Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 153791, 16 pages
http://dx.doi.org/10.1155/2014/153791

Research Article

A Layered Searchable Encryption Scheme with Functional Components Independent of Encryption Methods

Guangchun Luo, Ningduo Peng, Ke Qin, and Aiguo Chen

School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China

Correspondence should be addressed to Ningduo Peng; nindo_academia@163.com

Received 18 October 2013; Accepted 8 January 2014; Published 25 February 2014

Academic Editors: H.-E. Tseng and G. Wei

Searchable encryption technique enables the users to securely store and search their documents over the remote semitrusted server, which is especially suitable for protecting sensitive data in the cloud. However, various settings (based on symmetric or asymmetric encryption) and functionalities (ranked keyword query, range query, phrase query, etc.) are often realized by different methods with different searchable structures that are generally not compatible with each other, which limits the scope of application and hinders the functional extensions. We prove that asymmetric searchable structure could be converted to symmetric structure, and functions could be modeled separately apart from the core searchable structure. Based on this observation, we propose a layered searchable encryption (LSE) scheme, which provides compatibility, flexibility, and security for various settings and functionalities. In this scheme, the outputs of the core searchable component based on either symmetric or asymmetric setting are converted to some uniform mappings, which are then transmitted to loosely coupled functional components to further filter the results. In such a way, all functional components could directly support both symmetric and asymmetric settings. Based on LSE, we propose two representative and novel constructions for ranked keyword query (previously only available in symmetric scheme) and range query (previously only available in asymmetric scheme).

1. Introduction

Cloud storage provides an elastic, highly available, easily accessible, and cheap repository to users to store and use their data, and such a convenient way attracts more and more people. In many cases, the users require their sensitive data, such as business documents, to be secure against any adversary or even the cloud provider, and therefore all data must be encrypted before sending to the server [1]. However, traditional encryption schemes (e.g., DES) do not provide any functionality to the users such that searching for the desired documents by keywords, as a basic function for storage system, is quite impossible. The problem is that there is no way to know if there exists such keywords in an encrypted document without decryption, and apparently the server should not have the decryption key.

Searchable encryption technique provides a solution to such problem. It enables the users to encrypt their sensitive data and store it to the remote server, while retaining the ability to search by keywords. While searching, the user sends to the server a secret token (a transformation of the queried keywords); then the server uses the token to search over the encrypted data and returns the matched documents. During the process, the server does not know what the queried keywords and the document contents are, and therefore the privacy is guaranteed.

Many searchable encryption schemes have been proposed with various settings and functionalities. For symmetric searchable encryption schemes, the user encrypts, searches, and decrypts the documents using his/her private symmetric key. For asymmetric searchable encryption schemes, the data sender encrypts the documents using the user's public key, and the user searches and decrypts the documents using the private key. Beyond the basic keyword matching, many functions are also added to either symmetric or asymmetric setting, such as range query, phrase query, and fuzzy keyword query.

However, these functions are often realized by different methods with different searchable structures which are generally not compatible with each other. For example, the asymmetric encryption scheme introduced in [2] realized conjunctive, subset, and range queries. However, it is difficult to figure out how to apply this method to symmetric setting. Even for the same setting, such as the fuzzy query scheme introduced in [3] and the rank-ordered query scheme introduced in [4], it is difficult to figure out how to combine two methods together since the functions are constructed based on different indexing structures.

Layered searchable encryption (LSE) scheme aims to provide compatibility, flexibility, and security for various settings and functionalities. In this new framework, keywords are firstly transformed to tokens that are filtered by the core searchable component (symmetric or asymmetric setting), and then the tokens are dynamically converted to uniform mappings which are transmitted to many standalone functional components (e.g., ranked keyword query component, fuzzy query component, etc.) to further filter the results. Since all functional components are independent of each other and the interfaces are common, the functions are compatible with each other and directly support both symmetric and asymmetric settings, and adding or deleting a function is quite simple since each function is loosely coupled with the core searchable component. Furthermore, LSE supports combined query. For example, the query "SELECT ∗ WHERE keywords = "cloud, storage, encryption" AND "security classification > 5" ORDERED BY "keyword:cloud"" (to express the query, we adopt the SQL-like format used in database) is a combination of three functional components: basic query, range query, and ranked keyword query (in this paper, we will present the concrete construction for this example).

Furthermore, this framework is similar to the data stream processing architecture [5], where functional components could be treated as operator boxes and the whole scheme could be treated as a data-flow system by which all processes follow the popular boxes and arrows paradigm. Therefore, in comparison to the previous searchable encryption schemes, LSE is more suitable for distributed and parallel computing environment.

In this paper, our contributions are the following. (1) We propose a novel framework for designing searchable encryption scheme called layered searchable encryption (LSE), which enables combined query and provides compatibility, flexibility, and security for various settings and functionalities. The new framework consists of a core searchable component with a symmetric/asymmetric converter, many functional components, and a common interface with new security model. (2) We propose a concrete construction for LSE that could theoretically combine all possible functionalities which are proposed in the recent years, and prove its semantic security for the interface. (3) As a complement for the prior works, we formally define two new security models for ranked keyword query and range query, called semantic security against chosen ranked keyword attack (CRKA) and chosen range attack (CRA) respectively, which provide integral security models for cryptographic analysis. (4) Based on LSE, we propose two representative and novel constructions for ranked keyword query component (previously only available in symmetric scheme) and range query component (previously only available in asymmetric scheme) and prove them semantically secure under the new security models.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 presents the notations and preliminaries. Section 4 presents the layered searchable encryption scheme and the concrete construction. Section 5 discusses how to realize various functionalities and presents the concrete constructions for ranked keyword query and range query. Section 6 concludes this paper.

2. Related Work

Searchable encryption schemes are designed to help the users to securely search over the encrypted data by keywords. The first scheme was introduced in [6] by Song et al., and later on many index-based symmetric searchable encryption (SSE) schemes were proposed. Goh introduced the first secure index in [7], and they also built the security model for searchable encryption called Adaptive Chosen Keyword Attack (IND-CKA). In [8], Curtmola et al. introduced two constructions to realize symmetric searchable encryption: the first construction (named SSE-1) is nonadaptive and the second one (named SSE-2) is adaptive. A generalization for symmetric searchable encryption was introduced in [9], and a representative SSE system designed by Microsoft was introduced in [10]. Another type of searchable encryption named asymmetric searchable encryption (ASE) is public-key based, which allows the user to search over the data encrypted by some data senders using the public key of the user. The first scheme was introduced in [11] by Boneh et al. based on bilinear maps, and the improved definition was introduced in [12].

There are many functional extensions for the searchable encryption schemes beyond the basic precise keyword matching. For symmetric setting, the authors in [4, 13, 14] introduced ranked keyword search schemes based on order-preserving encryption technique or two-round protocol, which allows the server to only return the top-k relevant results to the user. In [15], Golle et al. introduced a scheme supporting conjunctive keyword search which allows the user to search multiple keywords in a single query. In [3, 16, 17], the authors introduced fuzzy keyword search schemes based on wildcard technique, which allows the user to submit only part of the precise keyword. Similar to fuzzy keyword search but different, the authors in [18, 19] introduced similarity search schemes based on wildcard technique, which allows the server to return the results similar to the queried keyword. In [20, 21], the authors introduced phrase query schemes based on trusted client-side server or binary search, which allows the user to query a phrase instead of multiple independent keywords. For asymmetric setting, the authors in [22, 23] introduced range query schemes. In addition, Boneh et al. also introduced conjunctive and subset query in [22] based on bilinear maps.

Note that most of these techniques are not compatible with each other due to specific data structure and mathematical property. However, in the following sections, we will prove that functional structures and searchable structures could be separately constructed, and asymmetric structures could be converted to symmetric structures such that a compatible all-in-one scheme is possible.

3. Notations and Preliminaries

We write $x \leftarrow_U X$ to denote sampling element x uniformly random from a set X and write $x \leftarrow \mathscr{A}$ to denote the output of an algorithm $\mathscr{A}$. We write $a \parallel b$ to denote the concatenation of two strings a and b. We write $|A|$ to denote its cardinality if A is a set and write $|a|$ to denote its bit length if a is a string. A function $\mu(k) : \mathbb{N} \rightarrow \mathbb{R}$ is negligible, if for every positive polynomial $p(\cdot)$ there exists an inter $N > 0$ such that for all $k > N$, $|\mu(k)| < 1/p(k)$. We write poly(k) and negl(k) to denote polynomial and negligible functions in k, respectively.

We write $\Delta = (w_1, \dots, w_n)$ to denote a dictionary of n words in lexicographic order. We assume that all words are of length polynomial in k. We write d to refer to a document that contains poly(k) words and write $|d|$ to denote the size of the document in bytes. In some cases, we also write d to denote the document identifier that uniquely identifies the document, such as a memory location. We write `X` to denote a component or a scheme and write `X.func(...)` to denote the corresponding function for the component or an algorithm in the scheme.

4. Layered Searchable Encryption Scheme

Layered searchable encryption scheme aims to combine symmetric and asymmetric searchable encryption schemes to provide a uniform model for functional extensions. Therefore, we first revisit the basic symmetric and asymmetric searchable encryption models and then build the layered searchable encryption model based on these two different settings. After that, we introduce the security model of the new framework, and finally we present the concrete construction.

4.1. Revisiting Searchable Encryption. We adopt the definition introduced by Curtmola et al. in [8] as a representative model for symmetric searchable encryption scheme. In this setting, the user who searches for the documents is also the data sender who encrypts the documents. Therefore, some efficient searching techniques, such as using a global index, are used and the searchable structure may be a single index file for all stored documents. For consistency with other definitions, we make a little modification for the original definition, and define the scheme as follows.

Definition 1 (symmetric searchable encryption). A symmetric searchable encryption (SSE) scheme is a collection of five polynomial-time algorithms `SSE` = (`Gen`, `Enc`, `Token`, `Search`, `Dec`) as follows.

> $K \leftarrow \texttt{Gen}(1^k)$ is a probabilistic algorithm that takes as input a security parameter k and outputs a secret key K. It is run by the user and the key is kept secret.
>
> $(\gamma, C) \leftarrow \texttt{Enc}(K, D)$ is a probabilistic algorithm that takes as input a secret key K and a document collection $D = (d_1, \dots, d_n)$ and outputs a searchable structure γ and a sequence of encrypted documents $C = (c_1, \dots, c_n)$. It enables a user to query some keywords and the server returns the matched documents. For instance, in an index-based symmetric searchable encryption scheme, γ is the secure index. It is run by the user and (γ, C) is sent to the server.
>
> $t \leftarrow \texttt{Token}(K, w)$ is a deterministic algorithm that takes as input a secret key K and a keyword w and outputs a search token t (also named trapdoor or capacity). It is run by the user.
>
> $C' \leftarrow \texttt{Enc}(C, \gamma, t)$ is a deterministic algorithm that takes as input the encrypted documents C, the searchable structure γ, and the search token t and outputs the matched documents (or identifiers) $C' = (c_1, \dots, c_m)$. It is run by the server and C' is sent to the user.
>
> $d \leftarrow \texttt{Dec}(K, c)$ is a deterministic algorithm that takes as input a secret key K and the encrypted document c and outputs the recovered plaintext d. It is run by the user.

We adopt the definition introduced by Boneh et al. in [11] as a representative model for asymmetric searchable encryption scheme. In this setting, the user generates the public key and the private key. The data sender encrypts the data using the public key, and the user searches and decrypts the data using the private key. The original definition only contains the searchable part, for consistency; we add two algorithms and define the asymmetric searchable encryption as follows.

Definition 2 (asymmetric searchable encryption). An asymmetric searchable encryption (ASE) scheme is a collection of seven polynomial-time algorithms `ASE` = (`Gen`, `PEKS`, `Enc`, `Token`, `Test`, `Search`, `Dec`) as follows.

> $K \leftarrow \texttt{Gen}(1^k)$ is a probabilistic algorithm that takes as input a security parameter k and outputs a public/private key pair $K = (K_{\text{pub}}, K_{\text{priv}})$. It is run by the user and only K_{priv} is kept secret.
>
> $s \leftarrow \texttt{PEKS}(K_{\text{pub}}; w)$ is a probabilistic algorithm that takes as input a public key K_{pub} and a word w and outputs a searchable structure s. It is run by the data sender and s is attached to the encrypted message, and the combination is sent to the server.
>
> $c \leftarrow \texttt{Enc}(K_{\text{pub}}; d)$ is a probabilistic algorithm that takes as input a public key K_{pub} and a document

(message) and outputs the ciphertext c. It is run by the data sender and c (followed by multiple searchable structures) is sent to the server.

$t \leftarrow \texttt{Token}(K_{\text{priv}}; w)$ is a deterministic algorithm that takes as input a private key K_{priv} and a keyword w and outputs a search token t. It is run by the user.

$b \leftarrow \texttt{Test}(K_{\text{pub}}; s; t)$ is a deterministic algorithm that takes as input the public key K_{pub}, a searchable structure $s \leftarrow \texttt{PEKS}(K_{\text{pub}}, w')$, and a search token $t \leftarrow \texttt{Token}(K_{\text{priv}}, w)$ and outputs $b = 1$ if $w = w'$ or $b = 0$ otherwise. It is run by the server.

$C' \leftarrow \texttt{Search}(K_{\text{pub}}; C; S; t)$ is a deterministic algorithm that takes as input the public key K_{pub}, the encrypted documents $C = (C_1, \ldots, C_n)$, the corresponding searchable structure set $S = (S_1, \ldots, S_n)$ (each S_i contains multiple searchable structures corresponding to the keywords of the document) and the search token t and outputs the matched documents $C' = (c_1, \ldots, c_m)$ (the documents' searchable structures satisfying $1 \leftarrow \texttt{Test}(K_{\text{pub}}, s, t)$). It is run by the server and C' is sent to the user.

$d \leftarrow \texttt{Dec}(K_{\text{priv}}; c)$ is a probabilistic algorithm that takes as input a private key K_{priv} and a ciphertext c and outputs the plaintext d. It is run by the user.

Unlike symmetric setting, the definition of asymmetric setting only works on a single document. For a document collection, it does not make any difference since the user could execute the encryption algorithm for each document, respectively.

By comparing the definitions of the two different settings, there exists a common link between the queried keywords and the matched documents: the searchable structure which is constructed using either symmetric key or the public key. Note that the structure is probabilistic in the asymmetric setting, or else the server could directly launch the chosen plaintext attack using the public key. However, we say that for symmetric and asymmetric settings, the searchable structures are both run-time deterministic. To prove this property, we first introduce a lemma as follows.

Lemma 3. *For asymmetric setting, if the token t generated using the private key K_{priv} is deterministic, then the searchable structure s encrypted using the public key K_{pub} is run-time deterministic when the the algorithm ASE.Test outputs 1, even if the encryption is probabilistic.*

Proof. Recall that the algorithm $t \leftarrow \texttt{Token}(K_{\text{priv}}, w)$ is deterministic and $s \leftarrow \texttt{PEKS}(K_{\text{pub}}, w)$ is probabilistic. However, for a single document, there only exists a single s that links to w. When $1 \leftarrow \texttt{Test}(K_{\text{pub}}, s, t)$, it implies that t matches s. We replace t with s; then the token $t = s$, which could be generated by the data sender, who could generate the token using the public key $t \leftarrow \texttt{Token}'(K_{\text{pub}}, w)$ which is in fact the algorithm $\texttt{PEKS}(K_{\text{pub}}, w)$. It seems that the data sender has indirectly generated the token without having the private key. Therefore, when the output of the test is 1, both the token and the searchable structure map to t, which is deterministic. □

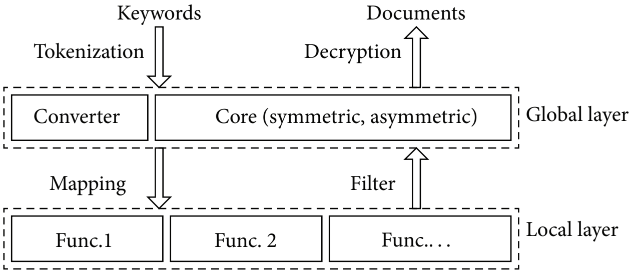


FIGURE 1: Architecture for layered searchable encryption scheme.

Based on the lemma above, we introduce a theorem which guides us to construct the converter in the layered searchable encryption scheme.

Theorem 4 (run-time invariance). *For both symmetric and asymmetric settings, if the search token t is deterministic, then the searchable structure is run-time deterministic.*

Proof. As proved in Lemma 3, the searchable structure is run-time deterministic for asymmetric setting. For symmetric setting, the searchable structure is encrypted using the symmetric key $\gamma \leftarrow \texttt{Enc}(K, D)$, which is probabilistic. The token $t \leftarrow \texttt{Token}(K, w)$ is deterministic. Similar to asymmetric setting, when executing the deterministic algorithm $\texttt{Search}$, the matched entries (probabilistic) map to t, and the mapping is deterministic (here, an entry is the encrypted data using symmetric encryption that contains the information about the matched document, such as the node in the inverted index [8]). In other words, the searchable structure is run-time deterministic because of the deterministic mapping. □

4.2. Scheme Definition. Our primary goal is to separate the functionalities from the searchable structures; therefore we consider to construct the basic searchable structures and various functions in different layers, as shown in Figure 1.

(i) Global layer: we name this layer "global" because all documents and all searchable structures are involved. In this layer, the basic searchable encryption scheme (symmetric or asymmetric) is executed and a global index could be constructed to improve search efficiency. The server receives the search tokens (each token is related to a keyword), executes the search procedure, and outputs the matched documents. Furthermore, the server converts the tokens (symmetric or asymmetric) to the corresponding mappings (another type of secret token) with uniform format and transfers the mappings with the matched documents (or identifiers) to the local layer.

(ii) Local layer: we name this layer "local" because functional structures are constructed for each document independently. In this layer, each matched document is further filtered by all functions (e.g., phrase query function) which execute separately. Only the documents that pass all filter tests are returned to the global layer and finally return to the user.

For both layers, the framework consists of three different components: the core symmetric and asymmetric searchable components which provide basic keyword search, one or more functional components which provides various functionalities, and a converter. The converter is an algorithm that provides a uniform interface for both symmetric and asymmetric settings and provides uniform inputs for all functions. We note that all components in the two layers execute the search algorithm on the server side, and no trusted third-party is required. Now we formally define the scheme as follows.

Definition 5 (layered searchable encryption). A layered searchable encryption (LSE) scheme is a collection of five polynomial-time algorithms $\texttt{LSE} = (\texttt{Gen}, \texttt{Enc}, \texttt{Token}, \texttt{Search}, \texttt{Dec})$ as follows.

$K \leftarrow \texttt{Gen}(1^k)$ is a probabilistic algorithm that takes as input a security parameter k and outputs either a symmetric encryption key $K = K_{\text{priv}}$ or an asymmetric encryption key pair $K = (K_{\text{pub}}, K_{\text{priv}})$. It is run by the user and only the public key K_{pub} is not kept secret.

$(C; G; L) \leftarrow \texttt{Enc}(K_e; D)$ is a probabilistic algorithm that takes as input an encryption key K_e ($K_e = K_{\text{priv}}$ for symmetric setting or $K_e = K_{\text{pub}}$ for asymmetric setting) and a document collection $D = (d_1, \ldots, d_n)$. It outputs n encrypted documents $C = (c_1, \ldots, c_n)$, a single (index-based) global searchable structure G or a sequence of global searchable structures $G = (G_1, \ldots, G_n)$ corresponding to n documents, and a sequence of local functional structures $L = (L_1, \ldots, L_n)$ corresponding to n encrypted documents. It is run by the data sender and (C, G, L) are sent to the server.

$T \leftarrow \texttt{Token}(K_{\text{priv}}, w)$ is a deterministic algorithm that takes as input a secret key K_{priv} and a set of keywords $W = (w_1, \ldots, w_o)$ with functional instructions and outputs the corresponding search tokens $T = (t_1, \ldots, t_o)$ with functional instructions. It is run by the user and T is sent to the server.

$C' \leftarrow \texttt{Search}(K_{\text{pub}}; C; G; L; T)$ is a deterministic algorithm that takes as input a public key K_{pub} (only for asymmetric setting), the encrypted documents C, the global searchable structure G, the local functional structure L, and the search token T and outputs the matched documents $C' = (c_1, \ldots, c_m)$. It is run by the server and C' is sent to the user.

$d \leftarrow \texttt{Dec}(K_{\text{priv}}, c)$ is a deterministic algorithm that takes as input a secret key K_{priv} and an encrypted document c, and outputs the plaintext d. It is run by the user.

Functional instructions are separately specified by the functionalities and are written as a single SQL-like query. For example, the query "SELECT $*$ WHERE keywords = "cloud, storage, encryption" AND "security classification > 5" ORDERED BY "keyword:cloud"" indicate that finding the documents that satisfying: containing the keywords "cloud, storage, encryption", the security classification of the documents > 5, sorting the matched documents by relevance score according to the keyword "cloud" and return the top-k relevant documents. Here we only write $W = (w_1, \ldots, w_o)$ (e.g., W = "cloud, storage, encryption") as a representation for any instruction that contains the keywords. Similarly, the tokens T are just a representation for all functional instructions.

A functional component (FC) is a module in LSE that provides a specific functionality. It generates a local functional structure L for each encrypted document and provides filter service while searching. FC is designed to be compatible with both symmetric and asymmetric settings. Therefore, a conversion for the document as well as the query is required. We formally define the FC as follows.

Definition 6 (functional component). A functional component (FC) is a collection of two polynomial-time algorithms $\texttt{FC} = (\texttt{Build}, \texttt{Filter})$ as follows.

$L_d \leftarrow \texttt{Build}(d; V_d)$ is an algorithm that takes as input a document d and the corresponding conversion V_d and outputs a functional structure L_d. It is run by the data sender and L_d is appended to the encrypted document.

$C' \leftarrow \texttt{Filter}(C; L; V_T)$ is an algorithm that takes as input the encrypted documents $C = (C_1, \ldots, C_x)$, the corresponding functional structure set $L = (L_1, \ldots, L_x)$, and the converted search tokens $V_T = (V_1, \ldots, V_x)$ and outputs a subset of documents C'. It is run by the server.

4.3. Security Model. The security of LSE relies on the algorithms used by the components. For example, if the symmetric searchable encryption scheme introduced in [8] is used as the core searchable component, then the core searchable structure guarantees that it is semantic secure against chosen keyword attack (CKA-secure). Similarly, the functional components have their individual security guarantees. Therefore, the whole LSE scheme does not have a uniform security model, and security models are built separately and each component could be analyzed independently. However, we could divide the security models into three parts: searchable component security, interface security, and functional component security. Searchable component security is guaranteed by the underlying core searchable encryption scheme. Therefore, we mainly discuss the other two security models.

The interface is common, and therefore the data that flow through the interface must be semantic secure. Informally speaking, it must guarantee that the adversary cannot distinguish the input and the output of each component from random strings. Semantic security against chosen plaintext attack (CPA) is very important for the interface, or else the security of some components will be correlated such that the loose coupling property is lost.

We first define the notion of plain trace, which is the direct information that could be captured from the data that flow through the interface.

Definition 7 (plain trace). Let $D = (d_1, \ldots, d_n)$ be a document collection. Let $N = (N_1, \ldots, N_n)$ (only for asymmetric setting) be a keyword-counter set where N_i is the number of keywords in d_i. Let the query history $W = (w_1, \ldots, w_p)$ be a sequence of queried keywords. Let the search pattern $\sigma(W)$ be a $p \times p$ binary matrix such that for $1 \leq i, j \leq p$, the ith row and jth column is 1 if $W_i = W_j$ and 0 otherwise. The plain trace $\pi(D, W) = (|d_1|, \ldots, |d_n|, N, \sigma(W))$.

Note that plain trace is different from the notion of trace introduced in [8] which further captures the logic links. We will explain the reason after the definition of the security model. We now present the security model for the interface.

Definition 8 (interface security against chosen plaintext attack, interface-CPA-secure). Let Σ be the layered searchable encryption scheme. Let $k \in \mathbb{N}$ be the security parameter. We consider the following probabilistic experiments where $\mathscr{A}$ is an adversary and $\mathscr{S}$ is a simulator.

> $\text{Real}_{\Sigma,\mathscr{A}}(k)$: the challenger runs $\texttt{Gen}(1^k)$ to generate the key $K = K_{\text{priv}}$ (symmetric) or $K = (K_{\text{priv}}, K_{\text{pub}})$ (asymmetric). The adversary $\mathscr{A}$ generates a document collection $D = (d_1, \ldots, d_n)$, a sequence of query $W = (w_1, \ldots, w_p)$, and receives $(C, G) \leftarrow \texttt{Enc}(K_e, D)$ and search tokens $T \leftarrow \texttt{Token}(K_{\text{priv}}, W)$ from the challenger. $\mathscr{A}$ generates a mapping V_T as the input for the functional component. Finally, $\mathscr{A}$ returns a bit b that is output by the experiment.
>
> $\text{Sim}_{\Sigma,\mathscr{A},\mathscr{S}}(k)$: given the plain trace $\pi(D, W)\mathscr{S}$ generates (C^*, G^*) and T^* and then sends the results to $\mathscr{A}$. $\mathscr{A}$ generates a mapping V_T^* as the input for the functional component. Finally, $\mathscr{A}$ returns a bit b that is output by the experiment.

We say that the interface of LSE is semantic secure against chosen plaintext attack, if for all PPT adversaries $\mathscr{A}$, there exists a PPT simulator $\mathscr{S}$ such that

$$\left|\Pr\left[\text{Real}_{\Sigma,\mathscr{A}}(k) = 1\right] - \Pr\left[\text{Sim}_{\Sigma,\mathscr{A},\mathscr{S}}(k) = 1\right]\right| \leq \text{negl}(k), \tag{1}$$

where the probabilities are over the coins of $\texttt{Gen}$.

Note that the functional structure L is not included here since the functional component is loosely coupled with the core. Therefore, the security of the functional component is separate from the framework and should be defined and analyzed separately.

The security model of the interface does not care about the search algorithm and the number of queries (therefore, only a single query sequence is presented). The reason is that the other information about the queried keywords and the documents are protected by the components. For example, if some documents are returned by one token, then the adversary could immediately infer that these documents have a common keyword (even the tokens and documents are indistinguishable from random in the interface), and such logic links could be hidden by generating multiple different tokens for one keyword (please refer to the adaptive construction in [8]) and the protection is guaranteed in the core searchable component.

Therefore, semantic security for the interface does not guarantee that the whole scheme is secure against chosen keyword attack or each component is secure under some other security models. However, it provides the basic security guarantee for the whole scheme and the independence for each component, and we will show such independence in the construction of the functional component later.

4.4. Concrete Construction. We first present the basic idea for the search process and the converter; then we present the template for constructing the functional component. Finally, we present the constructions for LSE (symmetric and asymmetric) in detail and prove the security of the interface.

4.4.1. Basic Idea. As shown in Figure 2, the basic search process is as follows. The user transforms his queried keywords W to tokens T using the private key. The server receives the tokens T and executes the search procedure over all encrypted documents $c_1, \ldots, c_n$. Each c_i $(1 \leq i \leq n)$ is linked to a global searchable structure G_i (if a global index is used, then only a single searchable structure G is used for all encrypted documents) and a local functional structure L_i, and only the global searchable structure $G/(G_1, \ldots, G_n)$ is used in this step. Then the tokens T are converted to the uniform tokens V_T, and both V_T and the matched x encrypted documents are transmitted to functional components $\text{FC}_1, \ldots, \text{FC}_e$ to further filter the results (e.g., phrase query filter). Each component outputs a subset of the input documents, and all components work serially since any document that does not pass the current filter will be unnecessary for the next filter. Finally, the matched encrypted documents $c_1, \ldots, c_m$ are returned to the user and the user decrypts them to obtain the plaintexts $d_1, \ldots, d_m$. In order to construct a functional component that supports both symmetric and asymmetric settings, a conversion is needed to transform the plaintext to a kind of ciphertext that is independent from the settings. We call this independent ciphertext as a one-to-one "mapping" since each word in the plaintext has a deterministic token in the ciphertext. In addition, in order to provide a uniform format for the functional components, a hash function is used, and we will show the detailed construction in the next section. Now we present the template for the functional component (FC) in Algorithm 1.

We note that, in order to obtain the loose coupling property, any specific parameter is not allowed. Therefore, the uniform mappings of the words become the ideal common parameter. Another advantage of the mapping is that the main information needed for any functionality is retained: the difference of each word and the order of all words in the document. Based on this information, the word frequency,

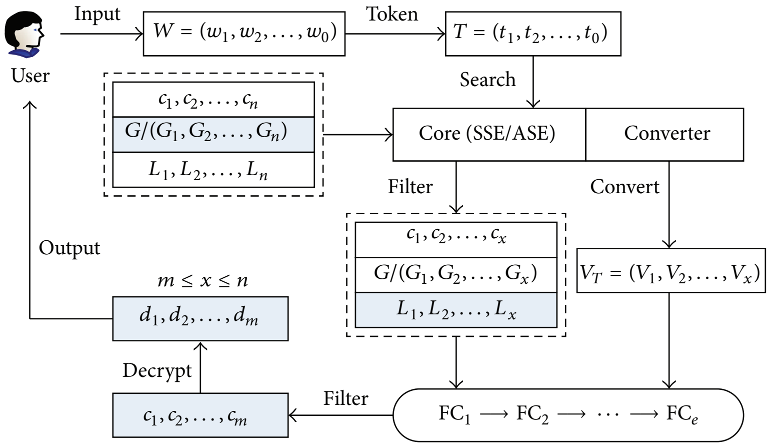

Input
$W = (w_1, w_2, \ldots, w_0)$
Token
$T = (t_1, t_2, \ldots, t_0)$
User
Search
$c_1, c_2, \ldots, c_n$
$G/(G_1, G_2, \ldots, G_n)$
$L_1, L_2, \ldots, L_n$
Core (SSE/ASE)
Converter
Filter
Convert
$c_1, c_2, \ldots, c_x$
$G/(G_1, G_2, \ldots, G_x)$
$L_1, L_2, \ldots, L_x$
$V_T = (V_1, V_2, \ldots, V_x)$
Output
$m \leq x \leq n$
$d_1, d_2, \ldots, d_m$
Decrypt
Filter
$c_1, c_2, \ldots, c_m$
$\mathrm{FC}_1 \longrightarrow \mathrm{FC}_2 \longrightarrow \cdots \longrightarrow \mathrm{FC}_e$

Figure 2: Search process of layered searchable encryption scheme.

```
Build($d, V_d$):
  (1) input a document $d$ and the mappings of all words $V_d$ in $d$.
  (2) specified according to the functionality.
  (3) output a local functional structure $L_d$.
Filter($C, L, V_T$):
  (1) input a set of encrypted document $C = (c_1, \ldots, c_x)$, the corresponding local functional
      structures $L = (L_1, \ldots, L_x)$, and the mappings of the queried keywords $V_T = (V_1, \ldots, V_x)$.
  (2) specified according to the functionality.
  (3) output a subset of the documents $C' \subseteq C$.
```

Algorithm 1: Template for functional component: FC.

rank, subset, and so forth could also be inferred without the plaintext, which facilitates the designs of the Token and Search algorithms.

4.4.2. Constructing Symmetric Part. For symmetric setting, the deterministic mapping of a document could be computed with ease. Let the tokens t_1, t_2, t_3 map to the words "day," "by," and "night," respectively. Then the deterministic mapping of a sentence could be written as

$$\text{“day by day, night by night”} \Longrightarrow t_1 t_2 t_1 t_3 t_2 t_3. \quad (2)$$

Both the Enc and Token algorithms could generate these mappings, and the main process is as follows. For each document d, scan all words and compute the corresponding tokens, which are further hashed to the fixed-size mappings.

Suppose there are n documents $D = (d_1, \ldots, d_n)$, n corresponding ciphertexts $C = (c_1, \ldots, c_n)$, e functional components $\mathrm{FC} = (F_1, \ldots, F_e)$, and o queried keywords $W = (w_1, \ldots, w_o)$. In addition, we define a hash function as follows:

$$f_h : \{0,1\}^* \longrightarrow \{0,1\}^l, \quad (3)$$

where l is the length of the mapping according to the hash function. For example, if we use MD5 [24] as f_h, then l is 128 bit. For clarity, we present the encryption scheme in Algorithm 2 and the search scheme in Algorithm 3 and finally present the complete scheme in Algorithm 4.

4.4.3. Constructing Asymmetric Part. For asymmetric setting, the data sender does not have the private key; therefore the mapping will fail while searching since any encryption using the public key is probabilistic (CPA security). For example, let e represent an encryption of a word, and the same sentence will become (note that both e_1 and e_3 map to the word "day")

$$\text{“day by day, night by night”} \Longrightarrow e_1 e_2 e_3 e_4 e_5 e_6. \quad (4)$$

Therefore, we delay the construction for such mapping after the construction of the searchable structure in algorithm Enc and use this searchable structure as an independent token for the corresponding word in algorithm Search. Recall that $s \leftarrow \mathtt{PEKS}(K_{\text{pub}}, w)$, then the tokens t_1, t_2, t_3 which map to the words "day," "by," and "night" will be transformed to s_1, s_2, s_3 when the test in the search algorithm outputs 1. Then we have

$$\text{“day by day, night by night”} \Longrightarrow s_1 s_2 s_1 s_3 s_2 s_3. \quad (5)$$

In this way, the data sender could construct the deterministic mapping for the document and indirectly obtain the deterministic tokens just using the public key. Similar to the symmetric setting, the process is as follows. For each document d, scan all words and compute the corresponding tokens according to searchable structures, which are further hashed

```
Input: the encryption key $K_e = K_{\text{priv}}$, the documents $D = (d_1, \ldots, d_n)$.
Output:
(1) $C$: encrypted documents $C = (c_1, \ldots, c_n)$.
(2) $G$: global searchable structure (index-based).
(3) $L$: local functional structures $L = (L_1, \ldots, L_n)$.
Method:
(1) compute $(\gamma, C) \leftarrow \texttt{SSE}\cdot\texttt{Enc}(K_e, D)$. Here $C = (c_1, \ldots, c_n)$.
(2) for each document $d_i \in D$ and the corresponding $c_i$ $(1 \le i \le n)$ do
(3)     scan $d_i$ for all $r$ words to form a word list $W = (w_1, \ldots, w_r)$.
(4)     for each word $w_k$ $(1 \le k \le r)$ in $W$ do
(5)         compute $t_k \leftarrow \texttt{SSE}\cdot\texttt{Token}(K_e, w_k)$.
(6)         compute $v_{ik} \leftarrow f_h(t_k)$.
(7)     end for
(8)     let $V_i = (v_{i1}, \ldots, v_{ir})$.
(9)     for each functional component $\texttt{FC}_\texttt{j}$ $(1 \le j \le e)$ do
(10)        compute $L_i^j \leftarrow \texttt{FC}_\texttt{j}\cdot\texttt{Build}(d_i, V_i)$.
(11)    end for
(12)    append $L_i = (L_i^1, \ldots, L_i^e)$ to $c_i$.
(13) end for
(14) let $G = \gamma$ and $L = (L_1, \ldots, L_n)$, and output $(C, G, L)$.
```

Algorithm 2: Encryption (symmetric): $\texttt{Enc}(K_e, D)$.

```
Input:
(1) $K_{\text{pub}}$: the public key is not available here.
(2) $C$: encrypted documents $C = (c_1, \ldots, c_n)$.
(3) $G$: global searchable structure (index-based).
(4) $L$: local functional structures $L = (L_1, \ldots, L_n)$.
(5) $T$: search tokens $T = (t_1, \ldots, t_o)$.
Output: matched documents $C' = (c_1, \ldots, c_m)$.
Method:
(1) compute $C' \leftarrow \texttt{SSE}\cdot\texttt{Search}(C, G, T)$. Here $C' = (c_1, \ldots, c_x)$.
(2) for each token $t_k$ $(1 \le k \le o)$, compute $v_k \leftarrow f_h(t_k)$.
(3) let $V_1 = V_2 = \cdots = V_x = (v_1, \ldots, v_o)$ and.
(4) for each functional component $\texttt{FC}_\texttt{j}$ $(1 \le j \le e)$ do
(5)     let $C' = (c_1, \ldots, c_x)$, then the corresponding $L' = (L_1, \ldots, L_x)$
        and $V_T = (V_1, \ldots, V_x)$. Let $L^j = (L_1^j, \ldots, L_x^j)$.
(6)     compute $C' \leftarrow \texttt{FC}_\texttt{j}\cdot\texttt{Filter}(C', L^j, V_T)$.
(7) end for
```

Algorithm 3: Search (symmetric): $\texttt{Search}(K_{\text{pub}}, C, G, L, T)$.

```
$\texttt{Gen}(1^k)$: compute $K = K_{\text{priv}} \leftarrow \texttt{SSE}\cdot\texttt{Gen}(1^k)$, and output $K$.
$\texttt{Enc}(K_e, D)$: described in Algorithm 2.
$\texttt{Token}(K_{\text{priv}}, W)$:
(1) for each keyword $w_k$ $(1 \le k \le o)$ in $W$ do
(2)     compute $t_k \leftarrow \texttt{SSE}\cdot\texttt{Token}(K_{\text{priv}}, w_k)$.
(3) end for
(4) output $T = (t_1, \ldots, t_o)$.
$\texttt{Search}(K_{\text{pub}}, C, G, L, T)$: described in Algorithm 3.
$\texttt{Dec}(K_{\text{priv}}, c)$: compute $d \leftarrow \texttt{SSE}\cdot\texttt{Dec}(K_{\text{priv}}, c)$, and output $d$.
```

Algorithm 4: LSE scheme: symmetric part.

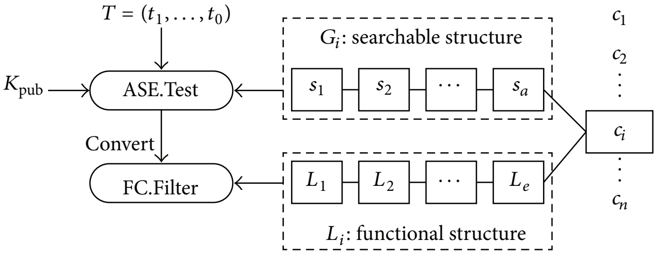


FIGURE 3: Data structure and search process for asymmetric setting.

TABLE 1: Searchable encryption schemes with various functionalities.

	Symm.	Asymm.	Ranked keyword	Range	Phrase	Fuzzy keyword	Similarity	Subset
Ranked keyword query [4, 13, 14, 25]	Yes	—	Yes	—	Possible	Possible	Possible	—
Range query [22, 23]	—	Yes	—	Yes	—	Possible	Possible	Possible
Phrase query [20, 21]	Yes	—	Possible	—	Yes	Possible	Possible	—
Fuzzy keyword query [3, 16, 17]	Yes	—	Possible	—	Possible	Yes	Possible	—
Wildcard query [26]	—	Yes	—	Possible	—	Yes	Possible	Possible
Similarity query [18, 19]	Yes	—	Possible	—	Possible	Possible	Yes	—
Subset query [22]	—	Yes	—	Possible	—	Possible	Possible	Yes
This paper	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

to the fixed-size mappings. While searching, the tokens are mapped to different searchable structures according to each document.

There are some differences from the symmetric counterpart, as shown in Figure 3. First, the searchable structures are appended to each encrypted data such that the global index is not available. Second, a public key is involved for the searchable structure. However, due to the conversion, the public key is unnecessary for the functional components.

Now we present the encryption scheme in Algorithm 5 and the search scheme in Algorithm 6 and finally present the complete scheme in Algorithm 7.

We note that the process of "find s" at line 5 in Algorithm 6 could be simply done by directly using the intermediate results from the algorithm `ASE.Search` at line 1.

4.4.4. Proof of Security. As we encapsulate the basic symmetric and asymmetric searchable encryptions in the global layer, the core is semantic secure against chosen keyword attack (CKA) [8, 9, 11]. The only thing we need is proving that the interface is CPA secure, and other functionalities are analyzed independently.

Theorem 9. *If the core symmetric or asymmetric component is semantic secure against chosen keyword attack (CKA-secure), then LSE is interface-CPA-secure.*

Proof. We briefly prove this theorem since the proof is straightforward. We claim that no polynomial-size distinguisher could distinguish (C, G, T, V_T) from equal-size random strings (C^*, G^*, T^*, V_T^*). As proved in [8, 11], the CKA-security of the core component guarantees that (C, G, T) are indistinguishable from (C^*, G^*, T^*). For symmetric setting, V_T is the hash of T which is indistinguishable from T^*. For asymmetric setting, V_T is the hash of the searchable structure S which is indistinguishable from random, say S^*. Therefore, the hash value V_T is indistinguishable from the hash value V_T^*. □

5. Realizing Various Functionalities

In this section, we show how to realize various functionalities based on LSE. We fist present the overview of the searchable encryption schemes with various functionalities and then propose two representative constructions for ranked keyword query and range query. Finally, we briefly discuss the methods for realizing the other functionalities.

5.1. Overview. As shown in Table 1, we present various functionalities for searchable encryption schemes: symmetric setting (Symm), asymmetric setting (Asym), ranked keyword query (Ranked keyword), range query (Range), phrase query (Phrase), fuzzy keyword query and wildcard query (Fuzzy keyword), similarity query (Similarity), and subset query

```
Input: encryption key $K_e = K_{\text{pub}}$, the documents $D = (d_1, \dots, d_n)$.
Output:
(1) $C$: encrypted documents $C = (c_1, \dots, c_n)$.
(2) $G$: global searchable structures $G = (G_1, \dots, G_n)$.
(3) $L$: local functional structures $L = (L_1, \dots, L_n)$.
Method:
for each document $d_i$ $(1 \le i \le n)$ in $D$ do
    compute $c_i \leftarrow$ ASE·Enc($K_{\text{pub}}, d_i$).
    scan $d_i$ for all $r$ words to form a word list $W = (w_1, \dots, w_r)$.
    extract $a$ distinct keywords $W' = (w_1, \dots, w_a)$ from $W$.
    for each word $w_x$ $(1 \le x \le a)$ in $W'$ do
        compute $s_{ix} \leftarrow$ ASE·PEKS($K_{\text{pub}}, w_x$).
        compute $h_{ix} \leftarrow f_h(s_{ix})$.
    end for
    let $G_i = (s_{i1}, \dots, s_{ia})$ map to $H_i = (h_{i1}, \dots, h_{ia})$ map to $W'$.
    for each word $w_y$ $(1 \le y \le r)$ in $W$ do
        find the $h \in H_i$ that the corresponding word $w_y \in W'$.
        set $v_{iy} = h$.
    end for
    let $V_i = (v_{i1}, \dots, v_{ir})$.
    for each functional component FC_j $(1 \le j \le e)$ do
        compute $L_i^j \leftarrow$ FC_j·Build($d_i, V_i$).
    end for
    append $L_i = (L_i^1, \dots, L_i^e)$ to $c_i$.
end for
output $C = (c_1, \dots, c_n)$, $G = (G_1, \dots, G_n)$, $L = (L_1, \dots, L_n)$.
```

Algorithm 5: Encryption (asymmetric): Enc(K_e, D).

```
Input:
(1) $K_{\text{pub}}$: the user's public key.
(2) $C$: encrypted documents.
(3) $G$: global searchable structures $G = (G_1, \dots, G_n)$.
(4) $L$: local functional structures $L = (L_1, \dots, L_n)$.
(5) $T$: the search tokens $T = (t_1, \dots, t_o)$.
Output: matched documents $C' = (c_1, \dots, c_m)$.
Method:
compute $C' \leftarrow$ASE·Search($K_{\text{pub}}, C, G, T$). Let $C' = (c_1, \dots, c_x)$.
for each $c_i \in C'$ and the functional structure $L_i$ $(1 \le i \le x)$ do
    let $G_i = (s_{i1}, \dots, s_{ia})$ denote the searchable encryptions of $c_i$,
    where $a$ is the number of keywords in $c_i$.
    for each $t_y$ $(1 \le y \le o)$ in $T$ do
        find $s \in G_i$ where ASE·Test($K_{\text{pub}}, s, t_y$) == yes.
        compute $v_{iy} \leftarrow f_h(s)$.
    end for
    let $V_i = (v_{i1}, \dots, v_{io})$, $L_i = (L_i^1, \dots, L_i^e)$.
end for
for each functional component FC_j $(1 \le j \le e)$ do
    let $C' = (c_1, \dots, c_x)$, then the corresponding $L' = (L_1, \dots, L_x)$
    and $V_T = (V_1, \dots, V_x)$. Let $L^j = (L_1^j, \dots, L_x^j)$.
    compute $C' \leftarrow$ FC_j·Filter($C', L^j, V_T$).
end for
```

Algorithm 6: Search (asymmetric): Search($K_{\text{pub}}, C, G, L, T$).

```
Gen($1^k$): compute $K = (K_{\text{pub}}, K_{\text{priv}}) \leftarrow$ ASE·Gen($1^k$), and output $K$.
Enc($K_e, D$): described in Algorithm 5.
Token($K_{\text{priv}}, W$):
for each keyword $w_k$ $(1 \leq k \leq o)$ in $W$ do
    compute $t_k \leftarrow$ ASE·Token($K_{\text{priv}}, w_k$).
end for
output $T = (t_1, \ldots, t_o)$.
Search($K_{\text{pub}}, C, G, L, T$): described in Algorithm 6.
Dec($K_{\text{priv}}, c$): compute $d \leftarrow$ ASE·Dec($K_{\text{priv}}, c$), and output $d$.
```

Algorithm 7: LSE scheme: asymmetric part.

(Subset). "Yes" means that the corresponding scheme directly supports such functionality. "Possible" means that the underlying data structure is compatible, and such functionality could be realized through minor modification of the original scheme. "—" means that realizing such functionality is quite challenging or the cost is relatively high.

5.2. Ranked Keyword Query. Ranked keyword query refers to a functionality that all matched documents are sorted according to some criteria, and only the top-k relevant documents will be returned to the user. The SQL query format is "ORDERED BY "keyword"." In [14], the authors introduced the computation for the relevance scores and proposed a comparing method over the encrypted scores based on order preserving symmetric encryption (OPSE) [27]. By using the same cryptographic primitive, the functional structure could record the encrypted relevance scores and setup an index with (token, score) pairs in order to obtain the score with $O(1)$ computation complexity.

5.2.1. Preliminaries. Order-preserving encryption (OPE) aims to encrypt the data in such a way that comparisons over the ciphertexts are possible. For $A, B \subseteq \mathbb{N}$, a function $f : A \rightarrow B$ is order-preserving if for all $i, j \in A$, $f(i) < f(j)$ if and only if $i < j$. We say an encryption scheme $\texttt{OPE} = (\texttt{Enc, Dec})$ is order-preserving if $\texttt{Enc}(K, \cdot)$ is an order-preserving function. In [28], Agrawal et al. proposed a representative OPE scheme that all numeric numbers are uniformly distributed. In [27], Boldyreva et al. introduced an order-preserving symmetric encryption scheme and proposed the security model. The improved definitions are introduced in [29]. Informally speaking, OPE is secure if the oracle access to OPE.Enc is indistinguishable from accessing to a random order-preserving function (ROPF). The security model is described as Pseudorandom Order-Preserving Function against Chosen Ciphertext Attack (POPF-CCA) [27].

A sparse look-up table is often managed by indirect addressing technique. Indirect addressing is also called FKS dictionary [30], which is used in symmetric searchable encryption scheme [8]. The addressing format is address, value, where the address is a virtual address that could locate the value field. Given the address, the algorithm will return the associated value in constant look-up time and return otherwise.

5.2.2. Construction. We build a sparse look-up table A that records the pair (keyword, relevance score) with all data encrypted. When queried, the server searches the relevance scores of all documents and finds the top-k relevant documents. Note that, in order to security use OPE scheme to encrypt relevance scores, a preprocessing is necessary.

We build an OPE table to preprocess all plaintexts and store the encrypted relevance scores as follows. Given a document collection $D = (d_1, \ldots, d_n)$. For each document d_k $(1 \leq k \leq n)$, scan it for o^k keywords. Compute the relevance score (based on word frequency) s_i^k $(1 \leq i \leq o^k)$ for each keyword $w_i^k \in W$ in d_k, and record a $o^k \times 3$ matrix for d_k with the ith line recording $R_i^k = (w_i^k, s_i^k, p_i^k)$, where p_i^k is the position where the first w_i^k occurs. For all documents, setup the OPE with $N = o^1 + o^2 + \cdots + o^n$ numbers $(s_1, \ldots, s_N)$. For each number s_j $(1 \leq j \leq N)$, the encryption is e_j. Transform the previous matrix to an OPE table with the ith line recording $R_i^k = (w_i^k, e_i^k, p_i^k)$ where e_i^k is the encryption of s_i^k.

For a document, it has at most $|d|/2 + 1$ keywords (note that each keyword is followed by a separator such as a blank). The look-up table is padded to $|d|/2 + 1$ entries in order to achieve semantic security. Now we present the concrete construction for ranked keyword query component in Algorithm 8.

5.2.3. Proof of Security. Informally speaking, the functional component must guarantee that given two documents' collection D_1, D_2 with equal size and $|D_1| = |D_2|$ then the challenger flips a coin b and encrypts D_b using LSE (the order of the ciphertexts are randomized). The adversary could query a keyword and receive the ordered document collection but he could not distinguish which one the challenger selected. By combining the security models defined in [8, 27], we formally define the notion of non-adaptive chosen ranked keyword attack (CRKA) as follows.

Definition 10 (semantic security against nonadaptive chosen ranked keyword attack, CRKA-secure). Let Σ be the functional component for ranked keyword query. Let $k \in \mathbb{N}$ be

```
Build$(d, V_d)$:
(1) input a document $d$ and the mapping $V_d = (v_1, \ldots, v_r)$ for $r$ words.
(2) let the entries of $d$ in OPE table be $((w_1, e_1, p_1), \ldots, (w_o, e_o, p_o))$.
(3) for each $i \in [1, o]$, build index $A[v_{p_i}] = e_i$.
(4) padding the remaining $|d|/2 + 1 - o$ entries with random strings.
(5) output a local functional structure $L_d = A$.
Filter$(C, L, V_T)$:
(1) input $n$ ciphertexts $C = (c_1, \ldots, c_n)$, the corresponding functional structures
    $L = (L_1, \ldots, L_n)$ and the mappings of the queried keywords
    $V_T = (V_1, \ldots, V_n) = (v_1, \ldots, v_n)$ (single keyword).
(2) for all $n$ functional structures, compute $r_1 = L_1[v_1], \ldots, r_n = L_n[v_n]$
    and select the top $k$ results $c_1, \ldots, c_k$ corresponding to $r_1, \ldots, r_k$.
(3) output $C' = (c_1, \ldots, c_k)$.
```

Algorithm 8: Ranked keyword query component.

the security parameter. We consider the following probabilistic experiments, where $\mathscr{A}$ is an adversary and $\mathscr{S}$ is a simulator.

> $\mathrm{Real}_{\Sigma,\mathscr{A}}(k)$: the challenger runs $\texttt{Gen}(1^k)$ to generate the key K. The adversary $\mathscr{A}$ generates a document collection $D = (d_1, \ldots, d_n)$ (the size of each document is fixed) and receives the encrypted documents $C = (c_1, \ldots, c_n)$ and functional structures $L = (L_1, \ldots, L_n)$ with random order from the challenger. $\mathscr{A}$ is allowed to query a keyword w, where $w \in d_1, \ldots, w \in d_n$ and receives a mapping v from the challenger. Finally, $\mathscr{A}$ returns a bit b that is output by the experiment.
>
> $\mathrm{Sim}_{\Sigma,\mathscr{A},\mathscr{S}}(k)$: given the number of documents n, the size of each document $|d|$, and the size of the mapping $|v|$, $\mathscr{S}$ generates C^*, L^*, and v^* and then sends the results to $\mathscr{A}$. Finally, $\mathscr{A}$ returns a bit b that is output by the experiment.

We say that the functional component is CRKA-secure, if for all PPT adversaries $\mathscr{A}$, there exists a PPT simulator $\mathscr{S}$ such that

$$\left|\Pr\left[\mathrm{Real}_{\Sigma,\mathscr{A}}(k) = 1\right] - \Pr\left[\mathrm{Sim}_{\Sigma,\mathscr{A},\mathscr{S}}(k) = 1\right]\right| \leq \mathrm{negl}(k), \tag{6}$$

where the probabilities are over the coins of $\texttt{Gen}$.

Theorem 11. *If LSE is interface-CPA-secure and the underlying OPE is POPF-CCA secure, then the ranked keyword query component is CRKA-secure.*

Proof. The simulator $\mathscr{S}$ generates C^*, L^*, and v^* as follows. As to C^*, $\mathscr{S}$ generates n random strings $c_1^*, \ldots, c_n^*$ of size $|d|$. As to L^*, let $m = |d|/2+1$; $\mathscr{S}$ generates m random strings $V^* = v_1^*, \ldots, v_m^*$ with each has size $|v|$. $\mathscr{S}$ generates an $m \times n$ matrix $E_{m\times n} = (e_{ij}^*)$, where each element e_{ij}^* is a random number. Then for each document, $\mathscr{S}$ generates an index $A_j^*[v_i^*] = e_{ij}^*$ $(1 \leq i \leq m,\ 1 \leq j \leq n)$. As to v^*, $\mathscr{S}$ randomly selects $v^* = v_i^* \in V^*$.

We claim that no polynomial-size distinguisher could distinguish (C, L, v) from (C^*, L^*, v^*). Since the encryption key K is kept secret from the adversary, the interface-CPA-security directly guarantees that C^* is indistinguishable from C. It also guarantees that v^* is indistinguishable from v. Upon receiving $v = v_i \in V$ or $v^* = v_i^* \in V^*$, the adversary $\mathscr{A}$ could invoke $\texttt{Filter}(C, L, v)$ or $\texttt{Filter}(C^*, L^*, v^*)$ to obtain $(r_1 = L_1[v_i] = e_1, \ldots, r_n = L_n[v_i] = e_n)$ or $(r_1^* = L_1^*[v_i^*] = e_1^*, \ldots, r_n^* = L_n^*[v_i^*] = e_n^*)$. POPF-CCA security guarantees that the set $(r_1, \ldots, r_n)$ is indistinguishable from $(r_1^*, \ldots, r_n^*)$; that is, the adversary is unable to distinguish the result of OPE from the result of a random order-preserving function. Therefore, L is indistinguishable from L^*. □

5.3. Range Query. Range query refers to a functionality that the server could test if the submitted keyword (integer) is within a range. The SQL query format is "WHERE '*x* operator *y*'." For example, the user submits an integer w, and the server could return the documents where the corresponding searchable fields a satisfying $a > w$.

Although OPE could be applied here to support range query (similar to ranked keyword query), we propose another solution to demonstrate that how to apply the methods used in asymmetric setting to LSE. In [2], the authors introduced a construction based on bilinear map (asymmetric setting), which is not compatible with symmetric setting. However, the idea of transforming the comparison into a predicate (e.g., $P_a(w) = 1$ if $a > w$ where P is a predicate) could be used, and the functional structure could record all possible predicates and provide predicate test using a bloom filter.

5.3.1. Preliminaries. A bloom filter [31] is a space-efficient probabilistic data structure that is used to test whether an element s is a member of a set $S = (s_1, \ldots, s_n)$. The set S is coded as an array B of m bits. Initially, all array bits are set to 0. The filter uses r independent hash functions $h_1, \ldots, h_r$ where each $h_i : \{0,1\}^* \rightarrow [1, m]$ for $1 \leq i \leq r$. For each element $s_k \in S$ where $1 \leq k \leq n$, set the bits at positions $h_1(s_k), \ldots, h_r(s_k)$ to 1. Note that, a location could be set to 1 multiple times. To determine if $s \in S$, just check whether the positions $h_1(s), \ldots, h_r(s)$ in B are all 1. If any bit is 0, then $s \notin S$. Otherwise, we say $s \in S$ with high probability (the probability could be adjusted by parameters until acceptable).

```
Build($d, V_d$):
input a range document $d = (> a_1, \geq a_1, \ldots, > a_{i-1}, \geq a_{i-1}, \geq a_i, = a_i, \leq a_i, < a_{i+1}, \leq a_{i+1}, \ldots, < a_N, \leq a_N)$
    and the mapping $V_d = (v_1, \ldots, v_{2N+1})$. Here $d$ is the transformed form for the label $a = a_i$.
initialize a bloom filter $B$ with all bits set to 0.
for ($k = 1; k \leq 2N + 1; k + +$) do
    compute $r$ codewords $y_1 = h_1(\overline{d} \parallel v_k), \ldots, y_r = h_r(\overline{d} \parallel v_k)$.
    insert the codewords $y_1, \ldots, y_r$ into the bloom filter $B$.
end for
output a local functional structure $L_d = B$.
Filter($C, L, V_T$):
input $n$ ciphertexts $C = (c_1, \ldots, c_n)$, the corresponding functional structures $L = (L_1, \ldots, L_n)$,
    the mappings of the queried keywords $V_T = (V_1, \ldots, V_n) = (v_1, \ldots, v_n)$ (single keyword),
    where $v_i (1 \leq i \leq n)$ is the mapping of "$> w$".
for ($i = 1; i \leq n; i + +$) do
    compute $r$ codewords $y_1 = h_1(\overline{d} \parallel v_i), \ldots, y_r = h_r(\overline{d} \parallel v_i)$.
    if all $r$ locations $y_1, \ldots, y_r$ in bloom filter $B_i = L_i$ are 1, then add $c_i$ to $C'$.
end for
output $C'$.
```

Algorithm 9: Range query component.

In addition, we write $\overline{d}$ to denote the identifier of a document d such as the cryptographic hash of the pathname, and write $x > (y_1, \ldots, y_n)$ to denote $x > y_1, \ldots, x > y_n$ for simplicity.

5.3.2. Construction. For range query, the document d is labeled by some numbers. Here we only consider a single label a. Therefore, the aim of range query is to enable the user to submit a number w to search for the documents that satisfying the SQL-like query such as "WHERE "$a > w$"". We consider the five basic range query operators "$>, \geq, <, \leq, =$." The other operators such as "$\in$" could be naturally derived from the basic operators.

We consider the whole range to be a sequence of N discrete numbers $A = (a_1, \ldots, a_N)$, where $a_1 < a_2 < \cdots < a_N$. Then we set five shared virtual documents $d'_1 = (> a_1, > a_2, \ldots, > a_{N-1})$, $d'_2 = (\geq a_1, \geq a_2, \ldots, \geq a_N)$, $d'_3 = (< a_2, < a_3, \ldots, < a_N)$, $d'_4 = (\leq a_1, \leq a_2, \ldots, \leq a_N)$, and $d'_5 = (= a_1, = a_2, \ldots, = a_N)$ for all user's documents. The virtual document could be encrypted by LSE' core as a normal document. Therefore, for any keyword such as "$> a_i$" where $1 \leq i \leq N-1$, there always exists a mapping v_i.

Based on the notion of virtual document, a label $a \in A$ for a user's document satisfying $a_{i-1} < a < a_{i+1}$ (or $a = a_i$) could be represented as $2N+1$ keywords $d = (> a_1, \geq a_1, \ldots, > a_{i-1}, \geq a_{i-1}, \geq a_i, = a_i, \leq a_i, < a_{i+1}, \leq a_{i+1}, \ldots, < a_N, \leq a_N)$, and these keywords are stored in a bloom filter B. Suppose the user queries a keyword "$> w$," where $a_1 < w < a_N$; then the query is transmitted to the bloom filter to test if "$> w$" $\in B$.

For example (we only consider the operator "$>$" here for simplicity), suppose we have two documents c_1, c_2 labeled 5, 10, respectively. Then the transformed sets are $B_1 = (> 1, > 2, \ldots, > 4)$ and $B_2 = (> 1, > 2, \ldots, > 9)$. If the user submits >7, then only B_2 matches the query, which is the same result as direct comparisons since $5 \ngtr 7$ and $10 > 7$, and then c_2 is returned. Similarly, the query ">3" will match both documents, and c_1, c_2 are returned.

Now we construct the secure version of the aforementioned scheme. Let $A = (a_1, \ldots, a_N)$ denote the domain of the label, and setup the bloom filter with r independent hash functions $h_1, \ldots, h_r$. The identifier $\overline{d}$ of a document is always bound to the document d or the ciphertext c. The concrete construction is presented in Algorithm 9.

The size of the bloom filter could be dramatically reduced if the domain is bucketized [32] for example, bucketizing the subrange $[10, 20)$ as tag 10 and the subrange $[20, 30)$ as tag 20. Then a query for ">13" could be mapped to the closest query ">10." In other words, the whole domain is divided to multiple subranges that the queried range is transformed to the approximate range. The optimization of the idea of bucketizing the range is introduced in [33]. In such way, the number of the data stored in the bloom filter will become smaller. However, this will induce inaccuracy for the query result.

5.3.3. Proof of Security. For simplicity without loss of generality, we only consider the operator "$>$" here, and the other operators are the same. Informally speaking, the functional component must guarantee that the adversary is unable to guess the queried range as well as the range in the ciphertext, and the basic game works as follows. Given two documents d_1, d_2 that are labeled with two numbers a_1, a_2, respectively, the challenger flips a coin b and encrypts (d_b, a_b). The adversary is allowed to adaptively query p keywords $W = (w_1, \ldots, w_p)$, where each $w_i \in W$ that $(a_1, a_2) > w_i$. Note that querying w_i that $a_1 > w_i$, $a_2 \ngtr w_i$ is not allowed since the document is immediately distinguished (only the document with $a_1 > w_i$ is matched and returned). We propose the notion of chosen range attack (CRA) and formally define the security model for semantic security as follows.

Definition 12 (semantic security against chosen range attack, CRA-secure). Let Σ be the functional component for ranked keyword query. Let $k \in \mathbb{N}$ be the security parameter. We consider the following probabilistic experiments, where $\mathscr{A}$ is an adversary and $\mathscr{S}$ is a simulator.

> $\mathrm{Real}_{\Sigma,\mathscr{A}}(k)$: the challenger runs $\texttt{Gen}(1^k)$ to generate the key K. The adversary $\mathscr{A}$ generates a document d and the labeled number a and receives the encrypted document c and the functional structure L. $\mathscr{A}$ is allowed to adaptively query p keywords $W = (>w_1, \ldots, >w_p)$. For each query "$> w_i$," $\mathscr{A}$ receives a mapping v_i from the challenger. Finally, $\mathscr{A}$ returns a bit b that is output by the experiment.
>
> $\mathrm{Sim}_{\Sigma,\mathscr{A},\mathscr{S}}(k)$: given the document size $|d|$, the cardinality of the range N, and the size of the mapping $|v|$, $\mathscr{S}$ generates c^*, L^*, and $V^* = (v_1^*, \ldots, v_p^*)$, and then sends the results to $\mathscr{A}$. Finally, $\mathscr{A}$ returns a bit b that is output by the experiment.

We say that the functional component is CRA-secure, if for all PPT adversaries $\mathscr{A}$, there exists a PPT simulator $\mathscr{S}$ such that

$$\begin{aligned} &\left|\Pr\left[\mathrm{Real}_{\Sigma,\mathscr{A}}(k) = 1\right] - \Pr\left[\mathrm{Sim}_{\Sigma,\mathscr{A},\mathscr{S}}(k) = 1\right]\right| \\ &\quad \leq \operatorname{negl}(k), \end{aligned} \tag{7}$$

where the probabilities are over the coins of $\texttt{Gen}$.

Theorem 13. *If LSE is interface-CPA-secure, then the ranked keyword query component is CRA-secure.*

Proof. The simulator $\mathscr{S}$ generates c^*, L^* and $V^* = (v_1^*, \ldots, v_p^*)$ as follows. As to c^*, $\mathscr{S}$ generates a random string of size $|d|$. As to L^*, $\mathscr{S}$ generates a random string $\overline{d}^*$ and $2N+1$ distinct and random strings $T = (t_1, \ldots, t_{2N+1})$. For each $t_i \in T$, $\mathscr{S}$ computes r codewords $y_1^* = h_1(\overline{d}^* \| t_i), \ldots, y_r^* = h_r(\overline{d}^* \| t_i)$ and inserts the codewords $y_1^*, \ldots, y_r^*$ into a bloom filter B^*. Let $L^* = B^*$. As to V^*, for each $v_i^* \in V^*$, $\mathscr{S}$ randomly selects a distinct $t_j \in T$ maps to v_i^*, such that $v_i^* = t_j$. Note that, if $v_x^* = v_y^*$ for some locations $1 \leq x, y \leq p$, the mapping is the same.

We claim that no polynomial-size distinguisher could distinguish (c, L, V) from (c^*, L^*, V^*). Since the encryption key K is kept secret from the adversary, the interface-CPA-security directly guarantees that c^* is indistinguishable from c. It also guarantees that each $v_i \in V$ is indistinguishable from the random string v_i^* such that V is indistinguishable from V^*. Therefore, the locations $(y_1, \ldots, y_r)$ of v_i in bloom filter B is indistinguishable from the locations $(y_1^*, \ldots, y_r^*)$ of v_i^* in bloom filter B^*. Therefore, $r \cdot (2N + 1)$ locations in B are indistinguishable from $r \cdot (2N + 1)$ locations in B^*. Thus, L is indistinguishable from L^*. □

5.4. Other Functionalities. Due to space limitation, we only discuss the above two representative functional components. We briefly introduce how to realize some other functionalities based on LSE as follows.

Phrase Query. It refers to a query with consecutive and ordered multiple keywords. For example, searching with phrase "operating system" requires that not only each keyword "operating" and "system" must exist in each returned document, but also the order that "operating" is followed by "system" must also be satisfied. In [21], the authors introduced a solution based on Nextword Index [34]. It allows the index to record the keyword position for each document and enables the user to query the consecutive keywords based on binary search over all positions. However, it has $O(\log n)$ computation complexity for each document. Based on LSE, this functionality could be realized using bloom filter (as demonstrated in range query scheme) which recording biword or more words based on Partial Phrase Indexes [35]. As a result, the scheme coude achieve approximately $O(1)$ computation complexity (note that, the index in the global layer could reduce a large number of results for multiple keywords).

Fuzzy Keyword Search. It refers to a functionality that the user submit a fragment of a keyword (or a keyword that does not exist in all documents) and the server could search for the documents with all possible keywords that are closed to the fragment. In [3], the authors introduced a wildcard-based construction that could handle fuzzy keyword search with arbitrary edit distance [36]. By using the same method, the functional structure could realize this functionality by recording and indexing the fuzzy set of all mappings instead of keywords.

Similarity Query. It refers to a functionality that the server could return to the user some documents containing keywords which are similar to the queried keyword. In both [18, 19], the authors realized this functionality based on fuzzy set. Therefore, although different methods are used, the construction of the fundamental component is similar to the construction of fuzzy keyword search scheme.

Subset Query. It refers to a functionality that the server could test if the queried message is a subset of the values in the searchable fields. For example, let S be a set that contains multiple e-mail addresses. If the user search for some encrypted mails containing Alice's e-mail a, then the server must have the ability to test if $a \in S$ without knowing any other information. A solution was also introduced in [2]. Similar to the range query scheme, this test could also be viewed as a predicate and therefore the solution is the same.

5.5. Performance Analysis. The algorithms of ranked keyword query component and range query component are coded in C++ programming language and the server is a Pentium Dual-Core E5300 PC with 2.6 GHz CPU. Each document is fixed to 10 KB with random words chosen from a dictionary, and the query is also some random keywords (random numbers). For bloom filter used in range query

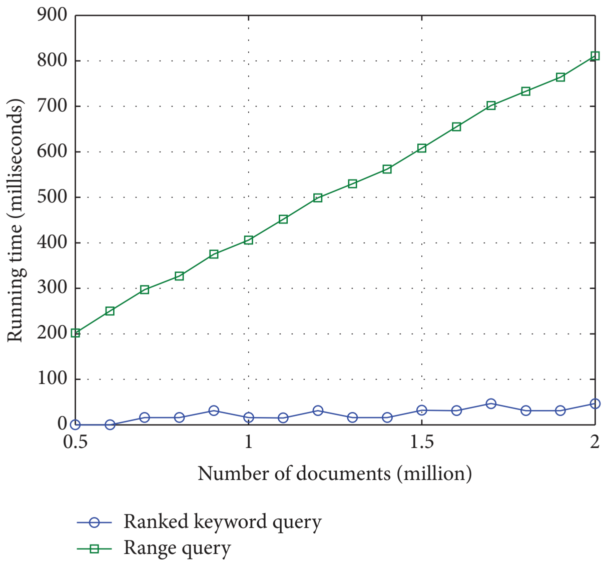


FIGURE 4: Time costs of the filter algorithms (single server, single query).

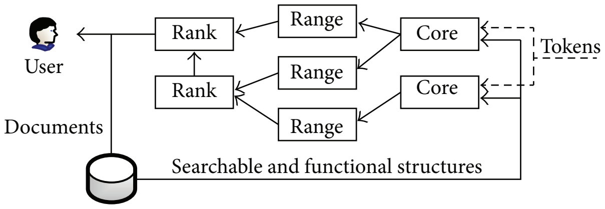


FIGURE 5: Deploying functional components to multiple servers.

component, the number of hash functions is set to 8. The time costs of the filter algorithms are shown in Figure 4.

Let n denote the number of documents. For ranked keyword query, the main operations are retrieving the relevance scores from the secure table managed by indirect addressing technique ($O(n)$ search complexity) and selecting the top-k scores ($O(n)$ computation complexity). For range query, the main operation is computing 8 hash values ($O(n)$ computation complexity). Note that the current document will be passed if any position in bloom filter is 0. Therefore, not all eight hash functions are executed all the time. The figure demonstrates that, even for a single server, the algorithms are both efficient. Note that, since the functional components are loosely coupled with each other, they could be deployed to different servers. For example, two core components (Core), two ranked keyword query components (Rank), and three range query components (Range) could be executed as a dataflow boxes as shown in Figure 5. Each box could be deployed to any server. The detailed methods are out of scope of this paper and we will not discuss this further.

6. Conclusions

Layered searchable encryption scheme provides a new way of thinking the relationship among the searchable structure, functionality and security. It separates the functionalities apart from the core searchable structure without loss of security. Therefore, the loose coupling property provides compatibility for symmetric and asymmetric settings and it also provides flexibility for adding or deleting various functionalities. Furthermore, following the popular boxes and arrows paradigm, the loose coupling property makes the scheme more suitable for distributed and parallel computing environment.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work is supported by the Science and Technology Department of Sichuan Province (Grant no. 2012GZ0088 and no. 2012FZ0064).

References

[1] H. Takabi, J. B. D. Joshi, and G.-J. Ahn, "Security and privacy challenges in cloud computing environments," *IEEE Security and Privacy*, vol. 8, no. 6, pp. 24–31, 2010.

[2] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Theory of Cryptography*, pp. 535–554, Springer, 2007.

[3] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proceeding of the Fuzzy Keyword Search over Encrypted Data in Cloud Computing (IEEE INFOCOM '10)*, Institute of Electrical and Electronics Engineers, March 2010.

[4] A. Swaminathan, Y. Mao, G.-M. Su et al., "Confidentiality-preserving rank-ordered search," in *Proceedings of the ACM Workshop on Storage Security and Survivability (StorageSS '07)*, pp. 7–12, Association for Computing Machinery, October 2007.

[5] D. J. Abadi, D. Carney, U. Çetintemel et al., "Aurora: a new model and architecture for data stream management," *VLDB Journal*, vol. 12, no. 2, pp. 120–139, 2003.

[6] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 44–55, May 2000.

[7] E. J. Goh, "Secure indexes," Tech. Rep., IACR ePrint Cryptography Archive, 2003.

[8] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*, pp. 79–88, November 2006.

[9] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *Proceedings of the 16th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT '10)*, pp. 577–594, Springer, 2010.

[10] S. Kamara, C. Papamanthou, and T. Roeder, "Cs2: a searchable cryptographic cloud storage system," Tech. Rep. MSR-TR-2011-58, Microsoft Research.

[11] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '04)*, pp. 506–522, Springer, 2004.

[12] M. Abdalla, M. Bellare, D. Catalano et al., "Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions," *Journal of Cryptology*, vol. 21, no. 3, pp. 350–391, 2008.

[13] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proceedings of the IEEE International Conference on Computer Communications (IEEE INFOCOM '11)*, pp. 829–837, Institute of Electrical and Electronics Engineers, 2011.

[14] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proceedings of the 30th IEEE International Conference on Distributed Computing Systems (ICDCS '10)*, pp. 253–262, June 2010.

[15] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Proceedings of the 2nd International Conference (ACNS '04)*, pp. 31–45, Springer, 2004.

[16] C. Bosch, R. Brinkman, P. Hartel, and W. Jonker, "Conjunctive wildcard search over encrypted data," in *Proceedings of the 8th VLDB Workshop on Secure Data Management*, pp. 114–127, Springer, 2011.

[17] J. Bringer and H. Chabanne, "Embedding edit distance to allow private keyword search in cloud computing," in *Proceedings of the 8th FTRA International Conference on Secure and Trust Computing, Data Management, and Application*, pp. 105–113, Springer, 2011.

[18] W. Cong, R. Kui, Y. Shucheng, and K. M. R. Urs, "Achieving usable and privacy-assured similarity search over outsourced cloud data," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '12)*, pp. 451–459, 2012.

[19] M. Kuzu, M. S. Islam, and M. Kantarcioglu, "Efficient similarity search over encrypted data," in *Proceedings of the IEEE 28th International Conference on Data Engineering (ICDE '12)*, pp. 1156–1167, 2012.

[20] S. Zittrower and C. C. Zou, "Encrypted phrase searching in the cloud," in *Proceedings of the IEEE Conference and Exhibition Global Telecommunications Conference (GLOBECOM '12)*, pp. 764–770, 2012.

[21] Y. Tang, D. Gu, N. Ding, and H. Lu, "Phrase search over encrypted data with symmetric encryption scheme," in *Proceedings of the 32nd International Conference on Distributed Computing Systems Workshops (ICDCSW '12)*, pp. 471–480, 2012.

[22] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. E. Skeith III, "Public key encryption that allows pir queries," in *Proceeding of the 27th Annual International Cryptology Conference (CRYPTO '07)*, pp. 50–67, Springer, 2007.

[23] E. Shi, J. Bethencourt, T.-H. H. Chan, D. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," in *Proceedings of the IEEE Symposium on Security and Privacy (SP '07)*, pp. 350–364, May 2007.

[24] R. Rivest, *The Md5 Message-Digest Algorithm*, Internet Request For Comments, 1992.

[25] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 8, pp. 1467–1479, 2012.

[26] S. Sedghi, P. Van Liesdonk, S. Nikova, P. Hartel, and W. Jonker, "Searching keywords with wildcards on encrypted data," in *Security and Cryptography For Networks*, pp. 138–153, Springer, 2010.

[27] A. Boldyreva, N. Chenette, Y. Lee, and A. Oneill, "Order-preserving symmetric encryption," in *Advances in Cryptology (EUROCRYPT '09)*, pp. 224–241, Springer, 2009.

[28] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '04)*, pp. 563–574, June 2004.

[29] A. Boldyreva, N. Chenette, and A. O'Neill, "Order-preserving encryption revisited: improved security analysis and alternative solutions," in *Proceedings of the Advances in Cryptology (CRYPTO '11)*, pp. 578–595, Springer, 2011.

[30] M. L. Fredman, E. Szemeredi, and J. Komlos, "Storing a sparse table with o(1) worst case access time," *Journal of the ACM*, vol. 31, no. 3, pp. 538–544, 1984.

[31] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

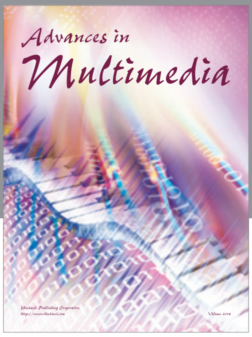
[32] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in *Proceedings of the ACM SIGMOD International Conference on Managment of Data (ACM SIGMOD '02)*, pp. 216–227, June 2002.

[33] B. Hore, S. Mehrotra, and G. Tsudik, "A privacy-preserving index for range queries," in *Proceedings of the Thirtieth international conference on Very large data bases*, pp. 720–731, VLDB Endowment, 2004.

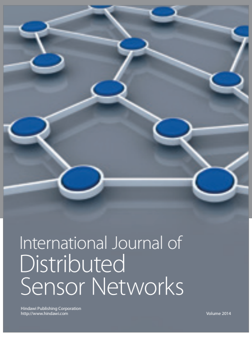
[34] H. E. Williams, J. Zobel, and P. Anderson, "What's next? index structures for efficient phrase querying," in *Australasian Database Conference*, pp. 141–152, 1999.

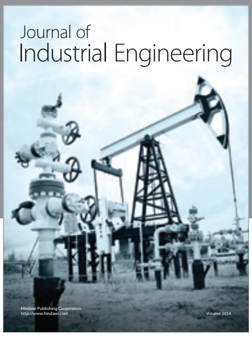
[35] C. Gutwin, G. Paynter, I. Witten, C. Nevill-Manning, and E. Frank, "Improving browsing in digital libraries with keyphrase indexes," *Decision Support Systems*, vol. 27, no. 1, pp. 81–104, 1999.

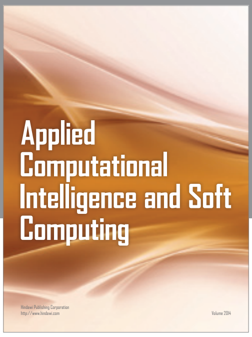
[36] V. Levenstein, "Binary codes capable of correcting spurious insertions and deletions of ones," *Problems of Information Transmission*, vol. 1, no. 1, pp. 8–17, 1965.

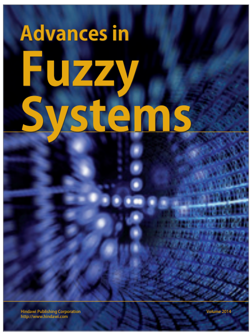

Advances in
Multimedia

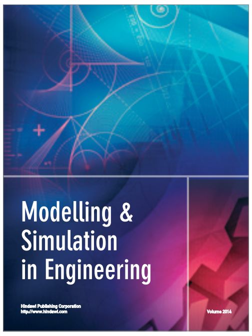
The Scientific
World Journal

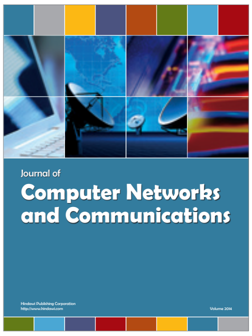

International Journal of
Distributed
Sensor Networks

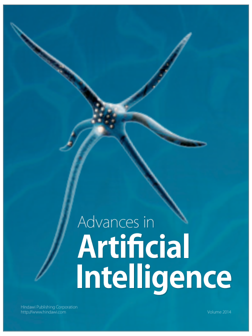

Journal of
Industrial Engineering

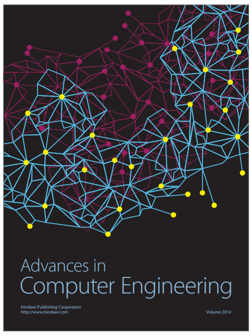

Applied
Computational
Intelligence and Soft
Computing


Advances in
Fuzzy
Systems

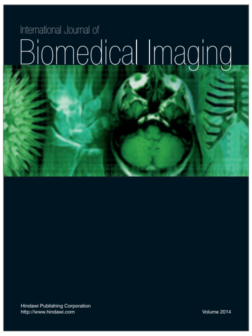

Modelling &
Simulation
in Engineering

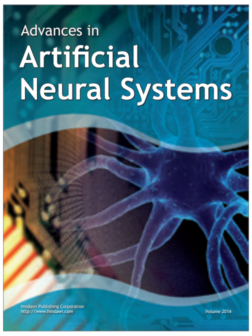

Journal of
Computer Networks
and Communications

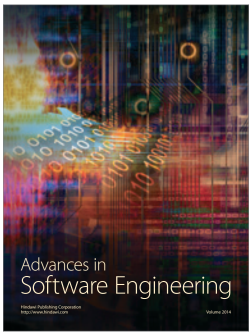

Advances in
Artificial
Intelligence


Advances in
Computer Engineering

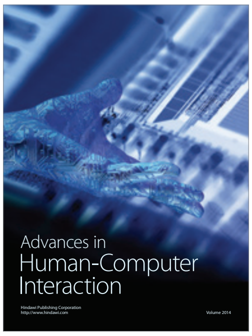
International Journal of
Computer Games
Technology

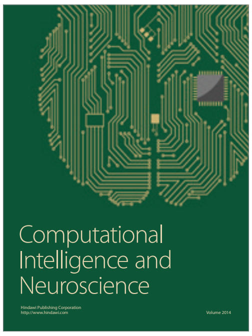

International Journal of
Biomedical Imaging


Advances in
Artificial
Neural Systems


Advances in
Software Engineering

Journal of
Robotics


Advances in
Human-Computer
Interaction


Computational
Intelligence and
Neuroscience

International Journal of
Reconfigurable
Computing

Journal of
Electrical and Computer
Engineering