Efficient Trapdoor Generation from Multiple Hashing in Searchable Symmetric Encryption

Takato Hirano(✉), Yutaka Kawai, and Yoshihiro Koseki

Mitsubishi Electric Corporation, Kamakura, Japan
Hirano.Takato@ay.MitsubishiElectric.co.jp,
Kawai.Yutaka@da.MitsubishiElectric.co.jp,
Koseki.Yoshihiro@ak.MitsubishiElectric.co.jp

Abstract. Searchable symmetric encryption (SSE) which can search encrypted data using encrypted keywords has been extremely studied. In Asiacrypt'10, Chase and Kamara formalized structured encryption which is a generalization of SSE, and its concrete schemes were proposed. An efficient SSE scheme (hereafter, Chase-Kamara scheme) which has a very simple encrypted index is obtained by simplifying the concrete schemes, and its adaptive security can be proved, easily. In the Chase-Kamara scheme, a search result for a keyword is represented as a bit string in which the i-th bit is 1 when the i-th document contains the keyword, and the encrypted index is built by directly masking the search result with each bit of the output of a pseudo-random function. Therefore, the Chase-Kamara scheme requires pseudo-random functions whose output lengths are longer than the number of documents that users would like to store. As a result, the trapdoor size of the Chase-Kamara scheme depends on the number of stored documents. In this paper, we propose a modified scheme whose trapdoor size does not depend on the number of stored documents. The modified scheme is constructed by using our multiple hashing technique which can transform a trapdoor of short length to that of long length without any secret information. We also show that the modified scheme achieves the same adaptive security as the Chase-Kamara scheme in the random oracle model.

Keywords: Searchable symmetric encryption
Chase-Kamara scheme · Trapdoor size · Multiple hashing

1 Introduction

1.1 Background

Nowadays, cloud services such as data storing on remote third-party providers give high data availability and reduce IT infrastructure costs of a company. From a viewpoint of security, company's sensitive data such as secret information or

C. Su and H. Kikuchi (Eds.): ISPEC 2018, LNCS 11125, pp. 160–175, 2018.
https://doi.org/10.1007/978-3-319-99807-7_10

privacy data of customers should be encrypted to be kept secret from people outside of the company when stored on the cloud. On the other hand, it is indispensable to search the stored data from a viewpoint of usability. However, data encrypting and keyword searching are incompatible in general, since keyword searching for encrypted data is intractable. Although there is a naive approach in which keyword searching is performed after decrypting encrypted data on the cloud, this is insufficient because malicious administrators or softwares on the cloud would steal the plain data or decryption keys when performed the decryption process. As a solution to these problems, searchable encryption has been proposed.

After the first searchable encryption scheme was proposed in [42], many concrete schemes have been constructed. Roughly speaking, searchable encryption schemes are typically classified into two types: symmetric-key type (e.g. [1,2,5,7–49]) and public-key type (e.g. [3,6]). This paper focuses on the former searchable encryption.

Searchable encryption of symmetric-key type is called *searchable symmetric encryption* or *SSE*. SSE consists of *document storing process* and *keyword searching process*, and these processes are performed by the same user since a unique secret key is used in typical SSE. In the document storing process, the user encrypts documents and generates an encrypted index from the secret key, and the server stores a pair of the encrypted documents and the encrypted index. In the keyword searching process, the user generates an encrypted query (called *trapdoor*) from the secret key and a keyword, and the server searches by applying the trapdoor to the encrypted index. Although the keyword searching cost in SSE is quite lower than that in public-key type, this cost becomes critical even in SSE as the number of stored documents increases. In order to reduce this cost, SSE schemes with useful indexes such as inverted index structure or Bloom filter have been constructed.

Security models for SSE also have been studied. Curtmola et al. [15,16] carefully extracted unavoidable information leaked from the document storing process and the keyword searching process of a typical SSE scheme, and formalized acceptable leakage information. Then, they defined that an SSE scheme is secure if information revealed from the processes of the SSE scheme is at most the acceptable leakage information. Their security model and its variants (e.g. [13]) are used in many SSE schemes. Especially, adaptive security definitions proposed in [13,15,16] is considered as one of the security goals in SSE literature.

1.2 Motivation

The SSE schemes (called SSE-1 and SSE-2) proposed by Curtmola et al. have search-friendly encrypted indexes such as inverted index structure [15,16]. Their schemes have had a big impact on constructing efficient SSE schemes. Especially, SSE-2 is constructed only from pseudo-random functions and achieves the adaptive security. Furthermore, the keyword searching process of SSE-2 is based on the binary searching operation, and therefore performed efficiently. However,

there is a problem that the trapdoor size of SSE-2 depends on the number of stored documents.

Chase and Kamara formalized structured encryption which is a generalization of SSE, and its concrete schemes were proposed [13]. An efficient SSE scheme (hereafter, Chase-Kamara scheme) which has a very simple structure is obtained by simplifying the concrete schemes. It is very easy to show that the Chase-Kamara scheme achieves the adaptive security, thanks to simplicity of its encrypted index structure. In the Chase-Kamara scheme, a search result for a keyword is represented as a bit string in which the i-th bit is 1 when the i-th document contains the keyword, and the encrypted index is built by directly masking the search result with each bit of the output of a pseudo-random function. Therefore, the Chase-Kamara scheme requires pseudo-random functions whose output lengths are longer than the number of documents that the user would like to store. As a result, the trapdoor size of the Chase-Kamara scheme depends on the number of stored documents. This trapdoor size becomes critical as the number of stored documents increases. For example, the trapdoor size is about 120MB when the number of stored documents is one billion. Thus, the Chase-Kamara scheme has the same trapdoor size problem as SSE-2.

Recently, Miyoshi et al. proposed the SSE scheme with a small encrypted index [36]. Their scheme is constructed by hierarchical Bloom filters, and achieves the adaptive security. However, in their scheme, the trapdoor size also depends on the number of stored documents, and the number of communication rounds between the user and the server is two. Therefore, their keyword searching process is inefficient although the encrypted index size is reasonable.

1.3 Our Contributions

In this paper, we focus on the trapdoor size problem of the Chase-Kamara scheme, and propose a modified scheme whose trapdoor size does not depend on the number of stored documents. The modified scheme is constructed by using our multiple hashing technique which can transform a trapdoor of *short length* to that of *long length* without any secret information. With this technique, the trapdoor size of the modified scheme depends only on the output length of a used hash function (e.g. 512-bit if SHA-256 is used) even if the number of stored documents is one billion. We can show that the modified scheme is adaptively secure in the random oracle model.

A key point of our modified scheme is to securely divide the trapdoor generation process of the Chase-Kamara scheme by using our multiple hashing technique. According to this modification of the trapdoor generation process, the encrypted index of the Chase-Kamara scheme is also slightly modified. Informally, in the Chase-Kamara scheme, the user generates a trapdoor of long length and the server searches the encrypted index by directly using the trapdoor. On the other hand, in our modified scheme, the user generates a trapdoor of short length, and the server transforms the trapdoor to a meaningful value of long length, which consists of hash values and corresponds to the trapdoor of the Chase-Kamara scheme. This transformation uses only the trapdoor sent by

the user, but not any secret information. After that, the server searches the encrypted index using the trapdoor and the meaningful value, similarly to the Chase-Kamara scheme.

We give a comparison result among the adaptively secure SSE schemes [13, 15, 36] and our modified scheme in Table 1, where ℓ and λ are the output lengths of a pseudo-random function and a hash function, respectively, n_D is the number of stored documents, n_w is the number of used keywords, $n_{\mathbf{D}(w)}$ is the number of documents containing the keyword w (i.e. the cardinality of the search result of w), $\Sigma_{\mathbf{D}(w)} = \sum_{i=1}^{n_w} n_{\mathbf{D}(w_i)}$, $m_{\mathbf{D}(w)} = \max_w(n_{\mathbf{D}(w)})$, and $\mathtt{PRF}$ and $\mathtt{HF}$ are the computation costs of a pseudo-random function and a hash function, respectively. Here, we assume that $\lambda < n_D$ and the binary complete-matching cost for N words is $\log N$. Note that these assumptions are reasonable in practical situations.

Table 1. Comparisons among related works [13, 15, 36] and our work.

Scheme	SSE-2 [15]	Chase-Kamara [13]	Miyoshi et al. [36]	Our work
Index size	$O(\ell n_w m_{\mathbf{D}(w)})$	$O(n_w n_D)$	$O(\ell(n_w \log n_w + \Sigma_{\mathbf{D}(w)}))$	$O(n_w n_D)$
Trapdoor size	$O(\ell m_{\mathbf{D}(w)})$	$O(n_D)$	$O(\ell n_{\mathbf{D}(w)})$	$O(\lambda)$
User's search cost	$O(m_{\mathbf{D}(w)}\mathtt{PRF})$	$O(\mathtt{PRF})$	$O(n_{\mathbf{D}(w)}\mathtt{PRF})$	$O(\mathtt{HF})$
Server's search cost	$O(m_{\mathbf{D}(w)} \log n_D)$	$O(\log n_D)$	$O(\mathtt{PRF} \log n_w + n_{\mathbf{D}(w)} \log m_{\mathbf{D}(w)})$	$O(\frac{n_D}{\lambda}\mathtt{HF} + \log n_w)$
Round	1	1	2	1

1.4 Related Works

Curtmola et al. proposed the SSE schemes (SSE-1 and SSE-2) whose encrypted indexes have search-friendly structures such as inverted index [15]. Their schemes have had a big impact on constructing efficient SSE schemes. Although SSE-2 achieves the adaptive security, the trapdoor size of SSE-2 depends on the number of stored documents.

The Chase-Kamara scheme [13] can build an encrypted index of a very simple structure, and therefore the keyword searching process is conducted efficiently. However, the trapdoor size depends on the number of stored documents.

The Miyoshi et al. scheme [36] can a construct small encrypted index by using hierarchical Bloom filters. However, the trapdoor size also depends on the number of stored documents, and the number of communication rounds between the user and the server is two.

While this paper focuses on constructing efficient SSE schemes, other useful functionalities for SSE have been studied, in addition to basic functionalities such as document storing and keyword searching: for example, document adding/deleting/updating functionalities (a.k.a. dynamic SSE) [9, 15,

20,23,26,28,29,37,38,43,45,47–49], flexible search functionalities [5,7,10,14, 18,21,27,30,31,34,35,37,41,46], localities [2,11,17], forward security [8], UC-security [32,33,40], multi-user settings [1,15,16,19,24,48], etc.

1.5 Organization

The rest of this paper is organized as follows. In Sect. 2, we recall cryptographic primitives and SSE definitions which are used throughout the paper. The Chase-Kamara scheme is given in Sect. 3, and its modified scheme is proposed in Sect. 4. We conclude in Sect. 5.

2 Preliminaries

In this section, we recall cryptographic primitives and SSE definitions which are used throughout the paper.

2.1 Notations and Basic Cryptographic Primitives

We denote the set of positive real numbers by $\mathbb{R}^+$. We say that a function $\mathtt{negl} : \mathbb{N} \to \mathbb{R}^+$ is negligible if for any (positive) polynomial p, there exists $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, it holds $\mathtt{negl}(n) < 1/p(n)$. If A is a probabilistic algorithm, $y \leftarrow A(x)$ denotes running A on input x with a uniformly-chosen random tape and assigning the output to y. $A^{\mathcal{O}}$ denotes an algorithm with oracle access to $\mathcal{O}$. If S is a finite set, $s \xleftarrow{u} S$ denotes that s is uniformly chosen from S. We denote the bit length of S by $|S|$, and the cardinality of S by $\#S$. For strings a and b, $a||b$ denotes the concatenation of a and b.

We recall the definition of pseudo-random functions. A function $f : \{0,1\}^\lambda \times \{0,1\}^k \to \{0,1\}^\ell$ is pseudo-random if f is polynomial-time computable in λ, and for any probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$, it holds

$$|\Pr[1 \leftarrow \mathcal{A}^{f_K(\cdot)}(1^\lambda) \mid K \xleftarrow{u} \{0,1\}^\lambda] - \Pr[1 \leftarrow \mathcal{A}^{g(\cdot)}(1^\lambda) \mid g \xleftarrow{u} \mathtt{F}[k,\ell]]| \leq \mathtt{negl}(\lambda),$$

where $\mathtt{F}[k,\ell]$ is the set of functions mapping $\{0,1\}^k$ to $\{0,1\}^\ell$.

We recall the definition of *left-or-right indistinguishability against the chosen plaintext attack (LOR-CPA)* for symmetric-key encryption [4]. A symmetric-key encryption scheme is secure in the sense of LOR-CPA if for any PPT adversary $\mathcal{A}$, it holds

$$\begin{aligned}|\Pr[1 \leftarrow \mathcal{A}^{\mathtt{Enc}_K(\mathcal{LR}(\cdot,\cdot,1))}(1^\lambda) \mid K \leftarrow \mathtt{Gen}(1^\lambda)] \\ - \Pr[1 \leftarrow \mathcal{A}^{\mathtt{Enc}_K(\mathcal{LR}(\cdot,\cdot,0))}(1^\lambda) \mid K \leftarrow \mathtt{Gen}(1^\lambda)]| \leq \mathtt{negl}(\lambda),\end{aligned}$$

where $\mathtt{Enc}_K(\mathcal{LR}(\cdot,\cdot,b))$ is the left-or-right oracle that takes an input (x_0,x_1) and outputs $C_0 \leftarrow \mathtt{Enc}_K(x_0)$ if $b = 0$ and $C_1 \leftarrow \mathtt{Enc}_K(x_1)$ if $b = 1$.

2.2 Definitions of SSE

We recall the definitions of SSE, formalized in [15]. Firstly, we give notions used in SSE literature.

- Let $D \in \{0,1\}^*$ be a document, and $\mathbf{D} = (D_1, \ldots, D_n)$ be a document collection. Let $\mathbf{C} = (C_1, \ldots, C_n)$ be a ciphertext collection of $\mathbf{D}$, where C_i is a ciphertext of D_i for $1 \leq i \leq n$. We assume that D_i and C_i contain the same unique identifier id_i.
- Let $w \in \{0,1\}^k$ be a keyword, and $\Delta \subseteq \{0,1\}^k$ be a set of possible keywords. Let $\Delta(\mathbf{D}) \subseteq \Delta$ be a set of keywords which are contained in some of $D_1, \ldots, D_n$. Throughout this paper, we assume that $\#\Delta$ is polynomially bounded in a security parameter λ.
- For $\mathbf{D} = (D_1, \ldots, D_n)$ and $w \in \Delta$, let $\mathbf{D}(w)$ be a set of identifiers of documents that contain w. Namely, $\mathbf{D}(w) = \{id_{i_1}, \ldots, id_{i_m}\}$ for $w \in \Delta(\mathbf{D})$ or $\emptyset$ for $w \notin \Delta(\mathbf{D})$. For a searching sequence $\mathbf{w} = (w_1, \ldots, w_q)$, let $\mathbf{D}(\mathbf{w}) = (\mathbf{D}(w_1), \ldots, \mathbf{D}(w_q))$.

An SSE scheme over Δ, $\mathtt{SSE} = (\mathtt{Gen}, \mathtt{Enc}, \mathtt{Trpdr}, \mathtt{Search}, \mathtt{Dec})$, is defined as follows.

- $K \leftarrow \mathtt{Gen}(1^\lambda)$: $\mathtt{Gen}$ is a probabilistic algorithm which takes a parameter 1^λ as an input and outputs a secret key K, where λ is a security parameter.
- $(\mathcal{I}, \mathbf{C}) \leftarrow \mathtt{Enc}(K, \mathbf{D})$: $\mathtt{Enc}$ is a probabilistic algorithm which takes a secret key K and a document collection $\mathbf{D}$ as input and outputs an encrypted index $\mathcal{I}$ and a ciphertext collection $\mathbf{C} = (C_1, \ldots, C_n)$.
- $T \leftarrow \mathtt{Trpdr}(K, w)$: $\mathtt{Trpdr}$ is a deterministic algorithm which takes a secret key K and a keyword w as input and outputs a trapdoor T.
- $S \leftarrow \mathtt{Search}(\mathcal{I}, T)$: $\mathtt{Search}$ is a deterministic algorithm which takes an encrypted index $\mathcal{I}$ and a trapdoor T as input and outputs an identifier set S.
- $D \leftarrow \mathtt{Dec}(K, C)$: $\mathtt{Dec}$ is a deterministic algorithm which takes a secret key K and a ciphertext C as input and outputs a plaintext D of C.

An SSE scheme is correct if for all $\lambda \in \mathbb{N}$, all $\mathbf{D}$, all $w \in \Delta(\mathbf{D})$, all K output by $\mathtt{Gen}(1^\lambda)$, and all $(\mathcal{I}, \mathbf{C})$ output by $\mathtt{Enc}(K, \mathbf{D})$, it holds $\mathtt{Search}(\mathcal{I}, \mathtt{Trpdr}(K, w)) = \mathbf{D}(w)$ and $\mathtt{Dec}(K, C_i) = D_i$ for $1 \leq i \leq n$.

We give security notions, *history*, *access pattern*, *search pattern*, *trace*, and *non-singular* [15].

- For a document collection $\mathbf{D} = (D_1, \ldots, D_n)$ and a searching sequence $\mathbf{w} = (w_1, \ldots, w_q)$, $H = (\mathbf{D}, \mathbf{w})$ is called *history*. This information is sensitive in SSE.
- $\alpha(H) = (\mathbf{D}(w_1), \ldots, \mathbf{D}(w_q))$ is called *access pattern* for a history $H = (\mathbf{D}, \mathbf{w})$. This information is appeared by performing the keyword searching processes.
- The following binary symmetric matrix $\sigma(H) = (\sigma_{i,j})$ is called *search pattern* for a history $H = (\mathbf{D}, \mathbf{w})$: for $1 \leq i \leq j \leq q$, $\sigma_{i,j} = 1$ if $w_i = w_j$, and $\sigma_{i,j} = 0$ otherwise. This information is appeared by performing the keyword searching processes because trapdoors are deterministically generated in SSE.

- $\tau(H) = (|D_1|, \ldots, |D_n|, \alpha(H), \sigma(H))$ is called *trace* for a history $H = (\mathbf{D}, \mathbf{w})$. This information is leaked while performing SSE protocols, and therefore considered as acceptable leakage information in SSE.
- H is called *non-singular* if (1) there exists a history $H' \neq H$ such that $\tau(H) = \tau(H')$, and (2) H' is computed from a given trace $\tau(H)$, efficiently. We assume that any history is non-singular throughout the paper.

Then, we give the adaptive security definition proposed in [15] (a.k.a. IND-CKA2), which is widely used in SSE literature.

Definition 1 ([15]). *Let* $\mathtt{SSE} = (\mathtt{Gen}, \mathtt{Enc}, \mathtt{Trpdr}, \mathtt{Search}, \mathtt{Dec})$, λ *be a security parameter,* $q \in \mathbb{N} \cup \{0\}$, *and* $\mathcal{A} = (\mathcal{A}_0, \ldots, \mathcal{A}_q)$ *and* $\mathcal{S} = (\mathcal{S}_0, \ldots, \mathcal{S}_q)$ *be probabilistic polynomial-time (PPT) algorithms. Here, we consider the following experiments* **Real** *and* **Sim***:*

$\mathbf{Real}_{\mathcal{A}}(1^\lambda)$:
$K \leftarrow \mathtt{Gen}(1^\lambda)$
$(\mathbf{D}, st_{\mathcal{A}}) \leftarrow \mathcal{A}_0(1^\lambda)$
$(\mathcal{I}, \mathbf{C}) \leftarrow \mathtt{Enc}(K, \mathbf{D})$
Let $\mathbf{t}_0 = \emptyset$
For $1 \leq i \leq q$:
$(w_i, st_{\mathcal{A}}) \leftarrow \mathcal{A}_i(st_{\mathcal{A}}, \mathcal{I}, \mathbf{C}, \mathbf{t}_{i-1})$
$t_i \leftarrow \mathtt{Trpdr}(K, w_i)$
Let $\mathbf{t}_i = \mathbf{t}_{i-1} || t_i$
Output $(\mathcal{I}, \mathbf{C}, \mathbf{t}_q, st_{\mathcal{A}})$

$\mathbf{Sim}_{\mathcal{A},\mathcal{S}}(1^\lambda)$:
$(\mathbf{D}, st_{\mathcal{A}}) \leftarrow \mathcal{A}_0(1^\lambda)$
$(\mathcal{I}, \mathbf{C}, st_{\mathcal{S}}) \leftarrow \mathcal{S}_0(\tau(\mathbf{D}))$
Let $\mathbf{w}_0 = \emptyset$ and $\mathbf{t}_0 = \emptyset$
For $1 \leq i \leq q$:
$(w_i, st_{\mathcal{A}}) \leftarrow \mathcal{A}_i(st_{\mathcal{A}}, \mathcal{I}, \mathbf{C}, \mathbf{t}_{i-1})$
Let $\mathbf{w}_i = \mathbf{w}_{i-1} || w_i$
$(t_i, st_{\mathcal{S}}) \leftarrow \mathcal{S}_i(st_{\mathcal{S}}, \tau(\mathbf{D}, \mathbf{w}_i))$
Let $\mathbf{t}_i = \mathbf{t}_{i-1} || t_i$
Output $(\mathcal{I}, \mathbf{C}, \mathbf{t}_q, st_{\mathcal{A}})$

We define that SSE *is adaptively secure if for any* λ, *any* q *of polynomial size, and any* $\mathcal{A} = (\mathcal{A}_0, \ldots, \mathcal{A}_q)$, *there exists the following PPT algorithm* $\mathcal{S} = (\mathcal{S}_0, \ldots, \mathcal{S}_q)$*: For any PPT distinguisher* $\mathcal{D}$, *it holds*

$$|\Pr[\mathcal{D}(\mathcal{I}, \mathbf{C}, \mathbf{t}_q, st_{\mathcal{A}}) = 1 \mid (\mathcal{I}, \mathbf{C}, \mathbf{t}_q, st_{\mathcal{A}}) \leftarrow \mathbf{Real}_{\mathcal{A}}(1^\lambda)] - \Pr[\mathcal{D}(\mathcal{I}, \mathbf{C}, \mathbf{t}_q, st_{\mathcal{A}}) = 1 \mid (\mathcal{I}, \mathbf{C}, \mathbf{t}_q, st_{\mathcal{A}}) \leftarrow \mathbf{Sim}_{\mathcal{A},\mathcal{S}}(1^\lambda)]| \leq \mathtt{negl}(\lambda).$$

3 The Chase-Kamara Scheme

In this section, we give the Chase-Kamara scheme which is directly obtained by simplifying the structured encryption schemes (especially, the associative structured encryption scheme for labeled data) proposed in [13].

Let $F : \{0,1\}^\lambda \times \{0,1\}^k \to \{0,1\}^\ell$ be a pseudo-random function, and SKE be a symmetric-key encryption scheme. Let n be the number of stored documents, that is, $\mathbf{D} = \{D_1, \ldots, D_n\}$. In the Chase-Kamara scheme, we restrict that $\ell \geq n$. Here, we use the following bit string $b_1 || \cdots || b_n || b_{n+1} || \cdots || b_\ell$ as another representation for $\mathbf{D}(w)$: $b_i = 1$ if $id_i \in \mathbf{D}(w)$, and $b_i = 0$ otherwise. For example, if $n = 3$, $\ell = 5$, and $\mathbf{D}(w) = \{id_1, id_3\}$, then we also regard $\mathbf{D}(w)$ as 10100. The encrypted index $\mathcal{I}$ built in the Chase-Kamara consists of $\{(key, val)\}$. Let the notation $\mathcal{I}[x]$ be y if there exists a pair (x, y) in $\mathcal{I}$, or $\perp$ otherwise. Then, the Chase-Kamara scheme is given as follows:

- $\texttt{Gen}(1^\lambda)$:
  1. Choose $K_1, K_2 \xleftarrow{u} \{0,1\}^\lambda$ and $K_3 \leftarrow \texttt{SKE.Enc}(1^\lambda)$.
  2. Output $K = (K_1, K_2, K_3)$.
- $\texttt{Enc}(K, \mathbf{D})$:
  1. Let $\mathcal{I} = \emptyset$.
  2. For $w \in \Delta$,
     (a) Compute $key = F(K_1, w)$ and $val = \mathbf{D}(w) \oplus F(K_2, w)$.
     (b) Append (key, val) to $\mathcal{I}$.
  1. For $D \in \mathbf{D}$, compute $C \leftarrow \texttt{SKE.Enc}(K_3, D)$.
  2. Output $\mathcal{I}$ and $\mathbf{C} = (C_1, \ldots, C_n)$.
- $\texttt{Trpdr}(K, w)$:
  1. Compute $T_1 = F(K_1, w)$ and $T_2 = F(K_2, w)$.
  2. Output $T = (T_1, T_2)$.
- $\texttt{Search}(\mathcal{I}, T)$:
  1. Parse $T = (T_1, T_2)$.
  2. Let $S = \emptyset$.
  3. If $\mathcal{I}[T_1] = \perp$ then output $\emptyset$.
  4. Compute $v = \mathcal{I}[T_1] \oplus T_2$.
  5. Parse $v = v_1 || \cdots || v_n || v_{n+1} || \cdots || v_\ell$, where $v_i \in \{0,1\}$ for $1 \leq i \leq \ell$.
  6. For $1 \leq i \leq n$, append id_i to S if $v_i = 1$.
  7. Output S.
- $\texttt{Dec}(K, C)$:
  1. Compute $D \leftarrow \texttt{SKE.Dec}(K_3, C)$.
  2. Output D.

The Chase-Kamara scheme is adaptively secure if SKE is LOR-CPA secure and F is a pseudo-random function. This security proof is very simple and straightforward (see [13]).

We observe that the Chase-Kamara scheme can perform the keyword searching process, efficiently, thanks to very simple structures of the encrypted index $\mathcal{I}$ and the trapdoor T. On the other hand, the trapdoor size, especially $|T_2|$, depends on the number of stored documents (that is, n). The trapdoor size becomes critical as n is increased. For example, $|T_2|$ is of about one billion bits (approximately, 120MB) when n is one billion.

4 The Proposed Scheme

In this section, we tackle to the trapdoor size problem of the Chase-Kamara scheme, and propose its modified scheme by using our multiple hashing technique which can transform a trapdoor of short length to that of long length. Our modified scheme can break the restriction $\ell \geq n$, where n is the number of stored documents and ℓ is the output length of the used pseudo-random function F.

4.1 Our Strategy

A key point of our modified scheme is to securely divide the trapdoor generation process of the Chase-Kamara scheme by using our multiple hashing technique. According to this modification of the trapdoor generation process, the encrypted index of the Chase-Kamara scheme is also slightly modified. In the keyword searching process of the Chase-Kamara scheme, the user generates a trapdoor $T = (T_1, T_2)$ of long length (especially, $T_2 = F_2(K, w)$) and the server searches the encrypted index $\mathcal{I}$ by directly using the trapdoor T. In order to address the trapdoor size problem, we modify this process as follows. The user generates a trapdoor of *short length*, and the server transforms the trapdoor to a meaningful value of *long length*, which consist of multiple hash values and correspond to the trapdoor of the Chase-Kamara scheme. Then, the server searches the encrypted index using the trapdoor and the hash values. This process can be achieved by using our multiple hashing technique. This technical overview is as follows.

As shown in Sect. 3, the encrypted index $\mathcal{I}$ of the Chase-Kamara scheme is constructed by

$$\{(key, val)\}_w = \{(F(K_1, w), \mathbf{D}(w) \oplus F(K_2, w))\}_{w \in \Delta},$$

where w is a keyword, $F : \{0,1\}^\lambda \times \{0,1\}^k \to \{0,1\}^\ell$ is a pseudo-random function, K_1 and K_2 are secret keys of F, and $\mathbf{D}(w)$ is a plain search result for a keyword w and represented as the special bit string form described in Sect. 3. A trapdoor T for a keyword w is computed by $T = (T_1, T_2) = (F(K_1, w), F(K_2, w))$, where $|T_2| = \ell \geq n$.

In order to address the above trapdoor size problem, we modify the encrypted index of the Chase-Kamara scheme by using the following multiple hashing technique. For a hash function $H : \{0,1\}^* \to \{0,1\}^\lambda$, we modify $\mathcal{I}$ as[1]

$$\{(key, val)\} = \{(F(K_1, w), \mathbf{D}(w) \oplus (h_{w,1}|| \cdots ||h_{w,N}))\}_{w \in \Delta},$$

where $N = \lceil n/\lambda \rceil$ and

$$h_{w,1} = H(H(K_2||w)||1), \ldots, h_{w,N} = H(H(K_2||w)||N).$$

In addition to the above modification of the encrypted index, we further modify the trapdoor $T = (T_1, T_2)$ as $(F(K_1, w), H(K_2||w))$.

Then, the keyword searching process in this modification is conducted as follows. For a trapdoor $T = (F(K_1, w), H(K_2||w))$, the server computes hash values $h_{w,1}, \ldots, h_{w,N}$ from $T_2 = H(K_2||w)$, and then checks its search result by $\mathcal{I}[T_1] \oplus (h_{w,1}|| \cdots ||h_{w,N}) (= \mathbf{D}(w))$, similarly to the keyword searching process of the Chase-Kamara scheme. Thus, this modification dramatically reduce the trapdoor size from $O(n)$ to $O(\lambda)$. For example, the trapdoor size is of 512 bits when we use SHA-256. We also observe an advantage that the server can generate arbitrary long values corresponding to T_2 (i.e. the keyword w) with no secret

[1] Later, we also modify *key* and T_1 as $H(K_1||w)$. Furthermore, we set $K_1 = K||0$ and $K_2 = K||1$ using a secret key K.

information. We believe that our multiple hashing technique would be applied to other SSE schemes which have the trapdoor size problem, due to its generality and simplicity. Our multiple hashing technique is summarized in Fig. 1.

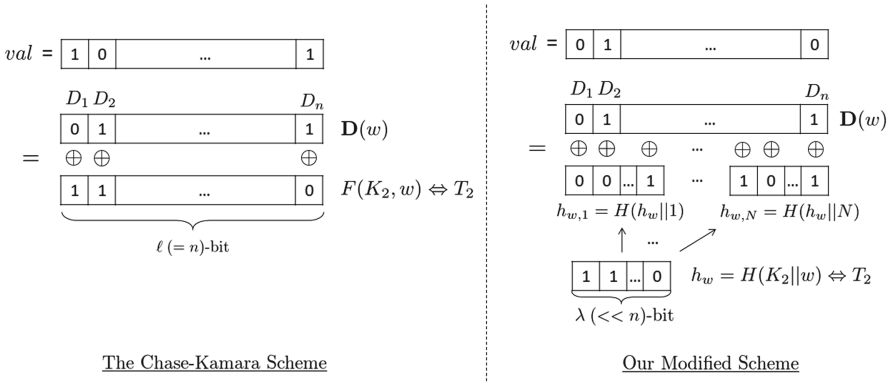

Fig. 1. Summary of our multiple hashing technique.

From a viewpoint of security, our multiple hashing technique leads that the server cannot infer not only hidden keywords from trapdoors, but also any information on relationships among elements of our encrypted index until received trapdoors, due to one-wayness of multiple hashing. As a result, we can also show its adaptive security from a similar strategy as the security proof of the Chase-Kamara scheme, but in the random oracle model since our proof strategy essentially requires randomness of hash functions.

4.2 Construction

Let $H : \{0,1\}^* \to \{0,1\}^\lambda$ be a hash function. Let $N = \lceil \frac{n}{\lambda} \rceil$ and $\mathbf{D}(w) = b_1||\cdots||b_n||b_{n+1}||\cdots||b_{\lambda N}$, where $b_1, \ldots, b_n$ are represented as the special bit form described in Sect. 3 and $b_{n+1} = \cdots = b_{\lambda N} = 0$. The modified scheme is proposed as follows:

- $\mathtt{Gen}(1^\lambda)$:
  1. Choose $K_1 \xleftarrow{u} \{0,1\}^\lambda$ and $K_2 \leftarrow \mathtt{SKE.Enc}(1^\lambda)$.
  2. Output $K = (K_1, K_2)$:
- $\mathtt{Enc}(K, \mathbf{D})$:
  1. Let $\mathcal{I} = \emptyset$.
  2. Compute $N = \lceil \frac{n}{\lambda} \rceil$.
  3. For $w \in \Delta$,
     (a) Compute $key = H(K_1||0||w)$.
     (b) Compute $h_w = H(K_1||1||w)$ and $h_{w,i} = H(h_w||i)$ for $1 \le i \le N$.
     (c) Compute $val = \mathbf{D}(w) \oplus (h_{w,1}||\cdots||h_{w,N})$.

(d) Append (key, val) to $\mathcal{I}$.
4. For $D \in \mathbf{D}$, compute $C \leftarrow \mathtt{SKE.Enc}(K_2, D)$.
5. Output $\mathcal{I}$ and $\mathbf{C} = (C_1, \ldots, C_n)$.

– $\mathtt{Trpdr}(K, w)$:
1. Compute $T_1 = H(K_1||0||w)$ and $T_2 = H(K_1||1||w)$.
2. Output $T = (T_1, T_2)$.

– $\mathtt{Search}(\mathcal{I}, T)$:
1. Parse $T = (T_1, T_2)$.
2. Let $S = \emptyset$.
3. If $\mathcal{I}[T_1] = \perp$ then output $\emptyset$.
4. Compute $N = \lceil \frac{n}{\lambda} \rceil$.
5. Compute $h'_1 = H(T_2||1), \ldots, h'_N = H(T_2||N)$.
6. Compute $v = \mathcal{I}[T_1] \oplus (h'_1||\cdots||h'_N)$.
7. Let $v = v_1||\cdots||v_n||v_{n+1}||\cdots||v_{\lambda N}$, where $v_i \in \{0,1\}$ for $1 \le i \le \lambda N$.
8. For $1 \le i \le n$, add id_i into S if $v_i = 1$.
9. Output S.

– $\mathtt{Dec}(K, C)$:
1. Compute $D \leftarrow \mathtt{SKE.Dec}(K_2, C)$.
2. Output D.

In the Chase-Kamara scheme, the user generates a trapdoor (T'_1, T'_2) for a keyword w', and the server searches the encrypted index $\mathcal{I}'$ by $\mathcal{I}'[T'_1] \oplus T'_2$. On the other hand, our modified scheme is that the user generates a trapdoor (T_1, T_2) for a keyword w, and the server transforms T_2 to the value $(h_{w,1}||\cdots||h_{w,N})$ and then searches the encrypted index $\mathcal{I}$ by $\mathcal{I}[T_1] \oplus (h_{w,1}||\cdots||h_{w,N})$.

Then, we can show the following security of the modified scheme.

Theorem 1. *The modified scheme is adaptively secure in the random oracle model if* `SKE` *is LOR-CPA secure.*

Before proving the security of our modified scheme, we give our proof strategy. Our security proof is straightforward, similarly to that of the Chase-Kamara scheme.

– **Simulation of $\mathcal{I}$:** From the leakage information $(|D_1|, \ldots, |D_n|)$ obtained by querying on $\mathbf{D}$, $\mathcal{S}$ chooses $k_i, r_{i,1}, \ldots, r_{i,N} \xleftarrow{u} \{0,1\}^\lambda$, and set $\mathcal{I} = \{(k_i, r_{i,1}||\cdots||r_{i,N})\}_{1 \le i \le \#\Delta}$. With this simulation, $\mathcal{S}$ cheats $\mathcal{A}$ as if $\mathcal{I} = \{(k_i, r_{i,1}||\cdots||r_{i,N})\}$ is generated in the real experiment.
– **Simulation of T:** If $\mathcal{A}$ queries on w_i, then for some j, $\mathcal{S}$ regards $r_{j,1}||\cdots||r_{j,N}$ as

$$\begin{aligned} r_{j,1}||\cdots||r_{j,N} &= \mathbf{D}(w_i) \oplus (\quad r'_{j,1} \quad ||\cdots||\quad r'_{j,N} \quad) \\ &= \mathbf{D}(w_i) \oplus (\ H(r_j||1)\ ||\cdots||\ H(r_j||N)\ ) \end{aligned}$$

by assigning some value $r_j \in \{0,1\}^\lambda$, and further regards r_j as $H(K_1||1||w_i)$. With this simulation, $\mathcal{S}$ cheats $\mathcal{A}$ as if r_j is obtained from $H(K_1||1||w_i)$ and $T = (k_j, r_j)$ is generated in the real experiment. In order to simulate the above completely, $\mathcal{S}$ computes $r'_{j,1}, \ldots, r'_{j,N}$ from $val_j = r_{j,1}||\cdots||r_{j,N}$ and the

leakage information $\mathbf{D}(w_i)$ obtained by querying on w_i, chooses $r_j \xleftarrow{u} \{0,1\}^\lambda$, and appends

Input	Output
$r_j \| \| 1$	$r'_{j,1}$
$\vdots$	$\vdots$
$r_j \| \| N$	$r'_{j,N}$

into a random oracle hash table $\mathcal{H}$.

Our formal proof with the above simulation is given as follows.

Proof. Let $\mathcal{H} = \{(input, output)\}$ be a random oracle hash table which is set to $\emptyset$, initially. A PPT simulator $\mathcal{S} = (\mathcal{S}_0, \ldots, \mathcal{S}_q)$ is constructed as follows.

$\mathcal{S}_0$'s simulation. For the leakage information $(|D_1|, \ldots, |D_n|)$ obtained from $\mathcal{A}$'s output $\mathbf{D} = (D_1, \ldots, D_n)$, $\mathcal{S}_0$ computes $N = \lceil \frac{n}{\lambda} \rceil$, and chooses random numbers $r_{1,1}, \ldots, r_{1,N}, \ldots, r_{\delta,1}, \ldots, r_{\delta,N} \xleftarrow{u} \{0,1\}^\lambda$, where $\delta = \#\Delta$. Let

$$\begin{aligned} R_1 &= r_{1,1} \| \cdots \| r_{1,N}, \\ &\vdots \\ R_\delta &= r_{\delta,1} \| \cdots \| r_{\delta,N}. \end{aligned}$$

$\mathcal{S}_0$ also chooses random numbers $k_1, \ldots, k_\delta \xleftarrow{u} \{0,1\}^\lambda$, and sets $\mathcal{I} = \{(k_i, R_i)\}_{1 \le i \le \delta}$. Further, $\mathcal{S}_0$ runs $SK \leftarrow \mathtt{SKE.Gen}(1^\lambda)$ and $C_i \leftarrow \mathtt{SKE.Enc}(SK, 0^{|D_i|})$ for $1 \le i \le n$. Then, $\mathcal{S}_0$ sends $\mathcal{I}$ and $\mathbf{C} = \{C_1, \ldots, C_n\}$ to $\mathcal{A}$.

$\mathcal{S}_i$'s simulation $(1 \le i \le q)$. For the leakage information $\alpha(\mathbf{D}, \mathbf{w}_i)$ and $\sigma(\mathbf{D}, \mathbf{w}_i)$ obtained from $\mathcal{A}$'s output w_i, $\mathcal{S}_i$ regards $\mathbf{D}(w_i)$ as $b_{i,1} \| \cdots \| b_{i,n} \| b_{i,n+1} (= 0) \| \cdots \| b_{i,\lambda N} (= 0)$, where $b_{i,j} = 1$ if $id_j \in \mathbf{D}(w_i)$ and $b_{i,j} = 0$ otherwise. After that, $\mathcal{S}_i$ checks whether $w_i \neq w_{i'}$ for any $w_{i'}$ $(1 \le i' < i)$. We note that this check can be efficiently done from the leakage information $\sigma(\mathbf{D}, \mathbf{w}_i)$.

If $w_i \neq w_{i'}$ for $1 \le i' < i$, $\mathcal{S}_i$ chooses $1 \le j \le \delta$ which has not been chosen yet, and computes $r'_{j,1} \| \cdots \| r'_{j,N} = \mathbf{D}(w_i) \oplus R_j$. Then, $\mathcal{S}_i$ chooses a random number $r_j \xleftarrow{u} \{0,1\}^\lambda$, appends

$$(r_j \| 1, r'_{j,1}), \ldots, (r_j \| N, r'_{j,N}),$$

into $\mathcal{H}$, and sends $T_i = (k_j, r_j)$ as a trapdoor of w_i to $\mathcal{A}$.

If there exists $i' < i$ such that $w_i = w_{i'}$, $\mathcal{S}_i$ merely re-sends $T_{i'} = (k_j, r_j)$, which has been already chosen in the i'-th simulation, to $\mathcal{A}$.

Analysis for $\mathcal{S}$'s simulation

- $\mathcal{I}$ and $(T_1, \ldots, T_q)$ output by $\mathcal{S}$ work correctly, similarly to **Real**.
- For any $1 \le i \le n$, $\mathcal{A}$ cannot distinguish C_i output by $\mathcal{S}_0$ from C_i output by **Real** since SKE is LOR-CPA secure.

- The probability that for any $1 \leq i \leq q$, $\mathcal{A}$ can query $K_1||0||w_i$ to the random oracle (i.e. $\mathcal{H}$) a priori (in other words, the probability that $\mathcal{A}$ can obtain its corresponding hash value k_j a priori), is negligible since $\mathcal{A}$ has no secret key and cannot infer it without querying on w_i.
- The probability that for any $1 \leq i \leq q$, $\mathcal{A}$ can query $K_1||1||w_i$ to $\mathcal{H}$ a priori (in other words, the probability that $\mathcal{A}$ can obtain its corresponding hash value r_j a priori), is negligible since $\mathcal{A}$ has no secret key and cannot infer it without querying on w_i.
- The probability that for any $1 \leq j \leq \delta$ and any $1 \leq i \leq N$, $\mathcal{A}$ can query $r_j||i$ to $\mathcal{H}$ a priori (in other words, the probability that $\mathcal{A}$ can obtain its corresponding hash value $r'_{j,i}$), is negligible since $\mathcal{A}$ cannot have r_j a priori for any $1 \leq j \leq \delta$ without querying on w_i.
- The probability that for any $1 \leq j \leq \delta$ and any $1 \leq i \leq N$, $\mathcal{A}$ can infer $r'_{j,i}$ from R_j, is negligible since $\mathcal{A}$ cannot have $\mathbf{D}(w_i)$ a priori for any $1 \leq i \leq q$ without querying on w_i.

From the above analysis, $\mathcal{A}$ and also any distinguisher $\mathcal{D}$ cannot distinguish (k_i, r_i) output by $\mathcal{S}$ from (key_i, val_i) output by **Real** for any $1 \leq i \leq \delta$. Thus, the modified scheme is adaptively secure in the random oracle model. □

5 Conclusion

In this paper, we have shown the Chase-Kamara encryption scheme which is obtained by simplifying the structured encryption schemes [13]. We have focused on the trapdoor size problem of the Chase-Kamara scheme, and proposed the modified scheme whose trapdoor size does not depend on the number of stored documents. The modified scheme is based on our multiple hashing technique which can transform a trapdoor of short length to that of long length. We have shown that the modified scheme is adaptively secure in the random oracle model.

A future work is to show that our modified scheme is adaptively secure from standard assumptions. We note that our modified scheme satisfies non-adaptive security if employed pseudo-random functions instead of hash functions in our modified scheme.

Acknowledgments. The authors would like to thank anonymous reviewers of ISPEC 2018 for their valuable comments.

References

1. Alderman, J., Martin, K.M., Renwick, S.L.: Multi-level access in searchable symmetric encryption. In: Brenner, M. (ed.) FC 2017. LNCS, vol. 10323, pp. 35–52. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70278-0_3
2. Asharov, G., Naor, M., Segev, G., Shahaf, I.: Searchable symmetric encryption: optimal locality in linear space via two-dimensional balanced allocations. In: STOC 2016 (2016)

3. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_30
4. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: FOCS 1997, pp. 394–403 (1997)
5. Boldyreva, A., Chenette, N.: Efficient fuzzy search on encrypted data. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 613–633. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46706-0_31
6. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_30
7. Bösch, C., Brinkman, R., Hartel, P., Jonker, W.: Conjunctive wildcard search over encrypted data. In: Jonker, W., Petković, M. (eds.) SDM 2011. LNCS, vol. 6933, pp. 114–127. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23556-6_8
8. Bost, R.: $\Sigma o \phi o \varsigma$ - forward secure searchable encryption. In: ACM CCS 2016, pp. 1143–1154 (2016)
9. Cash, D., et al.: Dynamic searchable encryption in very-large databases: data structures and implementation. In: NDSS 2014 (2014)
10. Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M.-C., Steiner, M.: Highly-scalable searchable symmetric encryption with support for boolean queries. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 353–373. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_20
11. Cash, D., Tessaro, S.: The locality of searchable symmetric encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 351–368. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_20
12. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005). https://doi.org/10.1007/11496137_30
13. Chase, M., Kamara, S.: Structured encryption and controlled disclosure. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 577–594. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_33
14. Chase, M., Shen, E.: Substring-searchable symmetric encryption. PETS 2015 **2015**(2), 263–281 (2015)
15. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: ACM CCS 2006, pp. 79–88 (2006)
16. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. J. Comput. Secur. **19**(5), 895–934 (2011)
17. Demertzis, I., Papamanthou, C.: Fast searchable encryption with tunable locality. In: ACM SIGMOD 2017, pp. 1053–1067 (2017)
18. Do, H.G., Ng, W.K.: Private boolean query processing on encrypted data. In: Lam, K.-Y., Chi, C.-H., Qing, S. (eds.) ICICS 2016. LNCS, vol. 9977, pp. 321–332. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50011-9_25
19. Dong, C., Russello, G., Dulay, N.: Shared and searchable encrypted data for untrusted servers. J. Comput. Secur. **19**(3), 367–397 (2011)
20. Etemad, M., Kupcu, A., Papamanthou, C.: Efficient dynamic searchable encryption with forward privacy. PETS 2018 **2018**(1), 5–20 (2018)

21. Faber, S., Jarecki, S., Krawczyk, H., Nguyen, Q., Rosu, M., Steiner, M.: Rich queries on encrypted data: beyond exact matches. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9327, pp. 123–145. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24177-7_7
22. Goh, E.-J.: Secure indexes. Cryptology ePrint Archive, Report 2003/216 (2003). http://eprint.iacr.org/2003/216
23. Hahn, F., Kerschbaum, F.: Searchable encryption with secure and efficient updates. In: ACM CCS 2014, pp. 310–320 (2014)
24. Hamlin, A., Shelat, A., Weiss, M., Wichs, D.: Multi-key searchable encryption, revisited. In: Abdalla, M., Dahab, R. (eds.) PKC 2018. LNCS, vol. 10769, pp. 95–124. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76578-5_4
25. Hayasaka, K., Kawai, Y., Koseki, Y., Hirano, T., Ohta, K., Iwamoto, M.: Probabilistic generation of trapdoors: reducing information leakage of searchable symmetric encryption. In: Foresti, S., Persiano, G. (eds.) CANS 2016. LNCS, vol. 10052, pp. 350–364. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48965-0_21
26. Hirano, T., et al.: Simple, secure, and efficient searchable symmetric encryption with multiple encrypted indexes. In: Ogawa, K., Yoshioka, K. (eds.) IWSEC 2016. LNCS, vol. 9836, pp. 91–110. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44524-3_6
27. Kamara, S., Moataz, T.: Boolean searchable symmetric encryption with worst-case sub-linear complexity. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10212, pp. 94–124. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56617-7_4
28. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In: ACM CCS 2012, pp. 965–976 (2012)
29. Kamara, S., Papamanthou, C.: Parallel and dynamic searchable symmetric encryption. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 258–274. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39884-1_22
30. Kissel, Z.A., Wang, J.: Generic adaptively secure searchable phrase encryption. PETS 2017 **2017**(1), 4–20 (2017)
31. Kurosawa, K.: Garbled searchable symmetric encryption. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 234–251. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45472-5_15
32. Kurosawa, K., Ohtaki, Y.: UC-secure searchable symmetric encryption. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 285–298. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32946-3_21
33. Kurosawa, K., Ohtaki, Y.: How to update documents *Verifiably* in searchable symmetric encryption. In: Abdalla, M., Nita-Rotaru, C., Dahab, R. (eds.) CANS 2013. LNCS, vol. 8257, pp. 309–328. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-02937-5_17
34. Kuzu, M., Islam, M.S., Kantarcioglu, M.: Efficient similarity search over encrypted data. In: IEEE ICDE 2012, pp. 1156–1167 (2012)
35. Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., Lou, W.: Fuzzy keyword search over encrypted data in cloud computing. In: IEEE INFOCOM 2010 (Mini-Conference), pp. 1–5 (2010)
36. Miyoshi, R., Yamamoto, H., Fujiwara, H., Miyazaki, T.: Practical and secure searchable symmetric encryption with a small index. In: Lipmaa, H., Mitrokotsa, A., Matulevicius, R. (eds.) NordSec 2017. LNCS, vol. 10674, pp. 53–69. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70290-2_4

37. Moataz, T., Shikfa, A.: Boolean symmetric searchable encryption. In: ASIACCS 2013, pp. 265–276 (2013)
38. Naveed, M., Prabhakaran, M., Gunter, C.A.: Dynamic searchable encryption via blind storage. In: IEEE S&P 2014, pp. 639–654 (2014)
39. Ogata, W., Koiwa, K., Kanaoka, A., Matsuo, S.: Toward practical searchable symmetric encryption. In: Sakiyama, K., Terada, M. (eds.) IWSEC 2013. LNCS, vol. 8231, pp. 151–167. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41383-4_10
40. Ogata, W., Kurosawa, K.: Efficient no-dictionary verifiable searchable symmetric encryption. In: Kiayias, A. (ed.) FC 2017. LNCS, vol. 10322, pp. 498–516. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70972-7_28
41. Shen, Y., Zhang, P.: Ranked searchable symmetric encryption supporting conjunctive queries. In: Liu, J.K., Samarati, P. (eds.) ISPEC 2017. LNCS, vol. 10701, pp. 350–360. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-72359-4_20
42. Song, D., Wagner, D., Perrig, A.: Practical techniques for searching on encrypted data. In: IEEE S&P 2000, pp. 44–55 (2000)
43. Stefanov, E., Papamanthou, C., Shi, E.: Practical dynamic searchable encryption with small leakage. In: NDSS 2014 (2014)
44. Taketani, S., Ogata, W.: Improvement of UC secure searchable symmetric encryption scheme. In: Tanaka, K., Suga, Y. (eds.) IWSEC 2015. LNCS, vol. 9241, pp. 135–152. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22425-1_9
45. van Liesdonk, P., Sedghi, S., Doumen, J., Hartel, P., Jonker, W.: Computationally efficient searchable symmetric encryption. In: Jonker, W., Petković, M. (eds.) SDM 2010. LNCS, vol. 6358, pp. 87–100. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15546-8_7
46. Wang, C., Ren, K., Yu, S., Urs, K.M.R.: Achieving usable and privacy-assured similarity search over outsourced cloud data. In: IEEE INFOCOM 2012, pp. 451–459 (2012). https://doi.org/10.1109/INFCOM.2012.6195784
47. Xu, P., Liang, S., Wang, W., Susilo, W., Wu, Q., Jin, H.: Dynamic searchable symmetric encryption with physical deletion and small leakage. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017. LNCS, vol. 10342, pp. 207–226. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60055-0_11
48. Yang, Y.J., Ding, X.H., Deng, R.H., Bao, F.: Multi-user private queries over encrypted databases. Int. J. Appl. Crypt. **1**(4), 309–319 (2009)
49. Yavuz, A.A., Guajardo, J.: Dynamic searchable symmetric encryption with minimal leakage and efficient updates on commodity hardware. In: Dunkelman, O., Keliher, L. (eds.) SAC 2015. LNCS, vol. 9566, pp. 241–259. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31301-6_15