SDKSE-KGA: A Secure Dynamic Keyword Searchable Encryption Scheme Against Keyword Guessing Attacks

Hongyuan Chen, Zhenfu Cao, Xiaolei Dong, Jiachen Shen

SDKSE-KGA: A Secure Dynamic Keyword Searchable Encryption Scheme Against Keyword Guessing Attacks

Hongyuan Chen[1,[0000−0003−4273−7185]], Zhenfu Cao[1,2,[0000−0002−5250−5030]*]
Xiaolei Dong[1,[0000−−0002−5844−0223]], Jiachen Shen[1,[0000−−0003−2376−5068]*]

1. Shanghai Key Laboratory of Trustworthy computing
East China Normal University, China
2. Cyberspace Security Research Center
Peng Cheng Laboratory, Shenzhen
and Shanghai Institute of Intelligent Science and Technology
Tongji University, China
51164500090@stu.ecnu.edu.cn, {zfcao,dongxiaolei,jcshen}@sei.ecnu.edu.cn

Abstract. A number of searchable encryption schemes have been widely proposed to solve the search problem in ciphertext domain. However, most existing searchable encryption schemes are vulnerable to keyword guessing attacks. During keyword guessing attacks, with the help of the cloud, an adversary will learn what keyword a given trapdoor is searching for, which leads to the disclosure of users' privacy information. To address this issue, we propose SDKSE-KGA: a secure dynamic keyword searchable encryption scheme which resists keyword guessing attacks. SDKSE-KGA has constant-size indexes and trapdoors and supports functionalities such as dynamic updating of keywords and files. Formal proofs show that it is Trapdoor-IND-CKA and Index-IND-CKA secure in the standard model.

Keywords: Searchable encryption · Dynamic · Keyword Guessing Attack · Trapdoor-IND-CKA · Index-IND-CKA.

1 Introduction

Searchable encryption is an effective way to solve the search problem in ciphertext domain. It not only protects users' privacy but also completes search task. During the process of searchable encryption, users need encrypt data before uploading it. Then, they use trapdoors of keywords to execute search task. So the cloud cannot get exact information about data and keywords.

Song et al. firstly proposed a searchable encryption scheme for mail system in [1]. Then, the concept of searchable encryption (SE) came into people's attention and aroused a series of researches [2–6]. According to the encryption methods, divide SE into searchable encryption scheme (SSE) and public-key encryption with keyword search (PEKS). SSE has the advantages of high efficiency and practicability. So people tend to research the functionality of SSE. [7, 8] realized

the function of multi-keyword search. [9,10] realized the function of fuzzy search. [11] realized the ranking function of search results. PEKS has the advantage of strong security. So people tend to improve search expressions and security. [12,13] implement access control for users search privileges. [14] has the traceability for malicious users. [15] implements the revocation of malicious users privileges. [16] implements the verification of search results.

Consider one dynamic mail system: For user Alice, she has many friends and business partners in real life. So her inbox may received all kinds of mails everyday. The inbox will store these mails into cloud servers. Considering the cloud is not fully trusted, all information should be encrypted. When Alice checks mails, she will filter mails generally and search for parts of them. The search keywords are determined by Alice herself, and she is likely to change keywords according to the actual life. This application scenario requires our searchable encryption scheme to support dynamic keywords.

Kamara et al. firstly proposed a dynamic searchable symmetric encryption scheme in [17]. They gave the definition of dynamic CKA2 security and constructed algorithm by reverse indexes. But this scheme has the disadvantage of information leakage. They offered an improved scheme in [18]. It uses red black tree as index tree to protect information. But this advantage is at the cost of reducing search efficiency. Hahn et al. presented a new scheme in [19]. It leaks nothing except the access pattern and requires the scenario to have huge data and a few keywords. Xia et al proposed a multi-keyword ranked search scheme in [20], which supports dynamic environment, too. It uses balanced binary tree as index tree and sorts search results for users. But it lacks trapdoor indistinguishable security. Later, they presented a new scheme in [21]. For mail systems, it has significant reduction in IO cost. But the size of index is large. It will make pressure on communication overhead.

Meanwhile, Byun et al. first introduced the concept of keyword guessing attack in [25]. In keyword guessing attacks, adverseries take advantage of the fact that the keywords that one user are likely to use commonly are very limited. So they make guesses of the keyword corresponding to a trapdoor. With the help of the cloud, they are able to verify whether the guess is correct and shortly they will know which keyword this trapdoor is searching for. It is a crucial attack and violates the goal of searchable encryption.

The concept of offline keyword guessing attacks was proposed in [25]. Then Yau et al. presented the concept of online keyword guessing attacks in [26]. Tang et al. proposed a public-key encryption supporting registered keyword search in [27]. But it requires the sender and the receiver to negotiate registered keywords before the system was established. Compare with it, our scheme relaxes the restrictions on communication between senders and receivers. Chen et al. proposed a searchable encryption under dual systems in [28]. There are multiple interactions between front server and back server. It prevents independent servers from getting complete information to withstand attacks. Compare with it, the cloud server in our scheme has less computational and communication pressure.

For the above mail system, we construct a dynamic searchable encryption scheme which resists keyword guessing attacks. Our contribution is summarized as follows:

1. Our $SDKSE-KGA$ supports dynamic management of both keywords and files. In the mail system, senders may send messages anytime and receivers may delete messages too. The receiver may add or delete keywords by the binary tree. Compared with other papers, the cost of updating keywords and files is negligible. In addition, the update operation is completely executed by the receiver, so there is no risk of leaking private data.
2. Our $SDKSE-KGA$ has Index-IND-CKA(Index Indistinguishable against Chosen Keyword Attack) security and Trapdoor-IND-CKA(Trapdoor Indistinguishable against Chosen Keyword Attack) security. We will demonstrate security under the standard model. Moreover, the indexes and trapdoors are of constant size which helps to reduce transmission overhead significantly.
3. Our $SDKSE-KGA$ resists keyword guessing attacks. Therefore, our scheme has higher security level compared with other searchable encryption scenarios. In this scheme, the search task is assigned to the cloud server and the receiver, The cloud server performs fuzzy search while the receiver accurate search, The cloud server is not able to obtain specific information of keywords, so it cannot launch the keyword guessing attacks.

The rest of our paper is organized as follows. In Section 2, we will introduce the system model and security model, and describe some symbols used in our construction. In Section 3, we will introduce the keyword tree and fuzzy mapping function in detail. Section 4 depicts $SDKSE-KGA$ scheme in detail. Section 5 and Section 6 will show security analysis and performance analysis of $SDKSE-KGA$. In the last section we will summarize this paper.

2 Definitions

2.1 System Model

There are three roles in our application scenario: mail senders, mail receivers and the cloud server. The sender is responsible for adding keywords to these files, encrypting files and generating exact indexes and fuzzy indexes for keywords, and uploading them to the cloud server. The receiver is responsible for managing all the keywords by constructing a binary tree, and generating fuzzy and exact trapdoors. After receiving a fuzzy trapdoor, the cloud server conducts fuzzy search upon fuzzy indexes and sends fuzzy results to the receiver. Then the receiver performs exact search on the fuzzy results based on the exact trapdoors to obtain final results. Fig 1 shows the system model.

Considering third-party cloud servers cannot be fully trusted, we hope the cloud server get as little information as possible. Moreover, with the help of the cloud, KGA will learn what keyword a given trapdoor is searching for, which leads to the disclosure of users privacy information. In our model, the cloud

server is only allowed to perform fuzzy search. Even if it has access to all the fuzzy indexes of keywords and some of legal fuzzy trapdoors, it is still unable to get the exact information of the search. Moreover, this model not only protects the security of keywords, but also resists keyword guessing attacks.

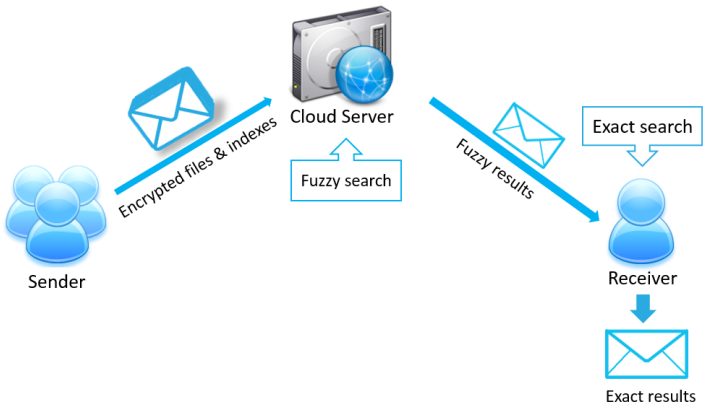

Fig. 1: System model

2.2 Security Model

In this part, we define $Index-IND-CKA$ security, $Trapdoor-IND-CKA$ security and adaptive KGA security. $Index-IND-CKA$ security means outside attackers cannot determine exact index $ExactIndex$ was generated by which keyword in case of they know nothing about the exact trapdoor of the given keywords. $Trapdoor-IND-CKA$ security means outside attackers cannot distinguish between the exact trapdoors of two challenge keywords [23]. The definitions of $Index-IND-CKA$ and $Trapdoor-IND-CKA$ are similar to these in [30]. We define the following security games to illustrate three kinds of security.

Game 1 : ($Index-IND-CKA$ security)
Setup. The challenger runs $Setup$ algorithm to obtain the public parameters and the master secret key. He retains the master secret key and gives the public parameters to the adversary $\mathcal{A}$.
Query phase 1. The adversary $\mathcal{A}$ adaptively selects keyword w to issue. The challenger generates ETd for w and sends it to $\mathcal{A}$.
Challenge. The adversary $\mathcal{A}$ selects target keywords w_0^* and w_1^*. Both of two target keywords has not queried before. Then, the challenger generates the exact index $ExactIndex$ for w_β^* and sends it to $\mathcal{A}$ where $\beta \in \{0,1\}$.

Query Phase 2. Repeat Query Phase 1.The adversary $\mathcal{A}$ continue to issue keywords except the target keywords w_0^* and w_1^*.
Guess. The adversary gives β' as the guess of β, if $\beta' = \beta$, then the adversary wins.
The advantage of $\mathcal{A}$ in this game is defined as follows:

$$Adv_{\mathcal{A}} = |Pr[\beta = \beta'] - \tfrac{1}{2}|$$

Definition 1. We say that SDKSE-KGA is Index-Indistinguishable security if $Adv_{\mathcal{A}}$ is negligible for any polynomial time attacker $\mathcal{A}$.

Game 2 : ($Trapdoor - IND - CKA$ security)
Setup. The challenger runs $Setup$ algorithm to obtain the public parameters and the master secret key. He retains the master secret key and gives the public parameters to the adversary $\mathcal{B}$.
Query phase 1. The adversary $\mathcal{B}$ adaptively selects keyword w to issue. The challenger generates ETd for w and sends it to $\mathcal{B}$.
Challenge. The adversary $\mathcal{B}$ selects target keywords w_0^* and w_1^*. Both of two target keywords has not queried before. Then, the challenger flips a coin $\beta \in \{0, 1\}$, generates the ETd for w_β^* and sends it to $\mathcal{B}$.
Query Phase 2. Repeat Query Phase 1. The adversary continue to issue keywords except the target keywords w_0^* and w_1^*.
Guess. The adversary gives β' as the guess of β, if $\beta' = \beta$, then the adversary wins.
The advantage of $\mathcal{B}$ in this game is defined as follows:

$$Adv_{\mathcal{B}} = |Pr[\beta = \beta'] - \tfrac{1}{2}|$$

Definition 2. We say that SDKSE-KGA is Trapdoor-Indistinguishable security if $Adv_{\mathcal{B}}$ is negligible for any polynomial time attacker $\mathcal{B}$.

Game 3 : (*Adaptive KGA* security)
Setup. The challenger runs this algorithm to obtain the public parameters and the master secret key. Then he retains the master secret key and gives the public parameters to the adversary $\mathcal{C}$.
Query phase 1. The adversary $\mathcal{C}$ queries the fuzzy trapdoor and fuzzy index of any keyword.
Challenge. The adversary selects the keyword w_0^* and w_1^* as challenge keywords, and neither keyword has been quried before. Then the challenger randomly selects the keyword $w_\beta^*(\beta \in \{0, 1\})$, generates ciphertext $FTd_{w_\beta^*}$ for it, and sends the trapdoor to $\mathcal{C}$.
Query Phase 2. Repeat Query Phase 1. The adversary $\mathcal{C}$ continue to query the fuzzy trapdoor and fuzzy index of keywords except the target keywords w_0^* and w_1^*.
Guess. The adversary gives β' as the guess of β, if $\beta' = \beta$, then the adversary wins.
The advantage of $\mathcal{C}$ in this game is defined as follows:

$$Adv_{\mathcal{C}} = |Pr[\beta = \beta'] - \frac{1}{2}|$$

Definition 3. We say that SDKSE-KGA is Adaptive KGA security if $Adv_{\mathcal{C}}$ is negligible for any polynomial time attacker $\mathcal{C}$.

2.3 Notations

This part we will illustrate some symbols used in this scheme. To manage all the keywords, we build a binary tree denoted by $\mathcal{T}$, use L to indicate the height of $\mathcal{T}$. And the height L is related to N which means the number of keywords. The fuzzy keyword mapped by the keyword w is expressed as w_f. $[I_1, ..., I_h]$ represents the location of keyword in the tree. The exact index and fuzzy index of keywords are respectively represented by $ExactIndex$ and $FuzzyIndex$. The exact trapdoor and fuzzy trapdoor of keywords are respectively represented by ETd and FTd.

Definition 3.$(SDKSE - KGA)$ A securely dynamic keyword searchable encryption scheme which resists keyword guessing attacks is a tuple of nine polynomial-time algorithms

$$SDKSE = (Setup, Encrypt, TDGen, FuzzySearch, ExactSearch, KWInsert, IndexInsert, KWDelete, IndexDelete)$$

such that

- $Setup(\lambda, N) \rightarrow (params, MSK)$: In this algorithm, input the security parameter λ and the number of keywords N, generate keyword tree $\mathcal{T}$, output public parameters of the scheme $params$ and master secret key MSK.
- $Encrypt(params, w) \rightarrow (FuzzyIndex,\ ExactIndex)$: In this algorithm, input $params$ and keyword w. Generate fuzzy index $FuzzyIndex$ and exact index $ExactIndex$ for w.
- $TDGen(MSK, w) \rightarrow (FTd, ETd)$: In this algorithm, generate fuzzy trapdoor FTd and exact trapdoor ETd for keyword w by MSK.
- $FuzzySearch(FuzzyIndex, FTd) \rightarrow (FuzzyCipher\ or\ \perp)$: In this algorithm, input fuzzy index $FuzzyIndex$ and fuzzy trapdoor FTd to match. If the match operation is successful, add these files associated with $FuzzyIndex$ to the fuzzy ciphertext set $FuzzyCipher$. If the operation is failed, output $\perp$.
- $ExactSearch(ExactIndex, ETd) \rightarrow (C\ or\ \perp)$: In this algorithm, input exact index $ExactIndex$ and exact trapdoor ETd for operation. If the operation is successful, output file set which contain keyword w. If the operation is failed, output $\perp$.
- $KWInsert(w)$: Insert new keyword w to the tree $\mathcal{T}$.
- $IndexInsert(w)$: Notify related files to update keyword list and generate encrypted keyword C for new keyword w.
- $KWDelete(w)$: Disable node bound to keyword w from tree $\mathcal{T}$.
- $IndexDelete(w)$: Notify related files to update keyword list and delete existing index of w.

3 Preliminaries

3.1 Keyword Tree

The receiver is responsible for constructing the binary tree $\mathcal{T}$. The tree $\mathcal{T}$ has two tasks: managing keywords dynamically and running fuzzy mapping function. Construct tree $\mathcal{T}$ based on the number of keywords N, height $L = \lceil log_2 N \rceil + 2$. Each leaf node may bind to one keyword. We call one leaf node that have not yet bound keyword as available node. The number of available nodes is denoted by $avlS$. In order to ensure the growth of the tree, we require $avlS \geq minS$, where $minS = 2^{L-2}$. Each leaf node has three states: disable, occupied, available. They are represented by $[0, 1, 2]$ respectively. Disable state means this leaf node is bound to one disable keyword. Occupied state means this leaf node is bound to one keyword. Available state means this leaf node has not been bound.

It is very easy to delete one keyword. We just need to set the state of the leaf node bound to this keyword to 0, and if this key is used again later, just change the state of the leaf node to 1. Adding keyword can be divided in two situations: If $avlS > minS$, select an appropriate available leaf node and bind it to the keyword. Then set its state value to 2. If $avlS = minS$, then generate child nodes of all available leaf nodes to double the number of available leaf nodes. The growth process is shown in Fig 2. Now $avlS > minS$, so we continue to add keywords.

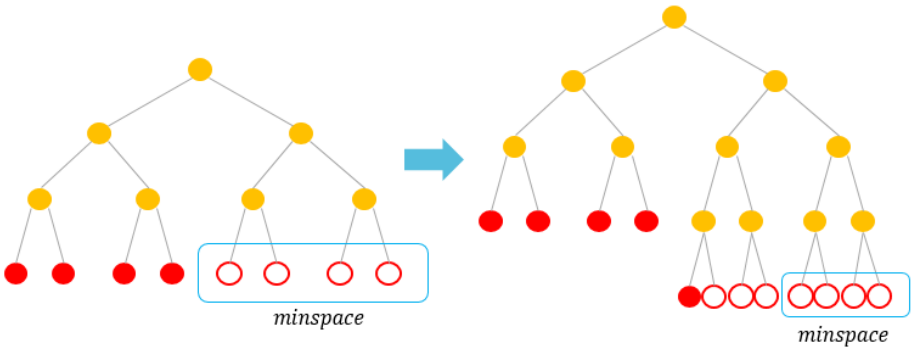

Fig. 2: Grow Tree

Now we design fuzzy mapping function to map each keyword to a fuzzy keyword. The position of the fuzzy keyword in the tree will be used to generate the pair of fuzzy index and fuzzy trapdoor. The cloud server searches upon the fuzzy index-trapdoor pair while the receiver searches upon the exact index-trapdoor pair. Now we introduce the fuzzy mapping function. For one leaf node in the binary tree, trace it up to n levels where n is a parameter defined by users,

the obtained node is the corresponding fuzzy node of it. If two leaf nodes have the same ancestor node after tracing the same layers, then these nodes share a fuzzy node.

3.2 Bilinear Map

In our scheme, we apply bilinear map to *FuzzySearch* and *ExactSearch* algorithm. The specific principle is as follows:

There is a composite group $\mathbb{G}$ with order $n = p_1p_2p_3p_4$ where p_1, p_2, p_3 and p_4 are distinct primes. Assume one of the generators of $\mathbb{G}$ is G, then the generators of $\mathbb{G}_{p_1}$, $\mathbb{G}_{p_2}$, $\mathbb{G}_{p_3}$ and $\mathbb{G}_{p_4}$ are G_1, G_2, G_3 and G_4 respectively. And $G_1 = G^{p_2p_3p_4}$, $G_2 = G^{p_1p_3p_4}$, $G_3 = G^{p_1p_2p_4}$, $G_4 = G^{p_1p_2p_3}$. We infer that for distinct i and j, $\forall R_i \in \mathbb{G}_{p_i}$, $R_j \in \mathbb{G}_{p_j}$, $e(R_i, R_j) = 1$ holds.

3.3 Complexity Assumptions

The security of our scheme is based on six complexity assumptions [22]. The hardness of these assumptions relies on the theorems proposed by [24].

In *Assumption* 1, given a group generator $\mathcal{G}$, input security parameter λ, then generate primes p_1, p_2, p_3, p_4, two groups $\mathbb{G}$, $\mathbb{G}_T$, and the bilinear map e. Set the integer $n = p_1p_2p_3p_4$. Select random element x from $\mathbb{G}_{p_1}$, similarly select G_3 from $\mathbb{G}_{p_3}$ and G_4 from $\mathbb{G}_{p_4}$. Set $D = \{\mathbb{G}, n, x, G_3, G_4\}$. $T_0 \in \mathbb{G}_{p_1p_2p_4}$, $T_1 \in \mathbb{G}_{p_1p_4}$ and $\beta \in \{0, 1\}$. Give (D, T_β) to the adversary $\mathcal{B}$, the adversary $\mathcal{B}$ outputs a guess β', if $\beta' = \beta$, then he succeeds. Define $Adv1_{\mathcal{G},\mathcal{B}}(\lambda)$ to denote the advantage of $\mathcal{B}$, $Adv1_{\mathcal{G},\mathcal{B}}(\lambda) = |Pr[\beta' = \beta] - \frac{1}{2}|$.

The following assumptions are very similar to *Assumption*1, so we only introduce their differences.
In *Assumption* 2,

$$D = \{\mathbb{G}, n, x, G_1G_2, G_3, H_2H_3, G_4\},$$

$$T_0 \in \mathbb{G}_{p_1p_2p_3}, T_1 \in \mathbb{G}_{p_1p_3}.$$

And $G_1 \xleftarrow{R} \mathbb{G}_{p_1}$, $G_2, H_2 \xleftarrow{R} \mathbb{G}_{p_2}$, $G_3, H_3 \xleftarrow{R} \mathbb{G}_{p_3}$, $G_4 \xleftarrow{R} \mathbb{G}_{p_4}$.
In *Assumption* 3,

$$D = \{\mathbb{G}, n, G_1, H_2H_3, G_3, G_4\},$$

$$T_0 = H_2H_3', T_1 \in \mathbb{G}_{p_1p_3}.$$

And $G_1 \xleftarrow{R} \mathbb{G}_{p_1}$, $H_2 \xleftarrow{R} \mathbb{G}_{p_2}$, $G_3, H_3, H_3' \xleftarrow{R} \mathbb{G}_{p_3}$, $G_4 \xleftarrow{R} \mathbb{G}_{p_4}$.
In *Assumption* 4,

$$D = \{\mathbb{G}, n, G_1, H_2H_4, G_3, G_4\},$$

$$T_0 \in \mathbb{G}_{p_2p_4}, T_1 \in \mathbb{G}_{p_4}.$$

And $G_1 \overset{R}{\leftarrow} \mathbb{G}_{p_1}$, $H_2 \overset{R}{\leftarrow} \mathbb{G}_{p_2}$, $G_3 \overset{R}{\leftarrow} \mathbb{G}_{p_3}$, $G_4, H_4 \overset{R}{\leftarrow} \mathbb{G}_{p_4}$.
In *Assumption* 5,

$$D = \{\mathbb{G}, n, x, G_1G_2, G_3, H_1H_2, I_2I_3, G_4\},$$

$$T_0 = e(G_1, H_1), T_1 \in \mathbb{G}_T.$$

And $x, G_1, H_1 \overset{R}{\leftarrow} \mathbb{G}_{p_1}$, $G_2, H_2, I_2 \overset{R}{\leftarrow} \mathbb{G}_{p_2}$, $G_3, I_3 \overset{R}{\leftarrow} \mathbb{G}_{p_3}$, $G_4 \overset{R}{\leftarrow} \mathbb{G}_{p_4}$.
In *Assumption* 6,

$$D = \{\mathbb{G}, n, G_1G_4, H_1H_2, I_2, I_3, I_4, J_1J_2J_4\},$$

$$T_0 \in J_1J_2'J_4', T_1 \in \mathbb{G}_{p_1p_2p_4}.$$

And $G_1, H_1, J_1 \overset{R}{\leftarrow} \mathbb{G}_{p_1}$, $H_2, I_2, J_2, J_2' \overset{R}{\leftarrow} \mathbb{G}_{p_2}$, $I_3 \overset{R}{\leftarrow} \mathbb{G}_{p_3}$, $G_4, I_4, J_4, J_4' \overset{R}{\leftarrow} \mathbb{G}_{p_4}$.

For *Assumption* $1 \sim 6$, we have the following definition:

Definition 4 : For any polynomial time, if $Adv - N_{\mathcal{G},\mathcal{B}}(\lambda)$ is a negligible function of λ, then we think the group generator $\mathcal{G}$ satisfies *Assumption* N, $N \in \{1, 2, 3, 4, 5, 6\}$.

4 Construction

In this section we will introduce *SDKSE-KGA* in detail.

Setup(λ, N) : First, the receiver builds the keyword tree $\mathcal{T}$ to manage initial keywords. For keyword w, encode it as $[I_1, \ldots, I_h]$ according to its position in the binary tree. Note $h = L - 1$. Next, runs group generator $\mathcal{G}$ and obtains $(p_1, p_2, p_3, p_4, \mathbb{G}, \mathbb{G}_T, e)$. Then, selects random elements $x, y, u_1, ..., u_h, \omega \leftarrow \mathbb{G}_{p_1}, G_3 \leftarrow \mathbb{G}_{p_3}, G_4 \leftarrow \mathbb{G}_{p_4}, R_4, R_{4,g}, R_{4,h}, R_{4,u_1}, ..., R_{4,u_h} \leftarrow \mathbb{G}_{p_4}$. G_3 is the generator of $\mathbb{G}_{p_3}$ and G_4 is the generator of $\mathbb{G}_{p_4}$ respectively. So a random element of $\mathbb{G}_{p_4}$ can be chosen by raising G_4 to random exponents from $\mathbb{Z}_n$. At last, set $n = p_1p_2p_3p_4$, $X = xR_{4,g}$, $Y = yR_{4,h}$, $U_1 = u_1R_{4,u_1}$, ..., $U_h = u_hR_{4,u_h}$, $E = e(g, \omega)$. The public parameters $params = [\mathbb{G}, n, X, Y, U_1, \ldots, U_h, R_3, R_4, E]$. The master private key $MSK = [x, y, u_1, \ldots, u_h, \omega]$. The receiver publishes the $params$ and retains the MSK for generate trapdoors later.

Encrypt$(params, w)$: w represents the keyword to be encrypted, parse it to $[I_1, ..., I_h]$. The sender selects random integer $s \leftarrow \mathbb{Z}_n$ and random elements $\overline{R}_4, \overline{R}_4' \leftarrow \mathbb{G}_{p_4}$. Picks random message M. Next, set

$$\begin{aligned} CT_0 &= ME^s, \\ CT_1 &= (H\textstyle\prod_{i=1}^{h} U_i^{I_i})^s \overline{R}_4, \\ CT_2 &= G^s \overline{R}_4'. \end{aligned}$$

Set $CT \leftarrow [CT_0, CT_1, CT_2] \in \mathbb{G}_T \times \mathbb{G}^3$. Define $ExactIndex = [M, CT]$.

Then, according to the fuzzy mapping function, the keyword w is mapped to w_f, parse it to $[I_1, ..., I_{h_f}]$. The sender selects random integer $s_f \leftarrow \mathbb{Z}_n$ and random elements $\overline{R}_{f,4}, \overline{R}_{f,4}' \leftarrow \mathbb{G}_{p_4}$. Picks random message M_f. Next, set

$$CT_{f,0} = M_f E^{s_f},$$
$$CT_{f,1} = (H\prod_{i=1}^{h_f} U_i^{I_i})^{s_f}\overline{R}_{f,4},$$
$$CT_{f,2} = G^{s_f}\overline{R}'_{f,4}.$$

Set $CT_f \leftarrow [CT_{f,0}, CT_{f,1}, CT_{f,2}] \in \mathbb{G}_T \times \mathbb{G}^3$. Define $FuzzyIndex = [M_f, CT_f]$.

TDGen(MSK, w) : w is the keyword to be retrieved. Parse w to $[I_1, ..., I_h]$. The receiver selects random integers $r_1, r_2 \leftarrow \mathbb{Z}_n$ and random elements

$$R_3^1, {R_3}^2, R_3^3, R_3^4, {R_3}^5, R_3^6 \leftarrow \mathbb{G}_{p_3}^4.$$

To obtain the exact trapdoor ETd of the keyword w. Set $Td_1 = x^{r_1}R_3^1$, $Td_2 = \omega(y\prod_{i=1}^{h} u_i^{I_i})^{r_1}{R_3}^2$, $Td_3 = u_h^{r_1}R_3^3$, $Td_4 = x^{r_2}R_3^4$, $Td_5 = \omega(y\prod_{i=1}^{h} u_i^{I_i})^{r_2}{R_3}^5$, $Td_6 = u_h^{r_2}R_3^6$. Set

$$ETd = [Td_1, Td_2, Td_3, Td_4, Td_5, Td_6].$$

Map the keyword w to w_f, parse it to $[I_1, ..., I_{h_f}]$. The receiver selects random integers $r_{f,1}, r_{f,2} \leftarrow \mathbb{Z}_n$ and random elements

$$R_{f,3}^1, {R_{f,3}}^2, R_{f,3}^3, R_{f,3}^4, {R_{f,3}}^5, R_{f,3}^6 \leftarrow \mathbb{G}_{p_3}^4.$$

To obtain the fuzzy trapdoor FTd of the keyword w. Set $Td_{f,1} = x^{r_{f,1}}R_{f,3}^1$, $Td_{f,2} = \omega(y\prod_{i=1}^{h_f} u_i^{I_i})^{r_{f,1}}{R_{f,3}}^2$, $Td_{f,3} = u_{h_f}^{r_{f,1}}R_{f,3}^3$, $Td_{f,4} = x^{r_{f,2}}R_{f,3}^4$, $Td_{f,5} = \omega(y\prod_{i=1}^{h_f} u_i^{I_i})^{r_{f,2}}{R_{f,3}}^5$, $Td_{f,6} = u_{h_f}^{r_{f,2}}R_{f,3}^6$. Set

$$FTd = [Td_{f,1}, Td_{f,2}, Td_{f,3}, Td_{f,4}, Td_{f,5}, Td_{f,6}].$$

FuzzySearch$(FuzzyIndex, FTd)$: Parse FTd to $[Td_{f,1}, Td_{f,2}, Td_{f,3}, Td_{f,4}, Td_{f,5}, Td_{f,6}]$. $FuzzyIndex = [M_f, CT_f]$, parse CT_f to $[CT_{f,0}, CT_{f,1}, CT_{f,2}]$. Compute

$$M'_f = CT_{f,0} \cdot \frac{e(Td_{f,1}, CT_{f,1})}{e(Td_{f,2}, CT_{f,2})}.$$

If $M_f = M'_f$, add all files containing exact keywords which mapping to w_f into the fuzzy result $FuzzyCipher$. Then $FuzzyCipher$ will be sent to the receiver.

ExactSearch$(ExactIndex, ETd)$: Parse ETd to $[Td_1, Td_2, Td_3, Td_4, Td_5, Td_6]$. $ExactIndex = [M, CT]$, parse CT to $[CT_0, CT_1, CT_2]$. Compute

$$M' = CT_0 \cdot \frac{e(Td_1, CT_1)}{e(Td_2, CT_2)}.$$

If $M = M'$, then output the file set C which contains the keyword w.

KWInsert(w) : Select an appropriate leaf node to bind the new keyword w in the binary tree.

IndexInsert(w) : Generate the index based on the location in the tree and add it into index list.

KWDelete(w) : Disable the keyword w in the binary tree.

IndexDelete(w) : Delete the index of w from the index list.

5 Security Proof

In this section, we will prove the security of $SDKSE - KGA$. Each keyword owns an exact trapdoor-index pair and a fuzzy trapdoor-index pair. The sender generates fuzzy indexes and exact indexes and sends them to the cloud. The receiver generates fuzzy trapdoors and exact trapdoors and sends fuzzy trapdoors to the cloud. Notice that in both $FuzzySearch$ and $ExactSearch$ algorithms, only if the location strings corresponding to the trapdoor and the index are identical, the match operation will succeed. Since the fuzzy trapdoors and fuzzy indexes are generated upon the position, which one-to-one mapped into the location string of the fuzzy node, the match operation will only succeed when the fuzzy trapdoor and fuzzy index are generated upon the same fuzzy node. On the other hand, fuzzy nodes and exact nodes are different from each other, so the match operation upon a fuzzy trapdoor and an exact index will always generates $\perp$. Therefore, even the cloud gets exact indexes, the privacy of users will not be destroyed.

Now we will prove our $SDKSE - KGA$ is $Index - IND - CKA$ and $Trapdoor - IND - CKA$ secure.

Theorem 1. *Our $SDKSE - KGA$ scheme is $Index - IND - CKA$ secure if a group generator $\mathcal{G}$ holds assumptions in [22].*

Proof. We will give the definitions of semi-functional indexes and semi-functional trapdoors for $ExactIndex$ and ETd , and show a series of games. Semi-functional indexes are composed by CT_0,CT_1,CT_2.

$$CT_0 = CT_0', CT_1 = CT_1' x_2^{r z_c}, CT_2 = CT_2' x_2^{r}.$$

where CT_0', CT_1' and CT_2' are components of CT generated in $Encrypt$ algorithm. And $x_2 \in \mathbb{G}_{p_2}$, $r, z_c \xleftarrow{R} \mathbb{Z}_N$. Semi-functional trapdoors are as follows:

$$Td_1 = Td'_1 x_2^{\gamma}, Td_2 = Td'_2 x_2^{\gamma z_1}, Td_3 = Td'_3 x_2^{\gamma z_2},$$

$$Td_4 = Td'_4 x_2^{\gamma'}, Td_5 = Td'_5 x_2^{\gamma' z_1'}, Td_6 = Td'_6 x_2^{\gamma' z_2'},$$

where Td_1', Td_2', Td_3', Td_4', Td_5', Td_6' are components of ETd generated in $TDGen$ algorithm, $x_2 \in \mathbb{G}_{p_2}$, and $\gamma, \gamma', z_1, z_1', z_2, z_2' \xleftarrow{R} \mathbb{Z}_N$.

In addition, we need to construct a series of games.

$\mathbf{Game_{Real}}$: **Game 1**.

$\mathbf{Game_{Restricted}}$: It is similar to $\mathbf{Game_{Real}}$ except that the adversary cannot query keywords which are prefixes of the challenge keyword modulus p_2.

$\mathbf{Game_k}$:$0 \leq k \leq q$, and q is the number of queries made by the adversary. The difference between $\mathbf{Game_k}$ and $\mathbf{Game_{Restricted}}$ are query results. The challenge index is semi-functional index in two games and the first k results of trapdoor are semi-functional trapdoors in $\mathbf{Game_k}$.

$\mathbf{Game_{Mhiding}}$: It selects random elements from $\mathcal{G}$ and constructs CT_0 of the challenge index.

$\mathbf{Game_{Random}}$: The second component and the third component of challenge

indexes are independent random elements in $\mathbb{G}_{p_1p_2p_4}$ in this game.

In $Game_{Random}$, the adversary knows nothing about keyword from the challenge index. So we need prove $Game_{Real}$ and $Game_{Random}$ are distinguishable. First step, the adversary selects keywords w_0 and w_1, $w_0 \neq w_1$ mod n and $w_0 \equiv w_1$ mod p_2. The simulator $\int$ factor n by computing $gcd(w_0 - w_1, N)$. But the assumption 1,2,3 will prove that n cannot be decomposed. As a result, $Game_{Real}$ and $Game_{Restricted}$ are distinguishable. Second step, we will prove $Game_{Restricted}$ and $Game_k$ are distinguishable. According to assumption 1, construct a new game. In this game, if $T = T_0$, the index generated by challenger is semi-functional index. In this case, the game is equal to $Game_0$ eventually. If $T = T_1$, the index generated by challenger is normal index and the game is equal to $Game_{Restricted}$. T_0 and T_1 have the same distribution in statics, so $Game_{Restricted}$ and $Game_k$ are distinguishable. Third step, we will prove the series games $Game_k (0 \leq k \leq q)$ are distinguishable. Use the same way to construct a new game according to assumption 5. The trapdoors sent by challenger are semi-functional trapdoors. If $T = T_0$, the game is equal to $Game_q$. If $T = T_1$, the game is equal to $Game_{Mhiding}$. So $Game_q$ and $Game_{Mhiding}$ are indistinguishable. Continue to deduce, we will get the conclusion that $Game_{Mhiding}$ and $Game_{Random}$ are indistinguishable by constructing the new game according to assumption 6. Finally, $Game_{Real}$ and $Game_{Random}$ are distinguishable. The proof is completed.

Theorem 2. *Our $SDKSE-KGA$ scheme is $Trapdoor-IND-CKA$ secure.*

Proof. In *Game* 2, the adversary selects target keywords w_0 and w_1, then receives $ETd_{w_\beta^*}$ from the challenger. As we all known, $ETd_w = [Td_1, Td_2, Td_3, Td_4, Td_5, Td_6]$ where $Td_1 = x^{r_1} R_3^1$, $Td_2 = \omega(y \prod_{i=1}^h u_i^{I_i})^{r_1} R_3^2$, $Td_3 = u_h^{r_1} R_3^3$, $Td_4 = x^{r_2} R_3^4$, $Td_5 = \omega(y \prod_{i=1}^h u_i^{I_i})^{r_2} R_3^5$, $Td_6 = u_h^{r_2} R_3^6$. $x, y, w, u_1, \ldots, u_h$ belong to public parameters, $R_3^1 \sim R_3^6$ are random elements selected from $\mathbb{G}_{p_3}^4$. So the adversary only infer the value of β from Td_2 or Td_5. According to the property of bilinear pairing, R_3^2 in Td_2 can be removed by elements of $\mathbb{G}_{p_i}$,$i \in [1, 2, 4]$. The location strings $[I_1, \ldots, I_h]$ of w_0 and w_1 are known to the adversary, he is able to compute $m_0 = y \prod_{i=1}^h u_i^{I_{i,0}}$ and $m_1 = y \prod_{i=1}^h u_i^{I_{i,1}}$. m_0 and m_1 are the elements in $\mathbb{G}_{p_1}$. In statistics, the distributions of m_0^r and m_1^r are exactly the same where r is a random element in $\mathbb{Z}_n$. So the adversary is not able to guess the value of β by m_0, m_1. In other words, the adversary should not be able to distinguish the trapdoors of w_0^* and w_1^*. The proof is completed.

Theorem 3. *Our $SDKSE - KGA$ scheme is Adaptive KGA secure.*

Proof. Case 1: If two challenge keywords will map to different fuzzy keywords, they will generate different fuzzy trapdoors. So the KGA security game is exactly the same as Trapdoor-IND security game. In this case, the advantage of adversary winning the game is negligible.

Case 2: If two challenge keywords will map to the same fuzzy keywords, they will generate the same fuzzy trapdoors. The challenge keywords w_0^* and w_1^* have

the same distribution in statistics. The adversary cannot determine β based on the fuzzy trapdoor. In other words, he cannot distinguish between w_0^* and w_1^*.

In both cases, the advantage of the adversary winning the game is negligible.

6 Performance

This section mainly gives the performance analysis of SDKSE-KGA. The *Setup* algorithm requires $h+2$ multiplications and one pairing, it takes $2(h+3)$ multiplications and 6 modular exponentiations to generate one exact trapdoor where h denotes the height of keyword tree in the scheme. It takes $h+2$ multiplications and 3 modular exponentiations to generate one index. For *Search* algorithm, it requires 2 pairings and 2 multiplications. The computational overhead of $KWInsert$ and $KWDelete$ are negligible.

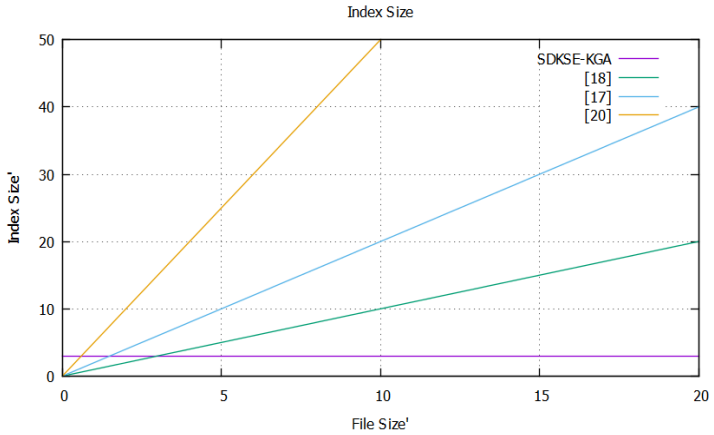

Fig. 3: Index Size

Our $SDKSE-KGA$ scheme supports keyword and file updating at the same time. To add a document, [20] and [17] need to iterate through keyword arrays and [18] needs to traverse a KRB tree. So the updating cost is very high. In addition, the index and trapdoors of our scheme are of constant size which reduces transmission overhead significantly. Table 1 shows the efficiency comparison between [20], [17], [18] and SDKSE-KGA and Fig 3 shows the comparison of the index sizes of different schemes.

Compared with other searchable encryption schemes which resist keyword guessing attacks, In terms of communication overhead, the size of index and trapdoor in SDKSE-KGA scheme is not affected by the number of files. Table 2 shows our advantages between this scheme and others. In this table, $\mathbb{G}$ represents a member of the group, *Pairing* means a bilinear pair operation, *Exp* means

Table 1: Comparisons with dynamic searchable schemes

Compare Items	[20]	[17]	[18]	$SDKSE-KGA$
Dynamic file	✓	✓	✓	✓
Dynamic Keyword	×	×	×	✓
Trapdoor-IND	✓	×	×	✓
Index Size	$O(n^2)$	$O(n)$	$O(n)$	$O(1)$
Trapdoor Size	$O(n^2)$	$O(1)$	$O(1)$	$O(1)$
Insert File	$O(n)$	$O(n)$	$O(n)$	$O(1)$
Insert Keyword	N/A	N/A	N/A	$O(n)$

power operation while Mul means multiplication operation. n is the number of all files.

Table 2: Comparisons with schemes resisting KGA

Schemes	Index Size	Search Overhead	KGA	Dynamic
[27]	$2\,\lvert\mathbb{G}\rvert$	$Pairing$	✓	×
[28]	$3\,\lvert\mathbb{G}\rvert$	$(7Exp+3Mul)\,n$	✓	×
[29]	$2\,\lvert\mathbb{G}\rvert$	$Pairing$	✓	×
SDKSE-KGA	$3\,\lvert\mathbb{G}\rvert$	$2Pairing$	✓	✓

7 Conclusion

In this paper, we proposed a secure dynamic searchable encryption scheme $SDKSE-KGA$ which resists keyword guessing attacks for mail systems. The complexity of the index and the trapdoor of $SDKSE-KGA$ are both constant size. Therefore, $SDKSE-KGA$ is capable of supporting dynamic management of mails and keywords and resisting keyword guessing attacks. In addition, it is both $Index-IND-CKA$ and $Trapdoor-IND-CKA$ secure.

8 Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (Grant No.61632012, 61672239, 61602180. and U1509219), in part by Natural Science Foundation of Shanghai (Grant No. 16ZR1409200), and in part

by "the Fundamental Research Funds for the Central Universities". Zhenfu Cao and Jiachen Shen are the corresponding authors.

References

1. Song D, Wagner D A, Perrig A, et al. Practical techniques for searches on encrypted data[J]. ieee symposium on security and privacy, 2000: 44-55.
2. Boneh D, Crescenzo G D, Ostrovsky R, et al. Public Key Encryption with Keyword Search[C]. theory and application of cryptographic techniques, 2004: 506-522.
3. Waters B, Balfanz D, Durfee G E, et al. Building an Encrypted and Searchable Audit Log.[C]. network and distributed system security symposium, 2004.
4. Curtmola R, Garay J A, Kamara S, et al. Searchable symmetric encryption: improved definitions and efficient constructions[C]. computer and communications security, 2006: 79-88.
5. Wang P, Wang H, Pieprzyk J. Threshold Privacy Preserving Keyword Searches[C]. International Conference on Current Trends in Theory and Practice of Computer Science. Springer Berlin Heidelberg, 2008:646-658.
6. Dong J P, Cha J, Lee P J. Searchable Keyword-Based Encryption[J]. Iacr Cryptology Eprint Archive, 2005, 2005.
7. Moataz T, Justus B, Ray I, et al. Privacy-Preserving Multiple Keyword Search on Outsourced Data in the Clouds[C]. Ifip Wg 11.3 Working Conference on Data and Applications Security and Privacy Xxviii. Springer-Verlag New York, Inc. 2014:66-81.
8. Yang Y, Liu X, Deng R. Multi-user Multi-Keyword Rank Search over Encrypted Data in Arbitrary Language[J]. IEEE Transactions on Dependable Secure Computing, 2017, PP(99):1-1.
9. Fu Z, Wu X, Guan C, et al. Toward Efficient Multi-Keyword Fuzzy Search Over Encrypted Outsourced Data With Accuracy Improvement[J]. IEEE Transactions on Information Forensics Security, 2017, 11(12):2706-2716.
10. Wang B, Yu S, Lou W, et al. Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud[C]. INFOCOM, 2014 Proceedings IEEE. IEEE, 2014:2112-2120.
11. Zhang W, Xiao S, Lin Y, et al. Secure Ranked Multi-keyword Search for Multiple Data Owners in Cloud Computing[C]. Ieee/ifip International Conference on Dependable Systems and Networks. IEEE, 2014:276-286.
12. Ameri M H, Delavar M, Mohajeri J, et al. A Key-Policy Attribute-Based Temporary Keyword Search scheme for Secure Cloud Storage[J]. IEEE Transactions on Cloud Computing, 2018, PP(99):1-1.
13. Liang X, Cao Z, Lin H, et al. Attribute based proxy re-encryption with delegating capabilities[C]. International Symposium on Information, Computer, and Communications Security. ACM, 2009:276-286.
14. Cui J, Zhou H, Zhong H, et al. AKSER: Attribute-based Keyword Search with Efficient Revocation in Cloud Computing[J]. Information Sciences, 2017, 423.
15. Hur J, Dong K N. Attribute-Based Access Control with Efficient Revocation in Data Outsourcing Systems[J]. IEEE Transactions on Parallel Distributed Systems, 2011, 22(7):1214-1221.
16. Cui H, Deng R H, Liu J K, et al. Attribute-Based Encryption with Expressive and Authorized Keyword Search[C]. Australasian Conference on Information Security and Privacy. Springer, Cham, 2017:106-126.

17. Kamara S, Papamanthou C, Roeder T, et al. Dynamic searchable symmetric encryption[C]. computer and communications security, 2012: 965-976.
18. Kamara S, Papamanthou C. Parallel and Dynamic Searchable Symmetric Encryption[C]. financial cryptography, 2013: 258-274.
19. Hahn F, Kerschbaum F. Searchable Encryption with Secure and Efficient Updates[C]. computer and communications security, 2014: 310-320.
20. Xia Z, Wang X, Sun X, et al. A Secure and Dynamic Multi-Keyword Ranked Search Scheme over Encrypted Cloud Data[J]. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(2): 340-352.
21. Miers I, Mohassel P. IO-DSSE: Scaling Dynamic Searchable Encryption to Millions of Indexes By Improving Locality[C]. Network and Distributed System Security Symposium. 2017.
22. Seo J H, Cheon J H. Fully Secure Anonymous Hierarchical Identity-Based Encryption with Constant Size Ciphertexts[J]. Iacr Cryptology Eprint Archive, 2011, 2011(2011):215-234.
23. Zhao Y, Chen X, Ma H, et al. A new trapdoor-indistinguishable public key encryption with keyword search[J]. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, vol. 3, no. 1/2, pp. 72-81, 2012.
24. Katz J, Sahai A, Waters B, et al. Predicate encryption supporting disjunctions, polynomial equations, and inner products[C]. theory and application of cryptographic techniques, 2008: 146-162.
25. Byun J W, Rhee H S, Park H A, et al. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data[C]. very large data bases, 2006: 75-83.
26. Wei-Chuen Yau,Raphael C.-W. Phan,Swee-Huay Heng,Bok-Min Goi. Keyword guessing attacks on secure searchable public key encryption schemes with a designated tester[J]. International Journal of Computer Mathematics, 2013,90(12)
27. Tang Q, Chen L. Public-key encryption with registered keyword search[C]. european public key infrastructure workshop, 2009: 163-178.
28. Chen R, Mu Y, Yang G, et al. Dual-Server Public-Key Encryption With Keyword Search for Secure Cloud Storage[J]. IEEE Transactions on Information Forensics and Security, 2016, 11(4): 789-798.
29. Lu Y, Li J. Efficient searchable public key encryption against keyword guessing attacks for cloud-based EMR systems[J]. Cluster Computing, 2018: 1-15.
30. Chen H, Cao Z, Dong Z, et al. SDKSE:A Secure Dynamic Keyword Searchable Encryption Scheme for Email Systems[C]. 2018 3rd International Conference on Security of Smart Cities, Industrial Control System and Communications, 2018.