Search Pattern Leakage in Searchable Encryption: Attacks and New Construction

Chang Liu, Liehuang Zhu, Mingzhong Wang, Yu-an Tan

changliu.bit@gmail.com; liehuangz@bit.edu.cn; wangmz@bit.edu.cn; victortan@yeah.net
Beijing Engineering Research Center of Massive Language Information Processing and Cloud Computing Application, School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China

Abstract

Searching on remote encrypted data (commonly known as *searchable encryption*) has become an important issue in secure data outsourcing, since it allows users to outsource encrypted data to an untrusted third party while maintains the capability of keyword search on the data.

Searchable encryption can be achieved using the classical method called oblivious RAM, but the resultant schemes are too inefficient to be applied in the real-world scenarios (e.g., cloud computing). Recently, a number of efficient searchable encryption schemes have been proposed under weaker security guarantees. Such schemes, however, still leak statistical information about the users' search pattern.

In this paper, we first present two concrete attack methods to show that the search pattern leakage will result in such a situation: an adversary who has some auxiliary knowledge can uncover the underlying keywords of user queries. To address this issue, we then develop a grouping-based construction (GBC) to transform an existing searchable encryption scheme to a new scheme hiding the search pattern. Finally, experiments based on the real-world dataset demonstrate the effectiveness of our attack methods and the feasibility of our construction.

Keywords: search pattern, searchable encryption, cloud computing, fake query, grouping-based construction

December 4, 2013

1. Introduction

Cloud computing has been increasingly applied to outsource data by cloud tenants to diminish the high overhead of data storage and management. In many cases, cloud tenants need to encrypt data before outsourcing them to prevent cloud administrators from accessing sensitive information, such as government documents, hospital records and personal emails. However, data encryption invalidates many data query functions, which will lead to inefficient data utilization. For instance, the cloud service provider cannot respond keyword search query over encrypted data. The purpose of searchable encryption is allowing a user to outsource data to a third party in a secure manner, and then retrieve documents containing the queried keyword. Therefore, searchable encryption plays an important role in cloud computing scenario [16].

Index, an auxiliary structure that accelerates the search process, has been widely studied in information retrieval fields. In general, each entry of the index is formed as a <keyword, document identifiers> tuple, so that all the documents containing the queried keyword can be easily located. We describe the general scenario of searchable encryption as follows: a data owner (e.g., Alice) has a set of documents to outsource to an untrusted third party (e.g., Carol). Alice first builds an index of all the keywords appeared in the documents, and then encrypts both the documents and the index. After that, she outsources the encrypted documents and index to Carol. An authorized data user (e.g., Bob) has a secret key, so that he is able to generate keyword search queries (a.k.a. trapdoors or tokens) by calling a trapdoor function. Once Carol receives a query from Bob, she can search in the index and return the (encrypted) search result[1] to Bob. Then Bob is able to decrypt the search result and retrieve needed documents. Note that if the secret key is shared appropriately (such as the Multi-user SSE scheme in [11]), there can be multiple data users. For example, all employees of a company shared a secret key to perform keyword search on their data.

It has been widely accepted in the literature that both outsourced data and search query should leak as little information as possible to the third party. We note that searchable encryption can be achieved using the classical method called oblivious RAM [19, 14], which attains the optimal security

[1]The search result is a collection of document identifiers whose corresponding documents contain the queried keyword.

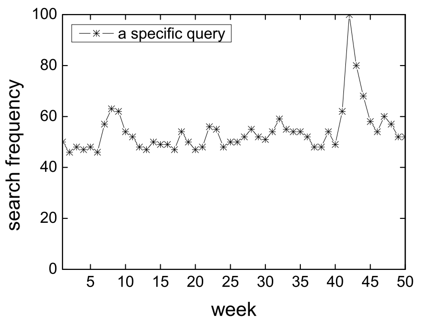
(i.e., nothing leaked to the third party). However, this kind of approaches is impractical due to the poly-logarithmic computation and communication overheads. Therefore a number of searchable symmetric encryption (SSE) schemes (e.g., [22, 13, 9, 11, 23, 18, 17]) have been proposed under weaker security guarantees for efficiency. However, the access pattern and the search pattern are leaked in their schemes. Informally, the access pattern is the information about which documents contain the queried keyword (i.e., the search result) for each of the user queries, while the search pattern is the information about whether any two queries are generated from the same keyword or not. In recent studies, [15] has discussed a concrete attack exploiting the access pattern to disclose the underlying keywords of user queries. However, the potential risks of the search pattern leakage are rarely studied in the literature. As it is noted in [17], a limitation of most known SSE schemes is the leakage of the search pattern. In fact, this limitation also exists in searchable asymmetric encryption schemes (e.g., [8, 6]). For the reasons above, we are motivated to investigate on the search pattern leakage issue and develop a new searchable encryption construction under more rigorous security requirements.

Intuitions. Let's begin with analyzing why search pattern is leaked in SSE schemes. To the best of our knowledge, the query algorithms of existing SSE schemes in the literature are mostly deterministic, which means the same keyword will always generate the same query. In this sense, an adversary can easily judge whether any two queries are generated from the same keyword or not, so as to obtain the users' search pattern. One may apply probabilistic query algorithms to solve this problem. However, simply making use of probabilistic query algorithms [21] still cannot hide the search pattern, because the entry touched in each search process discloses the search pattern as well. In other words, for the same keyword, its corresponding entry in the index must be the same, so that the adversary can disclose users' search pattern just by observing the entries touched in the index during the search process. For this reason, the search pattern is also leaked in searchable asymmetric encryption (a.k.a. public key encryption with keyword search, PEKS), whose query algorithms are probabilistic [8, 6, 20, 12]. Therefore, randomizing query algorithm only contributes to defend outer adversaries but not inner adversaries (e.g., cloud administrators). It is also worthwhile to point out that the access pattern might leak the search pattern in the same way (i.e., the same search result might mean the same queried keyword). We refer the reader to [15] for the details on how to hide the access pattern. The work of this paper focuses on the search pattern leakage caused by the queries and index. A delicate approach to hide the search pattern is launching several fake queries along with the real query, so that the adversary cannot identify the real query. Based on this idea, we develop a new searchable encryption construction where the number of sub-queries (i.e., the real query and the fake queries) for each query is parametrized by a confusion parameter k. We will give detailed description in Section 5.

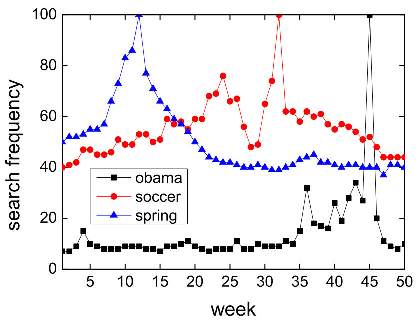
Our contributions. We outline the contributions of this paper as the following:

1. We address the search pattern leakage issue and demonstrate its potential risks in the practical applications by giving two concrete attack methods. In particular, with our attack methods, an adversary who has obtained users' search pattern can effectively uncover the underlying keywords of the user queries under the help of some public available knowledge.
2. We present a grouping-based construction (GBC) which transforms an existing index-based searchable encryption scheme to a new scheme hiding the search pattern. GBC is designed to be independent from the underlying searchable encryption scheme, so that most previous schemes [13, 9, 11, 23, 18, 17] can be used in GBC.
3. We prove that the resultant scheme of GBC satisfies a stronger security guarantee than any existing searchable encryption scheme. GBC reduces the search pattern leakage to the group pattern leakage.
4. Based on the real-world dataset [5], we test the performance of the proposed attack methods and the proposed construction. The experiment results indicate the effectiveness and feasibility of proposed attack methods and construction.

Organization of the paper. The remainder of the paper is organized as follows: Section 2 briefly surveys the motivations of hiding the search pattern. Section 3 introduces some preliminaries. We formalize two attack methods in Section 4. In Section 5 we will describe the grouping-based construction. Section 6 shows experimental studies for evaluating the performance of the proposed attack methods and construction. We review related work in Section 7 and conclude the paper in Section 8.



(a)



(b)

Figure 1: Search frequency[2]over time

2. Motivations

Throughout this paper, we treat the third party Carol as the adversary, which is consistent with most previous work [22, 13, 8, 9, 11, 23, 18, 17]. Carol behaves "honest-but-curious" in the searchable encryption scheme. On one hand, Carol follows the operations required in the scheme. On the other hand, Carol tries to deduce as much private information as possible by utilizing all kinds of attack methods. Carol has the ability of eavesdropping the network, so that she has access to all encrypted documents, index and queries. To make more effective attacks, Carol will draw support from some auxiliary knowledge, which can be legally obtained from other channels. Now we demonstrate that the search pattern leakage can lead to the keyword privacy leakage through several examples below.

Once the search pattern is leaked to Carol, the occurrence frequency of each query is known to Carol. Intuitively, the most vulnerable keywords are those of highest occurrence frequencies. For example, the keyword "Thanksgiving" is the hot keyword on Thanksgiving Day. Since Carol is able to identify the query with the highest occurrence frequency on Thanksgiving

[2]The search frequencies offered by Google Trends are normalized and displayed on a scale from 0 to 100, rather the absolute frequency numbers. We refer the reader to the site help of [2] for more details on how to normalize and scale data. In this paper, when we refer search frequencies (or occurrence frequencies) which are used for matching with the data in Google Trends, we assume these frequency numbers have been normalized and scaled properly.

Day, she knows the underlying keyword of that query is "Thanksgiving".

To make a more general attack, Carol records the occurrence frequency of a specific query over an interval of time. As it is shown in Figure 1(a), Carol records the occurrence frequency of a specific query in each week of the year 2012. She can also obtain auxiliary knowledge from a public web facility based on Google Search, called Google Trends [2] which shows how often a particular search-term is entered by users. For illustration, we show three sample keywords in Figure 1(b) and let the time interval be consistent with that recorded by Carol. What Carol needs to do is searching the best matched keyword by applying some methods of similarity measurement (e.g., Euclidean distance).

In some cases, data users might have specific background (such as IT, medicine, etc.), so the search habit of these users has discrepancy when compared with that of the general users. To attack the users with specific background, Carol needs to adjust the auxiliary knowledge accordingly. This is possible because Google Trends also offers statistics under variant categories, which can be treated as user backgrounds. Therefore the privacy of user searches will always be compromised once Carol obtains users' search pattern and an appropriate auxiliary knowledge. This paper just takes Google Trends for illustration, but never eliminates the possibility of the adversary using other kinds of auxiliary knowledge. The more similar the auxiliary knowledge is to the users' search habit, the more successful the attack will be.

Currently, more and more studies on users' search habits are open to the public (e.g., [1, 2, 3]), which indeed facilitates the attacks we have mentioned above. Consequently, the search pattern leakage will grievously threaten the keyword privacy. We will formalize two keyword attack methods in Section 4.

3. Preliminaries

Definition 1 defines the index-based searchable encryption, which covers both searchable symmetric encryption and searchable asymmetric encryption.

Definition 1 (Searchable Encryption)**.** *An index-based Searchable Encryption (SE) scheme is a tuple of 6 algorithms* $\mathsf{SE} = (\mathsf{KeyGen}, \mathsf{BuildIndex}, \mathsf{Encryption}, \mathsf{Query}, \mathsf{Search}, \mathsf{Decryption})$*:*

1. $\mathsf{KeyGen}(1^\lambda)$*: The key generation algorithm takes a security parameter λ as input, it outputs a secret key K.*
2. $\mathsf{BuildIndex}(\mathcal{D})$*: The index building algorithm takes a document collection $\mathcal{D} = \{D_1, ..., D_n\}$ as input, it outputs an index $\mathcal{I}$.*
3. $\mathsf{Encryption}(\mathcal{D}, \mathcal{I}, K)$*: The encryption algorithm takes a document collection $\mathcal{D}$, an index $\mathcal{I}$ and a secret key K as input, it outputs an encrypted document collection $\mathcal{C} = \{C_1, ..., C_n\}$ and a secure index $\mathcal{SI}$.*
4. $\mathsf{Query}(w, K)$*: The query algorithm takes a keyword w and a secret key K as input, it outputs an encrypted query q_w.*
5. $\mathsf{Search}(q_w, \mathcal{SI})$*: The search algorithm takes a query q_w and a secure index $\mathcal{SI}$ as input, it outputs a collection of document identifiers whose corresponding data file containing the keyword w, which denoted as $\mathcal{R}(w) = \{\mathsf{id}(w,1), ..., \mathsf{id}(w,p)\}$, where $\mathsf{id}(w,i)(1 \leq i \leq p)$ denotes the i-th identifier in $\mathcal{R}(w)$.*
6. $\mathsf{Decryption}(C_i, K)$*: The decryption algorithm takes an encrypted data file $C_i \in \mathcal{C}$ and a secret key K as input, it outputs D_i.*

Before the formal definition of the search pattern and the access pattern, we first give the definition of *history*.

Definition 2 (History)**.** *Let $\mathcal{D}$ be a document collection. An n-query history over $\mathcal{D}$ is a tuple $\mathcal{H} = (\mathcal{D}, \mathbf{w})$ where $\mathbf{w} = \{w_1, ..., w_n\}$ is a vector of underlying keywords of the n queries.*

Elements in the *history* are what users expect to hide from the adversary.

Definition 3 (Search Pattern)**.** *The search pattern over an n-query history $\mathcal{H} = (\mathcal{D}, \mathbf{w})$ is a $n \times n$ symmetric binary matrix $\tau_{\mathcal{H}}$ such that for $1 \leq i, j \leq n$, $\tau_{\mathcal{H}}[i][j] = 1$ if $w_i = w_j$, and 0 otherwise.*

Definition 4 (Access Pattern)**.** *The access pattern over an n-query history $\mathcal{H} = (\mathcal{D}, \mathbf{w})$ is a a tuple $\varphi_{\mathcal{H}} = (\mathcal{R}(w_1), ..., \mathcal{R}(w_n))$.*

Definition 5 defines the group pattern that is revealed by group-based construction (described in Section 5.2), where keywords are divided into groups.

Definition 5 (Group Pattern)**.** *The group pattern over an n-query history $\mathcal{H} = (\mathcal{D}, \mathbf{w})$ is a $n \times n$ symmetric binary matrix $\rho_{\mathcal{H}}$ such that for $1 \leq i, j \leq n$, $\rho_{\mathcal{H}}[i][j] = 1$ if w_i and w_j belong to the same subset, and 0 otherwise.*

Definition 6 explains the concept of CKA2-security from [10, 11], which has been considered as "state of the art".

Definition 6 (CKA2-security)**.** *Let* $\mathsf{SE} = (\mathsf{KeyGen}, \mathsf{BuildIndex}, \mathsf{Encryption}, \mathsf{Query}, \mathsf{Search}, \mathsf{Decryption})$ *be an index-based searchable encryption scheme and let* $\mathcal{L}$ *be a stateful algorithm. For an adversary* $\mathcal{A}$ *and a simulator* $\mathcal{S}$*, we consider the following probabilistic experiments:*

$\mathbf{Real}_{\mathcal{A}}^{\mathsf{SE}}(\lambda)$*: the challenger runs* $\mathsf{KeyGen}(1^\lambda)$ *to obtain a key* K*.* $\mathcal{A}$ *chooses* $\mathcal{D}$ *and receives* $(\mathcal{C}, \mathcal{SI})$ *such that* $(\mathcal{C}, \mathcal{SI}) \leftarrow \mathsf{Encryption}(\mathcal{D}, \mathcal{I}, K)$ *where* $\mathcal{I}$ *is the output of* $\mathsf{BuildIndex}(\mathcal{D})$*. Then* $\mathcal{A}$ *makes a polynomial number of adaptive queries and for each queried keyword* w *receives a query* $q_w \leftarrow \mathsf{Query}(w, K)$ *from the challenger. Finally,* $\mathcal{A}$ *returns a bit* b *that is output by the experiment.*

$\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{\mathsf{SE}}(\lambda)$*:* $\mathcal{A}$ *outputs* $\mathcal{D}$*. Given* $\mathcal{L}(\mathcal{D})$*,* $\mathcal{S}$ *generates and sends* $(\mathcal{C}, \mathcal{SI})$ *to* $\mathcal{A}$*. Then* $\mathcal{A}$ *makes a polynomial number of adaptive queries. For each queried keyword* w_i*, let* $\mathcal{H} = (\mathcal{D}, \mathbf{w})$ *denote the* i*-query history where* $\mathbf{w} = \{w_1, ..., w_i\}$*,* $\mathcal{S}$ *is given* $\mathcal{L}(\mathcal{H})$ *and returns an appropriate query* q_{w_i}*. Finally,* $\mathcal{A}$ *returns a bit* b *that is output by the experiment.*

We say that SE *is* $\mathcal{L}$*-secure against adaptive chosen-keyword attacks if for all* PPT *adversary* $\mathcal{A}$*, there exists a* PPT *simulator* $\mathcal{S}$ *such that*

$$|\Pr[\mathbf{Real}_{\mathcal{A}}^{\mathsf{SE}}(\lambda) = 1] - \Pr[\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{\mathsf{SE}}(\lambda) = 1]| \leq \mathrm{negl}(\lambda).$$

4. Two Attack Methods

In this section, we formalize two attack methods which exploit the search pattern to uncover the underlying keywords of user queries. Let q be a specific query that Carol wants to attack, Carol records the occurrence frequency of q in each period of time (e.g., day, week, month, etc.), which we denote as f_q^i for $1 \leq i \leq p$ where p is the total number of record items (e.g., $p = 50$ in Figure 1(a)). Thus Carol gets a frequency vector of q denoted as $V_q = (f_q^1, ..., f_q^p)$. Let $\mathcal{K}$ denote a dictionary of keywords and m denote the size of $\mathcal{K}$. In our attack methods, we assume Carol has auxiliary knowledge about users' search habits (e.g., Google Trends [2]). Let $V_{w_i} = (f_{w_i}^1, ..., f_{w_i}^p)$ denote the auxiliary frequency vector of the keyword $w_i \in \mathcal{K} (i \in \{1, ..., m\})$, where the time interval is consistent with V_q. Let $V = \{V_{w_1}, ..., V_{w_m}\}$ denote the set of all auxiliary frequency vectors. Let $Dist(V_q, V_{w_i})$ be a function measuring the similarity of V_q and V_{w_i} (e.g., Euclidean Distance, Cosine Distance, etc.).

Coral tries to identify the underlying keyword of q, which can be easily achieved using the following attack method. We call this attack method *The General Attack* and denote it as $\mathsf{ATK}^{General}$:

- $\mathsf{ATK}^{General}(V_q, V)$:

```
1. set $i^* = \arg\min_{i \in \{1,...,m\}} Dist(V_q, V_{w_i})$.
2. return $w_{i^*}$.
```

We have mentioned that users might have specific background while Carol does not know it in the beginning. To deal with this situation, we present *The Adaptive Attack* (denoted as $\mathsf{ATK}^{Adaptive}$) which dynamically adjust the auxiliary knowledge based on the previous rounds of attack. For example, if Carol has obtained five keywords "HIV", "leukocyte", "winter", "virus" and "glucose" for the first five rounds of attack, she will guess the users are medical-related and update the auxiliary frequency vector to an associated version. Here, we assume that the auxiliary knowledge offers statistics of user searches under different categories which refer to different user backgrounds.

Let $C = \{c_1, ..., c_r\}$ be the set of all possible categories in the auxiliary knowledge. Let $V_{w_i,c_j} = (f^1_{w_i,c_j}, ..., f^p_{w_i,c_j})$ $(1 \leq i \leq m, 1 \leq j \leq r)$ be the auxiliary frequency vector of w_i under the category c_j. Let $V_{c_j} = \{V_{w_1,c_j}, ..., V_{w_m,c_j}\}$ denote the set of all auxiliary frequency vectors under the category c_j. We assume each keyword in $\mathcal{K}$ is also labeled with a category in C according to the semantics of the keyword. We set a weight value on each of the category (denoted as $v_j (1 \leq j \leq r)$), which equals to the proportion of resultant keywords labeled with that category in the previous rounds of attack. *The Adaptive Attack* works as follows:

- $\mathsf{ATK}^{Adaptive}(\{V_{q_i}\}_{i=1,2,...}, \{V_{c_j}\}_{j \in \{1,...,r\}})$:

```
1. for $1 \leq j \leq r$, set $v_j = 0$.
2. randomly choose $V_{current}$ from $\{V_{c_j}\}_{j \in \{1,...,r\}}$.
3. set $\mathsf{ctr} = 1$.
4. output $w_{i^*} = \mathsf{ATK}^{General}(V_{q_{\mathsf{ctr}}}, V_{current})$.
5. let $c_{j^*}$ be the category labeled on $w_{i^*}$.
6. for $1 \leq j \leq r (j \neq j^*)$, let $v_j = \frac{v_j \cdot (\mathsf{ctr} - 1)}{\mathsf{ctr}}$.
```

```
 7. let $v_{j^*} = \frac{v_{j^*} \cdot (\mathsf{ctr}-1)+1}{\mathsf{ctr}}$.
 8. set $j_{max} = \arg\max_{j \in \{1,\ldots,r\}} v_j$.
 9. let $V_{current} = V_{c_{j_{max}}}$.
10. let $\mathsf{ctr} = \mathsf{ctr} + 1$
11. goto 4.
```

The Adaptive Attack might be inaccurate in the first few rounds of attack, since Carol have no idea about the exact user background in the beginning. With the increase of attack rounds, Carol can gradually identify the exact user background. We will show the attack accuracy in Section 6.

5. Our construction

In this section, we present new searchable encryption constructions to defend the previous attacks. For better understanding, we will first introduce a straightforward construction, and then our main construction. Both constructions are based on an existing index-based searchable encryption scheme. Before describing our constructions in detail, we introduce some additional notations. Let $\mathcal{W}$ ($\mathcal{W} \subseteq \mathcal{K}$) be the list of all distinct keywords contained in document collection $\mathcal{D}$ in alphabetical order and $|\mathcal{W}|$ be its size. Let w_i denote the i-th keyword in $\mathcal{W}$. Let k be the confusion parameter used in our constructions. Let SE be a specific index-based searchable encryption scheme that satisfies Definition 1.

5.1. The straightforward construction

In the straightforward construction, the query generated by Bob is a collection of k sub-queries, which includes one sub-query of the real keyword Bob wants to search for and $k-1$ sub-queries of randomly selected keywords. To prevent Carol from identifying the sub-query of the real keyword, Bob needs to place this sub-query at a random position in the query structure. When Carol receives a query (i.e., a collection of k sub-queries) from Bob, for each sub-query, she calls the search algorithm to obtain the sub-result (assumed to be encrypted). Then she can get the final search result in the form of k sub-results with the consistent order of the sub-queries, and send it to Bob. Since Bob knows the correct position of the real sub-query, he extracts the sub-result of the real sub-query and deletes other sub-results.

```
1. KeyGen($1^\lambda$): output $K \leftarrow$ SE.KenGen($1^\lambda$)
2. BuildIndex($\mathcal{D}$): output $\mathcal{I} \leftarrow$ SE.BuildIndex($\mathcal{D}$)
3. Encryption($\mathcal{D}, \mathcal{I}, K$): output $(\mathcal{C}, \mathcal{SI}) \leftarrow$ SE.Encryption($\mathcal{D}, \mathcal{I}, K$)
4. Query($k, w, K, \mathcal{W}$):
   (a) randomly choose $k-1$ keywords:
       $w_{i_1}, ..., w_{i_{k-1}} \in \mathcal{W} \backslash w$
   (b) randomly choose $b \in \{1, ..., k\}$
   (c) for $1 \leq j \leq b-1$:
           let $sq_j \leftarrow$ SE.Query($w_{i_j}, K$)
   (d) let $sq_b \leftarrow$ SE.Query($w, K$)
   (e) for $b+1 \leq j \leq k$:
           let $sq_j \leftarrow$ SE.Query($w_{i_{j-1}}$)
   (f) output $(b, \mathcal{Q} = \{sq_1, ..., sq_k\})$
5. Search($\mathcal{Q}, \mathcal{SI}$):
   (a) for each $sq_i$ in $\mathcal{Q}$:
           let $\mathcal{R}_i \leftarrow$ SE.Search($sq_i, \mathcal{SI}$)
   (b) output $\mathcal{R} = \{\mathcal{R}_1, ..., \mathcal{R}_k\}$
6. Extract($\mathcal{R}, b$): output $\mathcal{R}_b$
7. Decryption($C_i, K$): output $D_i \leftarrow$ SE.Decryption($C_i, K$)
```

Figure 2: The straightforward construction

Bob then uses his secret key to decrypt the sub-result and finishes the query process.

We use the Extract algorithm to represent the process of extracting the real sub-result from the search result structure. The details of the straightforward construction are shown in Figure 2.

Analysis. The main idea of the straightforward construction is blending the sub-query of the real keyword with several sub-queries of fake keywords which are randomly selected in $\mathcal{W}$. In this case, Carol cannot identify the real sub-query, thus the search pattern is hidden simultaneously. However, we observe that the construction is not robust in some cases. It is possible that some particular keywords are queried much more times than other keywords. In this case, Carol is able to figure out the real sub-queries of these keywords by performing set-intersection operation. For example, let the confusion parameter k be 3. Consider the query sequence $(w_1, w_2, w_1, w_1, w_1)$ whose corresponding queries received by Carol are, say

$$\mathcal{Q}_1 = \{sq(w_7), sq(w_1), sq(w_{38})\}$$

$$\mathcal{Q}_2 = \{sq(w_{34}), sq(w_{91}), sq(w_2)\}$$
$$\mathcal{Q}_3 = \{sq(w_1), sq(w_{51}), sq(w_{67})\}$$
$$\mathcal{Q}_4 = \{sq(w_8), sq(w_{77}), sq(w_1)\}$$
$$\mathcal{Q}_5 = \{sq(w_1), sq(w_{12}), sq(w_{83})\}$$

Through set-intersection operation, Carol can easily observe that $sq(w_1)$ appears four times among the five queries, so she believes that the real sub-query of the first, third, fourth and fifth query is $sq(w_1)$. Thus the view of Carol can be described as $View = \{\mathcal{Q}_1 = \{sq(w_1)\}, \mathcal{Q}_2 = \{sq(w_{34}), sq(w_{91}), sq(w_2)\}, \mathcal{Q}_3 = \{sq(w_1)\}, \mathcal{Q}_4 = \{sq(w_1)\}, \mathcal{Q}_5 = \{sq(w_1)\}\}$, which leaks Bob's search pattern to a certain degree. One may think that $sq(w_1)$ can also appear in a query when it is selected as a fake sub-query. Such a case, however, occurs with relatively low probability when (1) the keyword set size is large, or (2) k value is small.

5.2. Our main construction: the grouping-based construction (GBC)

To address the drawbacks of the straightforward construction, we now present our main construction, which we call the grouping-based construction (GBC). In GBC, we add an additional algorithm named Dividing, which divides $\mathcal{W}$ into $|\mathcal{W}|/k$ subsets $\{S_1, ..., S_{|\mathcal{W}|/k}\}$. Here, we assume $|\mathcal{W}|$ is an integral multiple of k.[3] Once a keyword $w \in \mathcal{W}$ is queried, each keyword that appears in the same subset with w will be selected, so that the final query also contains one real sub-query and $k-1$ fake sub-queries.

Clearly, the Dividing algorithm should make the frequency distribution of the resultant subsets as uniform as possible in order to minimize the leakage. Note that the adversary tries to use the distribution of searches (i.e., the auxiliary knowledge) to perform attacks, users can also use it in hiding the search pattern. In GBC, the Dividing algorithm takes the auxiliary knowledge

[3]For the case $|\mathcal{W}|$ is not an integral multiple of k, one can appropriately pad $\mathcal{W}$ with several keywords.

```
1. $\mathsf{KeyGen}(1^\lambda)$: output $K \leftarrow \mathsf{SE.KenGen}(1^\lambda)$
2. $\mathsf{BuildIndex}(\mathcal{D})$: output $\mathcal{I} \leftarrow \mathsf{SE.BuildIndex}(\mathcal{D})$
3. $\mathsf{Encryption}(\mathcal{D}, \mathcal{I}, K)$: output $(\mathcal{C}, \mathcal{SI}) \leftarrow \mathsf{SE.Encryption}(\mathcal{D}, \mathcal{I}, K)$
4. $\mathsf{Dividing}(k, \mathcal{W}, V)$:
   (a) sort all the keywords in $\mathcal{W}$ by their total occurrence frequency
       according to $V$. Let $\mathcal{W}'$ be the ranked list and $w'_i$ be the $i$-th
       keyword in $\mathcal{W}'$
   (b) divide $\mathcal{W}$ into $k$ sections as follows:
       for $1 \leq i \leq k$:
           set $\mathcal{W}'_i = \{w'_{(i-1)\cdot k+1}, ..., w'_{(i-1)\cdot k+k}\}$
   (c) initialize $|\mathcal{W}|/k$ empty subsets $S_1, ..., S_{|\mathcal{W}|/k}$, for $1 \leq i \leq$
       $|\mathcal{W}|/k$:
           for $1 \leq j \leq k$:
               random select $w$ from $\mathcal{W}'_j$
               insert $w$ into $S_i$
               delete $w$ from $\mathcal{W}'_j$
   (d) output $\{S_i\}_{i \in \{1,...,|\mathcal{W}|/k\}}$
5. $\mathsf{Query}(k, w, K, \{S_i\}_{i \in \{1,...,|\mathcal{W}|/k\}})$:
   (a) let $S_q$ be the subset containing $w$ and $S_q[j]$ be the $j$-th element
       in $S_q$. Suppose $w$ is the $b$-th element in $S_q$.
   (b) for $1 \leq j \leq k$:
           let $sq_j \leftarrow \mathsf{SE.Query}(S_q[j], K)$
   (c) output $(b, \mathcal{Q} = \{sq_1, ..., sq_k\})$
6. $\mathsf{Search}(\mathcal{Q}, \mathcal{SI})$:
   (a) for each $sq_i$ in $\mathcal{Q}$:
           let $\mathcal{R}_i \leftarrow \mathsf{SE.Search}(sq_i, \mathcal{SI})$
   (b) output $\mathcal{R} = \{\mathcal{R}_1, ..., \mathcal{R}_k\}$
7. $\mathsf{Extract}(\mathcal{R}, b)$: output $\mathcal{R}_b$
8. $\mathsf{Decryption}(C_i, K)$: output $D_i \leftarrow \mathsf{SE.Decryption}(C_i, K)$
```

Figure 3: The grouping-based construction (GBC)

V as one of the inputs (as it is shown in Figure 3). We first sort the keyword list W by keyword frequency according to V. Then we divide the ranked keyword list into k sections. We randomly extract one keyword from each of the sections and get a subset of k keywords. In this way, we can finally obtain $|\mathcal{W}|/k$ subsets with a relatively uniform frequency distribution.

The details of GBC are shown in Figure 3. Unlike the straightforward construction, which randomly selects fake sub-queries from the whole $\mathcal{W}$, GBC divides $\mathcal{W}$ into a number of subsets so that the generated queries will always be the same when the queried keywords belong to the same subset. This property avoids Carol's attack by performing set-intersection operation. For illustration, let k be 3 and suppose w_1 and w_2 belong to the subset (w_{33}, w_1, w_{74}) and (w_{85}, w_{41}, w_2) respectively. Let's consider the following view of Carol: $View = \{\mathcal{Q}_1 = \{sq(w_{33}), sq(w_1), sq(w_{74})\}, \mathcal{Q}_2 = \{sq(w_{85}), sq(w_{41}), sq(w_2)\}, \mathcal{Q}_3 = \{sq(w_{33}), sq(w_1), sq(w_{74})\}, \mathcal{Q}_4 = \{sq(w_{33}), sq(w_1), sq(w_{74})\}, \mathcal{Q}_5 = \{sq(w_{33}), sq(w_1), sq(w_{74})\}\}$. Although $\mathcal{Q}_1$, $\mathcal{Q}_3$, $\mathcal{Q}_4$ and $\mathcal{Q}_5$ are identical, it is not rational for Carol to guess the real sub-queries of the first, third, fourth and fifth queries are (1) $sq(w_1)$, because $sq(w_{33})$ and $sq(w_{74})$ also satisfy this situation; or (2) identical, because different keywords will generate the same query if these keywords are in the same subset.

As a result, the resultant searchable encryption scheme of GBC satisfies a stronger security property. Specifically, the search pattern in leakage function (see Definition 6) is replaced by the group pattern. We offer the formal proof in Appendix. In Section 6 we will show that this leakage reduction can successfully defend the attacks proposed in this paper. One of the main contributions of this paper is to discover the potential risks of the search pattern leakage, but we are not able to guarantee that replacing the search pattern with the group pattern is secure enough. We leave a better understanding of the group pattern leakage as an open problem.

Note that a data user has to maintain all the subsets locally to issue queries. There are many ways for a data owner to share all the subsets to data users. For example, (s)he can encrypt those subsets using the shared secret key and upload it to the server. A data user just downloads and decrypts it at a setup phase.

For both the straightforward construction and GBC, $k = 1$ implies an unchanged searchable encryption scheme SE, while $k = |\mathcal{W}|$ means each query contains sub-queries of all the keywords. The larger k is, the stronger security attained, but the higher communication and computation overhead required. Therefore, it is important to choose k properly according to the corresponding applications.

6. Experiments

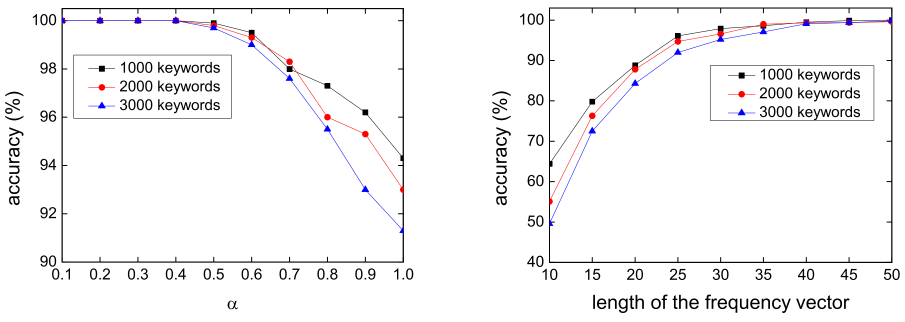
In this section, the proposed attack methods and construction are tested. Our test programs were implemented by Python-2.7.3. All experiments were performed on a computer with Intel Core i5-2400 CPU 3.10GHz and 4.00GB RAM. The operation system was Windows 7 Professional. Each data point presented in the experiment results was the mean of 10 executions.

In our experiments, all keywords were selected from Enron email dataset [5], which is a real-world dataset containing a total of about 500000 messages of about 150 users. We selected a subset of the emails as our corpus. It contains 96107 messages from the "Sent Mail" directories. The total number of distinct words in the corpus is 122426. We ranked these words with occurrence frequency and chose the top 3000 words as our keyword set. Here, all English stopwords [4] such as "a", "the" and "about" had been filtered away.

6.1. Performance of our attack methods

To demonstrate the feasibility and effectiveness of our attack methods, we tested their attack accuracy[4] under different parameter settings. We assumed that Carol had recorded the occurrence frequency of each query launched by users for consecutive weeks in the year 2011, thus she had obtained the frequency vector of each query. The length of the frequency vector was equal to the number of recorded weeks. Carol's auxiliary knowledge on statistics of users' search histories were extracted from Google Trends [2], which contains the frequency vectors of all the keywords in the keyword set. It is obvious that the attack accuracy will be 1 if the frequency vectors of users' queries perfectly match the auxiliary knowledge. However, in an actual scenario, the scale of users may be much smaller than Google's, thus the statistics of the real users' queries will be inevitably diverse from the auxiliary knowledge. Considering the significant differences between particular users' query behaviors (which is also noted by [15]), and no real-world query set on the Enron dataset has been published to the best of our observation, similar with the simulation of [15], we simulate the frequency vector of the user query of a

[4]The attack accuracy represents the proportion of keywords that are successfully attacked by the adversary.



(a) (b)

Figure 4: Attack accuracy of *The General Attack* for different choices of keyword set size.

particular keyword through adding Gaussian noise $\mathcal{N}(0, \alpha \cdot \sigma^2)$ to the frequency vector of this keyword in the auxiliary knowledge. Here, σ^2 is the variance of the frequency vector and α is a constant representing the noise level.[5]

Figure 4(a) shows the attack accuracy of *The General Attack* under different keyword set sizes and α values. In the test, the length of the frequency vector was fixed at 52 (i.e., 52 weeks in the year 2011), and the keyword set size was chosen to be 1000, 2000 and 3000 respectively. We can see that the attack accuracy was almost 100% when $\alpha \leq 0.5$ and started to decrease when $\alpha > 0.5$. The results indicate that *The General Attack* is quite accurate if data users' searches are well consistent with Google's statistic, and the attack is also successful if data users' searches have large deviation to Google's statistics since the attack accuracy was approximately 92% even when $\alpha = 1$. With the increase of keyword set size, the attack accuracy slightly decreased. This is because the larger size of keyword set inevitably incurs the higher probability of mismatch.

Figure 4(b) shows the attack accuracy of *The General Attack* under different keyword set sizes and lengths of the frequency vector. In this test, the α value was set at 0.5 and the keyword set size was chosen to be 1000, 2000 and 3000 respectively. We can see that the attack accuracy grew with the increase of the frequency vector length and reached 90% when the vector

[5]The larger α value implies the larger noise energy.

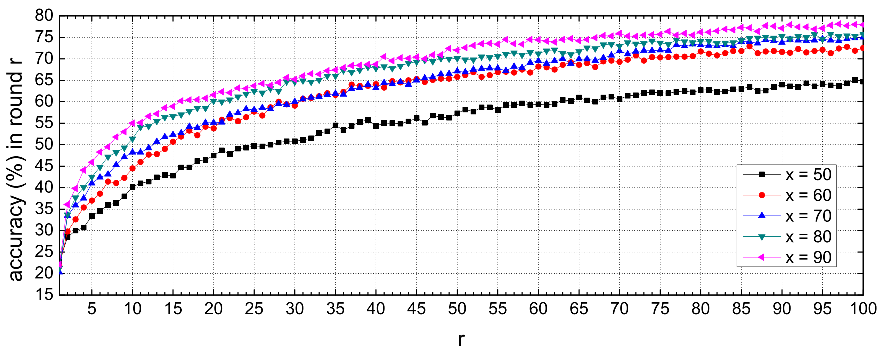


Figure 5: Attack accuracy of *The Adaptive Attack* for different choices of x.

length came to 25. In addition, the attack accuracy was higher than 50% even when the vector length was as short as 10. As with the previous test, the attack accuracy slightly decreased when the keyword set size increased.

To test the attack accuracy of *The Adaptive Attack*, we labeled 1000 keywords with six categories, which were "science", "health", "games", "sports", "food" and "non-classified". We also downloaded the associated frequency vectors under each of the six categories from Google Trends, which formed Carol's auxiliary knowledge. In this experiment, data users were assumed to be of a specific category. To simulate the users' searches, we assumed that users of a specific category searched the keywords of this very category with higher probability than keywords of other categories. We generated a sequence of 100 keywords, where x of them were labeled with the same category as the users. In the test, x was set to be 50, 60, 70, 80 and 90 respectively. The noise parameter α and the length of the frequency vector were set to be 0.5 and 52 respectively. According to *The Adaptive Attack* we have described in Section 4, during each round of the attack process Carol adjusts the utilized frequency vectors based on the previous rounds. From Figure 5 we can see that the attack accuracy in the first round was only about 20%, that is because Carol had no idea on the users' category in the beginning, thus the version of the auxiliary knowledge she used was very likely to mismatch the users' category. With the increase of rounds, Carol has significant probability to figure out the users' exact category, so that the attack accuracy in the 100th round came to about 65% when $x = 50$ and near

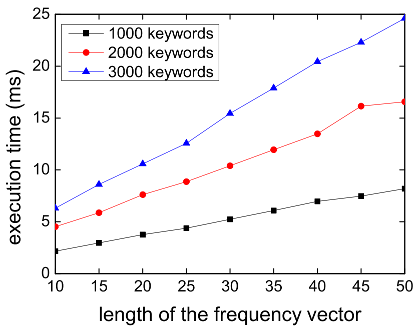


Figure 6: Execution time for search the best match.

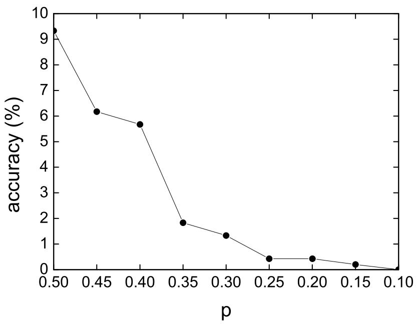


Figure 7: The attack accuracy of *The General Attack* for different p value.

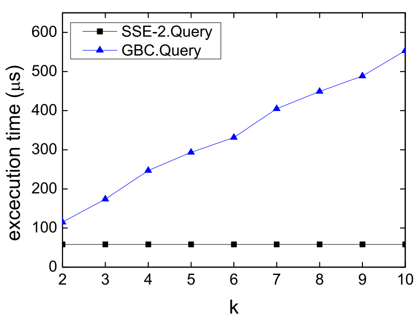
80% when $x = 90$. The larger x implies the higher speciality of user searches, so that Carol can identify the exact user category more easily. That's why the attack accuracy was higher when x was larger.

The experiment results suggest significant distinctions among the frequency vectors of each keyword, thus again we emphasize that any searchable encryption scheme leaking users' search pattern is vulnerable to the proposed attack methods.

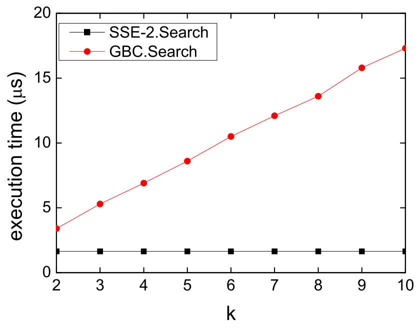
Figure 6 shows the execution time for searching the best match among the auxiliary frequency vectors under different lengths of the frequency vector. The keyword set size was set to be 1000, 2000 and 3000 respectively. We can see that although the time cost was linear to the length of the frequency vector and the number of keywords, it was in millisecond level, which demonstrates the efficiency of our attack methods.

6.2. Performance of GBC

This section evaluates the performance of GBC. In GBC, a user query is a set of $k-1$ fake sub-queries and one real sub-query. In order to apply *The General Attack*, Carol has to first guess the real sub-query for each of the user queries. Figure 7 reports the attack accuracy of *The Adaptive Attack* under a probability p of Carol correctly guessing the real sub-query. In the test, the keyword set size, the length of the frequency vector and the noise parameter α were 3000, 52 and 0.5 respectively. From Figure 7 we can see that the attack accuracy was approximately as low as 9% even Carol could correctly guess the real sub-query with a probability of 0.5. The attack accuracy was



(a)



(b)

Figure 8: Average query and search execution time in SSE-2 and GBC.

close to 0 when $p \leq 0.25$. Note that we didn't make any assumption of the approaches Carol might use to guess the real sub-query.

Obviously, the p value will be 1 if the search pattern is leaked to Carol. In GBC, only the group pattern is leaked, thus the p value would ideally be $1/k$, which means that Carol randomly chooses a sub-query as the "real" sub-query. The p value might slightly higher than $1/k$ if Carol is able to deduce some useful information from the group pattern. As the open problem we have discussed in Section 5, to figure out the actual p value one should quantify the leakage of the group pattern. Nevertheless, our experiment indeed demonstrates that GBC is a fairly effective way to defend the attack. We empirically suggest to choose k no less than 3.

We also evaluated the computation overhead of GBC. In our experiments, the SSE-2 scheme in [11] was chosen as the underlying searchable encryption scheme in GBC. Figure 8 reports the execution time for the Query and the Search algorithms under different choices of k value. Figure 8(a) shows that in GBC the execution time for the Query algorithm was linear to the k value. In addition, the time costs were about k times the execution time in SSE-2. Nevertheless, the Query algorithms of GBC quite time-saving, since the execution time was less than 1ms even when $k = 10$. In Figure 8(b), similarly, the execution time for the Search algorithm was linear to the k value, and it was about k times the execution time in SSE-2. In SSE-2, index was built using hash table, which achieved $O(1)$ look-up time. Therefore, the search time of GBC is fairly short even with a big k.

Besides the Query and the Search algorithms, the Dividing algorithm and the Extract algorithms also import computation overhead. However, the Dividing algorithm only needs executing once at the setup phase for each user, and the execution time of the Extract algorithms can be ignored. In conclusion, the additional computation overhead induced by GBC is acceptable. Obviously, the additional communication overhead (i.e., the fake queries and results) is also small when compared with the cost of outsourcing and retrieving the data files.

7. Related Work

The problems of searching on remote encrypted data have been widely studied in the literature, most of which focus on enhancing privacy guarantees and optimizing efficiency. The classical method proposed by Goldreich and Ostrovsky [19, 14], which called oblivious RAM, can resolve the problem without leaking any information to the third party. However, standing in the perspective of practical applications, such schemes are unacceptable due to their poly-logarithmic computation and communication overheads. A number of searchable encryption schemes [22, 13, 8, 9, 11, 23, 18, 17] have been proposed under weaker security guarantees for efficiency. Specifically, Song et al. [22] gave a practical solution attaining search time that is linear to the data size. This construction is not secure against statistical analysis, since the adversary can obtain the distribution information of the underlying plaintext through statistic approaches. To formalize security, Goh [13] formulated a security model as semantic security against adaptive chosen keyword attack (IND-CKA) and a slightly stronger IND2-CKA. He also developed an IND-CKA secure index called Z-IDX which utilizes Bloom filter [7] to build an index for each data file. The overhead of Z-IDX for testing whether a keyword belongs to a data file is $O(1)$, thus searching on the whole file collection needs $O(n)$ time. The security definition in [9] is similar with IND2-CKA except for the requirement that trapdoors should not leak any information of the queried keywords. As further related work, Curtmola et al. [11] put forward stronger security definitions. Their security definition requires nothing is leaked more than the length of the documents, the identifiers of the documents, the access pattern and the search pattern. In addition, both non-adaptive and adaptive adversaries are considered in their work.

The approaches mentioned above are in the scope of searching on symmetric key encrypted data, which thus called searchable symmetric encryption (SSE). For application sake, another research field of searchable encryption focuses on the public key setting. As pioneer work, Boneh et al. [8] proposed a searchable encryption scheme called PEKS (i.e., Public-key Encryption with Keyword Search), where trapdoor function is probabilistic. This property seems to contribute to hiding the search pattern. Unfortunately, not only trapdoors, but index and search outcomes leak the search pattern, which we have discussed in Section 1.

With the exception of oblivious RAM, all proposed searchable encryption schemes leak the search pattern. The trend of distributing searchable encryption schemes into cloud [16, 25, 23, 24] highlights the potential risks of search pattern leakage (as well as access pattern leakage). That is because large amounts of data centralizing into the cloud servers affiliates effective statistic attacks. Moreover, such attacks can be launched well under the massive computing power of cloud servers.

We are aware of a recent work of access pattern disclosure on searchable encryption proposed by Islam et al. [15], which is close to our work. In [15], the authors formulate a novel attack that exploits the access pattern leakage to disclose the underlying keywords of user queries. The adversary in their attack is assumed to have background knowledge on the keyword distribution of user's document collection. They also presented a mitigation approach to hide the access pattern. Our work is quite different with theirs due to: (1) their topic is the access pattern leakage while ours is the search pattern leakage; (2) the background knowledge used by an adversary is different; and (3) the core ideas of their solution and our construction are different (i.e., their idea is to import some false positives to make search outcomes turn into identical to a certain degree while ours is to generate fake queries along with the real query). Nevertheless, both their and our work demonstrate the underlying keywords of user queries can be recovered once the access pattern or the search pattern is leaked.

8. Conclusion

In this paper, we review searchable encryption schemes in the literature and point out the search pattern leakage issue. By giving two concrete attack methods, we demonstrate that the search pattern can be utilized to attack the underlying queried keywords. Motivated by this threat, we present a new searchable encryption construction based on the idea of generating fake queries. Finally, to clarify the performance of proposed attack methods and constructions, we give detailed experiments which are based on the real-world dataset.

9. Acknowledgements

The authors are grateful to Yuan Zhang and Seny Kamara for the discussions of this work. This work is supported by National Natural Science Foundation of China No.61272512, National 863 Programme No.2013AA01A212 and Beijing Municipal Natural Science Foundation No.4121001.

References

[1] AOL search-log. `http://www.gregsadetsky.com/aol-data/`.

[2] Google trends. `http://www.google.com/trends/`.

[3] Sogou search-log. `http://www.sogou.com/labs/dl/q-e.html/`.

[4] Stopword list. `http://www.ranks.nl/resources/stopwords.html/`.

[5] Enron email dataset, 2009. `http://www.cs.cmu.edu/~enron/`.

[6] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In *Advances in Cryptology - CRYPTO'05*, volume 3621 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2005.

[7] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[8] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Cryptology - EUROCRYPT'04*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, 2004.

[9] Y.-C. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *Applied Cryptography and Network Security (ACNS'05)*, volume 3531 of *Lecture Notes in Computer Science*, pages 391–421. Springer, 2005.

[10] M. Chase and S. Kamara. Structured encryption and controlled disclosure. In *Advances in Cryptology - ASIACRYPT'10*, volume 6477 of *Lecture Notes in Computer Science*, pages 577–594. Springer, 2010.

[11] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. *Journal of Computer Security*, 19(5):895–934, 2011.

[12] L. Fang, W. Susilo, C. Ge, and J. Wang. Public key encryption with keyword search secure against keyword guessing attacks without random oracle. *Information Sciences*, 238:221–241, 2013.

[13] E.-J. Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. `http://eprint.iacr.org/`.

[14] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious rams. *Journal of the ACM*, 43(3):431–473, 1996.

[15] M. S. Islam, M. Kuzu, and M. Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *Network and Distributed System Security Symposium (NDSS'12)*, 2012.

[16] S. Kamara and K. Lauter. Cryptographic cloud storage. In *Financial Cryptography Workshops*, volume 6054 of *Lecture Notes in Computer Science*, pages 136–149. pringer, 2010.

[17] S. Kamara, C. Papamanthou, and T. Roeder. Dynamic searchable symmetric encryption. In *ACM Conference on Computer and Communications Security (CCS'12)*, pages 965–976, 2012.

[18] M. Kuzu, M. S. Islam, and M. Kantarcioglu. Efficient similarity search over encrypted data. In *IEEE International Conference on Data Engineering (ICDE'12)*, pages 1156–1167, 2012.

[19] R. Ostrovsky. Efficient computation on oblivious rams. In *ACM Symposium on Theory of Computing (STOC'90)*, pages 514–523, 1990.

[20] H. S. Rhee, J. H. Park, and D. H. Lee. Generic construction of designated tester public-key encryption with keyword search. *Information Sciences*, 205:93–109, 2012.

[21] W. Sun and B. Wang and N. Cao and M. Li and W. Lou and Y.T. Hou and H. Li. Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. In *ACM SIGSAC symposium on Information, computer and communications security (ASIA CCS'13)*, pages 71-82, 2013.

[22] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searching on encrypted data. In *IEEE Symposium on Security and Privacy*, pages 44–55, 2000.

[23] C. Wang, N. Cao, K. Ren, and W. Lou. Enabling secure and efficient ranked keyword search over outsourced cloud data. *IEEE Transaction on Parallel Distributed Systems*, 23(8):1467–1479, 2012.

[24] C. Wang, K. Ren, S. Yu, and K. M. R. Urs. Achieving usable and privacy-assured similarity search over outsourced cloud data. In *IEEE INFOCOM*, pages 451–459, 2012.

[25] Y. Yang, H. Lu, and J. Weng. Multi-user privacy keyword search for cloud computing. In *IEEE International Conference on Cloud Computing Technology and Science (CloudCom'11)*, pages 264–271, 2011.

Appendix A. Security proof of GBC

We analyze the security of GBC under the "state of the art" security definition from [10, 11] (as defined in Section 3). The CKA2-security is parametrized by a leakage function $\mathcal{L}$, which is assumed to be leaked to an adversary. The definition guarantees that an adversary cannot learn more about $\mathcal{L}$ from a searchable encryption scheme that is $\mathcal{L}$-secure against adaptive chosen keyword attack. To make the security definition fit GBC, we appropriately modify the description of CKA2-security.

Definition 7 (modified CKA2-security)**.** *Let* $\mathsf{SE}^{\mathsf{GBC}} = (\mathsf{KeyGen}, \mathsf{BuildIndex}, \mathsf{Encryption}, \mathsf{Dividing}, \mathsf{Query}, \mathsf{Search}, \mathsf{Extract}, \mathsf{Decryption})$ *be an index-based searchable encryption scheme that results from applying random-dividing-based construction and let $\mathcal{L}$ be a stateful algorithm. Let k be the confusion parameter used in GBC. For an adversary $\mathcal{A}$ and a simulator $\mathcal{S}$, we consider the following probabilistic experiments:*

$\mathbf{Real}_{\mathcal{A}}^{\mathsf{SE}^{\mathsf{GBC}}}(\lambda)$*: the challenger runs* $\mathsf{KeyGen}(1^\lambda)$ *to obtain a key K. $\mathcal{A}$ chooses $\mathcal{D}$ and receives $(\mathcal{C}, \mathcal{SI})$ such that* $(\mathcal{C}, \mathcal{SI}) \leftarrow \mathsf{Encryption}(\mathcal{D}, \mathcal{I}, K)$ *where $\mathcal{I}$ is the output of* $\mathsf{BuildIndex}(\mathcal{D})$*. The challenger divides the keyword list $\mathcal{W}$ into $|\mathcal{W}|/k$ subsets $\{S_1, ..., S_{|\mathcal{W}|/k}\}$ by calling* $\mathsf{Dividing}(k, \mathcal{W}, V)$. *Then $\mathcal{A}$ makes a polynomial number of adaptive queries and for each queried keyword w receives a query* $\mathcal{Q}_w = \{sq_1, ..., sq_k\} \leftarrow \mathsf{Query}(k, w, K, \{S_i\}_{i \in \{1,...,|\mathcal{W}|/k\}})$ *from the challenger. Finally, $\mathcal{A}$ returns a bit b that is output by the experiment.*

$\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{\mathsf{SE}^{\mathsf{GBC}}}(\lambda)$*: $\mathcal{A}$ outputs $\mathcal{D}$. Given $\mathcal{L}(\mathcal{D})$, $\mathcal{S}$ generates and sends $(\mathcal{C}, \mathcal{SI})$ to $\mathcal{A}$. Then $\mathcal{A}$ makes a polynomial number of adaptive queries. For each queried keyword w_i, let $\mathcal{H} = (\mathcal{D}, \mathbf{w})$ denote the i-query history where $\mathbf{w} = \{w_1, ..., w_i\}$, $\mathcal{S}$ is given $\mathcal{L}(\mathcal{H})$ and returns to $\mathcal{A}$ an appropriate query $\mathcal{Q}_{w_i} = \{sq_1, ..., sq_k\}$. Finally, $\mathcal{A}$ returns a bit b that is output by the experiment.*

We say that $\mathsf{SE}^{\mathsf{GBC}}$ *is $\mathcal{L}$-secure against adaptive chosen-keyword attacks if for all* PPT *adversary $\mathcal{A}$, there exists a* PPT *simulator $\mathcal{S}$ such that*

$$|\Pr[\mathbf{Real}_{\mathcal{A}}^{\mathsf{SE}^{\mathsf{GBC}}}(\lambda) = 1] - \Pr[\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{\mathsf{SE}^{\mathsf{GBC}}}(\lambda) = 1]| \leq \mathrm{negl}(\lambda).$$

Theorem 1. *If* SE *is* $\mathcal{L}$*-secure against adaptive chosen keyword attack, then* $\mathsf{SE}^{\mathsf{GBC}}$ *as described above is* $\mathcal{L}'$*-secure against adaptive chosen keyword attack, where*

$$\mathcal{L}(\mathcal{D}) = (|D_1|, ..., |D_n|, \#\mathcal{SI})$$

$$\mathcal{L}(\mathcal{H}) = (\varphi_{\mathcal{H}}, \tau_{\mathcal{H}})$$

$$\mathcal{L}'(\mathcal{D}) = (|D_1|, ..., |D_n|, \#\mathcal{SI})$$

and

$$\mathcal{L}'(\mathcal{H}) = (\varphi_{\mathcal{H}}, \rho_{\mathcal{H}})$$

where $|D_i|$ *denotes the length of* D_i*,* $\#\mathcal{SI}$ *denotes the total number of entries in* $\mathcal{SI}$*, and* φ*,* τ *and* ρ *denote the access pattern, the search pattern and the group pattern, respectively.*

Before the proof,it should be noted that the access pattern φ in $\mathcal{L}'$ is a little bit different with that in $\mathcal{L}$ since the search result in $\mathsf{SE}^{\mathsf{GBC}}$ is a collection of k sub-results rather than a single search result in SE. For example, given an n-query history $\mathcal{H} = (\mathcal{D}, \mathbf{w})$ where $\mathbf{w} = \{w_1, ..., w_n\}$,

- in $\mathcal{L}(\mathcal{H})$:
$$\varphi_{\mathcal{H}} = (\mathcal{R}(w_1), ..., \mathcal{R}(w_n))$$
- in $\mathcal{L}'(\mathcal{H})$:
$$\varphi_{\mathcal{H}} = (\mathcal{R}_{1,1}, ..., \mathcal{R}_{1,k}, \mathcal{R}_{2,1}, ..., \mathcal{R}_{2,k}, ..., \mathcal{R}_{n,1}, ..., \mathcal{R}_{n,k})$$
  where $\mathcal{R}_{i,j}$ denotes the j-th sub-result of the i-th result structure.

Proof. Since SE is $\mathcal{L}$-secure against adaptive chosen keyword attack, there exists a simulator $\mathcal{S}$ that can simulate appropriate ciphertexts $\mathcal{C}^*$, secure index $\mathcal{SI}^*$ and search queries q_i^* (i=1,2,...), which are indistinguishable from the real ones (denoted as $\mathcal{C}$, $\mathcal{SI}$ and q_i) to a PPT adversary $\mathcal{A}$. Now we construct a simulator $\mathcal{S}'$ which makes use of $\mathcal{S}$ to simulate $\mathcal{C}^*$, $\mathcal{SI}^*$ and Q_i^* as follows:

- (Simulating $\mathcal{C}^*$ and $\mathcal{SI}^*$) Given $\mathcal{L}'(\mathcal{D})$, $\mathcal{S}'$ directly gives $\mathcal{L}'(\mathcal{D})$ to $\mathcal{S}$, and outputs $\mathcal{C}^*$ and $\mathcal{SI}^*$ that output by $\mathcal{S}$.

- (Simulating Q_i^*) Recall that $\tau_{\mathcal{H}}$ is a $|\mathbf{w}| \times |\mathbf{w}|$ binary matrix such that $\tau_{\mathcal{H}}[s][t] = 1$ if $w_s = w_t$ (and 0 otherwise), and $\rho_{\mathcal{H}}$ is a $|\mathbf{w}| \times |\mathbf{w}|$ binary matrix such that $\rho_{\mathcal{H}}[s][t] = 1$ if w_s and w_t belong to the same subset (and 0 otherwise). Let k be the confusion parameter. Given $\mathcal{L}'(\mathcal{H})$, for $1 \le s, t \le i$ (note that $|\mathbf{w}| = i$), $\mathcal{S}'$ construct a $\tau'_{\mathcal{H}}$ as follows:
  - if $\rho_{\mathcal{H}}[s][t] = 0$, then replaces $\rho_{\mathcal{H}}[s][t]$ with a zero matrix of size k, and
  - if $\rho_{\mathcal{H}}[s][t] = 1$, then replaces $\rho_{\mathcal{H}}[s][t]$ with a identity matrix of size k.

  Let the new matrix be $\tau'_{\mathcal{H}}$. We can see that the size of $\tau'_{\mathcal{H}}$ is $(k \cdot i) \times (k \cdot i)$. Let $\tau'_{\mathcal{H}}(j)$ denote the sub-matrix of τ' that covers the first j rows and columns of $\tau'_{\mathcal{H}}$. Let $\varphi_{\mathcal{H}}(j)$ denote the sub-tuple of $\varphi_{\mathcal{H}}$ that covers the firt j elements of $\varphi_{\mathcal{H}}$. $\mathcal{S}'$ respectively gives $(\varphi_{\mathcal{H}}(k \cdot (i-1)+1), \tau'_{\mathcal{H}}(k \cdot (i-1)+1)), ..., (\varphi_{\mathcal{H}}(k \cdot (i-1)+k), \tau'_{\mathcal{H}}(k \cdot (i-1)+k))$ to $\mathcal{S}$. As responses, $\mathcal{S}$ respectively returns the simulated search queries $q_{i,1}^*, ..., q_{i,k}^*$. Finally, $\mathcal{S}'$ outputs $Q_i^* = \{q_{i,1}^*, ..., q_{i,k}^*\}$.

Now we argue that $(\mathcal{C}^*, \mathcal{SI}^*, \{Q_i^*\}_{i=1,2,...})$ is computationally indistinguishable from $(\mathcal{C}, \mathcal{SI}, \{Q_i\}_{i=1,2,...})$ to a PPT adversary $\mathcal{A}$.

- ($\mathcal{C}^*$ and $\mathcal{C}$, $\mathcal{SI}^*$ and $\mathcal{SI}$) The CKA2-security of SE guarantees that $\mathcal{C}^*$ and $\mathcal{SI}^*$ are indistinguishable from $\mathcal{C}$ and $\mathcal{SI}$ to $\mathcal{A}$ respectively.
- ($\{Q_i^*\}_{i=1,2,...}$ and $\{Q_i\}_{i=1,2,...}$) For any individual sub-query $Q_i^* = \{q_{i,1}^*, ..., q_{i,k}^*\}$, the CKA2-security of SE guarantees that $q_{i,j}^*$ is indistinguishable from $q_{i,j}$ to $\mathcal{A}$. The CKA2-security of SE also guarantees that the outcome of $\mathsf{SE.Search}(q_{i,j}^*, \mathcal{SI}^*)$ is indistinguishable from the outcome of $\mathsf{SE.Search}(q_{i,j}, \mathcal{SI})$ to $\mathcal{A}$ (i.e., the resultant access patterns are indistinguishable between the real case and the simulation case). Thus Q_i^* is indistinguishable from Q_i to $\mathcal{A}$. Now we show that the resultant group pattern (denoted as ρ^*) in the simulation case is exactly the same with that in the real case (denoted as ρ).
  - If $\rho[s][t] = 0$ (i.e., $Q_s \neq Q_t$), as the description above, $\mathcal{S}'$ replaces the bit with a zero matrix of size k. The CKA2-security of SE requires $q_{s,1}^*, ..., q_{s,k}^*$ and $q_{t,1}^*, ..., q_{t,k}^*$ (generated from $\mathcal{S}$) satisfy (1) $q_{s,1}^* \neq q_{t,1}^*, ..., q_{s,k}^* \neq q_{t,k}^*$ and (2) $q_{s,r_1}^* \neq q_{s,r_2}^*$ and $q_{t,r_1}^* \neq q_{t,r_2}^*$ for any $1 \le r_1 \neq r_2 \le k$. Thus $Q_s^* \neq Q_t^*$ and $\rho^*[s][t] = 0$.

– If $\rho[s][t] = 1$ (i.e., $Q_s = Q_t$), as the description above, $\mathcal{S}'$ replaces the bit with a identity matrix of size k. The CKA2-security of SE requires $q^*_{s,1}, ..., q^*_{s,k}$ and $q^*_{t,1}, ..., q^*_{t,k}$ (generated from $\mathcal{S}$) satisfy (1) $q^*_{s,1} = q^*_{t,1}, ..., q^*_{s,k} = q^*_{t,k}$, and (2) $q^*_{s,r_1} \neq q^*_{s,r_2}$ and $q^*_{t,r_1} \neq q^*_{t,r_2}$ for any $1 \leq r_1 \neq r_2 \leq k$. Thus $Q^*_i = Q^*_j$ and $\rho^*[i][j] = 1$.

Thus we have $\rho^* = \rho$. Therefore, $\{Q^*_i\}_{i=1,2,...}$ is indistinguishable from $\{Q_i\}_{i=1,2,...}$ to $\mathcal{A}$.

For any PPT $\mathcal{A}$, $\mathcal{A}$ cannot distinguish the view in $\mathbf{Real}_{\mathcal{A}}^{\mathsf{SE}^{\mathsf{GBC}}}(\lambda)$ and the view in $\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{\mathsf{SE}^{\mathsf{GBC}}}(\lambda)$, so we have

$$|\Pr[\mathbf{Real}_{\mathcal{A}}^{\mathsf{SE}^{\mathsf{GBC}}}(\lambda) = 1] - \Pr[\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{\mathsf{SE}^{\mathsf{GBC}}}(\lambda) = 1]| \leq \text{negl}(\lambda).$$

$\square$