Query Recovery Attacks on Searchable Encryption Based on Partial Knowledge

Guofeng Wang[1], Chuanyi Liu[2(✉)], Yingfei Dong[3], Hezhong Pan[1], Peiyi Han[1], and Binxing Fang[2]

[1] School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China
{wangguofeng,hanpeiyi}@bupt.edu.cn

[2] School of Computer and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen, China
cy-liu04@mails.tsinghua.edu.cn

[3] Department of Electrical and Computer Engineering, University of Hawaii, Honolulu, USA
yingfei@hawaii.edu

Abstract. While Searchable Encryption (SE) is often used to support securely outsourcing sensitive data, many existing SE solutions usually expose certain information to facilitate better performance, which often leak sensitive information, e.g., search patterns are leaked due to observable query trapdoors. Several inference attacks have been designed to exploit such leakage, e.g., a query recovery attack can invert opaque query trapdoors to their corresponding keywords. However, most of these existing query recovery attacks assume that *an adversary knows almost all plaintexts* as prior knowledge in order to successfully map query trapdoors to plaintext keywords with a high probability. Such an assumption is usually impractical. In this paper, we propose new query recovery attacks in which an adversary only needs to have *partial knowledge of the original plaintexts*. We further develop a countermeasure to mitigate inference attacks on SE. Our experimental results demonstrate the feasibility and efficacy of our proposed scheme.

Keywords: Searchable encryption · Inference attacks
Query recovery attacks

1 Introduction

Due to security concerns, sensitive data is often encrypted before uploaded to cloud service providers (CSPs). Therefore, Searchable Encryption (SE) has become a critical technique for many secure applications, which allows a user to securely outsource its data to an untrusted cloud server, while maintaining various search functionalities.

Two Common SE Models. Currently, SE schemes mostly explore the trade-offs between query expressiveness, security, and efficiency. Oblivious RAMs

(ORAM) [1] satisfies security and query expressiveness but incurs many interactions for each read and write, which makes it impractical in deployment. Recently, many researchers focus on the *Encrypted-Index SE* model, summarized as follows. A user first encrypts some documents and generates a corresponding searchable index; it then uploads the encrypted documents and the encrypted index to a CSP. To search the documents, the user generates a search *trapdoor* to ask the CSP to search on the encrypted index and return corresponding results. While such a scheme achieves a good balance between security and efficiency, it loses some query expressiveness [2], and it requires modifying current cloud Application Programming Interface (API). In addition, most Encrypted-Index SE schemes leak certain sensitive information to the adversary (i.e., the curious cloud server) for better performance [3]. In the following, we focus on Encrypted-Index SE schemes that leak search patterns and access patterns.

On the other hand, to be compatible with legacy systems, SE schemes such as ShadowCrypt [4] and Mimesis Aegis [5] use the *Appended-Token SE* model, which encrypts each document using a conventional encryption method, and appends a sequence of tokens to the ciphertext. Because a token is deterministically generated by encrypting a corresponding keyword, the search operation for a keyword is conducted in two steps: generate the token of a keyword and request the server to search for the token. Many cloud industry solutions (such as Skyhigh [6], CipherCloud [7]) also advocate this approach. An Appended-Token SE scheme requires no modification on the CSP side. Because such a scheme provides no additional protection of token occurrence patterns, once encrypted documents and tokens are uploaded to a CSP, the count of each indexed keyword, its co-occurrence probabilities with other keywords, and the similarity between documents, can be easily learned by the CSP.

Limitation of Existing Query Recovery Attacks. Islam, Kuzu, and Kantarcioglu (IKK) [8] analyzed the implications of revealing search patterns and access patterns in SE. They showed that user queries can be inferred with a high success rate when an adversary knows all the original documents. Cash, Grubbs, Perry, and Ristenpart (CGPR) [9] proposed a simpler count attack, which outperformed the IKK attack in terms of efficiency and accuracy in the same scenarios. However, for a certain size of keyword vocabulary, both the IKK and CGPR attacks require almost the *complete knowledge* of plaintexts to achieve a good recovery rate of queries. For example, when knowing less than 80% of a document set, both the IKK attack and the CGPR count attack can invert almost no query trapdoors. In this paper, we focus on this issue and emphasize attacks with partial knowledge.

Our Contributions. Normally, an adversary rarely knows the entire document set of a victim, but it usually learns a subset, e.g., some well-circulated emails. Therefore, we focus on this practical case and develop query recovery attacks based on partial knowledge. We have made the following contributions in this paper.

1. *Document identification attack on Appended-Token SE schemes.* To attack an Appended-Token SE scheme, we first establish the mappings between the

known plaintext documents and the encrypted documents of a victim, which called *"document identification attack"*. This attack allows us to apply existing query recovery attack algorithms to invert tokens more accurately.
2. *Extended document identification attack on Encrypted-Index SE schemes.* In an Encrypted-Index SE scheme, before queries are issued, an adversary learns nothing except the sizes of the ciphertexts and the encrypted index. So, the adversary cannot directly perform the proposed document identification attack. However, the adversary can perform this attack in certain situations with some auxiliary information. For example, if the adversary knows widely-circulated emails, the document identification attack can be conducted according to specific protocols and related data items (such as senders and receivers in emails). With a sequence of query results, an inverted index can be built between search trapdoors and returned encrypted documents. Then, the adversary can remove irrelevant encrypted documents from the query results according to the identified encrypted documents, and use query recovery attack algorithms with prior knowledge to obtain query keywords more accurately.
3. *We propose a simple noise addition technique to mitigate the inference attacks.* We minimize information disclosure by spreading search tokens in the Appended-Token SE, to break the statistical relations between keywords and tokens. The proposed model achieves backwards compatibility with legacy systems. Our experimental results show its effectiveness.

The remainder of this paper is organized as follows. We introduce related work in Sect. 2, and present the query recovery attacks with partial knowledge in Sect. 3. We further present our evaluation in Sect. 4. We discuss mitigation methods in Sect. 5, and conclude this paper in Sect. 6.

2 Background and Related Work

In this section, we first introduce SE basics and common SE schemes, and point out their leakage models. We then discuss common inference attacks on SE.

2.1 SE Basics

First, we define terminologies and preliminaries used in this paper. Let n be the total number of documents in a collection $D = (D_1, D_2, ..., D_n)$. We denote the identifier of a document D_i by $ID(D_i)$. Let $D(w)$ be the ordered list consisting of the identifiers of all documents that contain the keyword w in set D. We use m to denote the total number of keywords in a dictionary, and use $W = (w_1, w_2, ..., w_m)$ to denote the set of keywords in a dictionary.

A trapdoor function f takes a keyword w as input, and emits a trapdoor that enables the server to search on the encrypted index while keeping the keyword hidden. A search pattern means the information that given two searches with the same results, we can determine whether the two searches use the same keyword. An access pattern refers to the information that may be leaked in query results. The returned results imply the document IDs containing the query keywords.

2.2 SE Models

In this section, we classify common SE schemes into two models: the Encrypted-Index SE model and the Appended-Token SE model, as shown in Fig. 1.

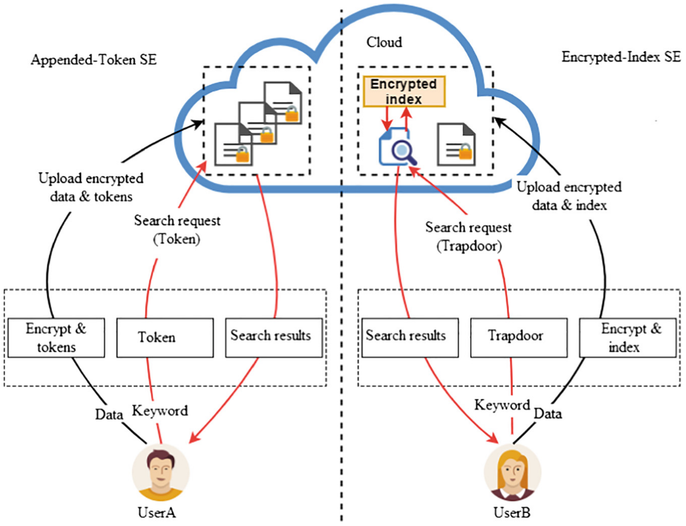

Fig. 1. Architecture for searchable encryption models.

Encrypted-Index SE Model. Curtmola et al. [3] first built an encrypted search mechanism using inverted indexes. For each keyword, it built a linked list of the IDs of the documents containing the keyword, improved the search efficiency to sub-linear time, and enhanced the security of SE. We classify this scheme into the Encrypted-Index SE model. In this scheme, the nodes of every linked list are encrypted with randomly generated keys and scrambled in a random order. A node contains a document identifier, a key used to decrypt the next node, and a pointer to the next encrypted node. Before queries are issued, the server learns nothing except the sizes of the documents and the index. For trapdoors that have been queried, the query results reveal the information about the occurrence counts of the hidden keywords and the co-occurrence patterns of multiple keywords. It can only support exact keyword searches and documents cannot be updated dynamically. Recently, advanced SE functions are further developed based on the above approach, such as multi-keyword SE [12], fuzzy keyword SE [13] and dynamic SE [14]. In summary, the Encrypted-Index SE model improves the search efficiency, but requires modifying current cloud APIs. After all keywords have been queried, the leakage of Encrypted-Index SE degenerates to the same level as the leakage of Appended-Token SE as depicted in the following section.

Appended-Token SE Model. As Fig. 1 shows, this model is compatible with legacy systems: the server can index and search the uploaded tokens. However, the model provides no additional protection of token occurrence patterns and gives the server the exact document-token matrix prior to search.

ShadowCrypt [4] and Mimesis Aegis [5] use this model to support SE in legacy applications. For a given keyword, ShadowCrypt uses a single pseudorandom function to generate a single search token. After that, it sorted the tokens in every document to disturb the correspondence between tokens and keywords. The Mimesis Aegis SE scheme is more secure than ShadowCrypt because it does not reveal a one-to-one correspondence between keywords and tokens. It uses Bloom filter and a family of pseudorandom functions to generate k distinct tokens for each keyword. However, the tokens are deterministically encrypted, that the server can learn the count of each unique indexed keyword and its co-occurrence probabilities with other keywords upon uploading (as in Shadow Nemesis [11]). In addition, the Bloom filter has a small error rate. There may be potential collisions that some tokens may correspond to more than one keyword. Due to the simplicity of the Appended-Token SE, several commercial encryption products from Skyhigh Networks [6], CipherCloud [7], Bitglass [15], and Virtue [16] use this model (or its variants) to support SE in their cloud services.

2.3 Inference Attacks on SE Models

An adversary can use inference attacks to obtain sensitive information against SE schemes based on its leakages and the adversary's prior knowledge.

Adversary's Prior Knowledge. We classify the adversary's prior knowledge into three types. (i) *Distributional document knowledge model:* The adversary has no a priori knowledge of the plaintext messages. In this scenario, an adversary may have the contextual information about the documents, such as whether they are emails or medical documents. (ii) *Full document knowledge model:* All documents are known to the adversary. While it is rare, it may happen sometimes, e.g., a user has a large corpus of emails stored at an email service, and it decides to encrypt all old emails using SE. (iii) *Partial document knowledge model:* As it is unlikely that an adversary knows all documents of a victim, it may only know a subset. In the following, we will focus on the effectiveness of our attacks in this scenario of partial knowledge.

Attack Modes. We classify attacks in two modes. (i) In a *passive attack*, the adversary intercepts communications between a user and a server, to count the frequencies of query keywords or the co-occurrence patterns of multiple keywords in a document set by observing query results. With known plaintext documents, the adversary can map opaque query trapdoors to plaintext keywords. The IKK attack [8] and the Shadow Nemesis attack [11] use co-occurrence matrixes and combination optimization algorithms to perform inference. (ii) In an *active attack*, an adversary proactively sends the client multiple plaintext documents with structured contents; the client will then create an encrypted index based on these documents and upload them to the cloud server. So, the attacker can observe the inserted documents contained in the user's query results to create mappings between the keywords and search trapdoors. CGPR's [9] active attack and ZKP [10] both use this attack mode.

Attack Algorithms. Different attack models are summarized in Table 1. IKK [8] first studied the empirical security of SE and analyzed the implications of revealing search patterns and access patterns. Let q be the number of unique query trapdoors observed. A $q \times q$ trapdoor co-occurrence matrix is built as C_q, where $C_q[i, j]$ represents the number of documents which the i-th trapdoor and the j-th trapdoor both hit. If the server has the prior knowledge of all indexed documents, a $m \times m$ keyword co-occurrence matrix C_m can be constructed, where $C_m[i, j]$ represents the number of documents in which the i-th keyword and the j-th keyword both appear. Then, a simulated annealing algorithm is used to find the best match of C_q to C_m, thus inverting the corresponding query trapdoors. When the information about plaintext is not accurate or only partial plaintext is known, the success rate of IKK query recovery attack is poor.

Table 1. Different attack schemes against SE. EISE represents Encrypted-Index SE, and ATSE represents Appended-Token SE.

Attack schemes	Attack methods	Prior knowledge	Attack SE
IKK [8]	Simulated annealing	Almost all documents	EISE and ATSE
CGPR [9]	Count or file injection	Almost all or partial documents	EISE and ATSE
ZKP [10]	File injection	No or partial documents	EISE and ATSE
Shadow Nemesis [11]	Graph matching	All or auxiliary documents	ATSE

CGPR [9] presented a simpler count attack without using optimization algorithms. It first calculates the number of documents in the query result of a search trapdoor, and then finds a unique keyword appeared in the same number of plaintext documents. If the unique keyword found, the mapping between the trapdoor and the keyword can be directly established. Based on the mappings, given an unknown search trapdoor q with a result length, it first selects the candidate keywords contained in the same number of plaintext documents. Then, to determine whether a keyword w in the candidate keyword set is corresponding to the trapdoor q, for each pair of identified keyword-trapdoor mapping w' and q', it computes the co-occurrence count c_1 of w and w' (the number of documents in which w and w' both appear) in known documents, and the co-occurrence count c_2 of q and q' (the number of documents which both the query q and q' match) in query results. If c_1 is not equal to c_2, then w will be removed from the candidate keyword set. Finally, only one remaining keyword meeting all the conditions can be mapped to the trapdoor q. However, CGPR requires almost the complete knowledge of a victim's documents to achieve a good query recovery rate. When only knowing a portion of the document set (e.g., less than 80%), both IKK and CGPR attacks perform poorly.

The Shadow Nemesis [11] launched inference attacks on the Appended-Token SE model. The attack creates a keyword co-occurrence matrix graph G and a token co-occurrence matrix graph H based on the auxiliary information and target data, respectively. As the Appended-Token SE model leaks the occurrence count of each indexed keyword and its co-occurrence probabilities with other keywords, which is sufficient to convert the attack to the well-known Weighted Graph Matching (WGM) problem. This method did not examine query recovery attacks with partial knowledge.

ZKP [10] used active attacks to infer query trapdoors. An attacker needs to inject known documents to a client, while the client must encrypt the received documents and generate corresponding search indexes. The number of injected documents is dependent on the size of keyword vocabulary. This assumption is often difficult to meet when a client only encrypts and indexes its own sensitive data, as in Virtue [16]. So, we do not investigate the active attack algorithms in the following.

3 Query Recovery Attacks with Partial Knowledge

In this section, we present our query recovery attacks with partial knowledge against two SE models.

3.1 Motivation

Although various attacks on SE have been investigated in different settings, there are still several interesting challenges to be addressed as follows.

(i) *To invert the query with a high accuracy, common query recovery attacks require almost the complete knowledge of a victim's documents, which is unrealistic in normal cases.* For example, an adversary needs to know almost all documents to achieve a high success rate in the CGPR [9] count attack. The IKK [8] attack and the Shadow Nemesis [11] attack consider a more realistic scenario, in which an adversary can collect publicly relevant data based on the distributional knowledge of the victim's documents. However, the adversary must have accurate keyword co-occurrence probabilities and corresponding keywords, which we believe the knowledge can only be obtained by an adversary that has access to all the documents.

(ii) *The recovery rate of queries is poor when an adversary only has partial knowledge of documents.* In this case, as the statistics of partially known documents do not match the statistics of query results on all documents, resulting in a low probability of success. In practice, the adversary usually has partial knowledge about a victim's document set. Therefore, we focus on this issue in our investigation.

3.2 Query Recovery Attacks Against Appended-Token SE Model

Prior to a search, the Appended-Token SE model leaks the count of each unique indexed keyword, its co-occurrence probabilities with other keyword, and the similarity of documents to a cloud server. When having the explicit knowledge of all documents of a victim, the adversary can get a consistent statistical distribution about keywords and tokens. The attacker can invert the underlying tokens to their respective keywords, even when no queries have been issued.

Document Identification Attack. However, if the adversary only has partial knowledge, as the statistics between keywords and tokens do not match well, it is very hard to invert the tokens. To address this issue, we first pre-established the mappings between the known documents and related encrypted documents, which we called *Document Identification Attack*. Then, the server can calculate the count of each token and its token co-occurrence probabilities with other tokens in the identified encrypted documents. Similarly, in the known documents corresponding to the identified encrypted documents, the server can obtain the count of each keyword and its keyword co-occurrence probabilities with other keywords. Finally, the server can build mappings between opaque tokens and plaintext keywords accurately.

Next, we describe our document identification attack algorithm in detail. Let $D = (D_1, D_2, ..., D_n)$ denote a collection of n plaintext documents. A keyword extraction algorithm takes a document D_i as input and outputs a vector W_i, where each component is a character string, namely a keyword w. We assume the keyword extraction algorithm is deterministic and known to the adversary. Let $W = (W_1, ..., W_n)$ be the ordered list of all keyword vectors.

For each known plaintext document D_i, we first choose the unique encrypted document in which the number of tokens is the same as the number of unique keywords in the keyword vector W_i. We name the mapping between the encrypted document and its corresponding plaintext document as a *base mapping*.

If the mapping is not unique, i.e., multiple candidate encrypted documents have the same number of tokens as the number of unique keywords in a plaintext document. We filter the candidate encrypted documents of the plaintext document by comparing the similarity of plaintext documents (i.e., the number of common keywords in two documents) and the similarity of encrypted documents (i.e., the number of common tokens in two encrypted documents) with help of base mappings. Our document identification attack algorithm is shown in Algorithm 1. In line 2, we build the similarity matrix of partial plaintext documents, C_k, where $C_k[i, j]$ represents the number of common keywords in two documents i and j, and the similarity matrix of all encrypted documents, C_t, where $C_t[i, j]$ is computed by counting the number of common tokens in two encrypted documents i and j. A document identification example is shown in Fig. 2, in which *PDoc* means "plaintext document", *EDoc* means "encrypted document". First, because only *PDoc1* has 101 keywords and *EDoc1* has 101 tokens, we have a unique mapping between them. Second, for *PDoc2*, we have two candidate encrypted documents *EDoc2* and *EDoc3*. As *PDoc2* has 25 common keywords with *PDoc1*, we find *EDoc3* has the same number of common tokens with *EDoc1*, while *EDoc2* only has 20 common tokens with *EDoc1*. So, we can determine the mapping between *EDoc3* and *PDoc2*.

Algorithm 1. Document Identification Attack algorithm

```
   input  : all encrypted document set $e$, partial plaintext document set $p$.
   output: mapping set between $e$ and $p$;
 1 initialize the base mapping set $K$;
 2 compute the similarity matrix of partial plaintext documents, $C_k$, and the
   similarity matrix of all encrypted documents, $C_t$;
 3 while size of $K$ is increasing do
 4     for each un-mapping plaintext document $d \in p - K$ do
 5         set candidate encrypted document set $S = \{s$ : the token count of $s$ is
           equal to the keyword count of $d$ $\}$;
 6         for $s \in S$ do
 7             for known base mapping $(d', s') \in K$ do
 8                 if $C_k[d, d'] \neq C_t[s, s']$ then
 9                     remove $s$ from $S$;
10         if one encrypted document $s$ remains in $S$ then
11             add $(d, s)$ to $K$
12 return the mapping set $K$;
```

Based on the document identification algorithm, we can utilize the CGPR count attack to build more mappings between tokens in the identified encrypted documents and keywords in the corresponding plaintext documents, which we called query recovery algorithm, as shown in Algorithm 2.

In the proposed query recovery algorithm, we first build a modified inverted index over the identified known documents. This is an $m \times n$ matrix I, where entry $I_{i,j} = 1$ iff document D_j contains keyword w_i. All other entries are set to zero. The rows are indexed by the keyword set, while the columns are indexed by the document set. In the same way, we build an $m \times n$ matrix I' for the identified encrypted documents, where entry $I'_{i,j} = 1$ iff document D_j contains token t_i. All other entries are set to zero. Based on matrix I and I', we then build a $m \times m$ keyword co-occurrence count matrix K', where $K'[i, j]$ represents the number of documents in which w_i and w_j both appear. Similarly, we build a $m \times m$ token co-occurrence count matrix T', where $T'[i, j]$ represents the number of encrypted documents in which token t_i and token t_j both appear.

3.3 Query Recovery Attacks Against Encrypted-Index SE Model

In the Encrypted-Index SE model, before queries are issued, the attacker learns nothing except the sizes of the documents and indexes. For trapdoors that have been queried, the query results reveal the information about the query keyword occurrence count and the keyword co-occurrence count of the queried keywords.

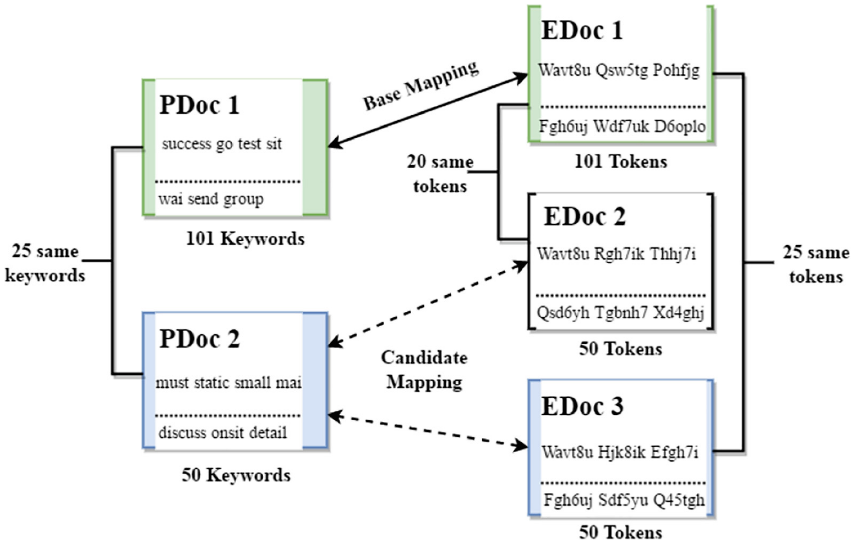

Fig. 2. An example of document identification attack. There are two known plaintext documents and three encrypted documents. As a result, *PDoc1* is mapped to *EDoc1*, *PDoc2* is mapped to *EDoc3*.

Algorithm 2. Query Recovery Attack algorithm

```
   input : Query token set $T$ in identified encrypted documents $e'$, keyword set
           $W$ in identified known plaintext documents $p'$.
   output: mapping set between $T$ and $W$;
1  initialize the base mapping set $G$;
2  compute the token co-occurrence matrix $T'$ for $T$ and the keyword
   co-occurrence matrix $K'$ for $W$;
3  while size of $G$ is increasing do
4      for each unknown token $t \in T - G$ do
5          build candidate keyword set $S = \{s$ : the occurrence count of $s$ in $p'$ is
           equal to the occurrence count of $t$ in $e'$ $\}$;
6          for $s \in S$ do
7              for known base mapping $(t', s') \in G$ do
8                  if $T'[t, t'] \neq K'[s, s']$ then
9                      remove $s$ from $S$;
10         if one keyword $s$ remains in $S$ then
11             add $(t, s)$ to $G$
12 return the mapping set $G$;
```

Initially, the attacker (i.e., the cloud server) cannot establish the base mappings between a known subset of plaintext documents and all encrypted documents based on the number of keywords. However, it can establish such mappings in specific scenes with auxiliary information, called *Extended Document Identification Attack*. For example, in the Enron [17] dataset, the public email contains auxiliary information, such as senders, receivers, and timestamps. If the attacker knows widely-circulated emails, it can make the association based on specific protocols and related data items. In this way, the attacker can construct the mappings between the identified encrypted documents and the corresponding plaintext documents. With the mappings, by counting the trapdoors and the returned results for a period of time, the attacker can build more mappings between trapdoors and corresponding keywords than the one without document identification. In fact, if the adversary intercepts a set of queries Q over a sufficient long period, it has a good chance to count most high-frequency keywords.

For queried trapdoors, the attacker can create an inverted index as shown in Fig. 3. The document IDs pointed by dotted arrows means that they are not belong to the constructed mappings of document identification. Then, the attacker performs document pruning to remove the document IDs that do not belong to the constructed document mappings. By this way, the attacker can remove the information that has nothing to do with the known subset of plaintexts. Finally, after performing the document identification and document pruning steps, the mappings between trapdoors and keywords can be built accurately using query recovery attack algorithms.

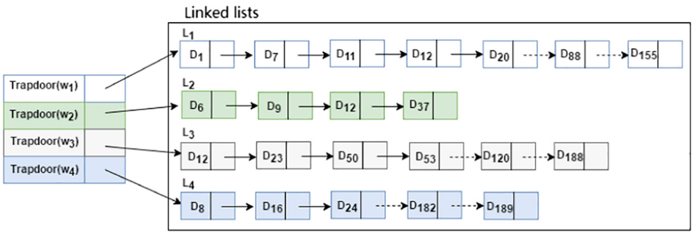

Fig. 3. The inverted index for queried results. D_i stands for document ID, L_i represents a linked list for keyword w_i.

4 Evaluation

We implemented a prototype system and conducted experiments to validate the effectiveness of the proposed document identification attacks: (1) in a single user case and in a multi-user case; (2) the improved success rate of query recovery attack. The configuration of the testing virtual machine includes an Intel 2.5 GHz dual-core with 8 GB memory. For each experiment on the Enron dataset [17], it took less than 5 min to complete, which shows the effectiveness of the proposed attack model.

4.1 Experimental Setup

We used the Enron [17] dataset available online as our test data. We chose emails from the "_sent_mail" folder of 73 employees, resulting in a total of 28,657 messages. There are about 49,835 distinct keywords in the whole dataset.

We extracted keywords from this dataset as follows: An email message is considered as one document. The first few lines of each email usually contain auxiliary information about the email, such as senders, receivers, and timestamps. We strip these lines off in a preprocessing step, because these lines are not part of the original email. The words in each email were first stemmed using the standard Porter stemming algorithm [18]; we remove 200 stop words [19] and duplicate keywords.

Given the set of n documents, the above process produces a set of distinct keywords for each document, resulting in n keyword sets. Assume there are a total of M distinct keywords in all the keyword sets, we then establish a fixed-size keyword vocabulary by taking the most frequent m keywords from these sets.

In our experiments, the adversary only knows a subset of emails. The leaked emails of different users are expected to vary significantly. Therefore, it is hard to adopt a methodology to capture which messages are more likely to be leaked. Without losing the generality, we randomly selected a subset of emails as the known documents for each setting. We present the concrete effect of document identification attack with partial knowledge against the Appended-Token SE model in the following. When attacking the Encrypted-Index SE model, we conduct the Extended Document Identification Attack with auxiliary information.

4.2 Effectiveness of Document Identification Attack

To achieve a high success rate, we first perform document identification attack, which establishes the mappings between the known subset of plaintext documents and the encrypted documents. We show that the attack works well even when just a small fraction of documents are known to the attacker. Note that the result of document identification attack is dependent on the randomly selected subset of known documents; however, we have repeated the experiments in the same setting many times, and the results are consistent.

Document Identification Attack in a Single-User Case. First, we consider the single-user SE scheme, such as in the ShadowCrypt [4] approach, which adds end-to-end encryption to cloud-based applications. It interposed itself between the interface of a legacy application and a user. As different users apply different keys to encrypt data and generate different query tokens, a user can only search for its documents.

Incidentally, an adversary may have partial knowledge about a victim's documents. We randomly selected a user from 73 employees, "allen-p", as the victim. There were 602 emails in its "_sent_mail" folder. While 69% of the emails contain less than 44 distinct keywords, only 8% of emails contain the unique number of keywords (mostly more than 100 keywords). The emails are named with different index numbers. We choose a proportion of the emails as partially known documents.

The experimental results of document identification attack in the single-user case are shown in Fig. 4, with different subsets of known documents. The x-axis represents the percentage of known documents, and the y-axis represents the number of documents that have been identified. The top (green) line with triangle markers represents the number of documents known to the attacker; the middle (red) curve with square markers represents the number of identified documents after the document identification attack; the bottom (blue) line with diamond markers represents the number of documents that have been identified in the base mappings. We can see that only a few documents are identified in the base mappings; with the proposed attack, we can map a large proportion of known documents to their corresponding encrypted documents. On average, we can identify about 81% of known plaintext documents with their encrypted versions. However, the attack is dependent on that at least one document is initially identified in the base mappings. This can be resolved by making an initial guess that maps a document to one in the candidate encrypted document set, and then runs the remainder of the algorithm. If the guess is wrong, the document similarity comparison algorithm detects inconsistency, and we then will try another candidate.

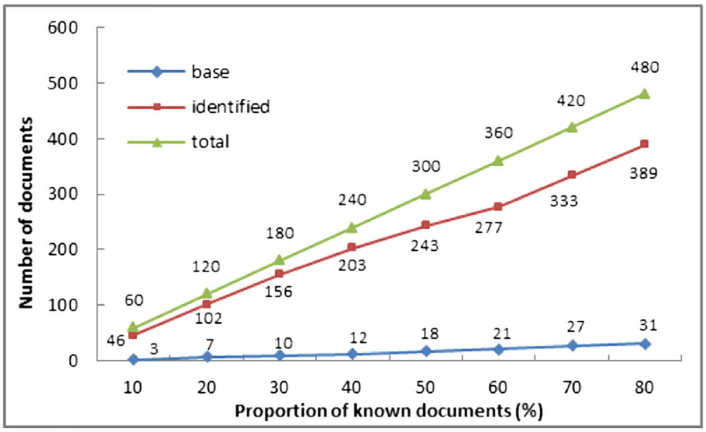

Fig. 4. Document identification results in a single-user case. There are 602 emails in the sent folder of the user. The top (green) line with triangle markers represents the number of documents known to the attacker. The middle (red) curve with square markers represents the number of documents that the attacker can map to specific encrypted documents. (Color figure online)

Document Identification Attack in a Multi-User Case. In a multi-user case, we take the Cloud Access Security Broker (CASB) [20] as the defacto architecture. In a CASB construction, a security control broker sits between cloud applications and a group of customers. Before confidential data is passed into the cloud, the broker intercepts and replaces it with random tokens or encrypted values. Furthermore, several pioneering companies such as Skyhigh Networks [6], CipherCloud [7] and Bitglass [15] have launched their commercial SE products based on CASB.

In this setting, multiple users may share some common documents, and an administrator allows a group of users to generate search tokens. For different users in the same group, the broker may use the same key to generate query tokens. A user query may be performed on the index of documents owned by the group. Initially, an attacker may have partial knowledge about a victim's document set. In the following experiment, assume that the attacker knows the same percentage of User allen-p's documents as in the single-user case, but the encrypted documents include the emails of multiple users. In the extreme setting with 73 users, there are 28,657 emails; 98% of the emails contain less than 252 distinct keywords, and 117 emails contain the unique number of keywords. The experimental results of the base mappings in document identification attack were shown in Fig. 5.

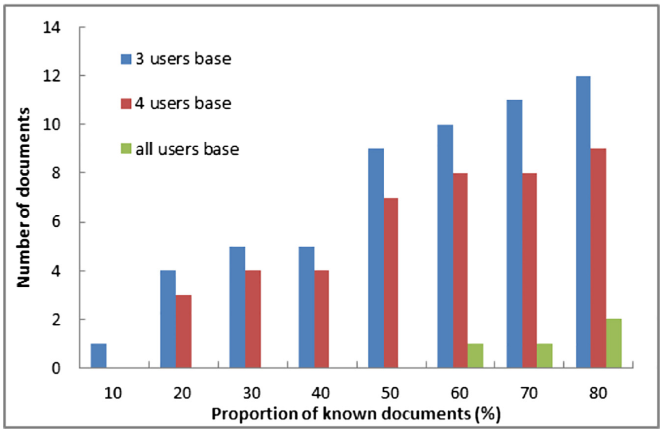

Fig. 5. Base mappings results in a multi-user case. The "3 users base" includes 2825 emails of allen-p, arnold-j and bass-e; the "4 users base" includes 3572 emails of allen-p, arnold-j, bass-e and farmer-d; and the "all users base" includes 28,657 emails of 73 users.

As shown in Fig. 5, in the settings of different proportions of known documents of User allen-p, the bars become shorter as more users are considered. That is, the identified documents in base mappings become fewer as more users are considered. Under the 73-user setting there does not exist a document in the base mappings until the attacker knows 60% of User allen-p's files. From Fig. 6 we can see that, even if the initial base mappings collection contains only one email, the final identified mappings collection can contain as many documents as the document identification result of the single-user case.

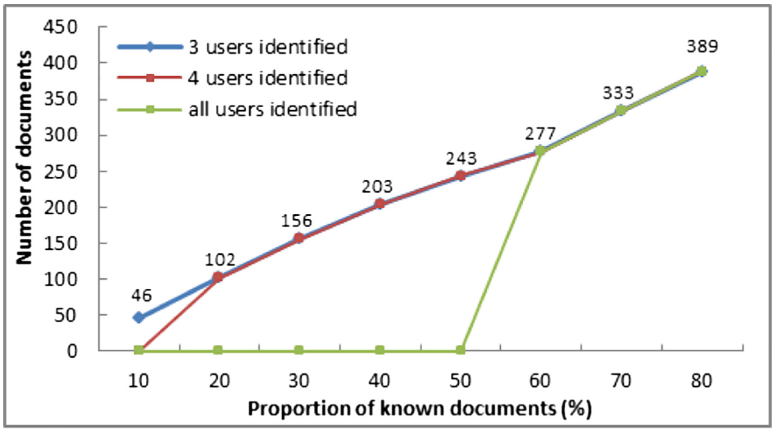

Fig. 6. Results of document identification attack in a multi-user case. Each curve represents the total number of identified documents in different settings.

4.3 Query Recovery Attacks on SE with Partial Knowledge

In this section, we show the experimental results of query recovery attacks after the document identification attack on various SE schemes with partial knowledge. Table 2 shows different numbers of keywords in different identified plaintext documents. In the attack against the Appended-Token SE model that exposing all the search tokens, we select the first 500 most frequent keywords in different identified plaintext documents as the keyword universe, and try to find their corresponding tokens in the identified encrypted documents. In the attack against the Encrypted-Index SE model, based on the selected keyword universe, we randomly selected 150 keywords as the query keyword set. Given the trapdoors of the query keyword set and their query results, we try to find their corresponding keywords in the keyword universe.

Table 2. Identified documents statistics.

Identified documents number	46	102	156	203	243	277	333	389
Keywords number	842	1554	1928	2164	2412	2756	2992	3292

Attack Against the Appended-Token SE Model. After the document identification attack, we use the identified encrypted documents and their corresponding plaintext documents to conduct a query recovery attack against the Appended-Token SE model. As shown in Fig. 7, the x-axis represents the percentage of documents known to the attacker. A "base" point at the bottom represents the number of keywords that have been identified in the base mappings, and the "identified" point at the top represents the total number of identified keywords after applying the keyword co-occurrence comparison algorithm. We use the first 500 most frequent keywords in the identified documents as our keyword universe. When we identify 46 documents of 60 known plaintext documents of User "allen-p", we can invert 44% (220 out of 500) tokens in the identified encrypted documents. When we identify 243 documents of 300 known plaintext documents of the user, we can invert 81% (465 out of 500) tokens. When we identify 389 documents of 480 known plain documents of the user, we can invert all 500 tokens. In contrast, without the proposed document identification attack, the query recovery rates in the settings of different proportional known documents are all close to zero.

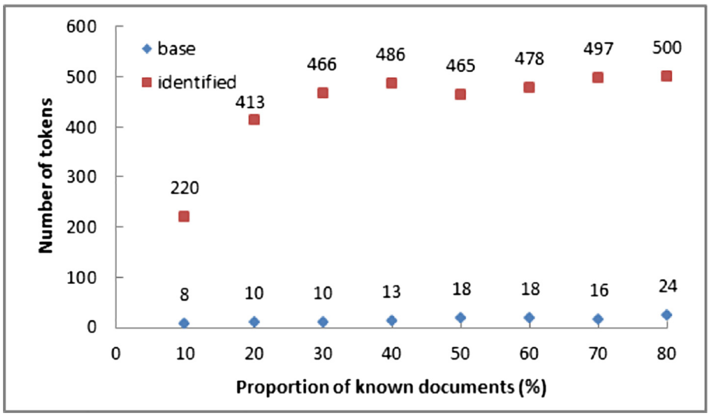

Fig. 7. Invert tokens under the Appended-Token SE model. A "base" point represents the number of tokens uncovered in the base mappings, and an "identified" point represents the number of tokens uncovered when 500 tokens are considered.

Attack Against the Encrypted-Index SE Model. In this setting, if the plaintext content contains timestamps, address or user information, the attacker can also use such auxiliary information to establish the document mappings. For example, in the Enron dataset, the public email contains auxiliary information such as senders, receivers, and timestamps. Since the sender and receiver information cannot be encrypted by the client to use the email service, the server can use this information to build the document identification attack. Using this extended document identification attack with auxiliary information, in a specific application such as email service, we can map any known documents to corresponding encrypted documents. Then, we count the first 500 most frequent keywords in the identified plaintext documents as the keyword universe. We refer $R_q = \{d_1, ..., d_n\}$ as the result sent by the server in response to a query q, such that $d_i = 1$ iff the i-th document contains the keyword corresponding to the query q; and $d_i = 0$ otherwise. For every d_i, if it does not belong to the identified encrypted documents, we set $d_i = 0$ to remove it from the query result. Then, for every query trapdoor, if the returned result contains a unique number of identified encrypted documents, the corresponding keyword has the same occurrence count in the identified known plaintext documents. The server can immediately invert the trapdoor by finding the keyword w such that $count(w) = count(q)$. We can then use a co-occurrence comparison algorithm to build other mappings.

As Fig. 8 shows, we randomly select a subset of 150 keywords from the 500 most frequent keywords in the identified plain documents as query keywords. When we identify 46 documents of known plaintext documents of User "allen-p", we can invert about 48% of the 150 trapdoors. When we identify 243 documents of known plaintext documents of the user, we can invert 93.3% of the 150 trapdoors. Eventually, if we identify 389 documents of known plaintext documents of the user, we can invert all 150 trapdoors. Note that the success rate of query recovery attack is dependent on the randomly selected query keyword set; however, we have repeated the experiments in the same setting many times, and the results are consistent. On the other hand, the success rates of query recovery attack in the settings of different proportions of known documents are all close to zero without the help of our document identification attack, as IKK and CGPR did.

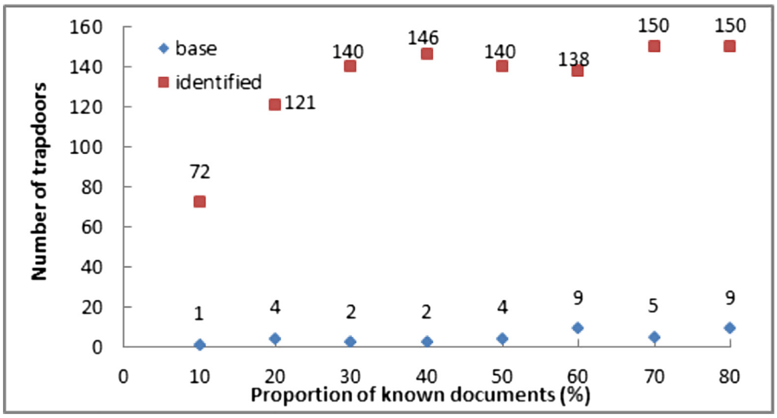

Fig. 8. Invert trapdoors for the Encrypted-Index SE Model. The "base" point represents the number of trapdoors uncovered in the base mappings, and the "identified" point represents the final number of trapdoors uncovered when 150 trapdoors are considered.

5 Mitigation

Inference attacks often use the frequency of keywords and the co-occurrence patterns of multiple keywords to guess the meanings of search trapdoors. So, the protection method needs to disrupt the frequency relationship between keywords and trapdoors. By adding noise to access patterns or search patterns, the observable statistics can be perturbed to a certain extent.

Add Noise to Access Pattern Leakage. To avoid causing an incomplete search result for a keyword, we cannot simply remove items from the search index to add noise to access patterns. An obvious way is padding the number of documents returned for a query. Since we can count the results of queries and filter out irrelevant documents, padding the index using bogus documents does not mitigate our attack effectively. IKK [8] uses the $(a, 0)$-secure index to thwart inference attacks. It aims to make query responses as similar as possible at the expense of increased false positives. Qualitatively, the $(a, 0)$-secure index guarantees that, for each keyword, there are at least other $(a-1)$ keywords that have exactly the same query results. However, it incurs extra communication costs and the client needs to detect and discard the false positives.

Add Noise to Search Pattern Leakage. To obscure the search patterns, an obvious way is to replace a keyword with multiple trapdoors. Liu et al. [21] proposed a grouping-based construction (GBC) to thwart inference attacks. In this scheme, the query generated by the client is a collection of k trapdoors, which includes one search trapdoor of the real keyword that the client wants to search for, and $(k-1)$ trapdoors of randomly selected keywords. GBC used "or" search function, which is not supported in some legacy applications.

Our Countermeasure. We outline an approach to add noises in search patterns and access patterns, and which can be applied to existing legacy applications. The basic idea is as follows. Assume we have a collection of documents D to be encrypted, and a set of keywords W to be queried, and set group size to 2. First, we sort the keywords in a descending order referring to keyword frequency. We map the first keyword and the last keyword to the same token T_1, the second keyword and the second-to-the-last keyword to the same token T_2, ..., until mapping the middle of the two to the same token T_i. In this way, the difference of the frequency of every query token is minimized. So, the adversary cannot perform the query recovery attack accurately based on the leakage of search patterns and access patterns. On the other hand, as the query results contain false positives, we need to filter out extra documents using a secondary map before returning it to the user. For space and efficiency, we simply mark the documents that contain at least one keyword of a group using a bitmap to build the secondary map. For most cloud services, a file often has its uploading timestamp as its attribute. So we can use this attribute to filter the extra results. First, in the index building process, we can sort the document set D_g that contain at least one keyword of a group g in a chronological order. For the group g, we build a bitmap in which location $L_i = 01$ if the i-th document of D_g only contains the less frequent keyword, $L_i = 10$ if the i-th document of D_g only contains the more frequent keyword, and $L_i = 11$ if the i-th document of D_g contains the two keywords in the group g. So, when querying a keyword belongs to a group, we can filter extra documents in the returned results based on the timestamps of the encrypted documents and the bitmap of the group.

Efficiency. We conducted experiments on our prototype to validate the efficiency of our countermeasure. We selected User "allen-p" as the victim. There were 602 emails in its "_sent_mail" folder. We count the frequency of every extracted keyword and selected the 500 most frequent keywords as our keyword universe. We group the first and the last one, the second and the second to the last, ..., until the middle of the two as a group. For keywords in each group, we map them to the same token. Then, we perform inference attacks with known documents and encrypted documents that contain query tokens. If the keywords of a group both appear in a document, then the count of tokens appended to the encrypted document is less than the count of keywords in the corresponding plaintext document, so that the document identification attack does not work well. The results show that, even if we know all documents, we can invert almost none of the query tokens. The experimental results show that our protective measures can effectively prevent query recovery attacks.

6 Conclusion

In this paper, we first introduce two searchable encryption models, including the Encrypted-Index searchable encryption model and the Appended-Token searchable encryption model, and related inference attacks. We then present our document identification attack and query recovery attack based on partial knowledge. We show that the attack is effective even when only a small fraction of documents is known to the attacker. We further design and validate a countermeasure to address this issue.

We plan to further investigate related query recovery attacks. Because the mappings in the document identification process can invert some tokens to their respective keywords, the unknown tokens associated with the remaining ciphertext can be guessed based on known tokens, related public documents, and co-occurrence algorithms, in order to invert as many tokens as possible. Moreover, we will design interactive SE constructions hiding access patterns to prevent inference attacks. A simple way is to keep the document identifiers encrypted in the query result of a search trapdoor, and decrypt it on the client side. The disadvantage is that the client has to spend an extra round-trip time to retrieve the documents.

Acknowledgments. This work is supported by the National High Technology Research and Development Program of China (863 Program) under Grant No. 2015AA016001, Production-Study-Research Cooperation Project in Guangdong Province under Grant No. 2016B090921001, Innovation projects in Shandong Province under Grant No. 2014ZZCX03411, and National Natural Science Foundation of China under Grant No. 61370068.

References

1. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. J. ACM (JACM) **43**(3), 431–473 (1996)
2. Bsch, C., Hartel, P., Jonker, W., et al.: A survey of provably secure searchable encryption. ACM Comput. Surv. (CSUR) **47**(2), 18 (2015)

3. Curtmola, R., Garay, J., Kamara, S., et al.: Searchable symmetric encryption: improved definitions and efficient constructions. J. Comput. Secur. **19**(5), 895–934 (2011)
4. He, W., Akhawe, D., Jain, S., et al.: Shadowcrypt: encrypted web applications for everyone. In: Proceedings of the 2014 ACM Special Interest Group on Security, Audit and Control, Scottsdale Arizona, USA, pp. 1028–1039 (2014)
5. Lau, B., Chung, S., Song, C., et al.: Mimesis aegis: a mimicry privacy shielda system's approach to data privacy on public cloud. In: Proceedings of the 23rd USENIX Security Symposium, SanDiego California, USA, pp. 33–48 (2014)
6. Skyhigh Networks. https://www.skyhighnetworks.com/
7. CipherCloud. https://www.ciphercloud.com/
8. Islam, M.S., Kuzu, M., Kantarcioglu, M.: Access pattern disclosure on searchable encryption: ramification, attack and mitigation. In: NDSS, vol. 20, p. 12 (2012)
9. Cash, D., Grubbs, P., Perry, J., et al.: Leakage-abuse attacks against searchable encryption. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 668–679. ACM (2015)
10. Zhang, Y., Katz, J., Papamanthou, C.: All your queries are belong to us: the power of file-injection attacks on searchable encryption. IACR Cryptology ePrint Archive, 2016:172 (2016)
11. Pouliot, D., Wright, C.V.: The shadow nemesis: inference attacks on efficiently deployable, efficiently searchable encryption. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1341–1352. ACM (2016)
12. Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M.-C., Steiner, M.: Highly-scalable searchable symmetric encryption with support for boolean queries. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 353–373. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_20
13. Li, J., Wang, Q., Wang, C., et al.: Fuzzy keyword search over encrypted data in cloud computing. In: INFOCOM, 2010 Proceedings IEEE, pp. 1–5. IEEE (2010)
14. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, pp. 965–976. ACM (2012)
15. Bitglass. http://www.bitglass.com/
16. Virtru. http://www.virtru.com/
17. Enron Email Dataset. www.cs.cmu.edu/~./enron/. Accessed 13 May 2015
18. Porter, M.: An algorithm for suffix striping. Program **14**(3), 130–137 (1980)
19. Common-English-Words. http://www.textfixer.com/tutorials/common-english-words.txt/
20. Gartner Report: How to Evaluate and Operate a Cloud Access Security Broker, 8 December 2015
21. Liu, C., Zhu, L., Wang, M., et al.: Search pattern leakage in searchable encryption: attacks and new construction. Inf. Sci. **265**, 176–188 (2014)