UC-Secure Searchable Symmetric Encryption

Kaoru Kurosawa and Yasuhiro Ohtaki

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

Abstract. For searchable symmetric encryption schemes (or symmetric-key encryption with keyword search), the security against passive adversaries (i.e. privacy) has been mainly considered so far. In this paper, we first define its security against active adversaries (i.e. reliability as well as privacy). We next formulate its UC-security. We then prove that the UC-security against non-adaptive adversaries is equivalent to our definition of privacy and reliability. We further present an efficient construction which satisfies our security definition (hence UC-security).

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

1 Introduction

We consider the following problem [8]: a client wants to store his files (or documents) in an encrypted form on a remote file server (in the store phase). Later (in the search phase), the client wants to efficiently retrieve some of the encrypted files containing (or indexed by) specific keywords, keeping the keywords themselves secret and not jeopardizing the security of the remotely stored files. For example, a client may want to store old email messages encrypted on a server managed by Google or another large vendor, and later retrieve certain messages while traveling with a mobile device. Such a scheme is called a searchable symmetric encryption (SSE) scheme because symmetric key encryption schemes are used.

For this problem, the security against passive adversaries (i.e. privacy) has been mainly considered so far. After a series of works [10, 9, 1, 8], Curtmola, Garay, Kamara and Ostrovsky [6, 7] showed a rigorous definition of security about the client's privacy against a passive server, and an efficient scheme which satisfies their definition.

However, an active adversary (i.e. a server) may forge the encrypted files and/or delete some of them. Even if the clients uses MAC to authenticate the encrypted files, a malicious server may replace $(C_i, \mathtt{MAC}(C_i))$ with some $(C_j, \mathtt{MAC}(C_j))$ in the search phase, where C_i is an encrypted file which should be returned. Then the client cannot detect cheating.

In this paper, we first formulate the security of *verifiable* SSE schemes against active adversaries by using the notion of privacy and reliability. Our definition of privacy is slightly stronger than "adaptive semantic security" of Curtmola et al.

[7, Definition 4.11]. Our definition of reliability means that even if the server is malicious, the client can receive the corresponding files correctly, or he outputs *fail* in the search phase.

We next formulate its UC-security, where UC means universal composability. (In the UC framework [3–5], the security of a protocol $\Sigma = (P_1, \cdots, P_n)$ is maintained under a general protocol composition if Σ is UC-secure.) We then prove that the UC-security against non-adaptive adversaries is equivalent to our definition of privacy and reliability.

We further present an efficient scheme which satisfies our definition (hence UC-security). The communication overhead of our search phase is proportional to N, where N is the number of stored files. (It is independent of the size of each file.) 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.

2 Verifiable Searchable Symmetric Encryption (SSE)

In this section, we define *verifiable* searchable symmetric encryption (verifiable SSE) scheme and its security.

- Let $\mathcal{D} = \{D_1, \cdots, D_N\}$ be a set of documents (or files).
- Let $\mathcal{W} = \{0,1\}^\ell$ be a set of keywords.
  (Hence ℓ denotes the bit length of each keyword.)
- Let $\mathtt{D}(w)$ denote the set of documents which contain a keyword $w \in \mathcal{W}$.

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. PPT means probabilistic polynomial time.

2.1 Verifiable SSE

A verifiable SSE scheme consists of six polynomial time algorithms

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

such that

- $K \leftarrow \mathtt{Gen}(1^k)$: is a probabilistic algorithm which generates a key K, where k is a security parameter.
- $(\mathcal{I}, \mathcal{C}) \leftarrow \mathtt{Enc}(K, \mathcal{D}, \mathcal{W})$: is a probabilistic encryption algorithm which outputs an encrypted index $\mathcal{I}$ and $\mathcal{C} = \{C_1, \cdots, C_N\}$, where C_i is a ciphertext of D_i.
- $t(w) \leftarrow \mathtt{Trpdr}(K, w)$: is a deterministic algorithm which outputs a trapdoor $t(w)$ for a keyword w.
- $(\mathtt{C}(w), Tag) \leftarrow \mathtt{Search}(\mathcal{I}, \mathcal{C}, t(w))$: is a deterministic search algorithm, where

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

- $\mathtt{accept/reject} \leftarrow \mathtt{Verify}(K, t(w), \tilde{\mathtt{C}}(w), Tag)$: is a deterministic verification algorithm which checks the validity of $\tilde{\mathtt{C}}(w)$.

– $D \leftarrow \mathtt{Dec}(K, C)$: is a deterministic decryption algorithm, where D is a document and C is a string.

For the set of documents $\mathcal{D} = \{D_1, \cdots, D_N\}$ and a keyword $w \in \mathcal{W}$, it must be that

– $D_i = \mathtt{Dec}(K, C_i)$ if C_i is a ciphertext of D_i.
– $\mathtt{Verify}(K, t(w), \mathtt{C}(w), Tag) = \mathtt{accept}$ if $(\mathcal{I}, \mathcal{C})$ is output by $\mathtt{Enc}(K, \mathcal{D}, \mathcal{W})$, $t(w)$ is output by $\mathtt{Trpdr}(K, w)$ for $w \in \mathcal{W}$, and $(\mathtt{C}(w), Tag)$ is output by $\mathtt{Search}(\mathcal{I}, \mathcal{C}, t(w))$.

The definition of usual searchable symmetric encryption (SSE) schemes [6, 7] is obtained by deleting Tag and $\mathtt{Verify}$ from the verifiable SSE schemes.

We next translate a $\mathtt{vSSE}$ into a protocol Σ_{vsse} which is a protocol between a client and a server. It consists of the store phase and the search phase as shown below, where the store phase is executed once, and the search phase is executed for polynomially many times.

Store phase:

1. The client generates a key $K \leftarrow \mathtt{Gen}(1^k)$ and keeps it secret.
2. On input $(\mathcal{D}, \mathcal{W})$, the client computes $(\mathcal{I}, \mathcal{C}) \leftarrow \mathtt{Enc}(K, \mathcal{D}, \mathcal{W})$ and store them to the server,
   where $\mathcal{D}$ is a set of documents, $\mathcal{W}$ is the set of keywords, $\mathcal{I}$ is an encrypted index and $\mathcal{C}$ is a ciphertext of $\mathcal{D}$.

Search phase:

1. On input a keyword $w \in \mathcal{W}$, the client computes a trapdoor $t(w) \leftarrow \mathtt{Trpdr}(K, w)$ and sends it to the server.
2. The server computes $(\mathtt{C}(w), Tag) \leftarrow \mathtt{Search}(\mathcal{I}, \mathcal{C}, t(w))$ and sends them to the client (where $\mathtt{C}(w)$ is defined by eq.(1)).
3. If the client received $(\tilde{\mathtt{C}}(w), Tag)$ from the server,
   then the client computes $\mathtt{Verify}(K, t(w), \tilde{\mathtt{C}}(w), Tag)$.
   – If the result is $\mathtt{accept}$, then the client decrypts each ciphertext C_i in $\mathtt{C}(w)$ to the document D_i by using $\mathtt{Dec}(K, \cdot)$, and outputs the set of such D_i as $\mathtt{D}(w)$.
   – Otherwise the client outputs *fail*.

2.2 Privacy

In the above protocol, the server learns $|D_i|$ from C_i for each i, and ℓ from $\mathcal{I}$ in the store phase. In the search phase, she also knows

$$\mathtt{List}(w) = \{i \mid D_i \text{ contains } w\}$$

for each queried keyword w. We require that the server should not be able to learn any more information.

Based on the work of Curtmola, Garay, Kamara and Ostrovsky [6, 7], this security notion is formulated as follows. We consider a real game $\mathtt{Game}_{real}$ and a simulation game $\mathtt{Game}_{sim}$ as shown below, where $\mathtt{Game}_{real}$ is played by a challenger and an adversary $\mathbf{A}$, and $\mathtt{Game}_{sim}$ is played by a simulator **Sim** as well.

Real Game ($\mathtt{Game}_{real}$)

- Adversary $\mathbf{A}$ chooses $(\mathcal{D}, \mathcal{W})$ and sends them to the challenger.
- The challenger generates $K \leftarrow \mathtt{Gen}(1^k)$,
  and then sends $(\mathcal{I}, \mathcal{C}) \leftarrow \mathtt{Enc}(K, \mathcal{D}, \mathcal{W})$ to $\mathbf{A}$.
- For $i = 1, \cdots, q$, do:
  1. $\mathbf{A}$ chooses a keyword $w_i \in \mathcal{W}$ and sends it to the challenger.
  2. The challenger sends a trapdoor $t(w_i) \leftarrow \mathtt{Trpdr}_K(w_i)$ to $\mathbf{A}$.
- $\mathbf{A}$ outputs a bit b.

Simulation Game ($\mathtt{Game}_{sim}$)

- $\mathbf{A}$ chooses $(\mathcal{D}, \mathcal{W})$ and sends them to the challenger.
- The challenger sends $|D_1|, \cdots, |D_N|$ and ℓ to simulator **Sim**,
  where $\mathtt{D} = \{D_1, \cdots, D_N\}$ and ℓ is the length of a keyword.
- **Sim** computes $(\mathcal{I}', \mathtt{C}')$ from $|D_1|, \cdots, |D_N|$ and ℓ,
  and sends them to the challenger.
- The challenger relays $(\mathcal{I}', \mathtt{C}')$ to $\mathbf{A}$.
- For $i = 1, \cdots, q$, do:
  1. $\mathbf{A}$ chooses $w_i \in \mathcal{W}$ and sends it to the challenger.
  2. The challenger sends $\mathtt{List}(w_i)$ to **Sim**.
  3. **Sim** computes $t(w_i)'$ from $\mathtt{List}(w_i)$ and sends it to the challenger.
  4. The challenger relays $t(w_i)'$ to $\mathbf{A}$.
- $\mathbf{A}$ outputs a bit b.

Definition 1. *We say that a verifiable SSE satisfies privacy if there exists a PPT simulator* **Sim** *such that*

$$|\Pr(\mathbf{A} \text{ outputs } b = 1 \text{ in } \mathtt{Game}_{real}) - \Pr(\mathbf{A} \text{ outputs } b = 1 \text{ in } \mathtt{Game}_{sim})| \quad (2)$$

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

"Adaptive semantic security" of Curtmola et al. [7, Definition 4.11] requires that for any PPT adversary $\mathbf{A}$, there exists a PPT **Sim** such that eq.(2) is negligible. On the other hand, our definition requires that there exists a PPT **Sim** such that for any PPT adversary $\mathbf{A}$, eq.(2) is negligible. Hence our definition is slightly stronger. This small change is important when we prove the relationship with UC-security. (See Remark 1 in the proof of Theorem 2.)

2.3 Reliability

In addition to the privacy, the server (an adversary $\mathbf{A}$) should not be able to forge $(\mathtt{C}(w), Tag)$ in the search phase. We formulate this security notion as follows.

Fix $(\mathcal{D}, \mathcal{W})$ and search queries $w_1, \cdots, w_q \in \mathcal{W}$ arbitrarily. In the store phase, suppose that the client generated K and then computed $(\mathcal{I}, \mathtt{C})$.

- We say that $\mathtt{C}(w)^*$ is invalid for $t(w)$ if $\mathtt{C}(w)^* \neq \mathtt{C}(w)$, where $(\mathtt{C}(w), Tag) \leftarrow \mathtt{Search}(\mathcal{I}, \mathcal{C}, t(w))$.
- We say that $\mathbf{A}$ wins if she can return $(\mathtt{C}(w_i)^*, Tag^*)$ for some query $t(w_i)$ such that $\mathtt{C}(w_i)^*$ is invalid for $t(w_i)$ and $\mathtt{Verify}(K, t(w_i), \mathtt{C}(w_i)^*, Tag) = \mathtt{accept}$.

Definition 2. *We say that a verifiable SSE satisfies reliability if for any PPT adversary* $\mathbf{A}$*,* $\Pr(\mathbf{A}$ *wins*$)$ *is negligible for any* $(\mathcal{D}, \mathcal{W})$ *and any search queries* $w_1, \cdots, w_q$*.*

3 UC-Secure SSE

3.1 UC Security

The security of a protocol $\Sigma = (P_1, \cdots, P_n)$ is maintained under a general protocol composition if Σ is secure in the universally composable (UC) security framework [3–5].

In this framework, there exists an environment $\mathcal{Z}$ which generates the input to all parties, reads all outputs, and in addition interacts with an adversary $\mathbf{A}$ in an arbitrary way throughout the computation.

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

The following universal composition theorem is proven in [3, 4]. Consider a protocol Σ that operates in a hybrid model of computation where parties can communicate as usual, and in addition have ideal access to (an unbounded number of copies of) some ideal functionality $\mathcal{F}$. Let ρ be a protocol that securely realizes $\mathcal{F}$ as sketched above, and let Σ^ρ be the composed protocol. That is, Σ^ρ is identical to Σ with the exception that each interaction with some copy of $\mathcal{F}$ is replaced with a call to (or an invocation of) an appropriate instance of ρ. Similarly, ρ-outputs are treated as values provided by the appropriate copy of $\mathcal{F}$. Then Σ and Σ^ρ have essentially the same input/output behavior. In particular, if Σ securely realizes some ideal functionality $\mathcal{G}$ given ideal access to $\mathcal{F}$, then Σ^ρ securely realizes $\mathcal{G}$ from scratch.

For more details, see [3, 4].

3.2 Ideal Functionality of Verifiable SSE

We define the ideal functionality $\mathcal{F}_{vSSE}$ of verifiable SSE protocols as follows.

Ideal Functionality $\mathcal{F}_{vSSE}$

Running with a dummy client P_1, a dummy server P_2 and an ideal world adversary **S**.

Store: Upon receiving input $(\textbf{store}, sid, D_1, \cdots, D_N, \mathcal{W})$ from P_1, verify that this is the first input from P_1 with $(\textbf{store}, sid)$.
If so, store $D_1, \cdots, D_N$, and send $|D_1|, \cdots, |D_N|$ and ℓ to **S**.
Otherwise ignore this input.

Search: Upon receiving $(\textbf{search}, sid, w)$ from P_1, send $\texttt{List}(w)$ to **S**.
1. If **S** returns OK, then send $\texttt{D}(w)$ to P_1.
2. If **S** returns $\perp$, then send $\perp$ to P_1.

Our $\mathcal{F}_{vSSE}$ provides an ideal world because

- The dummy client receives $\texttt{D}(w)$ correctly or $\perp$.
- The ideal world adversary **S** learns only $|D_1|, \cdots, |D_N|$ and ℓ in the store phase, and only $\texttt{List}(w)$ in the search phase.

(See the first paragraph of Sec.2.2.)

We say that a verifiable SSE protocol Σ_{vSSE} is UC-secure if it securely realizes the ideal functionality $\mathcal{F}_{vSSE}$.

4 Equivalence

In this section, we prove that the UC-security notion of SSE is equivalent to the definitions of privacy and reliability presented in Sec.2. In the UC framework, a non-adaptive adversary corrupts some parties at the beginning of the protocol execution.

Theorem 1. *A verifiable SSE scheme* vSSE *satisfies privacy and reliability if the corresponding protocol Σ_{vsse} is UC-secure against non-adaptive adversaries.*

(Proof) Assume that vSSE does not satisfy (one of) privacy or reliability. We show that Σ_{vsse} does not securely realize $\mathcal{F}_{vSSE}$.

This is done by constructing an environment $\mathcal{Z}$ and an adversary **A** such that for any ideal world adversary **S**, $\mathcal{Z}$ can tell whether it is interacting with **A** in Σ_{vsse}, or with **S** in the ideal world which interacts with $\mathcal{F}_{vSSE}$.

(I) Assume that vSSE does not satisfy the privacy property defined by Def.1. That is, for any simulator **Sim**, there exists an adversary **B** such that eq.(2) is non-negligible.

$\mathcal{Z}$ asks **A** or **S** to corrupt P_2 (server) so that P_2 relays each message which he received from P_1 (client) to $\mathcal{Z}$ (in the real world). Except for this, P_2 behaves honestly. $\mathcal{Z}$ then internally runs **B** as follows.

- If **B** sends $(\mathcal{D}, \mathcal{W})$ to the challenger, then
  1. $\mathcal{Z}$ activates P_1 (client) with input $(\textbf{store}, sid, \mathcal{D}, \mathcal{W})$.

2. In the real world,
   P_1 sends $(\mathcal{I}, \mathtt{C})$ to $P_2(= \mathbf{A})$, and $P_2(= \mathbf{A})$ relays it to $\mathcal{Z}$.
   In the ideal world,
   P_1 sends $(\mathbf{store}, sid, \mathcal{D}, \mathcal{W})$ to $\mathcal{F}_{vSSE}$.
   $\mathcal{F}_{vSSE}$ sends $|D_1|, \cdots, |D_N|$ and ℓ to $\mathbf{S}(= P_2)$.
   $\mathbf{S}(= P_2)$ computes $(\mathcal{I}', \mathtt{C}')$, and sends it to $\mathcal{Z}$.
3. $\mathcal{Z}$ sends $(\mathcal{I}, \mathtt{C})$ or $(\mathcal{I}', \mathtt{C}')$ to $\mathbf{B}$.

– If $\mathbf{B}$ sends w_i to the challenger, then
1. $\mathcal{Z}$ activates P_1 with input $(\mathbf{search}, sid, w_i)$.
2. In the real world,
   P_1 sends $t(w_i)$ to P_2, and P_2 relays it to $\mathcal{Z}$.
   In the ideal world,
   P_1 sends $(\mathbf{search}, sid, w_i)$ to $\mathcal{F}_{vSSE}$.
   $\mathcal{F}_{vSSE}$ sends $\mathtt{List}(w_i)$ to $\mathbf{S}(= P_2)$.
   $\mathbf{S}(= P_2)$ computes $t(w_i)'$, and sends it to $\mathcal{Z}$.
3. $\mathcal{Z}$ sends $t(w_i)$ or $t(w_i)'$ to $\mathbf{B}$.

Finally $\mathcal{Z}$ outputs 1 if and only if $\mathbf{B}$ outputs 1.

If $\mathcal{Z}$ interacts with Σ_{vsse} (i.e. the real world), then it is easy to see that $\mathtt{Game}_{real}$ is simulated for $\mathbf{B}$. On the other hand, suppose that $\mathcal{Z}$ interacts with $\mathbf{S}$ in the ideal world. Then $\mathtt{Game}_{sim}$ is simulated for $\mathbf{B}$ because the ideal functionality $\mathcal{F}_{vSSE}$ plays the role of the challenger and the ideal world adversary $\mathbf{S}$ plays the role of **Sim**.

Now from our assumption, for any ideal world adversary $\mathbf{S}$, there exists some $\mathbf{B}$ which can distinguish $\mathtt{Game}_{real}$ from $\mathtt{Game}_{sim}$. This means that for any ideal world adversary $\mathbf{S}$, there exists some $\mathcal{Z}$ which can distinguish Σ_{vsse} (the real world) from the ideal world.

(II) Assume that vSSE does not satisfy reliability, i.e. there exists an adversary $\mathbf{B}$ which breaks the reliability defined by Def.2. $\mathcal{Z}$ asks $\mathbf{A}$ to corrupt P_2 (server) so that P_2 behaves in the same way as $\mathbf{B}$. $\mathcal{Z}$ finally outputs 1 if and only if $\mathcal{Z}$ receives some set of documents $\mathcal{D}'$ from P_1 such that $\mathcal{D}' \neq \mathtt{D}(w)$ for some w.

If $\mathcal{Z}$ interacts with Σ_{vsse}, then $\mathbf{B}$ wins with non-negligible probability from our assumption. Hence $\mathcal{Z}$ outputs 1 with non-negligible probability. On the other hand, if $\mathcal{Z}$ interacts with $\mathbf{S}$ in the ideal world, $\mathcal{Z}$ never receives such $\mathcal{D}'$ from P_1. Hence $\mathcal{Z}$ never outputs 1. This means that $\mathcal{Z}$ can distinguish Σ_{vsse} from the ideal world for any ideal world adversary $\mathbf{S}$.

Q.E.D.

Theorem 2. *Σ_{vSSE} is UC-secure against non-adaptive adversaries if the underlying* vSSE *satisfies privacy and reliability.*

(Proof) Assume that Σ_{vSSE} does not securely realize $\mathcal{F}_{vSSE}$ against non-adaptive adversaries. That is, there exists some $\mathcal{Z}$ who can distinguish between the real world and the ideal world.

We show that vSSE does not satisfy (one of) privacy or reliability. Assume that vSSE satisfies privacy. (Otherwise the theorem is proven). Then there exists a simulator **Sim** which satisfies Def.1.

Suppose that the real world adversary **A** does not corrupt any party. Then it is easy to see that no $\mathcal{Z}$ can distinguish the real world from the ideal world. (Note that $\mathcal{Z}$ interacts only with P_1.)

Suppose that $\mathcal{Z}$ asks **A** to corrupt P_1 (client). Note that **A** can report the communication pattern of P_1 to $\mathcal{Z}$. Consider an ideal world adversary **S** who runs **A** internally by playing the role of P_2. Note that **S** can play the role of P_2 faithfully because P_2 has no interaction with $\mathcal{Z}$ and $\mathcal{F}_{vSSE}$. Hence it is easy to see that no $\mathcal{Z}$ can distinguish the real world from the ideal world in this case, too.

Suppose that $\mathcal{Z}$ asks **A** to corrupt P_2 (server), but P_2 can not break the *reliability* at all. That is, $\Pr(P_2 \text{ wins}) = 0$ in Def.2. **A** may report the communication pattern of P_2 to $\mathcal{Z}$. Then our ideal world adversary **S** behaves in the same way as the above mentioned **Sim**, where the ideal functionality $\mathcal{F}_{vSSE}$ plays the role of the challenger. In this case, no $\mathcal{Z}$ can distinguish between the real world and the ideal world from the definition of *privacy*.

Remark 1. Def.1 says that there exists a **Sim** such that for any interactive distinguisher ($\mathcal{Z}$ in the above case), eq.(2) is negligible. This is the point where the privacy definition of of Curtmola et al. [7, Definition 4.11] does not work.

Suppose that $\mathcal{Z}$ asks **A** to corrupt P_2 (server), and P_2 breaks the *reliability* with negligible probability. That is, $\Pr(P_2 \text{ wins})$ is negligible in Def.2. Then similarly to the above, no $\mathcal{Z}$ can distinguish between the real world and the ideal world.

Therefore it must be that $\mathcal{Z}$ asks **A** to corrupt P_2 (server), and P_2 breaks *reliability* with non-negligible probability. That is, $\Pr(P_2 \text{ wins})$ is non-negligible in Def.2. (Otherwise no $\mathcal{Z}$ can distinguish between the real world and the ideal world.) This means that `vSSE` does not satisfy *reliability*.

Q.E.D.

5 Construction

In this section, we construct an efficient *verifiable* SSE scheme which satisfies Def.1 and Def.2. Our scheme is based on SSE-2 of Curtmola et al. [6, 7]. (Note that SSE-2 is not verifiable).

5.1 Overview

We first illustrate SSE-2 of Curtmola et al. [6, 7] by using an example.

(Store phase:) The client constructs an array $\mathcal{I}$ as follows. Let π_K be a pseudo-random permutation, where K is the secret key of the client. Suppose that $\mathtt{D}(Austin) = (D_3, D_6, D_{10})$. That is, D_3, D_6 and D_{10} contains a keyword $Austin$. First the client computes

$$\mathtt{addr}_{Austin,i} = \pi_K(Austin, i)$$

for $i = 1, \cdots, N$. Next let

$$\mathcal{I}(\mathtt{addr}_{Austin,1}) = 3,\ \mathcal{I}(\mathtt{addr}_{Austin,2}) = 6,\ \mathcal{I}(\mathtt{addr}_{Austin,3}) = 10 \tag{3}$$

and

$$\mathcal{I}(\mathtt{addr}_{Austin,i}) = \mathtt{dummy} \tag{4}$$

for $i = 4, \cdots, N$. Do the same thing for all the other keywords. Finally the client stores $\mathcal{I}$ and $\mathcal{C} = \{C_1, \cdots, C_N\}$ to the server, where C_i is a ciphertext of D_i.

(Search phase:) Suppose that the client wants to retrieve the documents which contain *Austin*. Then the client sends

$$t(Austin) = (\mathtt{addr}_{Austin,1}, \cdots, \mathtt{addr}_{Austin,N})$$

to the server. From eq.(3) and eq.(4), the server sees that $\mathtt{List}(Austin) = \{3, 6, 10\}$. The server then returns (C_3, C_6, C_{10}) to the client. The client finally decrypts them to obtain (D_3, D_6, D_{10}).

The above scheme satisfies privacy, but not reliability. To achieve reliability, a naive approach is to replace C_i with $\hat{C}_i = (C_i, \mathtt{MAC}(C_i))$. The client stores the set of such $\hat{C}_i$ to the server. For a query $t(Austin)$, an (honest) server returns $(\hat{C}_3, \hat{C}_6, \hat{C}_{10})$ to the client. However, a malicious server would return $(\hat{C}_3, \hat{C}_6, \hat{C}_{11})$ or just $(\hat{C}_3, \hat{C}_6)$. Then the client cannot detect any cheating.

To overcome this problem, we construct $\mathcal{I}$ as follows.

$$\begin{aligned}
\mathcal{I}(\mathtt{addr}_{Austin,1}) &= (3, tag_1 = \mathtt{MAC}(\mathtt{addr}_{Austin,1}, C_3)) \\
\mathcal{I}(\mathtt{addr}_{Austin,2}) &= (6, tag_2 = \mathtt{MAC}(\mathtt{addr}_{Austin,2}, C_6)) \\
\mathcal{I}(\mathtt{addr}_{Austin,3}) &= (10, tag_3 = \mathtt{MAC}(\mathtt{addr}_{Austin,3}, C_{10}))
\end{aligned}$$

and

$$\mathcal{I}(\mathtt{addr}_{Austin,i}) = (\mathtt{dummy}, tag_i = \mathtt{MAC}(\mathtt{addr}_{Austin,i}, \mathtt{dummy}))$$

for $i = 4, \cdots, N$. For a query $t(Austin)$, the server returns (C_3, C_6, C_{10}) and $(tag_1, \cdots, tag_N)$ to the client.

The client checks the validity of each tag_i. This approach works because MAC binds the (query, answer) pair.

Another subtle point is that the index of each D_i should appear in $\mathcal{I}$ the same number of times, say max times. (Otherwise the simulator **Sim** in the definition of privacy cannot construct $\mathcal{I}'$ which is indistinguishable from $\mathcal{I}$. Remember that **Sim** must be able to construct $\mathcal{I}'$ only from $|D_1|, \cdots, |D_N|$ and ℓ.)

For this problem, Curtmola et al. described the following method in SSE-2 [7, Fig.2].

For each index i:

- let c be the number of entries in $\mathcal{I}$ that already contain i.
- for $1 \leq \ell \leq \mathtt{max} - c$, set $\mathcal{I}[\pi_K(0^\ell, n + \ell)] = i$.

The last line is strange because ℓ is used in two different meanings. (In [7], ℓ is also defined as the bit length of each keyword.) Hence it must be that

– for $1 \leq k \leq \mathtt{max} - c$, set $\mathcal{I}[\pi_K(0^\ell, n+k)] = i$.

Even so, the above line does not work as shown below. For simplicity, suppose that $c = \mathtt{max} - 1$ for $i = 1, \cdots, N$. Then we have $\mathcal{I}[\pi_K(0^\ell, n+1)] = N$ at the end because the entry of $\mathcal{I}[\pi_K(0^\ell, n+1)]$ is overwritten for $i = 1, \cdots, N$. This means that in $\mathcal{I}$, only N appears $\mathtt{max}$ times and the other each i appears $\mathtt{max} - 1$ times.

We will show how to fix this flaw, too.

5.2 Proposed Verifiable SSE

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 a decryption algorithm.

Remember that the set of documents is $\mathcal{D} = \{D_1, \cdots, D_N\}$, and the set of keywords is $\mathcal{W} = \{0,1\}^\ell$. Let $\pi : \{0,1\}^k \times \{0,1\}^{\ell+1+\log N} \to \{0,1\}^{\ell+1+\log N}$ be a pseudo-random permutation. For simplicity, we will write $y = \pi(x)$ instead of $y = \pi(K, x)$, where K is a key.

Let $MAC : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$ be a MAC (a tag generation algorithm). For simplicity, we write $tag = \mathtt{MAC}(m)$ instead of $tag = \mathtt{MAC}(K, m)$, where K is a key and m is a message.

Now the proposed verifiable SSE scheme is as follows.

$\mathtt{Gen}(1^k)$**:** Run G to generate a key K_0 of $\mathtt{SKE}$. Choose a key $K_1 \in \{0,1\}^k$ of π and a key $K_2 \in \{0,1\}^k$ of $\mathtt{MAC}$ randomly. Let $K = (K_0, K_1, K_2)$.

$\mathtt{Enc}(K, \mathcal{D}, \mathcal{W})$**:** First compute $C_i = E(K_0, D_i)$ for each $D_i \in \mathcal{D}$ and let $\mathcal{C} = \{C_1, \cdots, C_N\}$. Next let $\mathcal{I}$ be an array of size $2 \times 2^\ell N$ as follows.

1. First let
$$\mathcal{I}(i) \leftarrow (\mathtt{dummy}, \mathtt{MAC}(i, \mathtt{dummy}))$$
for all $i = 1, \cdots, 2 \cdot 2^\ell N$.
2. Next for each $w \in \{0,1\}^\ell$, suppose that $\mathtt{D}(w) = (D_{s_1}, \cdots, D_{s_m})$. Define $(s_{m+1}, \cdots, s_N)$ as
$$\{s_{m+1}, \cdots, s_N\} = \{1, \cdots, N\} \setminus \{s_1, \cdots, s_m\}$$
For $j = 1, \cdots, N$, do
$$\begin{aligned} &\mathtt{addr}_j \leftarrow \begin{cases} \pi(0, w, j)\ if\ 1 \leq j \leq m \\ \pi(1, w, j)\ if\ m+1 \leq j \leq N \end{cases} \\ &tag_j \leftarrow \mathtt{MAC}(\mathtt{addr}_j, C_{s_j}) \\ &\mathcal{I}(\mathtt{addr}_j) \leftarrow (s_j, tag_j). \end{aligned}$$

It is now easy to see that each index i appears 2^ℓ times in $\mathcal{I}$.

Example 1. Suppose that $N = 5$ and $\mathtt{D}(Austin) = (D_1, D_3, D_5)$. Then

$$\begin{aligned}
\mathcal{I}(\pi(0, Austin, 1)) &= (1, tag_1) \\
\mathcal{I}(\pi(0, Austin, 2)) &= (3, tag_2) \\
\mathcal{I}(\pi(0, Austin, 3)) &= (5, tag_3) \\
\mathcal{I}(\pi(1, Austin, 4)) &= (2, tag_4) \\
\mathcal{I}(\pi(1, Austin, 5)) &= (4, tag_5)
\end{aligned}$$

$\mathtt{Trpdr}(K, w)$**:** Output

$$t(w) = (\pi(0, w, 1), \cdots, \pi(0, w, N)).$$

$\mathtt{Search}(\mathcal{I}, \mathcal{C}, t(w))$**:** Parse $t(w)$ as $t(w) = (\mathtt{addr}_1, \cdots, \mathtt{addr}_N)$. Suppose that

$$\mathcal{I}(\mathtt{addr}_i) = (s_i, tag_i)$$

for $i = 1, \cdots, N$. First let $\mathtt{C}(w) \leftarrow$ *empty*. Next for $i = 1, \cdots, N$, add C_{s_i} to $\mathtt{C}(w)$ if $s_i \neq \mathtt{dummy}$. Finally let

$$Tag = (tag_1, \cdots, tag_N)$$

Output $(\mathtt{C}(w), Tag)$.

$\mathtt{Verify}(K, t(w), \tilde{\mathtt{C}}(w), Tag)$: Parse $t(w), \tilde{\mathtt{C}}(w)$ and Tag as

$$\begin{aligned}
t(w) &= (\mathtt{addr}_1, \cdots, \mathtt{addr}_N) \\
\tilde{\mathtt{C}}(w) &= (\tilde{C}_1, \cdots, \tilde{C}_m) \\
Tag &= (tag_1, \cdots, tag_N)
\end{aligned}$$

First let $X_i \leftarrow \tilde{C}_i$ for $i = 1, \cdots, m$. Next let $X_i \leftarrow \mathtt{dummy}$ for $i = m+1, \cdots, N$. Finally if $tag_i = \mathtt{MAC}(\mathtt{addr}_i, X_i)$ for $i = 1, \cdots, N$, then output `accept`. Otherwise output `reject`.

$\mathtt{Dec}(K, C)$**:** Output a document $D = E^{-1}(K_0, C)$ for a ciphertext C.

5.3 Security

We assume that the symmetric-key encryption scheme $\mathtt{SKE} = (G, E, E^{-1})$ is left-or-right (LOR) CPA secure as defined by [2]. The common counter mode with AES satisfies this condition, where AES is assumed to be a pseudo-random permutation. We also assume that `MAC` is unforgeable against chosen message attack.

Theorem 3. *The above scheme satisfies privacy (see Def.1).*

(Proof) We construct a simulator **Sim** as follows. In the store phase, **Sim** is given $|D_1|, \cdots, |D_N|$ and ℓ.

1. **Sim** runs $\mathtt{Gen}(1^k)$ to generate $K = (K_0, K_1, K_2)$.

2. Let $C'_i = E(K_0, 0^{|D_i|})$ for $i = 1, \cdots, N$, and let $\mathcal{C}' = \{C'_1, \cdots, C'_N\}$.
3. Construct $\mathcal{I}'$ as if $\mathtt{D}(w) = (D_1, \cdots, D_N)$ for all $w \in \{0,1\}^\ell$. This means that for each $w \in \{0,1\}^\ell$,

$$\mathcal{I}'(\pi(0, w, i)) = (i, tag_i) \text{ for } i = 1, \cdots, N \tag{5}$$
$$\mathcal{I}'(\pi(1, w, i)) = (\mathtt{dummy}, tag'_i) \text{ for } i = 1, \cdots, N \tag{6}$$

where $tag_i = \mathtt{MAC}(\pi(0, w, i), C'_i)$ and $tag'_i = \mathtt{MAC}(\pi(1, w, i), \mathtt{dummy})$.
That is,

$$\mathcal{I}'(\pi(0, w, 1)) = (1, tag_1), \qquad \mathcal{I}'(\pi(1, w, 1)) = (\mathtt{dummy}, tag'_1)$$
$$\vdots \qquad\qquad\qquad \vdots$$
$$\mathcal{I}'(\pi(0, w, N)) = (N, tag_N), \qquad \mathcal{I}'(\pi(1, w, N)) = (\mathtt{dummy}, tag'_N)$$

4. Return $(\mathcal{I}', \mathcal{C}')$.

In the search phase, for $i = 1, \cdots, q$, **Sim** is given

$$\mathtt{List}(w_i) = \{s_1, \cdots, s_m\}$$

(but not w_i). Then **Sim** returns

$$t(w_i)' = (\pi(0, i, s_1), \cdots, \pi(0, i, s_m), \pi(1, i, m+1), \cdots, \pi(1, i, N)). \tag{7}$$

We will prove that any **A** cannot distinguish between $\mathtt{Game}_{real}$ and $\mathtt{Game}_{sim}$ by using a series of games $\mathtt{Game}_0, \cdots, \mathtt{Game}_2$, where $\mathtt{Game}_0 = \mathtt{Game}_{real}$. Let

$$p_i = \Pr(\mathbf{A} \text{ outputs } b = 1 \text{ in } \mathtt{Game}_i).$$

- $\mathtt{Game}_1$ is the same as $\mathtt{Game}_0$ except for that C_i is replaced with C'_i of the above for $i = 1, \cdots, N$. Then $|p_0 - p_1|$ is negligible from our assumption on SKE.
- $\mathtt{Game}_2$ is the same as $\mathtt{Game}_1$ except for that $\mathcal{I}$ is replaced with $\mathcal{I}'$ of the above, and $t(w_i)$ is replaced with $t(w_i)'$ of the above.
  Note that the index i of each D_i appears 2^ℓ times in both $\mathcal{I}$ and $\mathcal{I}'$.
  Next on $t(w_i)'$, let

$$\mathtt{addr}_1 = \pi(0, i, s_1), \cdots, \mathtt{addr}_m = \pi(0, i, s_m),$$
$$\mathtt{addr}_{m+1} = \pi(1, i, m+1), \cdots, \mathtt{addr}_N = \pi(1, i, N).$$

  Then from eq.(5) and eq.(6), it is easy to see that

$$\mathcal{I}'(\mathtt{addr}_1) = (s_1, tag_1), \cdots, \mathcal{I}'(\mathtt{addr}_m) = (s_m, tag_m),$$
$$\mathcal{I}'(\mathtt{addr}_{m+1}) = (\mathtt{dummy}, tag'_{m+1}), \cdots, \mathcal{I}'(\mathtt{addr}_N) = (\mathtt{dummy}, tag'_N)$$

  The value of such $\mathcal{I}'(\mathtt{addr}_i)$ is the same as the real one. Further π is a pseudo-random permutation. Hence $|p_1 - p_2|$ is negligible.

Consequently $|p_0 - p_2|$ is negligible. Further it is clear that $\mathtt{Game}_2 = \mathtt{Game}_{sim}$. This means that $\mathtt{Game}_{real}$ and $\mathtt{Game}_{sim}$ are indistinguishable for any **A**.

Q.E.D.

Theorem 4. *The above scheme satisfies reliability (see Def.2).*

(Proof) Suppose that there exists an adversary **A** who breaks the reliability for some $(\mathcal{D}, \mathcal{W})$ and some search queries $w_1, \cdots, w_q$. We will show a forger **B** for the underlying MAC.

B runs **A** by playing the role of a client, where the input to the client is $(\mathcal{D}, \mathcal{W})$ in the store phase, and $w_1, \cdots, w_q$ in the search phase. In the search phase, **B** uses his MAC oracle to compute $\mathcal{I}$.

From our assumption, **A** returns $(\mathtt{C}(w)^*, Tag^*)$ for some query $t(w)$ such that $\mathtt{C}(w)^*$ is invalid for $t(w)$ and

$$\mathtt{Verify}(K, t(w), \mathtt{C}(w)^*, Tag^*) = \mathtt{accept} \tag{8}$$

with non-negligible probability, where $w \in \{w_1, \cdots, w_q\}$. Let

$$(\mathtt{C}(w), Tag) \leftarrow \mathtt{Search}(\mathcal{I}, \mathcal{C}, t(w)). \tag{9}$$

Then $\mathtt{C}(w)^* \neq \mathtt{C}(w)$ because $\mathtt{C}(w)^*$ is invalid for $t(w)$. Suppose that

$$\begin{aligned} t(w) &= (\mathtt{addr}_1, \cdots, \mathtt{addr}_N) \\ \mathtt{C}(w) &= (C_1, \cdots, C_k) \\ \mathtt{C}(w)^* &= (C_1^*, \cdots, C_m^*) \\ Tag^* &= (tag_1^*, \cdots, tag_N^*) \end{aligned}$$

Since $\mathtt{C}(w)^* \neq \mathtt{C}(w)$, there are three cases.

Case 1: $m = k$ and there exists some C_i^* such that $C_i^* \neq C_i$.
Case 2: $m < k$.
Case 3: $m > k$.

If (Case 1) occurs, then **B** outputs $(\mathtt{addr}_i, C_i^*)$ and tag_i^* as a forgery. We will show that this is a valid forgery on MAC. That is, $tag_i^* = \mathtt{MAC}(\mathtt{addr}_i, C_i^*)$ and **B** never queried $(\mathtt{addr}_i, C_i^*)$ to the MAC oracle.

First it is clear that $tag_i^* = \mathtt{MAC}(\mathtt{addr}_i, C_i^*)$ from eq.(8).

Next it is easy to see that **B** queried $(\mathtt{addr}_i, C_i)$ to his MAC oracle when computing $\mathcal{I}$ from eq.(9). It means that **B** did not query $(\mathtt{addr}_i, C_i^*)$ to the MAC oracle because **B** does not query $\mathtt{addr}_i$ to the MAC oracle more than once when computing $\mathcal{I}$. This means that **B** succeeds in forgery.

If (Case 2) occurs, then B outputs $(\mathtt{addr}_{m+1}, \mathtt{dummy})$ and tag_{m+1}^* as a forgery. If (Case 3) occurs, then B outputs $(\mathtt{addr}_m, C_m^*)$ and tag_m^* as a forgery. We can show that these are valid forgeries on MAC similarly.

This is against our assumption on MAC. Hence our scheme satisfies reliability.

Q.E.D.

Corollary 1. *The corresponding protocol Σ_{vSSE} is UC-secure against non-adaptive adversaries.*

(Proof) From Theorem 2.

Q.E.D.

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. Ran Canetti, Universally Composable Security: A New Paradigm for Cryptographic Protocols, Revision 1 of ECCC Report TR01-016 (2001)
4. Ran Canetti, Universally Composable Signatures, Certification and Authentication, Cryptology ePrint Archive, Report 2003/239 http://eprint.iacr.org/ (2003)
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. Y.Chang and M.Mitzenmacher: Privacy Preserving Keyword Searches on Remote Encrypted Data. ACNS 2005: pp.442–455 (2005)
9. Eu-Jin Goh: Secure Indexes. Cryptology ePrint Archive, Report 2003/216, http://eprint.iacr.org/ (2003)
10. D.Song, D.Wagner, A.Perrig: Practical Techniques for Searches on Encrypted Data. IEEE Symposium on Security and Privacy 2000: pp.44–55 (2000)