathcal{W}, \{(w, \mathcal{D}(w)) \mid w \in \mathcal{W}\})$: a PPT algorithm that outputs an encrypted index $\mathcal{I}$ and the set of encrypted documents $\mathcal{C} = \{C_1, \ldots, C_N\}$. This algorithm is run by the client in the store phase. He then stores $(\mathcal{I}, \mathcal{C})$ on the server.
- $t(w) \leftarrow \texttt{Trpdr}(K, w)$: a PPT algorithm that outputs a trapdoor $t(w)$ for $w \in \{0,1\}^*$. In *no-dictionary* scheme, w is not necessarily a keyword. This algorithm is run by the client in the search phase. $t(w)$ is sent to the server.
- $(\mathcal{C}(w), \mathsf{Proof}) \leftarrow \texttt{Search}(\mathcal{I}, \mathcal{C}, t(w))$: a PT algorithm that outputs the search result $\mathcal{C}(w)$ and Proof for the validity check. This algorithm is run by the server in the search phase. She then returns $(\mathcal{C}(w), \mathsf{Proof})$ to the client.
- $\texttt{accept/reject} \leftarrow \texttt{Verify}(K, t(w), \tilde{\mathcal{C}}, \mathsf{Proof})$: a PT algorithm that verifies the validity of the search result $\tilde{\mathcal{C}}$ based on Proof. This algorithm is run by the client in the search phase.
- $D \leftarrow \texttt{Dec}(K, C)$: a PT algorithm that decrypts C. The client applies this algorithm to each $C \in \tilde{\mathcal{C}}$ when $\texttt{Verify}(K, t(w), \tilde{\mathcal{C}}, \mathsf{Proof}) = \texttt{accept}$ in the search phase.

We say that a no-dictionary vSSE satisfies correctness if the following holds for any $K, \mathcal{D}, \mathcal{W}, \{(w, \mathcal{D}(w)) \mid w \in \mathcal{W}\}$ and any word $w \in \mathcal{W}$.

- If

$$\begin{aligned} (\mathcal{I}, \mathcal{C}) &\leftarrow \texttt{Enc}(K, \mathcal{D}, \mathcal{W}, \{(w, \mathcal{D}(w)) \mid w \in \mathcal{W}\}), \\ t(w) &\leftarrow \texttt{Trpdr}(K, w), \\ (\tilde{\mathcal{C}}, \mathsf{Proof}) &\leftarrow \texttt{Search}(\mathcal{I}, \mathcal{C}, t(w)), \end{aligned}$$

  then

$$\begin{aligned} \texttt{Verify}(K, t(w), \tilde{\mathcal{C}}, \mathsf{Proof}) &= \texttt{accept} \\ \{D_i \mid D_i \leftarrow \texttt{Dec}(K, C_i), C_i \in \tilde{\mathcal{C}}\} &= \mathcal{D}(w). \end{aligned}$$

An (not verifiable) SSE scheme is defined by omitting Proof and $\texttt{Verify}$.

1. Adversary $\mathbf{A}$ chooses $(\mathcal{D}, \mathcal{W})$ and sends them to challenger $\mathbf{C}$.
2. $\mathbf{C}$ generates $K \leftarrow \mathtt{Gen}(1^\lambda)$ and sends $(\mathcal{I}, \mathcal{C}) \leftarrow \mathtt{Enc}(K, \mathcal{D}, \mathcal{W}, \{(w, \mathcal{D}(w)) \mid w \in \mathcal{W}\})$ to $\mathbf{A}$.
3. For $i = 1, \ldots, q$, do:
   (a) $\mathbf{A}$ chooses a word $w_i \in \{0,1\}^*$ and sends it to $\mathbf{C}$.
   (b) $\mathbf{C}$ sends the trapdoor $t(w_i) \leftarrow \mathtt{Trpdr}(K, w_i)$ back to $\mathbf{A}$.
4. $\mathbf{A}$ outputs bit b.

Fig. 1. Real game $\mathbf{Game}_{real}$

2.2 Security Definition

We next define the security of no-dictionary vSSE schemes. Note that searched word w does not need to belong to the set $\mathcal{W}$.

Privacy. In a (v)SSE, the server should learn almost no information on $\mathcal{D}, \mathcal{W}$ and the search words w. Let $L_1(\mathcal{D}, \mathcal{W})$ denote the information that the server can learn in the store phase, and let $L_2(\mathcal{D}, \mathcal{W}, \mathbf{w}, w)$ denote that in the search phase, where w is the current search word and $\mathbf{w} = (w_1, w_2, \ldots)$ is the list of the past search words queried so far.

In most existing SSE schemes, $L_1(\mathcal{D}, \mathcal{W}) = (|D_1|, \ldots, |D_N|, |\mathcal{W}|)$, and $L_2(\mathcal{D}, \mathcal{W}, \mathbf{w}, w)$ consists of $\{j \mid D_j \in \mathcal{D}(w)\}$ and the search pattern

$$\mathtt{SPattern}((w_1, \ldots, w_{q-1}), w) = (sp_1, \ldots, sp_{q-1}),$$

where

$$sp_j = \begin{cases} 1 & \text{if } w_j = w, \\ 0 & \text{if } w_j \neq w. \end{cases}$$

The search pattern reveals which past queries are the same as w.

Let $L = (L_1, L_2)$. The client's privacy is defined by using two games: a real game $\mathbf{Game}_{real}$ and a simulation game $\mathbf{Game}^L_{sim}$, as shown in Figs.1 and 2, respectively. $\mathbf{Game}_{real}$ is played by a challenger $\mathbf{C}$ and an adversary $\mathbf{A}$, and $\mathbf{Game}^L_{sim}$ is played by $\mathbf{C}$, $\mathbf{A}$ and a simulator $\mathbf{S}$.

Definition 1 (L-privacy). *We say that a no-dictionary vSSE scheme has L-privacy, if there exists a PPT simulator* $\mathbf{S}$ *such that*

$$|\Pr[\mathbf{A} \textit{ outputs } b = 1 \textit{ in } \mathbf{Game}_{real}] - \Pr[\mathbf{A} \textit{ outputs } b = 1 \textit{ in } \mathbf{Game}^L_{sim}]| \quad (2)$$

is negligible for any PPT adversary $\mathbf{A}$.

Reliability. In an SSE scheme, a malicious server might cheat a client by returning a false result $\tilde{\mathcal{C}}^*(\neq \mathcal{C}(w))$ during the search phase. (Weak) reliability guarantees that the client can detect such a malicious behavior. Formally, reliability is defined by game $\mathbf{Game}_{reli}$ shown in Fig.3, which is played by an adversary $\mathbf{B} = (\mathbf{B}_1, \mathbf{B}_2)$ (malicious server) and a challenger $\mathbf{C}$. $\mathbf{B}_1$ and $\mathbf{B}_2$ are assumed to be able to communicate freely.

1. Adversary $\mathbf{A}$ chooses $(\mathcal{D}, \mathcal{W})$ and sends them to challenger $\mathbf{C}$.
2. $\mathbf{C}$ sends $L_1(\mathcal{D}, \mathcal{W})$ to simulator $\mathbf{S}$.
3. $\mathbf{S}$ computes $(\mathcal{I}, \mathcal{C})$ from $L_1(\mathcal{D}, \mathcal{W})$, and sends them to $\mathbf{C}$.
4. $\mathbf{C}$ relays $(\mathcal{I}, \mathcal{C})$ to $\mathbf{A}$.
5. For $i = 1, \ldots, q$, do:
   (a) $\mathbf{A}$ chooses $w_i \in \{0,1\}^*$ and sends it to $\mathbf{C}$.
   (b) $\mathbf{C}$ sends $L_2(\mathcal{D}, \mathcal{W}, \mathbf{w}, w_i)$ to $\mathbf{S}$, where $\mathbf{w} = (w_1, \ldots, w_{i-1})$.
   (c) $\mathbf{S}$ computes $t(w_i)$ from $L_2(\mathcal{D}, \mathcal{W}, \mathbf{w}, w_i)$ and sends it to $\mathbf{C}$.
   (d) $\mathbf{C}$ relays $t(w_i)$ to $\mathbf{A}$.
6. $\mathbf{A}$ outputs bit b.

Fig. 2. Simulation game $\mathbf{Game}_{sim}^{L}$

(Store phase)

1. $\mathbf{B}_1$ chooses $(\mathcal{D}, \mathcal{W})$ and sends them to $\mathbf{C}$.
2. $\mathbf{C}$ generates $K \leftarrow \texttt{Gen}(1^\lambda)$, and sends $(\mathcal{I}, \mathcal{C}) \leftarrow \texttt{Enc}(K, \mathcal{D}, \mathcal{W}, \{(w, \mathcal{D}(w)) \mid w \in \mathcal{W}\})$ to $\mathbf{B}_2$.

(Search phase) For $i = 1, \ldots, q$, do

1. $\mathbf{B}_1$ chooses $w_i \in \{0,1\}^*$ and sends it to $\mathbf{C}$.
2. $\mathbf{C}$ sends the trapdoor $t(w_i) \leftarrow \texttt{Trpdr}(K, w_i)$ to $\mathbf{B}_2$.
3. $\mathbf{B}_2$ returns $(\tilde{\mathcal{C}}_i^*, \mathsf{Proof}_i^*)$ to $\mathbf{C}$.
4. $\mathbf{C}$ computes

$$\texttt{accept/reject} \leftarrow \texttt{Verify}(K, t(w_i), \tilde{\mathcal{C}}_i^*, \mathsf{Proof}_i^*)$$

   and returns $\mathcal{D}(w_i)^* = \{D_i \mid D_i \leftarrow \texttt{Dec}(K, C_i), C_i \in \tilde{\mathcal{C}}_i^*\}$ to $\mathbf{B}_1$ if the result is `accept`, otherwise sends $\perp$ to $\mathbf{B}_1$.

Fig. 3. $\mathbf{Game}_{reli}$

Definition 2 (Reliability). *We say that* $\mathbf{B}$ *wins in* $\mathbf{Game}_{reli}$ *if* $\mathbf{B}_1$ *receives* $\mathcal{D}(w_i)^*$ *such that* $\mathcal{D}(w_i)^* \notin \{\mathcal{D}(w_i), \perp\}$ *for some* i*. We say that a no-dictionary vSSE scheme satisfies reliability if for any PPT adversary* $\mathbf{B}$*,*

$$\Pr[\mathbf{B} \textit{ wins in } \mathbf{Game}_{reli}]$$

is negligible.

Strong reliability was also defined in [25]. In strong reliability, the server has to answer a wrong pair $(\tilde{\mathcal{C}}^*, \mathsf{Proof}^*)(\neq (\mathcal{C}(w), \mathsf{Proof}))$ that will be accepted in the search phase to win the game.

Definition 3 (Strong Reliability). *We say that* $\mathbf{B}$ *strongly wins in* $\mathbf{Game}_{reli}$ *if there exists* i*, such that both* $\texttt{Verify}(K, t(w_i), \tilde{\mathcal{C}}_i^*, \mathsf{Proof}_i^*) = \texttt{accept}$ *and* $(\tilde{\mathcal{C}}_i^*, \mathsf{Proof}_i^*) \neq$

$(\mathcal{C}(w_i), \mathsf{Proof}_i)$ *hold. We say that a no-dictionary vSSE scheme satisfies strong reliability if for any PPT adversary* $\mathbf{B}$*,*

$$\Pr[\mathbf{B} \textit{ strongly wins in } \mathbf{Game}_{reli}]$$

is negligible.

3 Building Blocks

3.1 Cuckoo Hashing

Cuckoo Hashing [29] is a hashing algorithm with the advantage that the search time is constant. To store n keys, it uses two tables T_1 and T_2 of size m, and two independent random hash functions h_1 and h_2 with the range $\{1, \ldots, m\}$. Every key x is stored at one of two positions, $T_1(h_1(x))$ or $T_2(h_2(x))$. So we need to inspect at most two positions to search x.

It can happen that both possible places $T_1(h_1(x))$ and $T_2(h_2(x))$ of a given key x are already occupied. This problem is solved by allowing x to throw out the key (say y) occupying the position $T_1(h_1(x))$. Next, we insert y at its alternative position $T_2(h_2(y))$. If it is already occupied, we repeat the above steps until we find an empty position. If we failed after some number of trials, we choose new hash functions and rebuild the data structure.

Let $n = m(1 - \epsilon)$ for some $\epsilon \in (0, 1)$. Then the above algorithm succeeds with probability $1 - c(\epsilon)/m + O(1/m^2)$ for some explicit function $c(\cdot)$ [27]. The expected construction time of (T_1, T_2) is bounded above by [27]

$$2n\frac{1 - e^{\epsilon-1}}{(1 - e^{\epsilon-1}) + \epsilon}. \tag{3}$$

3.2 Pseudo-random Function

Let $\mathcal{R}$ be a family of all functions $f : \{0,1\}^* \to \{0,1\}^n$. We say that $F : \{0,1\}^\ell \times \{0,1\}^* \to \{0,1\}^n$ is a pseudo-random function if for any PPT distinguisher $\mathbf{D}$,

$$\left|\Pr[k \xleftarrow{\$} \{0,1\}^\ell : \mathbf{D}^{F(k,\cdot)} = 1] - \Pr[f \xleftarrow{\$} \mathcal{R} : \mathbf{D}^{f(\cdot)} = 1]\right|$$

is negligibly small.

It is well known that a pseudo-random function works as a MAC which is existentially unforgeable against chosen message attack.

4 Generic Transformation from SSE to vSSE

In this section, we show a generic method to transform any SSE (with only privacy) to a no-dictionary verifiable SSE. Namely, in our vSSE scheme, the server can return a proof of the search result even if the search word is not in the dictionary used in the store phase.

4.1 Construction

Let $(\texttt{Gen}_0, \texttt{Enc}_0, \texttt{Trpdr}_0, \texttt{Search}_0, \texttt{Dec}_0)$ be an SSE scheme. We construct a no-dictionary verifiable SSE $(\texttt{Gen}_1, \texttt{Enc}_1, \texttt{Trpdr}_1, \texttt{Search}_1, \texttt{Verify}_1, \texttt{Dec}_1)$ as follows. Let F be a pseudo-random function.

– $\texttt{Gen}_1(1^\lambda)$: Run $\texttt{Gen}_0(1^\lambda)$ to obtain K_0. Also randomly choose a key k of F. Output (K_0, k). We write $F_k(x)$ instead of $F(k, x)$.
– $\texttt{Enc}_1((K_0, k), \mathcal{D}, \mathcal{W}, \{(w, \mathcal{D}(w)) \mid w \in \mathcal{W}\})$: Let $\mathcal{W} = \{w_1, w_2, \ldots, w_{|\mathcal{W}|}\}$.
  1. Run $\texttt{Enc}_0(K_0, \mathcal{D}, \mathcal{W}, \{(w, \mathcal{D}(w)) \mid w \in \mathcal{W}\})$ to obtain $(\mathcal{I}_0, \mathcal{C})$. Note that $C_i \in \mathcal{C}$ is a ciphertext of each document $D_i \in \mathcal{D}$.
  2. Compute $key_j \leftarrow F_k(0 \| w_j)$ for all $w_j \in \mathcal{W}$.
  3. Construct cuckoo hash tables (T_1', T_2') of size $|\mathcal{W}|+1$ which store $\{key_j\}_{j=1}^{|\mathcal{W}|}$. Let (h_1, h_2) be the hash functions which are used to construct (T_1', T_2'). This means that
$$T_1'(h_1(key_j)) = key_j \;\text{ or }\; T_2'(h_2(key_j)) = key_j$$
for each key_j.
  4. Construct two tables (T_1, T_2) of size $|\mathcal{W}| + 1$ as follows.
```
For $a = 1, 2$, do
    For $i = 1, \ldots, |\mathcal{W}| + 1$, do
        If $T_a'(i) = key_j$ for some $key_j = F_k(0\|w_j)$, then
            $T_a(i) \leftarrow \langle key_j, F_k(a\|i\|key_j), F_k(3\|key_j\|\mathcal{C}(w_j))\rangle$
        Else
            $T_a(i) \leftarrow \langle null, F_k(a\|i\|null), null\rangle$
```
  5. Output $(\mathcal{I} = (\mathcal{I}_0, T_1, T_2, h_1, h_2), \mathcal{C})$.

  We note that for each $key_j = F_k(0\|w_j)$, it holds that
$$T_1(h_1(key_j)) = \langle key_j, F_k(1\|h_1(key_j)\|key_j), F_k(3\|key_j\|\mathcal{C}(w_j))\rangle$$
  or
$$T_2(h_2(key_j)) = \langle key_j, F_k(2\|h_2(key_j)\|key_j), F_k(3\|key_j\|\mathcal{C}(w_j))\rangle.$$
– $\texttt{Trpdr}_1((K_0, k), w)$: Compute $key \leftarrow F_k(0\|w)$ and $t_0(w) \leftarrow \texttt{Trpdr}_0(K_0, w)$. Output $t(w) = (key, t_0(w))$.
– $\texttt{Search}_1((\mathcal{I}_0, T_1, T_2, h_1, h_2), \mathcal{C}, t(w) = (key, token))$: Retrieve
$$\langle \alpha_1, \beta_1, \gamma_1\rangle \leftarrow T_1(h_1(key)),$$
$$\langle \alpha_2, \beta_2, \gamma_2\rangle \leftarrow T_2(h_2(key)).$$
  Let
$$\mathcal{C}^* = \begin{cases} \texttt{Search}_0(\mathcal{I}_0, \mathcal{C}, token) & \text{if } key \in \{\alpha_1, \alpha_2\} \\ \emptyset & \text{otherwise} \end{cases}$$
$$\mathsf{Proof} = \begin{cases} \gamma_1 & \text{if } key = \alpha_1 \\ \gamma_2 & \text{if } key = \alpha_2 \\ (\alpha_1, \beta_1, \alpha_2, \beta_2) & \text{otherwise} \end{cases}$$
  Output $(\mathcal{C}^*, \mathsf{Proof})$.

– $\texttt{Verify}_1((K_0, k), t(w) = (key, token), \mathcal{C}^*, \mathsf{Proof})$:
(Case 1) $\mathsf{Proof} = \gamma$.
If $\gamma = F_k(3\|key\|\mathcal{C}^*)$, then output `accept`. Otherwise output `reject`.
(Case 2) $\mathsf{Proof} = (\alpha_1, \beta_1, \alpha_2, \beta_2)$.
If $\mathcal{C}^* \neq \emptyset$ or $key \in \{\alpha_1, \alpha_2\}$ or $\beta_1 \neq F_k(1\|h_1(key)\|\alpha_1)$ or $\beta_2 \neq F_k(2\|h_2(key)\|\alpha_2)$, then output `reject`. Otherwise output `accept`.
– $\texttt{Dec}_1((K_0, k), C_i)$: Output $D_i \leftarrow \texttt{Dec}_0(K_0, C_i)$.

4.2 Example

Suppose that there are 7 keywords $\mathcal{W} = \{w_1, \ldots, w_7\}$ and 8 ciphertexts $\mathcal{C} = \{C_1, \ldots, C_8\}$ such that $\mathcal{C}(w_j)$ are given in Table 1. In the same table, $h_1(key_j)$ and $h_2(key_j)$ are the hash values which are used to construct the cuckoo hash tables (T_1', T_2') for the set $\{key_j = F_k(0\|w_j) \mid j = 1, \ldots, 7\}$.

Table 1. Example

keyword w_j	$\mathcal{C}(w_j)$	$h_1(key_j)$	$h_2(key_j)$
w_1	C_1, C_4, C_5, C_8	6	1
w_2	C_2	2	4
w_3	C_1, C_4	6	4
w_4	C_1, C_3, C_7	6	3
w_5	C_2, C_6	7	8
w_6	C_5, C_8	7	6
w_7	C_1	2	8

Then T_1 and T_2 are constructed as shown in Table 2.

(Case 1) Suppose that a client searches for a keyword $w_3 \in \mathcal{W}$.

1. The client sends trapdoor $(key_3, t_0(w_3))$ to the server.
2. Since $h_1(key_3) = 6, h_2(key_3) = 4$, the server retrieves
$$\langle \alpha_1, \beta_1, \gamma_1 \rangle = T_1(6) = \langle key_3, F_k(1\|6\|key_3), F_k(3\|key_3\|C_1, C_4) \rangle,$$
$$\langle \alpha_2, \beta_2, \gamma_2 \rangle = T_2(4) = \langle key_2, F_k(2\|4\|key_2), F_k(3\|key_2\|C_2) \rangle$$
from T_1 and T_2.
Because $\alpha_1 = key_3$, the server obtains the search result
$$\mathcal{C}^* = (C_1, C_4) \leftarrow \texttt{Search}_0(\mathcal{I}_0, \mathcal{C}, t_0(w_3))$$
$$\mathsf{Proof} = \gamma_1 = F_k(3\|key_3\|C_1, C_4).$$
and returns $(\mathcal{C}^*, \mathsf{Proof})$ to the client.
3. The client verifies if $\gamma_1 = F_k(3\|key_3\|\mathcal{C}^*)$.

(Case 2) Suppose that the client searches for $w \notin \mathcal{W}$.

Table 2. Tables (T_1, T_2)

i	$T_1(i)$
1	$\langle$ null , $F_k(1\|1)$, null $\rangle$
2	$\langle$ key_7, $F_k(1\|2\|key_7)$, $F_k(3\|key_7\|C_1)$ $\rangle$
3	$\langle$ null , $F_k(1\|3)$, null $\rangle$
4	$\langle$ null , $F_k(1\|4)$, null $\rangle$
5	$\langle$ null , $F_k(1\|5)$, null $\rangle$
6	$\langle$ key_3, $F_k(1\|6\|key_3)$, $F_k(3\|key_3\|C_1, C_4)$ $\rangle$
7	$\langle$ key_6, $F_k(1\|7\|key_6)$, $F_k(3\|key_6\|C_5, C_8)$ $\rangle$
8	$\langle$ null , $F_k(1\|8)$, null $\rangle$

i	$T_2(i)$
1	$\langle$ key_1, $F_k(2\|1\|key_1)$, $F_k(3\|key_1\|C_1, C_4, C_5, C_8)$ $\rangle$
2	$\langle$ null , $F_k(2\|2)$, null $\rangle$
3	$\langle$ key_4, $F_k(2\|3\|key_4)$, $F_k(3\|key_4\|C_1, C_3, C_7)$ $\rangle$
4	$\langle$ key_2, $F_k(2\|4\|key_2)$, $F_k(3\|key_2\|C_2)$ $\rangle$
5	$\langle$ null , $F_k(2\|5)$, null $\rangle$
6	$\langle$ null , $F_k(2\|6)$, null $\rangle$
7	$\langle$ null , $F_k(2\|7)$, null $\rangle$
8	$\langle$ key_5, $F_k(2\|8\|key_5)$, $F_k(3\|key_5\|(C_2, C_6))$ $\rangle$

1. The client computes $key \leftarrow F_k(0\|w)$ and $t_0(w) \leftarrow \texttt{Trpdr}_0(K_0, w)$. He sends $t(w) = (key, t_0(w))$ to the server.
2. Suppose that $h_1(key) = 5$ and $h_2(key) = 3$. Then the server retrieves
$$\langle \alpha_1, \beta_1, \gamma_1 \rangle = T_1(5) = \langle null, F_k(1\|5), null \rangle,$$
$$\langle \alpha_2, \beta_2, \gamma_2 \rangle = T_2(3) = \langle key_4, F_k(2\|3\|key_4), F_k(3\|key_4\|C_1, C_3, C_7) \rangle.$$
Because $key \notin \{\alpha_1, \alpha_2\}$, the server returns $\mathcal{C}^* = \emptyset$ and
$$\mathsf{Proof} = (\alpha_1, \beta_1, \alpha_2, \beta_2) = (null, F_k(1\|5), key_4, F_k(2\|3\|key_4)).$$
3. The client verifies if $key \notin \{\alpha_1, \alpha_2\}$, $\beta_1 = F_k(1\|h_1(key)\|\alpha_1)$ and $\beta_2 = F_k(2\|h_2(key)\|\alpha_2)$.

4.3 Efficiency

In our transformed scheme,

- In the store phase, the client takes the expected time $O(|\mathcal{W}|) + time(\texttt{Enc}_0)$ to run $\texttt{Enc}_1$ from eq.(3).
- In the search phase, the search time for $w \in \mathcal{W}$ is almost the same as that of the original scheme.
- The server takes only $O(1)$ time to prove that $w \notin \mathcal{W}$ because the search time is constant in cuckoo hashing.

To prove that $w \notin \mathcal{W}$, in the method of [31], the server takes $O(N \log N|\mathcal{W}|)$ time. In the concrete method (Algorithm 1+2) in [6], it takes $O(\log |\mathcal{W}|) + time(\texttt{Search}_0)$.

4.4 Security

Theorem 1. *If the SSE scheme has $L = (L_1, L_2)$-privacy and F is a pseudo-random function, then our vSSE scheme has $L' = (L'_1, L'_2)$-privacy such that*

$$L'_1(\mathcal{D}, \mathcal{W}) = L_1(\mathcal{D}, \mathcal{W}) \cup \{|\mathcal{W}|\},$$
$$L'_2(\mathcal{D}, \mathcal{W}, \mathbf{w}, w_i) = L_2(\mathcal{D}, \mathcal{W}, \mathbf{w}, w_i) \cup \{\texttt{SPattarn}(\mathbf{w}, w_i), [w_i \in \mathcal{W}]\}.$$

In the all existing SSE schemes, $|\mathcal{W}| \in L_1(\mathcal{D}, \mathcal{W})$ and $\{\texttt{SPattarn}(\mathbf{w}, w_i), [w_i \in \mathcal{W}]\} \subseteq L'_2(\mathcal{D}, \mathcal{W}, \mathbf{w}, w_i)$. (There may be some exceptions which use oblivious RAM. But such SSE schemes are inefficient.) So, the client's privacy in our vSSE scheme has the same level as that of the underlying SSE scheme.

Proof. Let $\mathbf{S}_0$ be a simulator of the underlining SSE scheme which has (L_1, L_2)-privacy. We construct a simulator $\mathbf{S}$ of our vSSE scheme which achieves (L'_1, L'_2)-privacy as follows.

(Store phase) In $\mathbf{Game}_{sim}$, $\mathbf{S}$ takes $L'_1(\mathcal{D}, \mathcal{W}) = L_1(\mathcal{D}, \mathcal{W}) \cup \{|\mathcal{W}|\}$ as an input. $\mathbf{S}$ runs $\mathbf{S}_0(L_1(\mathcal{D}, \mathcal{W}))$ and gets its output $(\mathcal{I}_0, \mathcal{C})$. Next $\mathbf{S}$ constructs T_1 and T_2 as follows. Note that the size of each T_1, T_2 is $m = |\mathcal{W}| + 1$.

- Choose random strings $key'_1, \ldots, key'_{|\mathcal{W}|}$, and construct the cuckoo hash tables (T'_1, T'_2) which store $(key'_{\pi(1)}, \ldots, key'_{\pi(|\mathcal{W}|)})$, where π is a random permutation. Let h_1, h_2 be the two hash functions which are used to construct (T'_1, T'_2).
- For $a = 1, 2$, do

```
For $i = 1, \ldots, |\mathcal{W}| + 1$, do
    If $T'_a(i) = key'_j$ for some $j$, then
        choose two random strings $r, r'$ and $T_a(i) \leftarrow \langle key'_j, r, r' \rangle$
    Else
        choose a random string $r$ and $T_a(i) \leftarrow \langle null, r, null \rangle$
```

$\mathbf{S}$ sends $(\mathcal{I}_0, T_1, T_2, h_1, h_2)$ and $\mathcal{C}$ to the challenger. Let $counter \leftarrow 1$.

(Search phase) In the ith search phase, $\mathbf{S}$ takes $L'_2(\mathcal{D}, \mathcal{W}, \mathbf{w}, w^*) = L_2(\mathcal{D}, \mathcal{W}, \mathbf{w}, w^*) \cup \{\texttt{SPattarn}(\mathbf{w}, w^*), [w^* \in \mathcal{W}])\}$ as an input. $\mathbf{S}$ first obtains $t_0(w^*)$ by running $\mathbf{S}_0(L_2(\mathcal{D}, \mathcal{W}, \mathbf{w}, w^*))$, and sets

$$key^*_i \leftarrow \begin{cases} key'_{counter} & \text{if } sp_j = 0 \text{ for all } j \text{ and } w^* \in \mathcal{W}, \\ key^*_j & \text{if } sp_j = 1 \text{ for some } j, \\ \text{a random string otherwise.} \end{cases}$$

$$counter \leftarrow \begin{cases} counter + 1 & \text{if } sp_j = 0 \text{ for all } j \text{ and } w^* \in \mathcal{W}, \\ counter & \text{otherwise.} \end{cases}$$

$\mathbf{S}$ outputs $(key^*_i, t_0(w^*))$ as a simulated trapdoor.

We will prove that there is no adversary $\mathbf{A}$ who can distinguish between $\mathbf{Game}_{real}$ and $\mathbf{Game}_{sim}$. We consider a game sequence $(\mathbf{Game}_{real}, \mathbf{Game}_{mid}, \mathbf{Game}_{sim})$.

$\mathbf{Game}_{mid}$ is the same as $\mathbf{Game}_{real}$ except that all values of $F_k(\cdot)$ are replaced with random strings. For $i \in \{real, mid, sim\}$, define

$$P_i = \Pr[\ \mathbf{A} \text{ outputs } b = 1 \text{ in } \mathbf{Game}_i].$$

Then $|P_{real} - P_{mid}|$ is negligible because F is a pseudorandom function. We can also see that $|P_{mid} - P_{sim}|$ is negligible from the (L_1, L_2)-privacy of the underlying SSE scheme. Consequently, $|P_{real} - P_{sim}|$ is negligible. □

Theorem 2. *Our vSSE scheme satisfies strong reliability if F is a pseudorandom function.*

Proof. We look at the pseudorandom function F as a MAC. Suppose that there exists an adversary $\mathbf{B} = (\mathbf{B}_1, \mathbf{B}_2)$ who can break the strong reliability of our vSSE scheme, and $\mathbf{B}$ runs the search phase q times. Let $(\tilde{\mathcal{C}}_i^*, \widetilde{\mathsf{Proof}}_i)$ be $\mathbf{B}_2$'s response to $t(w_i) = (key_i, t_0(w_i))$ in the ith search phase, and let

$$(\mathcal{C}(w_i), \mathsf{Proof}_i) = \mathtt{Search}_1(\mathcal{I}, \mathcal{C}, t(w_i)).$$

From the definition, $\mathbf{B}$ strongly wins if there exists $i \in \{1, \ldots, q\}$ such that

$$\begin{aligned} &(\tilde{\mathcal{C}}_i^*, \widetilde{\mathsf{Proof}}_i) \neq (\mathcal{C}(w_i), \mathsf{Proof}_i) \\ &\text{and} \quad \mathtt{Verify}_1(K, (key_i, t_0(w_i)), \tilde{\mathcal{C}}_i^*, \widetilde{\mathsf{Proof}}_i) = \mathtt{accept}. \end{aligned} \tag{4}$$

By using $\mathbf{B}$, we will construct a forger $\mathbf{F}$ against the MAC, where $\mathbf{F}$ has oracle access to F_k.

$\mathbf{F}$ at first randomly chooses $J \in \{1, \ldots, q\}$. Then, $\mathbf{F}$ runs $\mathbf{B}$ by playing the role of the challenger $\mathbf{C}$ (see Fig.3) until the $(J-1)$th search phase. During this simulation, when $\mathbf{C}$ needs to compute $F_k(x)$ for some x, $\mathbf{F}$ queries x to its oracle F_k to obtain $F_k(x)$.

In the Jth search phase, we have the following three cases.

(1) $\widetilde{\mathsf{Proof}}_J = \tilde{\gamma}$.
In this case, $\mathbf{F}$ outputs $m' = (3 \| key_J \| \tilde{\mathcal{C}}_J^*)$ and $tag' = \tilde{\gamma}$ as a forgery of the MAC F.

(2) $\mathsf{Proof}_J = \gamma$ and $\widetilde{\mathsf{Proof}}_J = (\tilde{\alpha}_1, \tilde{\beta}_1, \tilde{\alpha}_2, \tilde{\beta}_2)$.
Since $\mathsf{Proof}_J = \gamma$, there exists $a \in \{1, 2\}$ such that $T_a(h_a(key_J)) = \langle key_J, F_k(a \| h_a(key_J) \| key_J), \ldots \rangle$. For this a, $\mathbf{F}$ outputs $m' = (a \| h_a(key_J) \| \tilde{\alpha}_a)$ and $tag' = \tilde{\beta}_a$ as a forgery.

(3) $\mathsf{Proof}_J = (\alpha_1, \beta_1, \alpha_2, \beta_2)$ and $\widetilde{\mathsf{Proof}}_J = (\tilde{\alpha}_1, \tilde{\beta}_1, \tilde{\alpha}_2, \tilde{\beta}_2)$.
If there exists $a \in \{1, 2\}$ s.t. $(\alpha_a, \beta_a) \neq (\tilde{\alpha}_a, \tilde{\beta}_a)$, then, $\mathbf{F}$ outputs $m' = (a \| h_a(key_J) \| \tilde{\alpha}_a)$ and $tag' = \tilde{\beta}_a$ as a forgery. Otherwise $\mathbf{F}$ outputs "fail."

Now $\mathbf{F}$ succeeds in forgery if $\mathbf{B}$ strongly wins and $\mathbf{F}$ correctly predicts i which satisfies eq.(4), i.e., eq.(4) holds in $i = J$. Since $\mathbf{F}$ predicts J correctly with probability $1/q$, we obtain that

$$\Pr[\mathbf{F} \text{ succeeds in forgery}] \geq \Pr[\mathbf{B} \text{ strongly wins in } \mathbf{Game}_{reli}] \times \frac{1}{q}.$$

□

We prove the UC-security of the transformed scheme in Appendix.

References

1. L. Ballard, S. Kamara, F. Monrose: Achieving Efficient Conjunctive Keyword Searches over Encrypted Data. ICICS 2005, pp.414-426 (2005)
2. N. Baric and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Advances in Cryptology: Proc. EUROCRYPT, volume 1233 of LNCS, pages 480–494. Springer-Verlag, 1997.
3. M. Bellare, A. Desai, E. Jokipii, P. Rogaway: A Concrete Security Treatment of Symmetric Encryption. FOCS 1997: pp.394–403 (1997)
4. M. Bellare, R. Guerin, P. Rogaway: XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions. CRYPTO 1995: pp.15–28
5. S. Bellovin and W. Cheswick: Privacy-Enhanced Searches Using Encrypted Bloom Filters, Cryptology ePrint Archive, Report 2006/210, http://eprint.iacr.org/ (2006)
6. R. Bost, P.-A. Fouque, D. Pointcheval: Verifiable Dynamic Symmetric Searchable Encryption Optimality and Forward Security, Cryptology ePrint Archive, Report 2016/62, http://eprint.iacr.org/ (2016)
7. J. W. Byun, D. H. Lee, and J. Lim: Efficient conjunctive keyword search on encrypted data storage system. EuroPKI, pp.184–196 (2006)
8. J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. CRYPTO 2002, pp.61–76 (2002)
9. R. Canetti, Universally Composable Security: A New Paradigm for Cryptographic Protocols, Proc. of 42nd FOCS, 2001. Full version is available at http://eprint.iacr.org/2000/067.
10. R. Canetti: "Universally Composable Signatures, Certification and Authentication," Cryptology ePrint Archive, Report 2003/239 (2003), http://eprint.iacr.org/
11. R. Canetti: "Universally Composable Security: A New Paradigm for Cryptographic Protocols," Cryptology ePrint Archive, Report 2000/067 (2005), http://eprint.iacr.org/
12. D. Cash, S. Jarecki, C.S. Jutla, H. Krawczyk, M. Rosu, M. Steiner: Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries. CRYPTO (1) 2013, pp.353–373 (2013)
13. D. Cash, J. Jaeger, S. Jarecki, C.S. Jutla, H. Krawczyk, M.-C. Rosu, M. Steiner: Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation. NDSS 2014
14. D. Cash, S. Tessaro: The Locality of Searchable Symmetric Encryption. EUROCRYPT 2014: pp.351–368
15. Y. Chang and M. Mitzenmacher: Privacy Preserving Keyword Searches on Remote Encrypted Data. ACNS 2005: pp.442–455 (2005)
16. R. Curtmola, J.A. Garay, S.Kamara, R.Ostrovsky: Searchable symmetric encryption: improved definitions and efficient constructions. ACM Conference on Computer and Communications Security 2006: pp.79–88 (2006).
17. Full version of [16]: Cryptology ePrint Archive, Report 2006/210, http://eprint.iacr.org/ (2006)
18. E.-J. Goh: Secure Indexes. Cryptology ePrint Archive, Report 2003/216, http://eprint.iacr.org/ (2003)
19. P. Golle, J. Staddon, B.R. Waters: Secure Conjunctive Keyword Search over Encrypted Data. ACNS 2004, pp.31–45 (2004)
20. S. Kamara and C. Papamanthou: Parallel and Dynamic Searchable Symmetric Encryption. FC 2013

21. S. Kamara, Charalampos Papamanthou, Tom Roeder: Dynamic searchable symmetric encryption. ACM Conference on Computer and Communications Security 2012: pp.965–976
22. K. Kurosawa: Garbled Searchable Symmetric Encryption. Financial Cryptography 2014: pp.234–251
23. K. Kurosawa, Y. Ohtaki: UC-Secure Searchable Symmetric Encryption. Financial Cryptography 2012: pp.285–298
24. K. Kurosawa, Y. Ohtaki: How to Update Documents Verifiably in Searchable Symmetric Encryption. CANS 2013: pp.309–328
25. The final version of [23]. Cryptology ePrint Archive, Report 2015/251 (2015)
26. K.Kurosawa, K.Sasaki, K.Ohta, K.Yoneyama: UC-Secure Dynamic Searchable Symmetric Encryption Scheme. IWSEC 2016: pp.73–90
27. R. Kutzelnigg: "Bipartite random graphs and cuckoo hashing," Fourth Colloquium on Mathematics and Computer Science. Discrete Mathematics and Theoretical Computer Science. pp.403–406 (2006)
28. M. Naveed, M. Prabhakaran and C. Gunter: Dynamic Searchable Encryption via Blind Storage. IEEE Security & Privacy 2014
29. R. Pagh, F.F. Rodler: Cuckoo Hashing. ESA 2001: pp.121–133 (2001)
30. D. Song, D. Wagner, A. Perrig: Practical Techniques for Searches on Encrypted Data. IEEE Symposium on Security and Privacy 2000: pp.44–55 (2000)
31. S. Taketani, W. Ogata: "Improvement of UC Secure Searchable Symmetric Encryption Scheme," Advances in Information and Computer Security, the 10th International Workshop on Security, IWSEC 2015, LNCS Vol.9241, pp.135–152 (2015)
32. P. Wang, H. Wang, J. Pieprzyk: Keyword Field-Free Conjunctive Keyword Searches on Encrypted Data and Extension for Dynamic Groups. CANS 2008: pp.178–195

A UC-Security for No-Dictionary vSSE

If a protocol is secure in the universally composable (UC) security framework, its security is maintained even if the protocol is combined with other protocols [9–11]. The UC security is defined based on *ideal functionality* $\mathcal{F}$. Kurosawa and Ohtaki introduced an ideal functionality of vSSE [23, 25]. Taketani and Ogata [31] generalized it in order to handle the general leakage functions $L = (L_1, L_2)$ as shown in Fig.4.

In the no-dictionary verifiable SSE setting, the real world is described as follows. We assume a real adversary, $\mathbf{A}^{\mathrm{uc}}$, can control the server arbitrarily, and the client is always honest. For simplicity, we ignore session id.

In the store phase, an environment, $\mathbf{Z}$, chooses $(\mathcal{D}, \mathcal{W})$ and sends them to the client. The client computes $K \leftarrow \mathtt{Gen}(1^\lambda)$ and $(\mathcal{I}, \mathcal{C}) \leftarrow \mathtt{Enc}(K, \mathcal{D}, \mathcal{W}, \{(w, \mathcal{D}(w)) \mid w \in \mathcal{W}\})$, and sends $(\mathcal{I}, \mathcal{C})$ to the server. The client stores K [2] and the server stores $(\mathcal{I}, \mathcal{C})$. In the search phase, $\mathbf{Z}$ chooses a word $w \in \{0,1\}^*$ and sends it to the client. The client computes $t(w) \leftarrow \mathtt{Trpdr}(K, w)$ and sends it to the server. The server, who may be controlled by real adversary $\mathbf{A}^{\mathrm{uc}}$, returns $(\tilde{\mathcal{C}}^*, \widetilde{\mathsf{Proof}})$ to the client. If $\mathtt{Verify}(K, t(w), \tilde{\mathcal{C}}^*, \widetilde{\mathsf{Proof}})$ outputs $\mathtt{accept}$, then the client decrypts all $\tilde{C}_i \in \tilde{\mathcal{C}}^*$, and sends the list of plaintexts $\tilde{\mathcal{D}}(w) = (\tilde{D}_1, \tilde{D}_2, \ldots)$ to $\mathbf{Z}$. If

[2] he may forget $\mathcal{D}, \mathcal{W}, \mathcal{C}, \mathcal{I}$.

Store: Upon receiving the input $(\textbf{store}, sid, D_1, \ldots, D_N, \mathcal{W})$ from the dummy client, verify that this is the first input from the client with $(\textbf{store}, sid)$.
If it is, then store $\mathcal{D} = \{D_1, \ldots, D_N\}$, and send $L_1(\mathcal{D}, \mathcal{W})$ to $\mathbf{S}^{\mathrm{uc}}$. Otherwise, ignore this input.

Search: Upon receiving $(\textbf{search}, sid, w)$ from the client, send $L_2(\mathcal{D}, \mathcal{W}, \mathbf{w}, w)$ to $\mathbf{S}^{\mathrm{uc}}$. Note that in a no-dictionary vSSE scheme, the client may send $w \notin \mathcal{W}$. If $\mathbf{S}^{\mathrm{uc}}$ returns `accept`, then send $\mathcal{D}(w)$ to the client. If $\mathbf{S}^{\mathrm{uc}}$ returns `reject`, then send $\perp$ to the client.

Fig. 4. Ideal functionality $\mathcal{F}^{L}_{vSSE}$

$\mathtt{Verify}(K, t(w), \tilde{\mathcal{C}}^*, \widetilde{\mathsf{Proof}})$ outputs `reject`, then $\perp$ is sent to $\mathbf{Z}$. After the store phase, $\mathbf{Z}$ outputs a bit b.

On the other hand, the ideal world is described as follows.

In the store phase, $\mathbf{Z}$ sends $(\mathcal{D}, \mathcal{W})$ to the dummy client. The dummy client sends $(\textbf{store}, \mathcal{D}, \mathcal{W})$ to functionality $\mathcal{F}^{L}_{vSSE}$ (see Fig.4). In the search phase, $\mathbf{Z}$ sends w to the dummy client. The dummy client sends $(\textbf{search}, w)$ to $\mathcal{F}^{L}_{vSSE}$, and receives $\mathcal{D}(w)$ or $\perp$ (according to ideal adversary $\mathbf{S}^{\mathrm{uc}}$'s decision), which is relayed to $\mathbf{Z}$. At last, $\mathbf{Z}$ outputs a bit b

In both worlds, $\mathbf{Z}$ can communicate with $\mathbf{A}^{\mathrm{uc}}$ (in the real world) or $\mathbf{S}^{\mathrm{uc}}$ (in the ideal world) in an arbitrary way.

UC-security of no-dictionary vSSE scheme is defined as follows.

Definition 4 (UC-security with leakage L). *We say that no-dictionary vSSE scheme has universally composable (UC) security with leakage L against non-adaptive adversaries, if for any PPT real adversary $\mathbf{A}^{\mathrm{uc}}$, there exists a PPT ideal adversary (simulator) $\mathbf{S}^{\mathrm{uc}}$, and for any PPT environment $\mathbf{Z}$,*

$$|\Pr[\mathbf{Z} \textit{ outputs } 1 \textit{ in the real world}] - \Pr[\mathbf{Z} \textit{ outputs } 1 \textit{ in the ideal world}]|$$

is negligible.

We can show a weak equivalence of UC security and privacy with reliability.

Theorem 3. *If a no-dictionary vSSE scheme satisfies L-privacy and strong reliability for some L, it has UC security with leakage L against non-adaptive adversaries.*

Proof. Assume that the scheme satisfies L-privacy and strong reliability.

We consider four games $\mathbf{Game}_0, \ldots, \mathbf{Game}_3$. Let

$$p_i = \Pr[\mathbf{Z} \text{ outputs } 1 \text{ in } \mathbf{Game}_i]$$

for a fixed $\mathbf{A}^{\mathrm{uc}}$. $\mathbf{Game}_0$ is equivalent to the real world in the definition of UC security. So,

$$p_0 = \Pr[\mathbf{Z} \text{ outputs } 1 \text{ in the real world}].$$

$\mathbf{Game}_1$ is different from $\mathbf{Game}_0$ in the following points.

- In the store phase, the client records $(\mathcal{D}, \mathcal{W}, \mathcal{I})$ as well as the key K.
- In the search phase, if $\mathbf{A}^{\mathrm{uc}}$ instructs the server to return $(\tilde{\mathcal{C}}^*, \widetilde{\mathsf{Proof}})$ such that $(\tilde{\mathcal{C}}^*, \widetilde{\mathsf{Proof}}) \neq (\mathcal{C}^*, \mathsf{Proof}) \leftarrow \mathtt{Search}(\mathcal{I}, \mathcal{C}, t(w))$, then the server returns `reject` to the client. Otherwise the server returns `accept`.
- If the client receives `accept` from the server, he sends $\mathcal{D}(w)$ to $\mathbf{Z}$. Otherwise, he sends $\perp$ to $\mathbf{Z}$.

$\mathbf{Game}_1$ is the same as $\mathbf{Game}_0$ until $\mathbf{A}^{\mathrm{uc}}$ instructs the server to return $(\tilde{\mathcal{C}}^*, \widetilde{\mathsf{Proof}})$ such that

$$\mathtt{Verify}(K, t(w), \tilde{\mathcal{C}}^*, \widetilde{\mathsf{Proof}}) = \mathtt{accept} \text{ and } (\tilde{\mathcal{C}}^*, \widetilde{\mathsf{Proof}}) \neq (\mathcal{C}^*, \mathsf{Proof}).$$

The above condition is the (strongly) winning condition of $\mathbf{B}$ in $\mathbf{Game}_{reli}$. So, we can obtain

$$|p_0 - p_1| \leq \max_{\mathbf{B}} \Pr[\mathbf{B} \text{ strongly wins in } \mathbf{Game}_{reli}].$$

From the assumption, $|p_0 - p_1|$ is negligibly small.

In $\mathbf{Game}_2$, we split the client into two entities, client1 and client2, as follows. (See Fig. 5(a).)

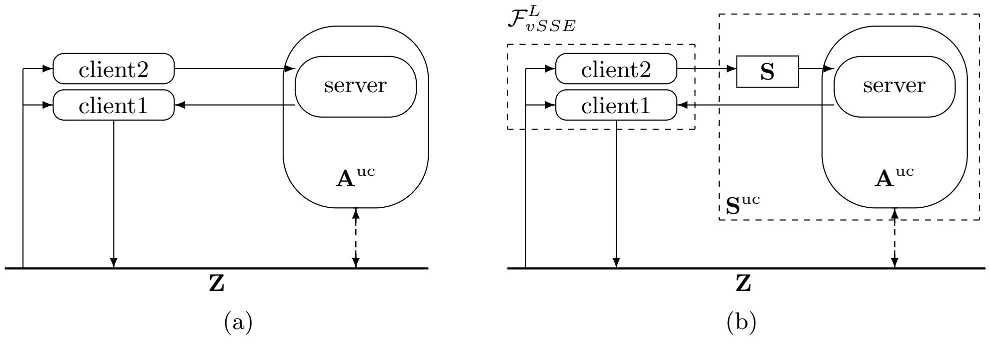

Fig. 5. (a) $\mathbf{Game}_2$, (b) $\mathbf{Game}_3$

- Both client1 and client2 receive all input from $\mathbf{Z}$.
- In the store/search phase, only client2 sends $(\mathcal{I}, \mathcal{C})/t(w)$ to the server.
- In the search phase, only client1 receives `accept/reject` from the server, and sends $\mathcal{D}(w)/\perp$ to $\mathbf{Z}$.

This change is conceptual only. Therefore $p_2 = p_1$.

Now, we look at $(\mathbf{Z}, \text{client1}, \text{server}, \mathbf{A}^{\mathrm{uc}})$ and client2 as an adversary $\mathbf{A}$ and a challenger $\mathbf{C}$ in the real game of privacy, respectively. Then, from the assumption, there exists a simulator $\mathbf{S}$ such that Eq.(2) is negligible.

In $\mathbf{Game}_3$, client2 plays the role of the challenger in the simulation game of privacy; he sends $L_1(\mathcal{D},\mathcal{W})$ or $L_2(\mathcal{D},\mathcal{W},\mathbf{w},w)$ to the simulator $\mathbf{S}$, and then $\mathbf{S}$ sends its outputs (the simulated message) to the server. (See Fig. 5(b).) Again, we look at $(\mathbf{Z}, \text{client1}, \text{server}, \mathbf{A}^{\text{uc}})$ as $\mathbf{A}$. Then $\mathbf{Game}_3$ is the simulation game and $\mathbf{Game}_2$ is the real game. Therefore

$$|p_3 - p_2| \le |\Pr[\mathbf{A} \text{ outputs 1 in } \mathbf{Game}_{real}] - \Pr[\mathbf{A} \text{ outputs 1 in } \mathbf{Game}^{L}_{sim}]|,$$

and it is negligible from the assumption.

In $\mathbf{Game}_3$, (client1, client2) behaves exactly the same way as $\mathcal{F}^{L}_{vSSE}$ in the ideal world. So, considering $(\mathbf{S}, \text{server}, \mathbf{A}^{\text{uc}})$ as a simulator $\mathbf{S}^{\text{uc}}$, we obtain

$$p_3 = \Pr[\mathbf{Z} \text{ outputs 1 in the ideal world}]$$

for this simulator. Consequently, we can say that for any $\mathbf{A}^{\text{uc}}$ there exists $\mathbf{S}^{\text{uc}}$ such that $|p_0 - p_3| = |\Pr[\mathbf{Z} \text{ outputs 1 in the real world}] - \Pr[\mathbf{Z} \text{ outputs 1 in the ideal world}]|$ is negligible. □

Theorem 4. *If a no-dictionary vSSE scheme has UC security with leakage L against non-adaptive adversaries for some L, it has satisfies L-privacy and reliability.*

This theorem is shown by the following lemmas.

Lemma 1. *If* `vSSE` *has UC security with leakage L against non-adaptive adversaries for some L,* `vSSE` *has satisfies L-privacy.*

Proof. Assume that the scheme has UC security with leakage L.

Consider a real adversary $\mathbf{A}_0^{\text{uc}}$ who sends $\mathbf{Z}$ all inputs that the corrupted server receives from the client. That is, $(\mathcal{I},\mathcal{C})$ and $t(w)$ are sent to $\mathbf{Z}$ in the store phase and the search phase, respectively. From the assumption, there exists an ideal adversary $\mathbf{S}_0^{\text{uc}}$ for such $\mathbf{A}_0^{\text{uc}}$, and any environment $\mathbf{Z}$ cannot distinguish the real world and the ideal world (Fig. 6). That is,

$$|\Pr[\mathbf{Z} \text{ outputs 1 in the real world}] - \Pr[\mathbf{Z} \text{ outputs 1 in the ideal world}]|$$

is negligible for any $\mathbf{Z}$. Note that $\mathbf{S}_0^{\text{uc}}$ can compute and send simulated $(\tilde{\mathcal{I}},\tilde{\mathcal{C}})$ and $\tilde{t}(w)$ to $\mathbf{Z}$.

Now we consider restricted environments $\mathbf{Z}_0$ that do not use the answer from the client/dummy client to distinguish the worlds. Namely, in the real world, $\mathbf{Z}_0$ sends $(\mathcal{D},\mathcal{W})$ and w to the client and receives $(\mathcal{I},\mathcal{C}) \leftarrow \mathtt{Enc}(K,\mathcal{D},\mathcal{W},\{(w,\mathcal{D}(w)) \mid w \in \mathcal{W}\})$ and $t(w) \leftarrow \mathtt{Trpdr}(K,w)$ from $\mathbf{A}_0^{\text{uc}}$ in the store phase and the search phase, respectively, and outputs a bit at last. This situation is exactly the same as $\mathbf{A}$ in $\mathbf{Game}_{real}$ (Fig. 7(a)). On the other hand, in the ideal world, $\mathbf{Z}_0$ sends $(\mathcal{D},\mathcal{W})$ and w to the dummy client and receives $(\tilde{\mathcal{I}},\tilde{\mathcal{C}})$ and $\tilde{t}(w)$ from $\mathbf{S}_0^{\text{uc}}$ in each phase, and outputs a bit. This situation is exactly the same as $\mathbf{A}$ in $\mathbf{Game}_{sim}$

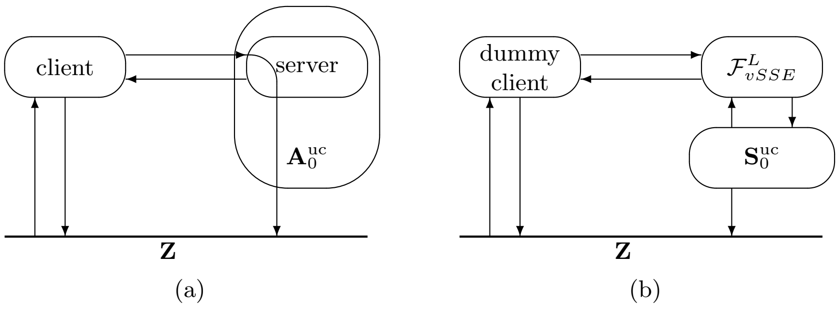

Fig. 6. (a) $\mathbf{A}_0^{\mathrm{uc}}$, (b) $\mathbf{S}_0^{\mathrm{uc}}$

(Fig. 7(b)). Therefore,

$$\begin{aligned}
&\max_{\mathbf{A}} |\Pr[\mathbf{A} \text{ outputs 1 in } \mathbf{Game}_{real}] - \Pr[\mathbf{A} \text{ outputs 1 in } \mathbf{Game}_{sim}]| \\
&= \max_{\mathbf{Z}_0} |\Pr[\mathbf{Z}_0 \text{ outputs 1 in the real world}] - \Pr[\mathbf{Z}_0 \text{ outputs 1 in the ideal world}]| \\
&\le \max_{\mathbf{Z}} |\Pr[\mathbf{Z} \text{ outputs 1 in the real world}] - \Pr[\mathbf{Z} \text{ outputs 1 in the ideal world}]| \\
&= negl.
\end{aligned}$$

□

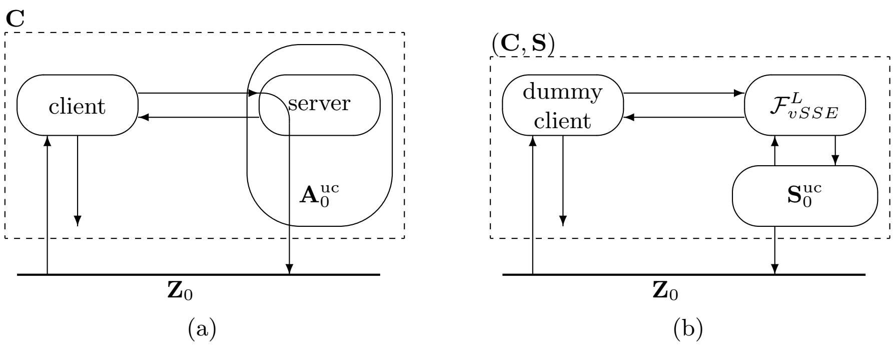

Fig. 7. $\mathbf{Z}_0$ in (a)real and (b)ideal world

Lemma 2. *If* `vSSE` *has UC security with leakage L against non-adaptive adversaries for some L,* `vSSE` *has satisfies reliability.*

Proof. We fix an arbitrary adversary $\mathbf{B} = (\mathbf{B}_1, \mathbf{B}_2)$ of reliability game. Consider a real adversary $\mathbf{A}_{\mathbf{B}}^{\mathrm{uc}}$ such that $\mathbf{A}_{\mathbf{B}}^{\mathrm{uc}}$ interacts with the client like $\mathbf{B}_2$ (by controlling the server), while $\mathbf{A}_{\mathbf{B}}^{\mathrm{uc}}$ interacts with $\mathbf{Z}$ like $\mathbf{B}_1$ (Fig. 8(a)). More precisely, at the beginning of each phase, $\mathbf{A}^{\mathrm{uc}}_{\mathbf{B}}$ suggests which $(\mathcal{D}, \mathcal{W})$ or w the environment should send to the client.

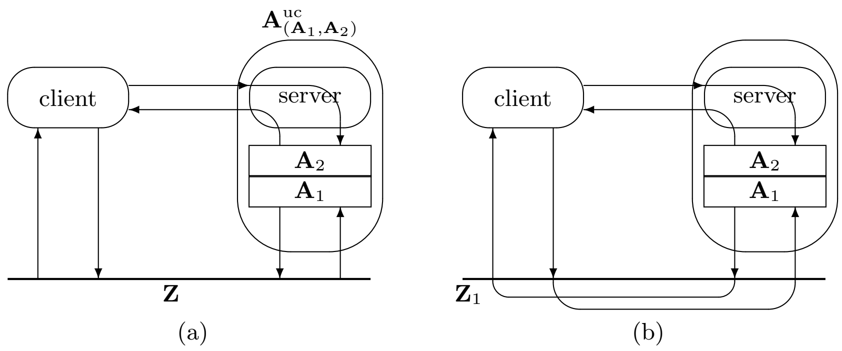

Fig. 8. (a) $\mathbf{A}^{\mathrm{uc}}_{\mathbf{B}}$, (b) $\mathbf{Z}_1$

If the scheme has UC security with leakage L, there exists an ideal adversary, $\mathbf{S}^{\mathrm{uc}}_{\mathbf{B}}$, and any environment $\mathbf{Z}$ cannot distinguish the real world and the ideal world.

Next, consider a simple environment $\mathbf{Z}_1$ performs as follows (Fig. 8(b)). At the beginning of each phase, $\mathbf{Z}_1$ sends the client/dummy client $(\mathcal{D}, \mathcal{W})$ or w suggested by $\mathbf{A}^{\mathrm{uc}}_{\mathbf{B}}$. When $\mathbf{Z}_1$ receives a message from the client/dummy client, $\mathbf{Z}_1$ relays it to $\mathbf{A}^{\mathrm{uc}}_{\mathbf{B}}$. If $\mathbf{Z}_1$ receives $\tilde{\mathcal{D}}(w) \notin \{\mathcal{D}(w), \perp\}$ as a reply of w, then outputs 1.

It is clear that

$$\Pr[\mathbf{Z}_1 \text{ outputs 1 in the real world}] = \Pr[\mathbf{B} \text{ wins in } \mathbf{Game}_{reli}].$$

On the other hand, in the ideal world, $\mathbf{Z}_1$ never receives $\tilde{\mathcal{D}}(w) \notin \{\mathcal{D}(w), \perp\}$ from $\mathcal{F}^{L}_{vSSE}$ through the client. Therefore,

$$\Pr[\mathbf{Z}_1 \text{ outputs 1 in the ideal world}] = 0.$$

Hence

$$\begin{aligned}&\Pr[\mathbf{B} \text{ wins in } \mathbf{Game}_{reli}]\\ &\ = \left|\Pr[\mathbf{Z}_1 \text{ outputs 1 in the real world}] - \Pr[\mathbf{Z}_1 \text{ outputs 1 in the ideal world}]\right|,\end{aligned}$$

which is negligible for any $\mathbf{B}$ from the assumption. □

Corollary 1. *Our transformed scheme is UC-secure with leakage $L' = (L'_1, L'_2)$ if the original SSE scheme has $L = (L_1, L_2)$-privacy, where L and L' are given in Theorem 1.*