How to Construct UC-Secure Searchable Symmetric Encryption Scheme

Kaoru Kurosawa Yasuhiro Ohtaki

Ibaraki University, Japan
E-mail. {kurosawa, y.ohtaki}@mx.ibaraki.ac.jp

Abstract

A searchable symmetric encryption (SSE) scheme allows a client to store a set of encrypted files on an untrusted server in such a way that he can efficiently retrieve some of the encrypted files containing (or indexed by) specific keywords keeping the keywords and the files secret. In this paper, we first extend the model of SSE schemes to that of *verifiable* SSE schemes, and formulate the UC security. We then prove its weak equivalence with privacy and reliability. Finally we show an efficient verifiable SSE scheme which is UC-secure.

Keywords: searchable symmetric encryption, UC-security, symmetric-key encryption

1 Introduction

In the model of searchable symmetric encryption (SSE) schemes, a client stores a set of encrypted files C_j on an untrusted server in the store phase. Later in the search phase, he can efficiently retrieve some of the encrypted files containing (or indexed by) specific keywords, keeping the keywords and the files secret (ideally without leaking any information to the server).

The first SSE scheme was proposed by Song, Wagner, Perrig [18]. Since then, single keyword search SSE schemes [6, 9, 11, 14], dynamic SSE schemes [15, 16, 17], multiple keyword search SSE schemes [3, 4, 8, 12, 13, 19] and more [10] have been studied extensively by many researchers.

In particular, for single keyword search SSE schemes, Curtmola, Garay, Kamara and Ostrovsky [6, 7] showed a rigorous definition of privacy of the

A preliminary version of this paper appeared in [14].

client[1] after a series of works [1, 9, 11, 18]. They also constructed SSE-2, and claimed that it satisfies their definition of privacy.

However, privacy is not sufficient. A malicious server may return incorrect search results to the client. Chang and Mitzenmacher showed how to detect such cheating by assuming that the client can always tell whether a file D_j is associated with a given keyword w_i or not by checking D_j [9, Sec.5].

On the other hand, even if a protocol Π is secure in a "stand-alone" setting where only a single protocol instance runs in isolation, it may not be secure in a more complex setting. To this problem, Canetti introduced a notion of universal composability (UC), and proved the UC composition theorem which states that if a protocol Π is UC secure, then its security is preserved under a general protocol composition operation [5].

In this paper,

- We first extend the model of SSE schemes to that of *verifiable* SSE schemes, and define the reliability without assuming the assumption that Chang and Mitzenmacher made in [9, Sec.5].

  Note that their assumption does not hold if the files are pictures or videos, for example. Also if a malicious server replaces $(C_i, \mathtt{MAC}(C_i))$ with some $(C_j, \mathtt{MAC}(C_j))$ in the search phase, then their method cannot detect such cheating.

- We next formulate the UC security of verifiable SSE schemes, and prove its weak equivalence with privacy and reliability.

- Finally we show an efficient verifiable SSE scheme, and prove that it is UC-secure.

We also point out a flaw of SSE-2, and show how to fix the flaw.

2 Preliminaries

2.1 Notation

If X is a string, then $|X|$ denotes the bit length of X. If X is a set, then $|X|$ denotes the cardinality of X. Let $x \overset{\$}{\leftarrow} X$ denote sampling an element from X at random and assigning it to x. PPT means probabilistic polynomial time.

[1] "adaptive semantic security" [7, Definition 4.11]

Let λ be the security parameter. We say that a function $\epsilon(\lambda)$ is negligible if it vanishes faster than the inverse of any polynomial in λ.

2.2 Pseudorandom Permutation and Function

Let $\mathsf{Perm}(\mathcal{X})$ be the set of all permutations on a set $\mathcal{X}$. We say that a polynomial time computable function $f : \{0,1\}^\lambda \times \mathcal{X} \to \mathcal{X}$ is a pseudorandom permutaton if $f(k,\cdot)$ is a permutation on $\mathcal{X}$ for any $k \in \{0,1\}^\lambda$ and

$$|\Pr(B^{f(k,\cdot)} = 1 : k \overset{\$}{\leftarrow} \{0,1\}^\lambda) - \Pr(B^\pi = 1 : \pi \overset{\$}{\leftarrow} \mathsf{Perm}(\mathcal{X}))|$$

is negligible for any PPT distinguisher B. We also say that π is a random permutation if $\pi \overset{\$}{\leftarrow} \mathsf{Perm}(\mathcal{X})$.

We say that a polynomial time computable function $f : \{0,1\}^\lambda \times \{0,1\}^{\ell_1} \to \{0,1\}^{\ell_2}$ is a pseudorandom function if

$$|\Pr(B^{f(k,\cdot)} = 1 : k \overset{\$}{\leftarrow} \{0,1\}^\lambda) - \Pr(B^{\mathsf{RO}} = 1)|$$

is negligible for any PPT distinguisher B, where $\mathsf{RO} : \{0,1\}^{\ell_1} \to \{0,1\}^{\ell_2}$ is the random oracle.

We sometimes write $f_k(\cdot)$ instead of $f(k,\cdot)$, where k is a key.

2.3 Symmetric-Key Encryption

Let $\mathtt{SKE} = (G, E, E^{-1})$ be a symmetric-key encryption scheme, where G is a key generation algorithm, E is an encryption algorithm and E^{-1} is the decryption algorithm. Let $\mathtt{left}(m_0, m_1) = m_0$ and $\mathtt{right}(m_0, m_1) = m_1$. We say that $\mathtt{SKE}$ is Left-or-Right (LOR) secure [2] if

$$|\Pr(A^{E_k(\mathtt{left}(\cdot,\cdot))} = 1) - \Pr(A^{E_k(\mathtt{right}(\cdot,\cdot))} = 1)|$$

is negligible for any PPT adversary A who queries (m_0, m_1) such that $|m_0| = |m_1|$ to the oracle, where $k \overset{\$}{\leftarrow} G(1^\lambda)$.

It is known that the counter mode is LOR secure if the underlying block cipher (say, AES) is a pseudorandom permutation [2].

3 Verifiable Searchable Symmetric Encryption

In this section, we extend the model of searchable symmetric encryption (SSE) schemes to that of *verifiable* SSE schemes. We also define its *reliability* as well as its privacy.

Let $\mathcal{D} = \{D_1, \cdots, D_n\}$ be a set of files, and $\mathcal{W} = \{w_1, \cdots, w_m\}$ be a set of keywords. Let $\texttt{Index} = (e_{i,j})$ be an $m \times n$ binary matrix such that

$$e_{i,j} = \begin{cases} 1 & \text{if a keyword } w_i \text{ is contained in a file } D_j \\ 0 & \textit{otherwise} \end{cases} . \tag{1}$$

Let $\texttt{D}(w)$ be the set of files D_j which contain a keyword w, and $\texttt{C}(w)$ be the set of ciphertexts C_j of $D_j \in \texttt{D}(w)$, and $\texttt{List}(w)$ be the set of indexes j of $D_j \in \texttt{D}(w)$. Namely

$$\begin{aligned} \texttt{List}(w_i) &= \{j \mid e_{i,j} = 1\}, \\ \texttt{D}(w_i) &= \{D_j \mid j \in \texttt{List}(w_i)\}, \\ \texttt{C}(w_i) &= \{C_j \mid j \in \texttt{List}(w_i)\}. \end{aligned}$$

Example 3.1 *Let* $\mathcal{D} = \{D_1, \cdots, D_5\}$, $\mathcal{W} = \{w_1, w_2\}$ *and*

$$\texttt{Index} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}. \tag{2}$$

Then

$$\begin{aligned} \texttt{List}(w_1) &= \{1, 3, 5\}, \\ \texttt{D}(w_1) &= \{D_1, D_3, D_5\}, \\ \texttt{C}(w_1) &= \{C_1, C_3, C_5\} \end{aligned}$$

and

$$\begin{aligned} \texttt{List}(w_2) &= \{2, 4\}, \\ \texttt{D}(w_2) &= \{D_2, D_4\}, \\ \texttt{C}(w_2) &= \{C_2, C_4\}. \end{aligned}$$

3.1 Verifiable SSE

A verifiable SSE scheme consists of six polynomial time algorithms

$$\texttt{vSSE} = (\texttt{Gen}, \texttt{Enc}, \texttt{Trpdr}, \texttt{Search}, \texttt{Dec}, \texttt{Verify})$$

such that

- $K \leftarrow \texttt{Gen}(1^\lambda)$: is a probabilistic algorithm which generates a key K.

- $(\mathcal{C}, \mathcal{I}) \leftarrow \texttt{Enc}(K, \mathcal{D}, \mathcal{W}, \texttt{Index})$: is a probabilistic encryption algorithm which outputs an encrypted index $\mathcal{I}$ and the set of ciphertexts $\mathcal{C} = (C_1, \cdots, C_n)$, where $\mathcal{D} = (D_1, \cdots, D_n)$ and C_j is a ciphertext of D_j for $j = 1, \ldots, n$.

  We assume that $\mathcal{I}$ includes an authenticator Tag_i related to a keyword w_i for $i = 1, \ldots, m$.

- $t(w) \leftarrow \texttt{Trpdr}(K, w)$: is a deterministic algorithm which outputs a trapdoor $t(w)$ for a keyword w.

- $(\texttt{C}(w), \boxed{Tag}) \leftarrow \texttt{Search}(\mathcal{I}, \mathcal{C}, t(w))$: is a deterministic search algorithm.

- $\boxed{\texttt{accept/reject} \leftarrow \texttt{Verify}(K, t(w), \texttt{C}'(w), Tag')}$: is a deterministic verification algorithm which checks the validity of $(\texttt{C}'(w), Tag')$.

- $D_j \leftarrow \texttt{Dec}(K, C_j)$: is a deterministic decryption algorithm.

(Correctness) Suppose that

$$K \leftarrow \texttt{Gen}(1^\lambda),\ (\mathcal{C}, \mathcal{I}) \leftarrow \texttt{Enc}(K, \mathcal{D}, \mathcal{W}, \texttt{Index}),$$

$$t(w) \leftarrow \texttt{Trpdr}(K, w),\ (\texttt{C}(w), \boxed{Tag}) \leftarrow \texttt{Search}(\mathcal{I}, \mathcal{C}, t(w)).$$

Then it must be that

$$\boxed{\texttt{accept} \leftarrow \texttt{Verify}(K, t(w), \texttt{C}(w), Tag)}$$

and

$$\texttt{D}(w) = \{D_j \mid D_j = \texttt{Dec}(K, C_j), C_j \in \texttt{C}(w)\}.$$

The definition and the correctness of SSE schemes [6, 7] are obtained by deleting the boxed parts.

We next translate a verifiable SSE scheme into a protocol $\Pi = (client, server)$ which consists of a store phase and a search phase. The store phase is shown in Fig.1 and is executed once. The search phase is shown in Fig.2 and is executed polynomially many times.

Store phase:

1. On input $(\mathcal{D}, \mathcal{W}, \texttt{Index})$, the client generates a key $K \leftarrow \texttt{Gen}(1^\lambda)$. He computes
$$(\mathcal{C}, \mathcal{I}) \leftarrow \texttt{Enc}(K, \mathcal{D}, \mathcal{W}, \texttt{Index})$$
and sends them to the server.

2. The server receives $(\mathcal{C}, \mathcal{I})$ and then stores them.

Figure 1: Store Phase

3.2 Privacy

In this subsection, we formulate *privacy* based on the work of Curtmola, Garay, Kamara and Ostrovsky [6, 7].

In the store phase of any (verifiable) SSE scheme, the number of keywords m, the number of files n and $|D_j|$ for $j = 1, \ldots, n$ are leaked to the server from $(\mathcal{C}, \mathcal{I})$. In each search phase, $\texttt{List}(w)$ is leaked to the server, where w is the search keyword, because otherwise the server cannot return $\texttt{C}(w)$ to the client.

Further suppose that the client sends $t(w_1)$ to the server in the first search phase and the tenth search phase. Then the server sees that the keywords the client searched in the first search phase and in the tenth search phase are the same.

We call these leaked information the minimum leakage. The notion of *privacy* requires that the server should not be able to learn any more information.

Formally, we consider a real game $\texttt{Game}_{real}$ and a simulation game $\texttt{Game}_{sim}$. The real game $\texttt{Game}_{real}$ is played by two PPT players, a distinguisher B and a challenger, as follows.

($\texttt{Game}_{real}$: Store phase)

1. A distinguisher B chooses $(\mathcal{D}, \mathcal{W}, \texttt{Index})$ and sends them to the challenger.

2. The challenger generates $K \leftarrow \texttt{Gen}(1^\lambda)$, and
   sends $(\mathcal{C}, \mathcal{I}) \leftarrow \texttt{Enc}(K, \mathcal{D}, \mathcal{W}, \texttt{Index})$ to B.

Search phase:

1. On input a keyword w, the client computes a trapdoor $t(w) \leftarrow \texttt{Trpdr}(K, w)$ and sends it to the server.
2. The server computes $(\texttt{C}(w), Tag) \leftarrow \texttt{Search}(\mathcal{I}, \mathcal{C}, t(w))$ and returns them to the client.
3. If the client receives $(\texttt{C}'(w), Tag')$ from the server,

   then the client computes

$$\texttt{accept}/\texttt{reject} \leftarrow \texttt{Verify}(K, t(w), (\texttt{C}'(w), Tag')).$$

   - If the result is `accept`, then the client decrypts

$$D_j \leftarrow \texttt{Dec}(K, C_j)$$

     for each $C_j \in \texttt{C}'(w)$, and outputs $\texttt{D}(w) = \{D_j \mid C_j \in \texttt{C}'(w)\}$.
   - Otherwise the client outputs `reject`.

Figure 2: Search Phase

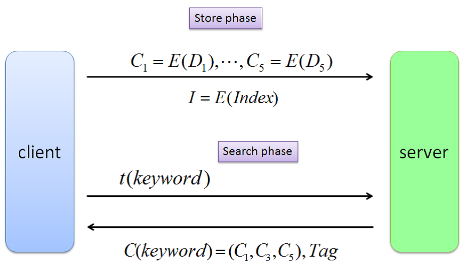
Store phase
$C_1 = E(D_1), \cdots, C_5 = E(D_5)$
$I = E(Index)$
client
Search phase
server
$t(keyword)$
$C(keyword) = (C_1, C_3, C_5), Tag$

Figure 3: Verifiable SSE scheme

($\texttt{Game}_{real}$: **Search phase**) For $i = 1, \cdots, q$, do:

1. B chooses a keyword $\tilde{w}_i$ and sends it to the challenger.
2. The challenger sends a trapdoor $t(\tilde{w}_i) \leftarrow \texttt{Trpdr}_K(\tilde{w}_i)$ to B.

Finally B outputs a bit b. (See Fig.4 and Fig.5.)

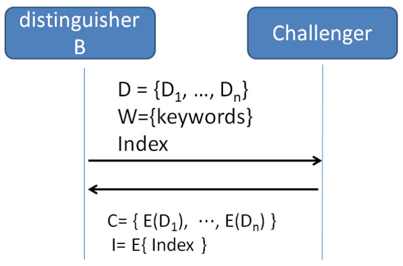

Figure 4: $\texttt{Game}_{real}$: Store phase

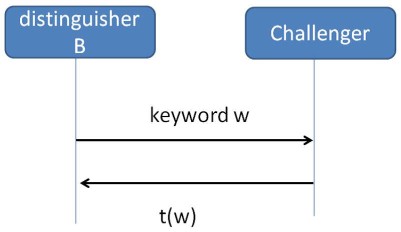

Figure 5: $\texttt{Game}_{real}$: Search phase

The simulation game $\texttt{Game}_{sim}$ is played by three PPT players, a distinguisher B and a challenger and a simulator Sim, as described below.

($\texttt{Game}_{sim}$: Store phase)

1. B chooses $(\mathcal{D}, \mathcal{W}, \texttt{Index})$ and sends them to the challenger.
2. The challenger sends $m, n, |D_1|, \cdots, |D_n|$ to Sim.

   He sets $c_0 \leftarrow 0$ and $J \leftarrow \emptyset$.
3. Sim computes $(\texttt{C}', \mathcal{I}')$ from m, n and $|D_1|, \cdots, |D_n|$,

   and sends them to the challenger.
4. The challenger returns $(\texttt{C}', \mathcal{I}')$ to B.

($\texttt{Game}_{sim}$: Search phase) For $i = 1, \cdots, q$, do:

1. B chooses a keyword $\tilde{w}_i$ and sends it to the challenger.
2. If $(\tilde{w}_i, c) \in J$ for some c, then the challenger sends c to Sim.

   Otherwise he sends $\texttt{List}(\tilde{w}_i)$ to Sim. He then sets $c_0 \leftarrow c_0 + 1$ and $J \leftarrow J \cup \{(\tilde{w}_i, c_0)\}$.
3. Sim returns $t'(\tilde{w}_i)$ to the challenger.
4. The challenger returns $t'(\tilde{w}_i)$ to B.

Finally B outputs a bit b. (See Fig.6 and Fig.7.)

Then we define the advantage of privacy as

$$\mathsf{Adv}^{priv}_{\mathsf{Sim}}(B) = |\Pr(b = 1 \text{ in } \texttt{Game}_{real}) - \Pr(b = 1 \text{ in } \texttt{Game}_{sim})|. \tag{3}$$

Further let

$$\mathsf{Adv}^{priv}_{\mathsf{Sim}} = \max_{B} \mathsf{Adv}_{\mathsf{Sim}}(B).$$

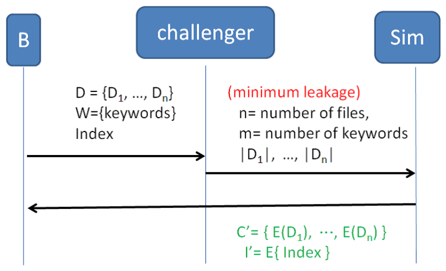

Figure 6: $\texttt{Game}_{sim}$: Store phase

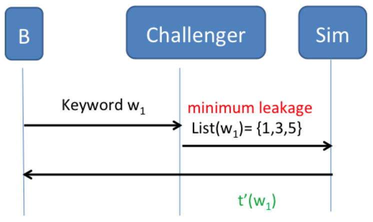

Figure 7: $\texttt{Game}_{sim}$: Search phase

Definition 3.1 *We say that a (verifiable) SSE scheme satisfies privacy if there exists a PPT simulator* Sim *such that* $\mathsf{Adv}_{\mathsf{Sim}}^{priv}$ *is negligible.*

"Adaptive semantic security" of Curtmola et al. [7, Definition 4.11] requires that for any PPT distinguisher B, there exists a PPT Sim such that eq.(3) is negligible. On the other hand, our definition requires that there exists a PPT Sim such that for any PPT distinguisher B, eq.(3) is negligible. Hence our definition is slightly stronger. This small change is important when we prove the relationship with UC-security.

3.3 Reliability

In this subsection, we formulate *reliability* of verifiable SSE schemes. Suppose that a malicious server returned an incorrect search result. Then the *reliability* requires that the client can detect this cheating.

Consider the following attack game among three PPT algorithms, the client, A_1 and A_2, where $A = (A_1, A_2)$ is an adversary. (See Fig.8.) We assume that A_1 and A_2 can communicate freely.

(Store phase)

1. A_1 chooses $(\mathcal{D}, \mathcal{W}, \texttt{Index})$ and sends them to the client.
2. The client generates $K \leftarrow \texttt{Gen}(1^\lambda)$, and then

   sends $(\mathcal{C}, \mathcal{I}) \leftarrow \texttt{Enc}(K, \mathcal{D}, \mathcal{W}, \texttt{Index})$ to A_2.

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

1. A_1 chooses a keyword $\tilde{w}_i$ and sends it to the client.
2. The client sends a trapdoor $t(\tilde{w}_i) \leftarrow \texttt{Trpdr}_K(\tilde{w}_i)$ to A_2.

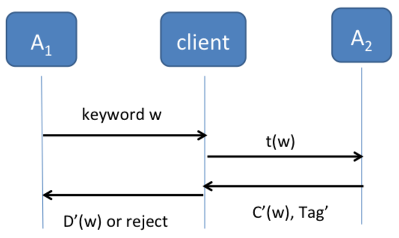

Figure 8: Attack game on reliability

3. A_2 returns $(\mathtt{C}'(\tilde{w}_i), Tag'_i)$ to the client.
4. The client returns $\mathtt{D}'(\tilde{w}_i)$ or `reject` to A_1 according to Fig.2.

We say that the adversary (A_1, A_2) succeeds if A_1 receives $\mathtt{D}'(\tilde{w}_i)$ such that $\mathtt{D}'(\tilde{w}_i) \neq \mathtt{D}(\tilde{w}_i)$ for some $\tilde{w}_i$. In this case, we let A_1 output 1. Otherwise A_1 outputs 0.

Then we define the advantage of reliability by

$$\mathsf{Adv}^{auth}(A_1, A_2) = \Pr((A_1, A_2) \text{ succeeds }).$$

Further let

$$\mathsf{Adv}^{auth} = \max_{(A_1,A_2)} \mathsf{Adv}^{auth}(A_1, A_2).$$

Definition 3.2 *We say that a verifiable SSE scheme satisfies reliability if* Adv^{auth} *is negligible.*

Further we say that the adversary (A_1, A_2) strongly succeeds if the client accepts $(\mathtt{C}'(\tilde{w}_i), Tag'_i)$ such that $(\mathtt{C}'(\tilde{w}_i), Tag'_i) \neq (\mathtt{C}(\tilde{w}_i), Tag_i)$ for some $\tilde{w}_i$, where

$$(\mathtt{C}(\tilde{w}_i), Tag_i) \leftarrow \mathtt{Search}(\mathcal{I}, \mathcal{C}, t(\tilde{w}_i)).$$

In this case, we let A_1 output 1. Otherwise A_1 outputs 0. Then we define the advantage of strong reliability by

$$\mathsf{Adv}^{sauth}(A_1, A_2) = \Pr((A_1, A_2) \text{ strongly succeeds }).$$

Further let

$$\mathsf{Adv}^{sauth} = \max_{(A_1,A_2)} \mathsf{Adv}^{sauth}(A_1, A_2).$$

Definition 3.3 *We say that a verifiable SSE scheme satisfies strong reliability if* Adv^{sauth} *is negligible.*

It is easy to see that a verifiable SSE scheme satisfies reliability if it satisfies strong reliability.

4 UC Secure Verifiable SSE Scheme

In this section, we define the universally composable (UC) security [5] of verifiable SSE schemes. More precisely, we define the ideal functionality of verifiable SSE schemes.

4.1 UC Framework

In general, even if a protocol Π is secure in a "stand-alone" setting where only a single protocol instance runs in isolation, it may not be secure in a more complex setting. To this problem, Canetti introduced a notion of universal composability (UC), and proved the UC composition theorem which states that if a protocol Π is UC secure, then its security is preserved under a general protocol composition operation [5].

In the UC framework of a given protocol Π, the real world and the ideal world of Π are defined. In the ideal world, a real world adversary A can corrupt some of the parties who run the protocol Π. In the ideal world, each party is replaced with a dummy party, and the run of Π is replaced with an ideal functionality $\mathcal{F}$. Also an ideal world adversary S can corrupt some dummy parties.

Then the protocol Π is called UC-secure if no environment $\mathcal{Z}$ can distinguish the real world and the ideal world, where $\mathcal{Z}$ generates the input to all parties, receives all their outputs, and in addition interacts with A or S in an arbitrary way.

More precisely, Π is said to securely realize the ideal functionality $\mathcal{F}$ if for any adversary A, there exists an ideal world adversary S such that no environment $\mathcal{Z}$ can tell whether it is interacting with A and the parties running the protocol, or with S and the dummy parties that interact with $\mathcal{F}$ in the ideal world.

4.2 Real World

In this subsection, we describe the real world of a verifiable SSE scheme in the UC framework. For simplicity, we ignore session id.

(Store phase)

1. An environment $\mathcal{Z}$ chooses $(\mathcal{D}, \mathcal{W}, \texttt{Index})$ and sends them to the client.
2. The client generates a key $K \leftarrow \texttt{Gen}(1^\lambda)$.

   He computes

$$(\mathcal{C}, \mathcal{I}) \leftarrow \texttt{Enc}(K, \mathcal{D}, \mathcal{W}, \texttt{Index})$$

and sends them to the server.

3. The server receives $(\mathcal{C}, \mathcal{I})$ and then stores them.

(Search phase)

1. $\mathcal{Z}$ chooses a keyword w, and sends it to the client.

2. The client computes a trapdoor $t(w) \leftarrow \texttt{Trpdr}(K, w)$ and sends it to the server.

3. The server returns $(\texttt{C}'(w), Tag')$ to the client.

4. The client computes

$$\texttt{accept/reject} \leftarrow \texttt{Verify}(K, t(w), \texttt{C}'(w), Tag').$$

   - If the result is `accept`, then the client decrypts

$$D_j \leftarrow \texttt{Dec}(K, C_j)$$

     for each $C_j \in \texttt{C}(w)$, and sends $\texttt{D}(w) = \{D_j \mid C_j \in \texttt{C}(w)\}$ to $\mathcal{Z}$.
   - Otherwise the client sends `reject` to $\mathcal{Z}$.

An adversary A can control the server arbitrarily. (We assume that the client is honest.) $\mathcal{Z}$ can communicate with A in an arbitrary way. Finally $\mathcal{Z}$ outputs 1 or 0. (See Fig.9.)

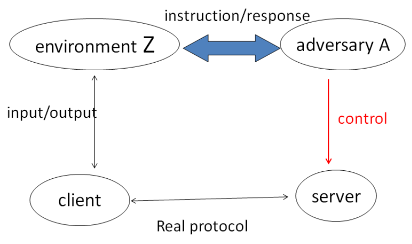

Figure 9: Real world of verifiable SSE scheme

4.3 Ideal World

In this subsection, we introduce the ideal functionality $\mathcal{F}_{\text{SSE}}$, and describe the ideal world of a verifiable SSE scheme. For simplicity, we ignore session id.

(Store phase) (See Fig.10.)

1. $\mathcal{Z}$ chooses $(\mathcal{D}, \mathcal{W}, \texttt{Index})$ and sends them to the dummy client.
2. The dummy client relays $(\mathcal{D}, \mathcal{W}, \texttt{Index})$ to $\mathcal{F}_{\text{SSE}}$.
3. $\mathcal{F}_{\text{SSE}}$ records $(\mathcal{D}, \mathcal{W}, \texttt{Index})$. It then sends $m, n, |D_1|, \cdots, |D_n|$ to the ideal adversary S.

   It sets $c_0 \leftarrow 0$ and $J \leftarrow \emptyset$.

(Search phase) (See Fig.11.)

1. $\mathcal{Z}$ chooses a keyword w, and sends it to the dummy client.
2. The dummy client relays w to $\mathcal{F}_{\text{SSE}}$.
3. If $(w, c) \in J$ for some c, then $\mathcal{F}_{\text{SSE}}$ sends c to S.

   Otherwise it sends $\texttt{List}(w)$ to S. It then sets $c_0 \leftarrow c_0 + 1$ and $J \leftarrow J \cup \{(w, c_0)\}$.
4. S returns `accept` or `reject` to $\mathcal{F}_{\text{SSE}}$.
5. $\mathcal{F}_{\text{SSE}}$ sends

$$X = \begin{cases} \texttt{D}(w) & \text{if } \mathcal{F}_{\text{SSE}} \text{ received } \texttt{accept} \text{ from } \mathsf{S} \\ \texttt{reject} & \text{if } \mathcal{F}_{\text{SSE}} \text{ received } \texttt{reject} \text{ from } \mathsf{S} \end{cases}$$

   to the dummy client.
6. The dummy client relays X to $\mathcal{Z}$.

$\mathcal{Z}$ can communicate with S in an arbitrary way. (See Fig.12.) Finally $\mathcal{Z}$ outputs 1 or 0.

Note that

- The ideal world adversary S learns only the minimum leakage (which is shown in Sec.3.2).
- The dummy client receives $\texttt{D}(w)$ or `reject`. It means that he never receives $\texttt{D}'(w)$ such that $\texttt{D}'(w) \neq \texttt{D}(w)$.

Therefore this is the ideal world.

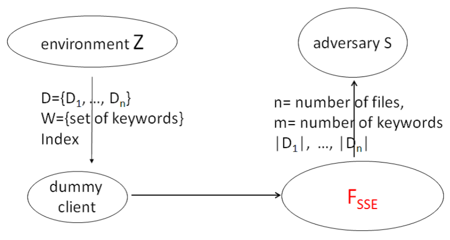

Figure 10: Ideal world (store)

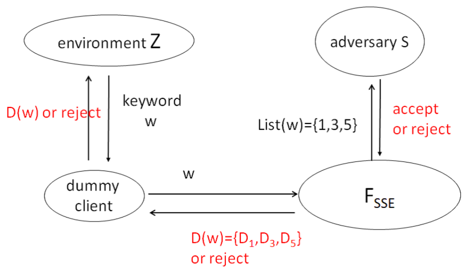

Figure 11: Ideal world (search)

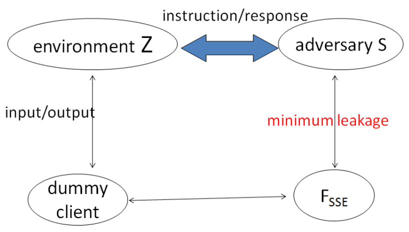

Figure 12: Ideal world

4.4 UC Security

Let

$$\begin{aligned} \mathrm{P}_{real} &= \Pr(\mathcal{Z} \text{ outputs 1 in the real world}), \\ \mathrm{P}_{ideal} &= \Pr(\mathcal{Z} \text{ outputs 1 in the ideal world}), \\ \mathsf{Adv}_{\mathsf{S}}^{uc}(\mathcal{Z}, \mathsf{A}) &= |\mathrm{P}_{real} - \mathrm{P}_{ideal}|. \end{aligned}$$

Then we say that a verifiable SSE scheme securely realizes the ideal functionality $\mathcal{F}_{\mathrm{SSE}}$ if for any real world adversary A, there exists an ideal world adversary S such that $\mathsf{Adv}_{\mathsf{S}}^{uc}(\mathcal{Z}, \mathsf{A})$ is negligible for any environment $\mathcal{Z}$.

Now we have the following from the UC composition theorem [5]. Let Σ be a larger protocol which uses a verifiable SSE scheme Π as a sub-protocol. Let Σ' be a variant of Σ such that Π is replaced with the ideal functionality $\mathcal{F}_{\mathrm{SSE}}$. Then Σ is as secure as Σ' if Π securely realizes $\mathcal{F}_{\mathrm{SSE}}$.

Namely for any adversary A against Σ, there exists an adversary S against Σ' such that no environment $\mathcal{Z}$ can distinguish between (Σ, A) and (Σ', S).

5 Weak Equivalence

In this section, we prove the following Theorems.

Theorem 5.1 *If a verifiable SSE scheme* Π *securely realizes* $\mathcal{F}_{\rm SSE}$*, then* Π *satisfies privacy and reliability.*

Theorem 5.2 *If a verifiable SSE scheme* Π *satisfies privacy and strongly reliability, then* Π *securely realizes* $\mathcal{F}_{\rm SSE}$*.*

5.1 Proof of Theorem 5.1

(Proof of privacy) In the real world (see Fig.9), consider an adversary A_0 such that A_0 sends each message Y that the server received from the client to $\mathcal{Z}$. (Namely $Y = (\mathcal{C}, \mathcal{I})$ in the store phase, and $Y = t(w)$ in the search phase.) Look at $\mathcal{Z}$ as a distinguisher and (client, server, A) as a challenger as shown in Fig.13. Then this real world can be seen as the real game of privacy.

On the other hand, in the ideal world, there exists an adversary S which sends almost the same Y to $\mathcal{Z}$ because $\mathcal{Z}$ cannot distinguish between the real world and the ideal world from our assumption. Now look at $\mathcal{Z}$ as a distinguisher, S as a simulator and (dummy-client, $\mathcal{F}_{\rm SSE}$) as a challenger as shown in Fig.14. Then this real world can be seen as the ideal game of privacy.

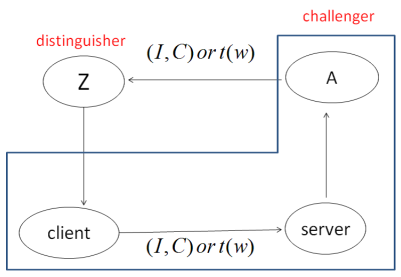

Figure 13: Real world (privacy)

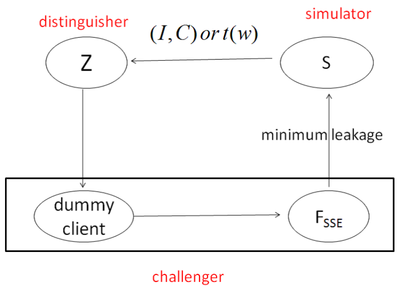

Figure 14: Ideal world (privacy)

This means that

$$\mathsf{Adv}_S^{priv}(\mathcal{Z}) = \mathsf{Adv}_{\mathsf{S}}^{uc}(\mathcal{Z}, \mathsf{A}_0).$$

Therefore we have

$$\begin{aligned}\mathsf{Adv}_S^{priv} &= \max_{\mathcal{Z}} \mathsf{Adv}_S^{priv}(\mathcal{Z}) \\ &= \max_{\mathcal{Z}} \mathsf{Adv}_S^{uc}(\mathcal{Z}, A_0) \\ &= \textit{negligible}\end{aligned}$$

from our assumption. Hence Π satisfies privacy.

(Proof of reliability) For an adversary (A_1, A_2) on reliability (see Sec.3.3), consider an environment $\mathcal{Z}$ and a real world adversary A such that $(\mathcal{Z}, \mathsf{A}) = (A_1, A_2)$. Then there exists an ideal world adversary S such that

$$\mathsf{Adv}_S^{uc}(\mathcal{Z}, A) = |\mathrm{P}_{real} - \mathrm{P}_{sim}| = \textit{negligible}$$

from our assumption.

In the ideal world, $\mathcal{Z}$ never receives $\mathtt{D}'(w)$ such that $\mathtt{D}'(w) \neq \mathtt{D}(w)$ for any search keyword w (see Fig.11). Therefore

$$\begin{aligned}\mathrm{P}_{ideal} &= \Pr(\mathcal{Z} \text{ outputs } 1) \\ &= \Pr(A_1 \text{ outputs } 1) \\ &= 0.\end{aligned}$$

Hence P_{real} =negligible. This means that

$$\mathsf{Adv}^{auth}(A_1, A_2) = \mathrm{P}_{real} = \textit{negligible.}$$

Therefore Π satisfies reliability.

5.2 Proof of Theorem 5.2

We say that $(\mathtt{C}'(w), Tag')$ is invalid if $(\mathtt{C}'(w), Tag') \neq (\mathtt{C}(w), Tag)$, where

$$(\mathtt{C}(w), Tag) \leftarrow \mathtt{Search}(\mathcal{I}, \mathcal{C}, t(w)). \tag{4}$$

Fix a real world adversary A arbitrarily. In the following, we consider a series of games $\mathtt{Game}_0, \cdots, \mathtt{Game}_3$, where $\mathtt{Game}_0$ is the real world. Let

$$p_i = \Pr(\mathcal{Z} \text{ outputs 1 in } \mathtt{Game}_i).$$

$(\mathtt{Game}_1)$ In this game, we modify $\mathtt{Game}_0$ as follows.

In the store phase, the client records $(\mathcal{D}, \mathcal{W}, \mathtt{Index})$.

In the search phase, suppose that $\mathcal{Z}$ sends a keyword w to the client.

1. If A instructs the server to return an invalid $(\mathtt{C}'(w), Tag')$, then the server returns `reject` to the client. [2]

   Otherwise the server returns `accept` to the client.
2. If the client receives `reject` from the server, then he sends `reject` to $\mathcal{Z}$.

   If the client receives `accept` from the server, then he sends $\mathtt{D}(w)$ to $\mathcal{Z}$.

See Fig.15.

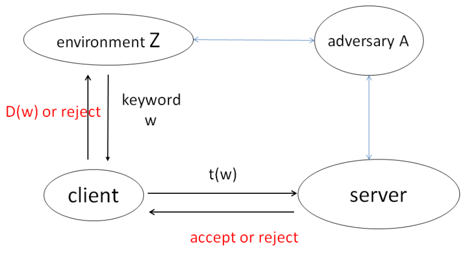

Figure 15: Game 1

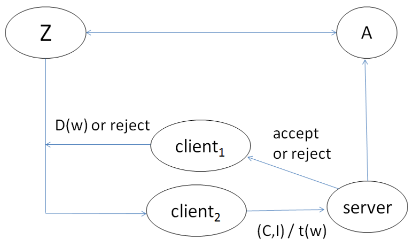

Figure 16: Game 2

Let BAD be the event that the client accepts an invalid $(\mathtt{C}'(w), Tag')$ in $\mathtt{Game}_0$. Then it holds that

$$|p_0 - p_1| \leq \Pr(\mathsf{BAD})$$

Now consider an adversary (A_1, A_2) on the reliability such that $(A_1, A_2) = (\mathcal{Z}, (\mathsf{A}, server))$ in $\mathtt{Game}_0$. Then we can see that

$$\Pr(\mathsf{BAD}) = \mathsf{Adv}^{sauth}(A_1, A_2)$$

Therefore we have

$$|p_0 - p_1| \leq \mathsf{Adv}^{sauth}.$$

$(\mathtt{Game}_2)$ In this game, we modify $\mathtt{Game}_1$ as follows. We replace the client with $(client_1, client_2)$ such that the server receives a message from $client_2$, and sends `accept` or `reject` to $client_1$. (See Fig.16.) Namely;

- Both of $client_1$ and $client_2$ receive the (same) input from $\mathcal{Z}$.
- In each phase, if the client sends Y to the server, $client_2$ sends Y to the server.

[2]The server first compute eq.(4).

- In the search phase, $client_1$ receives `accept` or `reject` from the server, and sends $\mathtt{D}(w)$ or `reject` to $\mathcal{Z}$.

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

($\mathtt{Game}_3$) In this game, we modify $\mathtt{Game}_2$ as follows. Since Π satisfies privacy from our assumption, there exists a simulator Sim such that $\mathsf{Adv}^{priv}_{\mathsf{Sim}} = negligible$.

Now in $\mathtt{Game}_3$, $client_2$ plays the role of the challenger in the simulation game of privacy. Namely he sends the minimum leakage to Sim. Sim then sends its outputs (the simulated message) to the server.

Further look at $(\mathcal{Z}, client_1, server, \mathsf{A})$ as a distinguisher of the privacy game (see Fig.17). Then $\mathtt{Game}_3$ is the simulation game and $\mathtt{Game}_2$ is the real game. Therefore it holds that

$$|p_3 - p_2| \leq \mathsf{Adv}^{priv}_{\mathsf{Sim}}.$$

In $\mathtt{Game}_3$, $(client_1, client_2)$ behaves exactly in the same way as the ideal functionality $\mathcal{F}_{\mathrm{SSE}}$. Further look at $(\mathsf{A}, server, \mathsf{Sim})$ as the ideal world adversary S (see Fig.18). Then $\mathtt{Game}_3$ can be seen as the ideal world of the UC framework.

Therefore we have

$$\begin{aligned} \mathsf{Adv}^{uc}_{\mathsf{S}}(\mathcal{Z}, A) &= |p_0 - p_3| \\ &\leq \mathsf{Adv}^{sauth} + \mathsf{Adv}^{priv}_{\mathsf{Sim}} \end{aligned}$$

for any $\mathcal{Z}$. Finally Adv^{sauth} and $\mathsf{Adv}^{priv}_{\mathsf{Sim}}$ are negligible from our assumption. Hence Π securely realizes $\mathcal{F}_{\mathrm{SSE}}$.

6 On SSE-2 of Curtmola et al.

In [6, 7], Curtmola et al. presented an SSE scheme called SSE-2, and claimed that it satisfies privacy.

In this section, we first point out a flaw of SSE-2. We next show how to fix the flaw. Then we prove the privacy of the modified SSE-2.

6.1 Flaw of SSE-2

We first illustrate SSE-2 by using Example 3.1.

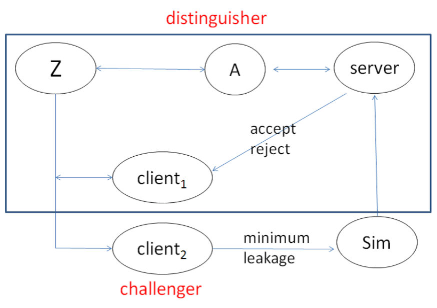

Figure 17: Game 3

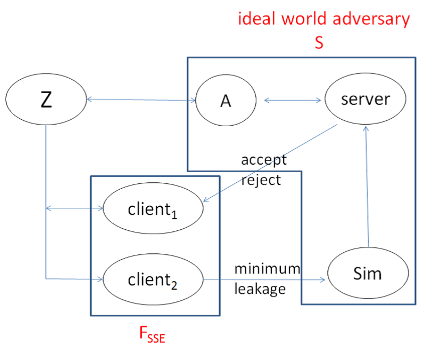

Figure 18: Game 3 = ideal world

(Store phase:) Let $\pi_k = \pi(k, \cdot)$ be a pseudorandom permutation, where k is a key. Then the client constructs an array $\mathcal{I}$ as follows. Initially, let $\mathcal{I}(x) = 0$ for all x. Next

- Since $\texttt{List}(w_1) = (1, 3, 5)$, set

$$\mathcal{I}(\pi_k(w_1, 1)) = 1,\ \mathcal{I}(\pi_k(w_1, 2)) = 3,\ \mathcal{I}(\pi_k(w_1, 3)) = 5, \qquad (5)$$

- Since $\texttt{List}(w_2) = (2, 4)$, set

$$\mathcal{I}(\pi_k(w_2, 1)) = 2,\ \mathcal{I}(\pi_k(w_2, 2)) = 4 \qquad (6)$$

The client stores $\mathcal{I}$ and $\mathcal{C} = \{C_1, \cdots, C_5\}$ to the server, where C_j is a ciphertext of D_j.

(Search phase:) Suppose that the client wants to retrieve the files which contain w_1. Then the client sends

$$t(w_1) = (\pi_k(w_1, 1), \ldots, \pi_k(w_1, 5))$$

to the server.

From eq.(5), the server sees that $\texttt{List}(w_1) = (1, 3, 5)$. The server then returns $\texttt{C}(w_1) = \{C_1, C_3, C_5\}$ to the client.

The client finally decrypts them to obtain $\texttt{D}(w_1) = \{D_1, D_3, D_5\}$.

The above scheme, however, does not satisfy privacy. The server sees that each file contains just one keyword because each file index $j \in \{1, \ldots, 5\}$ appears once in $\mathcal{I}$.

To solve this problem, the client does the following in the store phase of SSE-2 [7, Fig.2]. For each file D_j:

(a) Let c be the number of entries in $\mathcal{I}$ that already contain j.

(This means that j appears c times in $\mathcal{I}$.)

(b) For $1 \leq h \leq m - c$, set $\mathcal{I}[\pi_k(0^\ell, n + h)] = j$,

where ℓ is the bit length of each keyword.

In the above example, $n = 5, m = 2$, and $c = 1$ for $j = 1, \ldots, 5$. Therefore the above procedure says that for $j = 1, \ldots, 5$, set $\mathcal{I}[\pi_k(0^\ell, 5 + 1)] = j$. This means that the client sets

$$\begin{aligned} \mathcal{I}[\pi_k(0^\ell, 6)] &\leftarrow 1 \\ &\vdots \\ \mathcal{I}[\pi_k(0^\ell, 6)] &\leftarrow 5 \end{aligned}$$

At the end, we have $\mathcal{I}[\pi_k(0^\ell, 6)] = 5$ only ! Namely only 5 appears twice, but each $j \in \{1, \ldots, 4\}$ appears once in $\mathcal{I}$.

Therefore the above scheme (which is SSE-2) still does not satisfy privacy.

6.2 Modified SSE-2

In this subsection, we show how to fix the flaw of SSE-2.

We assume that $\mathcal{W} = \{0, 1\}^\ell$ for some $\ell = O(\log_2 \lambda)$, where λ is the security parameter. Hence $m = 2^\ell$. Let $\mathtt{SKE} = (G, E, E^{-1})$ be a symmetric-key encryption scheme.

(Store phase:) The client takes $(\mathcal{D}, \mathcal{W}, \mathtt{Index})$ as an input, where $\mathcal{W} = \{0, 1\}^\ell$ and $\mathtt{Index} = (e_{i,j})$ is defined by eq.(1). Let $\pi_k = \pi(k, \cdot)$ be a pseudorandom permutation on

$$\mathcal{X} = \{0, 1\} \times \{0, 1\}^\ell \times \{1, \ldots, n\}. \tag{7}$$

1. The client generates $k_e \leftarrow G(1^\lambda)$ and computes $C_j = E_{k_e}(D_j)$ for $j = 1, \ldots, n$. Let $\mathcal{C} = (C_1, \ldots, C_n)$.
2. The client chooses a key k of π randomly, and sets

$$\mathcal{I}(\pi_k(1, w_i, j)) \leftarrow e_{i,j} \tag{8}$$
$$\mathcal{I}(\pi_k(0, w_i, j)) \leftarrow 1 - e_{i,j} \tag{9}$$

for $i = 1, \ldots, m$ and $j = 1, \ldots, n$, where $m = 2^\ell$ and $w_i \in \mathcal{W}$.

The client stores $(\mathcal{C}, \mathcal{I})$ to the server.

(Search phase:) The client takes a keyword $w \in \mathcal{W}$ as an input.

1. Let $a_j = \pi_k(1, w, j)$ for $j = 1, \ldots, n$. Then the client sends
$$t(w) = (a_1, \ldots, a_n)$$
to the server.
2. From $t(w) = (a_1, \ldots, a_n)$, the server sets
$$\mathtt{List}(w) = \{j \mid \mathcal{I}(a_j) = 1\}.$$
(Remember eq.(8).) She then returns $\mathtt{C}(w) = \{C_j \mid j \in \mathtt{List}(w)\}$ to the client.
3. For each $C_j \in \mathtt{C}(w)$, the client computes $D_j = E^{-1}(C_j)$ and outputs $\mathtt{D}(w) = \{D_j \mid C_j \in \mathtt{C}(w)\}$.

Consider Example 3.1. Then the client sets

$$\begin{array}{ll}
\mathcal{I}(\pi(1, w_1, 1)) = 1, & \mathcal{I}(\pi(0, w_1, 1)) = 0 \\
\mathcal{I}(\pi(1, w_1, 2)) = 0, & \mathcal{I}(\pi(0, w_1, 2)) = 1 \\
\mathcal{I}(\pi(1, w_1, 3)) = 1, & \mathcal{I}(\pi(0, w_1, 3)) = 0 \\
\mathcal{I}(\pi(1, w_1, 4)) = 0, & \mathcal{I}(\pi(0, w_1, 4)) = 1 \\
\mathcal{I}(\pi(1, w_1, 5)) = 1, & \mathcal{I}(\pi(0, w_1, 5)) = 0
\end{array}$$

For a search keyword w_1, the client sends

$$\begin{array}{rcl}
t(w_1) & = & (a_1, a_2, a_3, a_4, a_5) \\
 & = & (\pi(1, w_1, 1), \pi(1, w_1, 2), \pi(1, w_1, 3), \pi(1, w_1, 4), \pi(1, w_1, 5)).
\end{array}$$

to the server. The server sees that

$$\mathcal{I}(\pi(1, w_1, 1)) = \mathcal{I}(\pi(1, w_1, 3)) = \mathcal{I}(\pi(1, w_1, 5)) = 1.$$

Hence the server returns $\mathtt{C}(w_1) = \{C_1, C_3, C_5\}$ to the client.

6.3 Privacy of Modified SSE-2

Theorem 6.1 *The above SSE scheme satisfies privacy if* SKE *is LOR secure.*

(Proof) We construct a simulator Sim in the simulation game $\texttt{Game}_{sim}$ of the privacy game as follows.

(Store phase:)

1. Sim is given $n, m(= 2^{\ell})$ and $|D_1|, \cdots, |D_n|$ from the challenger.
2. Sim generates $k_e \leftarrow G(1^{\lambda})$ and computes $C'_j = E_{k_e}(0^{|D_j|})$ for $j = 1, \ldots, n$. Let $\mathcal{C}' = (C'_1, \ldots, C'_n)$.
3. Sim chooses a key k of psuedorandom permutation π on $\mathcal{X}$ randomly, where $\mathcal{X}$ is given by eq.(7).

   For $i = 1, \ldots, m$ and $j = 1, \ldots, n$, Sim sets

$$\begin{aligned} \mathcal{I}'(\pi_k(1, w_i, j)) &\leftarrow 1, & (10) \\ \mathcal{I}'(\pi_k(0, w_i, j)) &\leftarrow 0. & (11) \end{aligned}$$

Sim returns $(\mathcal{C}', \mathcal{I}')$ to the challenger.

(Search phase:) Sim sets $c_1 \leftarrow 0$ and $L \leftarrow \emptyset$.

For $i = 1, \ldots, q$, do:

- Suppose that Sim is given $\texttt{List}(w)$ for some $w \in \mathcal{W}$ by the challenger. Then let $c_1 \leftarrow c_1 + 1$. For $j = 1, \ldots, n$, set

$$a_j = \begin{cases} \pi_k(1, c_1, j) & if \quad j \in \texttt{List}(w) \\ \pi_k(0, c_1, j) & if \quad j \notin \texttt{List}(w) \end{cases}$$

  Let

$$\begin{aligned} t'(w) &\leftarrow (a_1, \ldots, a_n) \\ L &\leftarrow L \cup \{(c_1, t'(w))\} \end{aligned}$$

  Sim returns $t'(w)$ to the challenger.

- Suppose that Sim is given $c \in \{1, 2, \ldots\}$ by the challenger. Then Sim finds $(c, t) \in L$, and returns t to the challenger.

We will prove that no distinguisher B can distinguish between $\texttt{Game}_{real}$ and $\texttt{Game}_{sim}$ by using a series of games $\texttt{Game}_0, \cdots, \texttt{Game}_4$, where $\texttt{Game}_0 = \texttt{Game}_{real}$. Let

$$p_i = \Pr(B \text{ outputs } b = 1 \text{ in } \texttt{Game}_i).$$

- `Game`$_1$ is the same as `Game`$_0$ except for that C_j is replaced with $C'_j = E_{k_e}(0^{|D_j|})$ for $j = 1, \ldots, n$. Then $|p_1 - p_0|$ is negligible because `SKE` is LOR secure.

- `Game`$_2$ is the same as `Game`$_1$ except for that π_k is replaced with a random permutation $\bar{\pi}$ on $\mathcal{X}$. Then $|p_2 - p_1|$ is negligible because π_k is a pseudorandom permutation.

- `Game`$_3$ is the same as `Game`$_2$ except for the following. In the store phase, the client records `Index` $= (e_{i,j})$, and sets

$$\mathcal{I}(\bar{\pi}(1, w_i, j)) \leftarrow 1, \qquad (12)$$
$$\mathcal{I}(\bar{\pi}(0, w_i, j)) \leftarrow 0. \qquad (13)$$

for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. In the search phase, the client set

$$a_j = \begin{cases} \bar{\pi}(1, w, j) & if \quad j \in \texttt{List}(w) \\ \bar{\pi}(0, w, j) & if \quad j \notin \texttt{List}(w) \end{cases}$$

for $j = 1, \ldots, n$. Then it is easy to see that $p_3 = p_2$ because $\bar{\pi}$ is a random permutation.

- `Game`$_4$ is the same as `Game`$_3$ except for that $\bar{\pi}$ is replaced with a pseudorandom permutation π_k. Then $|p_4 - p_3|$ is negligible.

It is easy to see that $p_4 = \Pr(b = 1 \text{ in } \texttt{Game}_{sim})$. Therefore

$$\begin{aligned} \mathsf{Adv}^{priv}_{\mathsf{Sim}}(B) &= |\Pr(b = 1 \text{ in } \texttt{Game}_{real}) - \Pr(b = 1 \text{ in } \texttt{Game}_{sim})| \\ &= |p_0 - p_4| \\ &\leq |p_0 - p_1| + |p_1 - p_2| + |p_2 - p_3| + |p_3 - p_4| \\ &= \textit{negligible} \end{aligned}$$

for any distinguisher B. Therefore the SSE scheme satisfies privacy.

Q.E.D.

6.4 Efficiency of Modified SSE-2

In the modified SSE-2 scheme,

$$\begin{aligned} |\mathcal{I}| &= 2mn \\ |t(w)| &= n(\log_2 m + \log_2 n + 1) \end{aligned}$$

7 More Efficient (Verifiable) SSE Scheme

In this section, we first show a more efficient SSE scheme than the modified SSE-2. We next extend it to a verifiable SSE scheme.

Then we prove the privacy and the strong reliability of the verifiable SSE scheme. This means that the verifiable SSE scheme securely realizes the ideal functionality $\mathcal{F}_{\mathrm{SSE}}$ from Theorem 5.2.

7.1 More Efficient SSE Scheme

In this subsection, we show a more efficient SSE scheme than the modified SSE-2.

Similarly to (the modified) SSE-2, we assume that $\mathcal{W} = \{0,1\}^{\ell}$ for some $\ell = O(\log_2 \lambda)$, where λ is the security parameter. Hence $m = 2^{\ell}$. Let $\mathtt{SKE} = (G, E, E^{-1})$ be a symmetric-key encryption scheme.

(Store phase:) The client takes $(\mathcal{D}, \mathcal{W}, \mathtt{Index})$ as an input, where $\mathcal{W} = \{0,1\}^{\ell}$ and $\mathtt{Index} = (e_{i,j})$ is defined by eq.(1). Let $\pi_k = \pi(k, \cdot)$ be a pseudorandom permutation on $\mathcal{W} = \{0,1\}^{\ell}$, and

$$g \ : \ \{0,1\}^{\lambda} \times \{0,1\}^{\ell} \to \{0,1\}^{n}$$

be a psuedorandom function.

1. The client generates $k_e \leftarrow G(1^{\lambda})$ and computes $C_j = E_{k_e}(D_j)$ for $j = 1, \ldots, n$. Let $\mathcal{C} = (C_1, \ldots, C_n)$.
2. The client chooses k and k' randomly, where k is a key of π and k' is a key of g. The client sets

$$\mathcal{I}(\pi_k(w_i)) = (e_{i,1}, \ldots, e_{i,n}) \oplus g_{k'}(w_i) \tag{14}$$

for $i = 1, \ldots, m$, where $m = 2^{\ell}$, $w_i \in \mathcal{W}$ and $\oplus$ denotes the bitwise XOR.

The client stores $(\mathcal{C}, \mathcal{I})$ to the server.

(Search phase:) The client takes a keyword $w \in \mathcal{W}$ as an input.

1. The client sends

$$t(w) = (\pi_k(w), g_{k'}(w)) \tag{15}$$

to the server.

2. The server computes
$$(e_1, \ldots, e_n) = \mathcal{I}(\pi_k(w)) \oplus g_{k'}(w). \tag{16}$$
She then returns $\mathtt{C}(w) = \{C_j \mid e_j = 1\}$ to the client.
3. For each $C_j \in \mathtt{C}(w)$, the client computes $D_j = E^{-1}(C_j)$ and outputs $\mathtt{D}(w) = \{D_j \mid C_j \in \mathtt{C}(w)\}$.

Consider Example 3.1. Let $w_1 = 0$ and $w_2 = 1$. Then the client sets
$$\begin{aligned} \mathcal{I}(\pi_k(0)) &= (10101) \oplus g_{k'}(0) \\ \mathcal{I}(\pi_k(1)) &= (01010) \oplus g_{k'}(1). \end{aligned}$$
If the search keyword is $w_1 = 0$, then the client sends
$$t(0) = (\pi_k(0), g_{k'}(0))$$
to the server. The server computes
$$\mathcal{I}(\pi_k(0)) \oplus g_{k'}(0) = (10101)$$
and returns $\mathtt{C}(w_1) = \{C_1, C_3, C_5\}$ to the client.

The privacy will be proved by Corollary 7.1.

(Remark) Our SSE scheme can be viewed as a variant of $Scheme1$ of Chang and Mitzenmacher [9]. Let
$$\begin{aligned} F &: \{0,1\}^\lambda \times \{0,1\}^\ell \to \{0,1\}^\lambda \\ G &: \{0,1\}^\lambda \times \{1, \ldots, n\} \to \{0,1\} \end{aligned}$$
be two psuedorandom functions. Let $F_k = F(k, \cdot)$ and $G_k = G(k, \cdot)$. Then in $Scheme1$, eq.(14) is replaced with
$$\mathcal{I}(\pi_k(w_i)) = (e_{i,1}, \ldots, e_{i,n}) \oplus (G_{r_i}(1), \ldots, G_{r_i}(n))$$
where $r_i = F_{k'}(w_i)$. Eq.(15) is replaced with
$$t(w) = (p = \pi_k(w), r = F_{k'}(p))$$
and eq.(16) is replaced with
$$(e_1, \ldots, e_n) = \mathcal{I}(p) \oplus (G_r(1), \ldots, G_r(n)). \tag{17}$$
However, we cannot prove the privacy of $Scheme1$. The reason is as follows. In the store phase, the simulator Sim sends $\mathcal{I}(p)$ to the challenger. In the search phase, Sim is given $(e_1, \ldots, e_n)$ by the challenger. Then Sim must be able to compute r which satisfies eq.(17) because she must send $t(w) = (p, r)$ to the challenger. It is, however, impossible.

7.2 Extension to Verifiable SSE Scheme

In this subsection, we extend the above SSE scheme to a verifiable SSE scheme.

As before, we assume that $\mathcal{W} = \{0,1\}^\ell$ for some $\ell = O(\log_2 \lambda)$, where λ is the security parameter. Hence $m = 2^\ell$. Let $\texttt{SKE} = (G, E, E^{-1})$ be a symmetric-key encryption scheme.

(Store phase:) The client takes $(\mathcal{D}, \mathcal{W}, \texttt{Index})$ as an input, where $\mathcal{W} = \{0,1\}^\ell$. Let $\pi_k = \pi(k, \cdot)$ be a pseudorandom permutation on $\mathcal{W} = \{0,1\}^\ell$, and

$$\begin{aligned} g &: \{0,1\}^\lambda \times \{0,1\}^\ell \to \{0,1\}^n \\ \texttt{MAC} &: \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\lambda \end{aligned}$$

be two psuedorandom functions.

1. The client generates $k_e \leftarrow G(1^\lambda)$ and computes $C_j = E_{k_e}(D_j)$ for $j = 1, \ldots, n$. Let $\mathcal{C} = (C_1, \ldots, C_n)$.
2. The client chooses k, k' and k_m randomly, where k is a key of π, k' is a key of g and k_m is a key of $\texttt{MAC}$.
3. The client first computes

$$\begin{aligned} t(w_i) &= (\pi_k(w_i), g_{k'}(w_i)) \\ \texttt{C}(w_i) &= \{C_j \mid e_{i,j} = 1\} \\ Tag_i &= \texttt{MAC}_{k_m}((t(w_i), \texttt{C}(w_i))) \end{aligned}$$

   for $i = 1, \ldots, m$, where $m = 2^\ell$ and $w_i \in \mathcal{W}$.
4. He next sets

$$\mathcal{I}(\pi_k(w_i)) = ((e_{i,1}, \ldots, e_{i,n}) \oplus g_{k'}(w_i), Tag_i)$$

   for $i = 1, \ldots, m$.

The client stores $(\mathcal{C}, \mathcal{I})$ to the server.

(Search phase:) The client takes a keyword $w \in \mathcal{W}$ as an input.

1. The client sends
$$t(w) = (\pi_k(w), g_{k'}(w))$$
   to the server.

2. Let
$$\mathcal{I}(\pi_k(w)) = (X, Tag).$$
The server computes
$$(e_1, \ldots, e_n) = X \oplus g_{k'}(w).$$
She then returns $\mathtt{C}(w) = \{C_j \mid e_j = 1\}$ and Tag to the client.
3. The client receives $(\mathtt{C}'(w), Tag')$. He checks if
$$Tag' = \mathtt{MAC}_{k_m}((t(w), \mathtt{C}'(w))). \tag{18}$$
If eq.(18) does not hold, then he outputs `reject`. Otherwise he computes $D_j = E^{-1}(C_j)$ for each $C_j \in \mathtt{C}'(w)$, and outputs $\mathtt{D}(w) = \{D_j \mid C_j \in \mathtt{C}'(w)\}$.

7.3 Privacy of Our Verifiable SSE Scheme

Theorem 7.1 *The above verifiable SSE scheme satisfies privacy if* SKE *is LOR secure.*

(Proof) We construct a simulator Sim in the simulation game $\mathtt{Game}_{sim}$ of privacy as follows.

(Store phase:) 1. Sim is given m, n and $|D_1|, \cdots, |D_n|$ from the challenger.

2. Sim generates $k_e \overset{\$}{\leftarrow} G(1^\lambda)$ and computes $C'_j = E_{k_e}(0^{|D_j|})$ for $j = 1, \ldots, n$. Let $\mathcal{C}' = (C'_1, \ldots, C'_n)$.
3. Sim chooses
$$\begin{aligned} \overline{\mathtt{index}_i} &\overset{\$}{\leftarrow} \{0,1\}^n \\ Tag_i &\overset{\$}{\leftarrow} \{0,1\}^\lambda \end{aligned}$$
for $i = 1, \ldots, m$.
4. Sim sets
$$\mathcal{I}'(i) = (\overline{\mathtt{index}_i}, Tag_i)$$
for $i = 1, \ldots, m$.

Sim returns $(\mathcal{C}', \mathcal{I}')$ to the challenger.

(Search phase:) Sim sets $c_1 \leftarrow 0$ and $L \leftarrow \emptyset$. It chooses a key k of the pseudorandom permutation π randomly.

For $i = 1, \ldots, q$, do:

- Suppose that Sim is given $\texttt{List}(w)$ for some $w \in \mathcal{W}$ by the challenger. Then let $c_1 \leftarrow c_1 + 1$. For $j = 1, \ldots, n$, set

$$e_j = \begin{cases} 1 & if \quad j \in \texttt{List}(w) \\ 0 & if \quad j \notin \texttt{List}(w) \end{cases}$$

  Suppose that $\mathcal{I}'(\pi_k(c_1)) = (X, Tag^*)$. Then let

$$\begin{aligned} Y &\leftarrow (e_1, \ldots, e_n) \oplus X \\ t'(w) &\leftarrow (\pi_k(c_1), Y) \\ L &\leftarrow L \cup \{(c_1, t'(w), Tag^*)\} \end{aligned}$$

  Sim returns $(t'(w), Tag^*)$ to the challenger.

- Suppose that Sim is given $c \in \{1, 2, \ldots\}$ by the challenger. Then Sim finds $(c, t, Tag) \in L$, and returns (t, Tag) to the challenger.

We will prove that no distinguisher B can distinguish between $\texttt{Game}_{real}$ and $\texttt{Game}_{sim}$ by using a series of games $\texttt{Game}_0, \cdots, \texttt{Game}_4$, where $\texttt{Game}_0 = \texttt{Game}_{real}$. Let

$$p_i = \Pr(B \text{ outputs } b = 1 \text{ in } \texttt{Game}_i).$$

- $\texttt{Game}_1$ is the same as $\texttt{Game}_0$ except for that C_j is replaced with $C'_j = E_{k_e}(0^{|D_j|})$ for $j = 1, \ldots, n$. Then $|p_1 - p_0|$ is negligible because $\texttt{SKE}$ is LOR secure.

- $\texttt{Game}_2$ is the same as $\texttt{Game}_1$ except for that each π_k is replaced with a random permutation $\bar{\pi}$. Then $|p_2 - p_1|$ is negligible.

- $\texttt{Game}_3$ is the same as $\texttt{Game}_2$ except for that each $\overline{\texttt{index}}_i$ is replaced with a random string of length n. Then $|p_3 - p_2|$ is negligible because g is a pseudorandom function.

- $\texttt{Game}_4$ is the same as $\texttt{Game}_3$ except for that each Tag_i is replaced with a random string of length λ. Then $|p_4 - p_3|$ is negligible because $\texttt{MAC}$ is a pseudorandom function.

It is easy to see that

$$|p_4 - \Pr(b = 1 \text{ in } \texttt{Game}_{sim})|$$

is negligible. Therefore

$$\begin{aligned}
\mathsf{Adv}^{priv}_{\mathsf{Sim}}(B) &= |\Pr(b = 1 \text{ in } \texttt{Game}_{real}) - \Pr(b = 1 \text{ in } \texttt{Game}_{sim})| \\
&= |p_0 - \Pr(b = 1 \text{ in } \texttt{Game}_{sim})| \\
&\leq |p_0 - p_1| + |p_1 - p_2| + |p_2 - p_3| + |p_3 - p_4| \\
&\quad + |p_4 - \Pr(b = 1 \text{ in } \texttt{Game}_{sim})| \\
&= \textit{negligible}
\end{aligned}$$

for any distinguisher B. Therefore the verifiable SSE scheme satisfies privacy.

Q.E.D.

Corollary 7.1 *The SSE scheme of Sec.7.1 satisfies privacy.*

The proof is almost the same as that of Theorem 7.1.

7.4 Strong Reliability of Our Verifiable SSE Scheme

Theorem 7.2 *The above verifiable SSE scheme satisfies strong reliability.*

(Proof) Suppose that there exists an adversary (A_1, A_2) who strongly succeeds in the attack game of reliability with nonnegligible probability. We will construct a distinguisher B which can distinguish between $\texttt{MAC}$ and the random oracle $\mathsf{RO} : \{0,1\}^* \to \{0,1\}^\lambda$.

Let $\mathcal{O}$ denote the oracle $\texttt{MAC}_{k_m}(\cdot)$ or $\mathsf{RO}(\cdot)$. B runs (A_1, A_2) by playing the role of the client except for the following.

(Store phase) For $i = 1, \ldots, m$, B queries $(t(w_i), \texttt{C}(w_i))$ to $\mathcal{O}$, and receives Tag_i.

(Search phase) Suppose that B sends $t(w)$ to A_2 (as the client), and A_2 returns $(\texttt{C}'(w), Tag')$. B queries $(t(w), \texttt{C}'(w))$ to $\mathcal{O}$, and receives Tag^*. B then checks if

$$Tag' = Tag^* = \mathcal{O}(t(w), \texttt{C}'(w)) \tag{19}$$

instead of eq.(18).

Finally B outputs 1 if the adversary (A_1, A_2) strongly succeeds, and 0 otherwise. Recall that the adversary strongly succeeds if the client accepts $(\mathtt{C}'(\tilde{w}_i), \widetilde{Tag}_i')$ such that $(\mathtt{C}'(\tilde{w}_i), \widetilde{Tag}_i') \neq (\mathtt{C}(\tilde{w}_i), \widetilde{Tag}_i)$ for some $\tilde{w}_i$, where

$$(\mathtt{C}(\tilde{w}_i), \widetilde{Tag}_i) \leftarrow \mathtt{Search}(\mathcal{I}, \mathcal{C}, t(\tilde{w}_i)).$$

Suppose that $\mathcal{O}(\cdot) = \mathtt{MAC}_{k_m}(\cdot)$. Then

$$\begin{aligned} p_0 &= \Pr(B \text{ outputs } 1) \\ &= \Pr((A_1, A_2) \text{ strongly succeeds}) \\ &= \textit{nonnegligible} \end{aligned}$$

from our assumption.

On the other hand, suppose that $\mathcal{O} = \mathsf{RO}$. The client never accepts $(\mathtt{C}'(\tilde{w}_i), \widetilde{Tag}_i')$ such that $\mathtt{C}'(\tilde{w}_i) = \mathtt{C}(\tilde{w}_i)$ and $\widetilde{Tag}_i' \neq \widetilde{Tag}_i$. Therefore if the adversary (A_1, A_2) strongly succeeds, then the client accepts $(\mathtt{C}'(\tilde{w}_i), \widetilde{Tag}_i')$ such that $\mathtt{C}'(\tilde{w}_i) \neq \mathtt{C}(\tilde{w}_i)$ for some $\tilde{w}_i$.

Note that B never queries $(t(w), \mathtt{C}'(w))$ such that $\mathtt{C}'(w) \neq \mathtt{C}(w)$ to $\mathcal{O}$ in the store phase. Therefore

$$\Pr(eq.(19) \text{ holds}) = 1/2^\lambda$$

for $(t(w), \mathtt{C}'(w))$ such that $\mathtt{C}'(w) \neq \mathtt{C}(w)$. Hence

$$\begin{aligned} p_1 &= \Pr(B \text{ outputs } 1) \\ &= \Pr((A_1, A_2) \text{ strongly succeeds}) \\ &\leq q/2^\lambda, \end{aligned}$$

where q is the number of keywords which A_1 sends to the client. Since q is bounded by some polynomial in λ, $q/2^\lambda$ is negligible.

Consequently $|p_0 - p_1|$ is nonnegligible. This is against that $\mathtt{MAC}$ is a pseudorandom function. Q.E.D.

7.5 UC Security

Corollary 7.2 *Our verifiable SSE scheme securely realizes the ideal functionality $\mathcal{F}_{\text{SSE}}$.*

(Proof) This corollary is obtained from Theorem 5.2, Theorem 7.1 and Theorem 7.2.

Q.E.D.

Table 1: Comparison of Efficiency

scheme	privacy	reliability	UC	$\lvert\mathcal{I}\rvert$	$\lvert t(w)\rvert$
Modified SSE-2	○	×	×	$2mn$	$n(\log_2 m + \log_2 n + 1)$
Sec.7.1	○	×	×	mn	$n + \log_2 m$
Sec.7.2	○	○	○	$m(n + \lambda)$	$n + \log_2 m$

8 Comparison

We show a comparison of the modified SSE-2, our SSE scheme of Sec.7.1 and our verifiable SSE scheme of Sec.7.2 in Table 1. As can be seen, our SSE scheme has shorter $|\mathcal{I}|$ and $|t(w)|$ than the modified SSE-2, and our verifiable SSE scheme has slightly larger $|\mathcal{I}|$ and $|t(w)|$ than our SSE scheme.

9 Summary

In this paper, we first extended the model of SSE schemes to that of *verifiable* SSE schemes, and define the (strong) reliability. We next formulated the UC security of verifiable SSE schemes, and proved its weak equivalence with both privacy and reliability. Finally we showed an efficient verifiable SSE scheme, and proved that it is UC-secure. We also pointed out a flaw of SSE-2, and showed how to fix the flaw.

In our verifiable SSE scheme, the communication overhead of the search phase is $n+\log_2 m = n+O(\log \lambda)$, where n is the number of stored encrypted files and λ is the security parameter. It will be an open problem to construct a UC-secure scheme such that the communication overhead of the search phase is sublinear in n in the standard model.

References

[1] S.Bellovin and W.Cheswick: Privacy-Enhanced Searches Using Encrypted Bloom Filters, Cryptology ePrint Archive, Report 2006/210, http://eprint.iacr.org/ (2006)

[2] M. Bellare, A. Desai, E. Jokipii, P. Rogaway: A Concrete Security Treatment of Symmetric Encryption. FOCS 1997: pp.394–403 (1997)

[3] Lucas Ballard, Seny Kamara, Fabian Monrose: Achieving Efficient Conjunctive Keyword Searches over Encrypted Data. ICICS 2005, pp.414–426 (2005)

[4] J. W. Byun, D. H. Lee, and J. Lim: Efficient conjunctive keyword search on encrypted data storage system. EuroPKI, pp.184–196 (2006)

[5] Ran Canetti, Universally Composable Security: A New Paradigm for Cryptographic Protocols, Cryptology ePrint Archive, Report 2000/067 http://eprint.iacr.org/ (2005)

[6] 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)

[7] Full version of the above: Cryptology ePrint Archive, Report 2006/210, http://eprint.iacr.org/ (2006)

[8] David Cash, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel Rosu, Michael Steiner: Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries. CRYPTO 2013.

[9] Y.Chang and M.Mitzenmacher: Privacy Preserving Keyword Searches on Remote Encrypted Data. ACNS 2005: pp.442–455 (2005)

[10] David Cash, Stefano Tessaro: The Locality of Searchable Symmetric Encryption. EUROCRYPT 2014: pp.351-368

[11] Eu-Jin Goh: Secure Indexes. Cryptology ePrint Archive, Report 2003/216, http://eprint.iacr.org/ (2003)

[12] Philippe Golle, Jessica Staddon, Brent R. Waters: Secure Conjunctive Keyword Search over Encrypted Data. ACNS 2004, pp.31–45 (2004)

[13] Kaoru Kurosawa: Garbled Searchable Symmetric Encryption. Financial Cryptography 2014: pp.234–251

[14] Kaoru Kurosawa, Yasuhiro Ohtaki: UC-Secure Searchable Symmetric Encryption. Financial Cryptography 2012: pp.285–298

[15] Seny Kamara, Charalampos Papamanthou, Tom Roeder: Dynamic searchable symmetric encryption. ACM Conference on Computer and Communications Security 2012: pp.965–976

[16] Seny Kamara and Charalampos Papamanthou: Parallel and Dynamic Searchable Symmetric Encryption. FC 2013

[17] Kaoru Kurosawa, Yasuhiro Ohtaki: How to Update Documents Verifiably in Searchable Symmetric Encryption. CANS 2013: pp.309–328

[18] D.Song, D.Wagner, A.Perrig: Practical Techniques for Searches on Encrypted Data. IEEE Symposium on Security and Privacy 2000: pp.44–55 (2000)

[19] Peishun Wang, Huaxiong Wang, Josef Pieprzyk: Keyword Field-Free Conjunctive Keyword Searches on Encrypted Data and Extension for Dynamic Groups. CANS 2008: pp.178–195