IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON CYBER-PHYSICAL-SOCIAL COMPUTING AND NETWORKING

Received December 4, 2017, accepted January 23, 2018, date of publication February 1, 2018, date of current version May 9, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2800684

Leakage Models and Inference Attacks on Searchable Encryption for Cyber-Physical Social Systems

GUOFENG WANG[1], CHUANYI LIU[2], YINGFEI DONG[3], KIM-KWANG RAYMOND CHOO[4], (Senior Member, IEEE), PEIYI HAN[1], HEZHONG PAN[1], AND BINXING FANG[2]

[1]School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, 100876, China
[2]School of Computer and Technology, Harbin Institute of Technology, Shenzhen, 518055, China
[3]Department of Electrical Engineering, University of Hawaii, Honolulu, HI 96822, USA
[4]Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249-0631, USA

Corresponding author: Chuanyi Liu (cy-liu04@mails.tsinghua.edu.cn)

This work was supported in part by the National High Technology Research and Development Program of China 863 Program under Grant 2015AA016001, in part by the Production-Study-Research Cooperation Project in Guangdong Province under Grant 2016B090921001, in part by the Innovation projects in Shandong Province under Grant 2014ZZCX03411, and in part by the National Natural Science Foundation of China under Grant 61370068.

ABSTRACT Searchable encryption (SE) schemes, such as those deployed for cyber-physical social systems, may be vulnerable to inference attacks. In such attacks, attackers seek to learn sensitive information about the queries and data stored on the (cyber-physical social) systems. However, these attacks are often based on strong (impractical) assumptions (e.g., the complete knowledge of documents or known document injection) using access-pattern leakage. In this paper, we first identify different leakage models based on the leakage profiles of common SE schemes, and then design inference methods accordingly. In particular, based on the leakage models, we show that some information leakage allows a very powerful attack with little prior knowledge. We then propose new inference attacks in which an adversary only needs to have a partial knowledge of target documents. Unlike previous attacks, the proposed inference algorithms perform effective document-mapping attacks before query recovery attacks, in the sense that they are more efficient and scalable without requiring optimization overheads. We then use experiments to demonstrate their effectiveness.

INDEX TERMS Inference attacks, searchable encryption, leakage models, partial knowledge.

I. INTRODUCTION

Encrypting data on the client side prior to outsourcing to cyber-physical cloud servers can be an effective way to protect mission-critical applications. However, client controlled encryption that hides all information conflicts with the search functions in cloud applications. While oblivious RAM (ORAM) [1] can support encrypted search functions and hide all information, it is inefficient to be deployed in practice. Searchable encryption (SE) [2]–[7] achieves good balance between security and efficiency, and has been the subject of several research efforts [8]–[10].

In existing SE constructions, functionality, security, and efficiency often conflict with each other [11]. Many SE schemes use only symmetric cryptography [2]–[4], and they are mainly used to allow a single user to search its own documents. An asymmetric SE scheme [5] allows a user to search the documents of multiple users, but is less efficient. Typically, SE schemes leak certain information to obtain better performance. For example, *index-based SE* schemes [2]–[5] allow an adversary (e.g., a curious cloud server) to learn search and access patterns by observing statistics. In addition, to perform search operations on an encrypted index using specific algorithms, index-based SE schemes require some compromise on cloud's Application Programming Interface (API), which hinders their broad deployment. To achieve backward compatibility with legacy systems, *token-based SE* schemes [6]–[7] append a sequence of searchable tokens to the cihpertext of a document, which allows the application server to use its original search functions. Many commercial products [12]–[14] also use such scheme to support SE functions in cloud services. However, a token-based SE scheme exposes token occurrence patterns

and leaks more sensitive information than an index-based SE scheme.

Islam *et al.* [15] first investigated the implications of access-pattern disclosure in the SE schemes. They presented their attack model as an optimization problem and used simulated annealing [16] to find the best match of query trapdoors and keywords. Cash *et al.* [17] proposed leakage abuse attacks that exploited the leakage profiles of various SE schemes to infer more sensitive information about queries and documents. The Shadow Nemesis [18] launched inference attacks on token-based SE schemes. It used the Weighted Graph Matching (WGM) [19] problem to invert search tokens. For a certain size of keyword vocabulary, both the above inference attacks require almost the complete knowledge of target documents to achieve a high query recovery accuracy. However, knowing all documents of a victim is unrealistic for an adversary in normal cases. Furthermore, in a dynamic SE scheme, the victim can bring interference information (add and remove documents, or pad the dataset with bogus documents) to the dataset. When only knowing a small proportion of a document corpus, the above attacks can invert almost no search trapdoors. Giraud, *et al.* [20] presented effective partial knowledge attacks against token-based SE using token occurrence patterns. However, their attacks are not applied to index-based SE schemes.

In this paper, we investigate the leaked information of various SE schemes and propose five *leakage types* that can be used by adversaries, including the number of keywords per document, the order of keywords per document, the similarity of different documents, the keyword occurrence frequency across all documents, and the keyword co-occurrence frequency across all documents. We then present different *leakage models* based on the leakage types of SE schemes. According to the leakage models, we propose new inference attacks in which an adversary only needs to know a small proportion of target documents. The attack methods first find the correspondence between known documents and their ciphertexts revealed by the SE schemes, then omit the not-mapped documents from query results and known documents, such that they can infer the queries with a high success rate. In addition, our attacks can be applied in a wide range of systems, such as cyber-physical social systems. This paper contributes to the literature, as follows.

1) *We examine the leaked information of provably secure SE schemes, and propose five leakage models that may be exploited by adversaries*. Existing attacks on SE schemes usually pay attention to the characteristics of keywords, such as keyword occurrence frequencies and keyword co-occurrence frequencies, without considering the characteristics of documents, such as the number of keywords per document, the order of keywords in a document, and the similarity of different documents. Our attacks show that such extra leakage allows a very powerful attack with little prior knowledge. We classify different SE constructions according to the above leakage types, and point out the limitations of current inference attacks on various SE schemes.
2) *We design new passive inference attacks in which an adversary only needs to know a portion of target documents*. We explore different attack methods that exploit the proposed leakage models, respectively. In particular, we propose new inference attacks in which an adversary only needs to know a portion of a victim's documents and does not need to inject known documents. Our inference algorithms build document mappings to get rid of not-mapped documents before a query recovery step, such that they can achieve a high accuracy with good performance, and are also applied to dynamic or padding SE schemes.
3) *We conduct experimental evaluations to demonstrate the effectiveness of the proposed attacks*. We investigate the leakage models that may be exploited by our inference attacks on SE, and develop attack experiments that are efficient and scalable without common optimization overheads. Finally, our experimental results show the feasibility and efficacy of the proposed scheme.

We organized this paper as follows. We introduce the background information in Sec. 2. The leakage models of SE schemes are introduced in Sec. 3. In Sec. 4, we present attacks against index-based SE schemes and token-based SE schemes. We present our evaluation in Sec. 5. We further discuss related work in Sec. 6 and conclude this paper in Sec. 7.

II. BACKGROUND INFORMATION

In this section, we introduce the background information and define the notations used in this paper. We let n denote the total number of documents in a collection $D = \{D_1, D_2, \ldots, D_n\}$, and $ID(D_i)$ is the identifier of a document D_i. Let $D(w)$ denote a lexicographically ordered list, consisting of the IDs of all documents in D that contain the keyword w. It can also represent the outcome of a search for w. We denote $|D(w)|$ as the number of documents in $D(w)$. We use m to denote the total number of keywords in a dictionary $W = \{w_1, w_2, \ldots, w_m\}$.

A SE scheme consists of encryption, search, and possibly update algorithms. The encryption algorithm E takes a secret key K and a document D_i as inputs, and generates a ciphertext $E_K(D_i)$. A search operation takes a secret key k and a keyword w as inputs, and outputs a query trapdoor TD using a pseudo-random function f. The update algorithm takes K and D_i as inputs, and outputs an update message. A SE scheme is dynamic if it includes the update algorithm. For a sequence of query keywords $(w_1, w_2, \ldots, w_i)$ of a user, we refer to the sequence $(D(w_1), \ldots, D(w_i))$ as an access pattern. We refer to the search pattern of a user as any information that can be derived from whether two arbitrary queries were performed for the same keyword or not.

A keyword extraction function takes an input document D_i and outputs a keyword vector V_i. We assume the keyword extraction function is deterministic and known to the adversary. A typical keyword extraction procedure will

first parse D_i into words, drop stop-words (e.g., "to" and "about"), stem the remaining words, and remove duplicate words. Let $V = (V_1, \ldots, V_n)$ be the ordered list of all the keyword vectors of D.

In the inference attacks on common SE schemes, we classify the prior knowledge of an adversary into two types: (i) Complete knowledge of target documents: An adversary knows all the document corpus. (ii) Partial knowledge of target documents: An adversary knows a subset of documents. We also classify attack modes into two types: (i) In a passive attack, an adversary relies on statistical analysis of the leakages of SE schemes and prior knowledge to get sensitive information. (ii) In an active attack, an adversary can perform any operation on the server. For example, an adversary can proactively inject known documents with specific contents to a user's dataset, then observe the query results to get sensitive information.

We define the following terminologies to characterize the leakage profiles of various SE schemes.

Keyword number $Num(D_i)$ - the number of unique keywords contained in a document D_i.

Document similarity $Sim(D_i, D_j)$ - the number of the common keywords in both document D_i and document D_j.

Keyword order $Ord(w_i)$ - the first appearance order of a query keyword w_i in a document. In other words, we can determine the position of the query keyword in the keyword vector of a document from the SE leakage profile.

Keyword frequency $Fre(w_i)$ - the occurrence frequency of keyword w_i in a document set D (or the number of documents contain the keyword w_i over $|D|$).

Keyword co-occurrence frequency $Co - Fre(w_i, w_j)$ - the co-occurrence frequency of keywords w_i and w_j in a document set D (or the number of documents in which keyword w_i and keyword w_j both appear over $|D|$).

A. Our Attack Model

Based on the leakages models of common SE schemes and prior knowledge, an adversary can perform inference attacks to obtain sensitive information about user documents and queries.

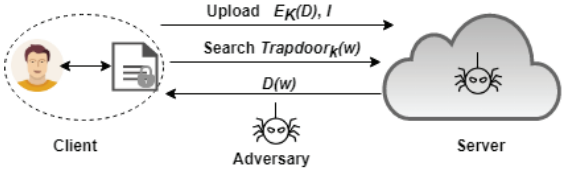

FIGURE 1. Inference attack model in the SE architecture.

Normally, an adversary rarely has a complete knowledge of the entire document set of a victim, but it can easily learn a subset, e.g., some well-circulated emails. Then, the adversary can intercept the communications between the victim and the server (or it is a curious server), to get the leaked information of a SE scheme, as shown in Fig. 1. Particularly, in a multi-user SE scheme, multiple users may send messages to each other, and a user may search the documents encrypted by other users. If the server colludes with a malicious user who shares some documents with an honest user, the honest user ends up sharing some documents with adversaries. So our inference attacks are very practical.

We build new passive inference attacks against different leakage models of common SE schemes in which an adversary only needs to know a small part of the target document set. The attacks work on any schemes that allow the required amount of leakage or more. Before performing a query recovery attack with statistical query results and prior partial knowledge, *the adversary seeks to find the correspondence of known documents to the encrypted documents (or encrypted document IDs) revealed by the SE schemes*. After obtaining as many mappings between known documents and encrypted documents as possible, the algorithms get rid of not-mapped documents from the query results and known documents. *In the query recovery step, the adversary seeks to find the correspondence of keywords in the mapped known documents to the search trapdoors whose query results containing only mapped encrypted documents*.

III. SE LEAKAGE MODELS

Given a collection of n documents $D = (D_1, \ldots, D_n)$, and its keyword collection $W = (w_1, \ldots, w_m)$, an index-based SE scheme contains the following steps.

1) A user generates a searchable encrypted index $I = BuildIndex_k(D = (D_1, \ldots, D_n), W = (w_1, \ldots, w_m))$ using a secret key k, and the encrypted documents $E_K(D)$ using a secret key K independently, and then uploads them together to a server.
2) Thereafter, when searching for a keyword w, the user applies the keyword to a trapdoor function f and sends generated trapdoor $TD = Trapdoor_k(w)$ to the server.
3) With the trapdoor TD, the server can search on the encrypted index I using specific algorithms and return the corresponding encrypted documents $D(w) = Search(I, TD)$.

Curtmola, *et al.* [4] introduced the security analysis of SE using leakage profiles. A leakage profile of a SE scheme is a function taking an interaction between a client and a server over Q queries as input and characterizing what an adversary actually "sees" by taking part in the execution of the SE scheme. How much information leaked from index-based SE is depend on how the index is constructed and encrypted, and how the queries are performed given search trapdoors.

Song, *et al.* [2] constructed the first practical SE scheme without leaking the information of plaintext documents, but used no formal security definition for SE. For each keyword in a document, it constructs a searchable ciphertext in order, which leaks the $Num(D_i)$ information. Each keyword is processed into a random value, so that the same two keywords are encrypted into different values, thus hiding the $Sim(D_i, D_j)$ information. However, given a sequence of query responses, an adversary can learn the information of $Ord(w_i)$, $Fre(w_i)$ and $Co - Fre(w_i, w_j)$. We called this the $LM3$ leakage model. Mylar [21] and CryptDB [22]

also used this scheme to construct their SE functions. However, CryptDB sorted the keywords of a document before encryption, thus hiding the $Ord(w_i)$ information.

Goh [3] used Bloom Filter (BF) to generate a searchable index for each document. Each keyword in a document is encrypted deterministically, and the encrypted values are inserted into the BF by setting corresponding bits to 1. This process makes the BFs of various documents different, even for the documents with the same keyword set. It avoids leaking the $Sim(D_i, D_j)$ information and the $Ord(w_i)$ information before queries are issued. However, regardless of padding, the size of the BF index per document is proportional to the number of distinct keywords in the document. The server can learn the $Num(D_i)$ information by counting the bits which are set to 1 in the BF. In a query process, an adversary can learn the information of $Fre(w_i)$ and $Co - Fre(w_i, w_j)$. We called this the $LM2$ leakage model.

Curtmola, *et al.* [4] first constructed an encrypted inverted index, which built an index on all documents instead of each document. For each keyword w, it built a linked list consisting of $|D(w)|$ nodes. A node contains three fields $< a||b||c >$, where a represents an identifier of a document containing the keyword w, b is a key used to decrypt the next node, and c is a pointer to the next encrypted node. All nodes are scrambled in a random order and encrypted with random keys. Before queries are issued, the server learns nothing except the sizes of the documents and the index. During a series of queries, the scheme lets nothing but the search and access patterns be known to the adversary, thus leaking the information of $Fre(w_i)$ and $Co-Fre(w_i, w_j)$. We called this the $LM1$ leakage model. Their provable security definitions for SE are widely used as the standard definitions [11], [23].

The above SE schemes are based on symmetric encryption. The PEKS scheme [5] used asymmetric encryption to generate encrypted indexes. Each keyword of a document is encrypted into a random value using a user's public key, and appended to the encrypted document. In this way, multiple users can use a user's public key to generate encrypted indexes, and only the user having the private key can generate a legitimate query trapdoor to search on the encrypted indexes. This scheme leaks the information of $Num(D_i)$ and $Ord(w_i)$, but hides the $Sim(D_i, D_j)$ information. In addition, the query trapdoors and results leak the $Fre(w_i)$ information and the $Co - Fre(w_i, w_j)$ information.

As mentioned earlier, common SE schemes usually come in two classes: the index-based SE and the token-based SE. The former is more secure than the latter, but requires modification at the cloud provider. In the following, we introduce the token-based SE schemes that can be deployed efficiently.

Given the keyword vector $V_i = (w_1, w_2, \ldots, w_{ci})$ of a document D_i, where ci represents the number of unique keywords contained in the document D_i, a token-based SE scheme contains the following steps.

1) A user extracts the keywords from D_i, generates a token for each keyword by deterministically encrypting the keyword using f and k, then encrypts document D_i using the secret key $K(K \neq k)$. Finally, she appends a sequence of ci sorted tokens to the ciphertext, and uploads $S = (..., E_K(D_i)||f_k(w_1), \ldots, f_k(w_{ci}), ...)$ to the cloud, where "||" denotes the concatenation operation.
2) To search for a keyword w, the user applies w to f and sends generated token $TK = f_k(w)$ to the server.
3) With the token TK, the server can search on the uploaded data S, using the original functions and return corresponding encrypted documents $D(w) = Search(S, TK)$.

From the above steps, we can see that token-based SE provides no additional protection of token occurrence patterns. For each document, the token set $(f_k(w_1), \ldots, f_k(w_{ci}))$ leaks the information about $Num(D_i)$ and $Sim(D_i, D_j)$. According to the upload data S, we can also build token-document mappings, i.e., for each keyword w_i, all the documents containing w_i can be deduced. Similarly, the number of documents in which keyword w_i and keyword w_j both appear can also be inferred. Thus the token-based SE scheme reveals the information of $Fre(w_i)$ and $Co - Fre(w_i, w_j)$ without the help of query information.

ShadowCrypt [6] appended sorted tokens to encrypted contents to disturb the order between tokens and keywords. Mimesis aegis [7] changed the one-to-one mapping between keywords and tokens to a one-to-many mapping, increasing the difficulty of statistical analysis. However, the mappings between keywords and their corresponding tokens can also be deduced by an adversary, as in [18]. From these two methods, an adversary cannot learn the $Ord(w_i)$ information. Other commercial SE products from Skyhigh [12], CipherCloud [13], and Virtue [14] also use this model or its variants to support SE functions in some cloud services. We called this the $LM4$ leakage model. It should be noted that when all the keywords are queried, the $LM1$, $LM2$ and $LM3$ leakage models also expose the above information.

Some commercial products [12], [13] make use of the scheme in which tokens are placed in the keyword appearance order to support searching for specific phrases, thus leaking the $Ord(w_i)$ information. We called this the $LM5$ leakage model. Note that the $LM3$ leakage model degenerates to the $LM5$ leakage model after all keywords have been queried.

TABLE 1. Leakage models of common SE schemes.

Models	Num	Ord	Sim	Fre	Co-Fre	Schemes
LM1	N	N	N	Y	Y	[4][23]
LM2	Y	N	N	Y	Y	[3][22]
LM3	Y	Y	N	Y	Y	[2][5][21]
LM4	Y	N	Y	Y	Y	[6-7][12-14]
LM5	Y	Y	Y	Y	Y	[12][13]

Based on the above analysis, we summarize the following leakage models $LM1 - LM5$ of common SE schemes, in which $LM1$ reveals the least information, as shown in Table 1. It should be noted that two very different schemes may be classified into the same leakage model, and an adversary may get information from other sources (e.g., the document size and the index size). In addition, an attack against a

specific leakage model can also be effective to other leakage models. For example, the attack against $LM3$ can also be applied to $LM5$, and the attack against $LM2$ can also be applied to $LM3$, $LM4$, $LM5$.

IV. INFERENCE ATTACKS ON SE SCHEMES

In this section, we present the inference attacks against index-based SE schemes and token-based SE schemes respectively. In the attacks against the leakage modes ($LM1$, $LM2$ and $LM3$) of index-based SE schemes, an adversary must make use of the query information. On the other hand, an adversary can launch attacks against the leakage models ($LM4$ and $LM5$) of token-based SE schemes without query information.

A. ATTACK AGAINST LM3

When having complete knowledge of all documents, an adversary can map underlying trapdoors to respective keywords by comparing the keyword distribution and the query information statistics [15]–[18]. If the adversary has only partial knowledge, as the distribution between keywords and trapdoors do not match well, it is very hard to invert the search trapdoors. To address this issue, we first perform document mappings between the known documents and their ciphertexts. To depict our inference methods, we first set forth the following definition.

Definition 1: For the encrypted document set, given a known document D_i, if there exists a unique encrypted document in which the leaked number of keywords is equal to the number of plaintext keywords in D_i, then we can build the mapping between the identified encrypted document and the plaintext document D_i as a base mapping directly.

According to Definition 1, if we have the information of the number of keywords in a document, we can always get the base mapping set M. Prior to searching, the $LM3$ leakage model leaks the number of keywords in a document and the order of keywords of a document. From this leakage model, we can build the base mapping set according to Definition 1. If multiple encrypted documents leak the same number of keywords as the number of plaintext keywords of a plaintext document, we can use the leaked information from query responses to filter the candidate encrypted documents. To depict the attack algorithm against $LM3$, we define *the union set of unique keywords* in a document set $D = \{D_1, D_2, \ldots, D_n\}$ as:

$$KeyUni(D) = V_1 \cup V_2 \cup \ldots \cup V_n \tag{1}$$

We define *the difference set of unique documents* that belong to a document set S but do not belong to D as:

$$DocDiff(S, D) = S - D \tag{2}$$

In addition, we give the following definition.

Definition 2: Under a SE scheme leaking the keyword order information, if a query result of a search trapdoor contains an encrypted document D_i that belongs to the mapped document set, then according to the match location of the query in D_i, we can determine the query keyword directly by finding the keyword at the same location in the corresponding plaintext document.

The document mapping algorithm using query information against the $LM3$ leakage model (such as [2], [5], and [21] schemes) is shown in Algorithm 1, in which $R(q)$ represents the query result of a query q, $r(L)$ denotes a query hits the location L of an encrypted document r, and $d(L)$ denotes the L-th keyword of the keyword vector V_d of the plaintext document d. $Set(E)$ represents the set of mapped encrypted documents, while $Set(P)$ represents the set of corresponding mapped plaintext documents.

Algorithm 1 Document Mapping Algorithm on $LM3$

```
input : all encrypted document set e, plaintext
        document set p;
output: mapping set between p and e;
initialize the base mapping set M;
while size of M is increasing do
    for each query q do
        for each not-mapped plaintext document
        d ∈ p − M do
            set candidate encrypted document set S = {s
            : leaked keyword count of s equals the
            keyword count of d};
            if R(q) ∩ Set(E) ≠ ∅ then
                get query keyword k from Set(P)
                according to Definition 2;
                if k in d then
                    get appearance location L of k in d;
                    for r ∈ S do
                        if q not hit r(L) then
                            remove r from S;
                else if k not in d then
                    S = S − R(q);
            else if R(q) ∩ Set(E) = ∅ then
                for r ∈ S do
                    if r ∈ R(q) then
                        get match location L of q in r;
                        get L location keyword
                        k′ = d(L);
                        if k′ ∈ KeyUni(Set(P)) then
                            remove r from S;
        if one document s remains in S then
            add (d, s) to M;
return the mapping set M;
```

An example of document mapping attack against the $LM3$ leakage model is shown in Fig. 2, in which *Pdoc* represents ''plaintext document'', and *Edoc* represents ''encrypted document''. First, when a query for 'shape' hits a mapped encrypted document, we can obtain the keyword from the corresponding plaintext document. Second, we have two

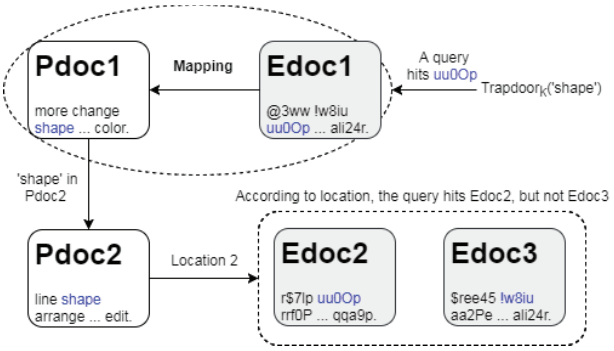

FIGURE 2. A document mapping example in the *LM*3 leakage model. There exist two known plaintext documents, and three encrypted documents. According to built mappings, *Pdoc*2 is mapped to *Edoc*2.

candidate encrypted documents *Edoc*2 and *Edoc*3 for *Pdoc*2. As the keyword 'shape' belongs to *Pdoc*2 at location 2, and the query hits location 2 in *Edoc*2, not *Edoc*3, so we can determine the unique mapping between *Edoc*2 and *Pdoc*2.

Based on the document mapping algorithm against the *LM*3 leakage model, we first delete the encrypted documents (or *ID*s) in query results that do not belong to the mapped document set. Then, referring to the count attack [17], we build the mappings between the query trapdoors that hit more than one mapped encrypted document and the keywords in the mapped plaintext documents according to the information of keyword frequency and keyword co-occurrence frequency, which we called query recovery algorithm, as shown in Algorithm 2. We first initialize the base mapping set Q between trapdoors contained in a unique number of mapped encrypted documents and keywords in the same number of mapped plaintext documents. Then we build a trapdoor co-occurrence count matrix T', where $T'[i, j]$ represents the number of mapped encrypted documents trapdoor t_i and t_j both hit. Similarly, a keyword co-occurrence count matrix K' can also be built, where $K'[i, j]$ represents the number of mapped plaintext documents in which keyword w_i and keyword w_j both appear.

Algorithm 2 Query Recovery Algorithm

input : query trapdoors T and results in mapped encrypted documents e', keyword set W in mapped known documents p';
output: mapping set between keywords in W and trapdoors in T;

```
initialize the base mapping set Q;
compute the matrix T' for trapdoors in e' and the K' for keywords in p';
while size of Q is increasing do
    for each unknown trapdoor t ∈ T − Q do
        build candidate keyword set S = {s : the occurrence count of s in p' equals to the occurrence count of t in e'};
        for s ∈ S do
            for known base mapping (t', s') ∈ Q do
                if T'[t, t'] ≠ K'[s, s'] then
                    remove s from S;
        if one keyword s remains in S then
            add (t, s) to Q
return the mapping set Q;
```

B. ATTACK AGAINST LM2

Prior to searching, the *LM*2 leakage model only leaks the keyword number information. From this leakage model, we can build the base mapping set according to Definition 1.

If multiple encrypted documents are mapped to a plaintext document according to the number of keywords, we use the leaked information from the query process to identify the unique encrypted document. We define the *intersection set* of unique keywords in a document set $D = \{D_1, D_2, \ldots, D_n\}$ as:

$$KeyInter(D) = V_1 \cap V_2 \cap \ldots \cap V_n \tag{3}$$

Similarly, equation (1), equation (2) and equation (3) also apply to search trapdoors in encrypted versions. In addition, we give the following definition.

Definition 3: If a query result of a search trapdoor contains an encrypted document set E' that is a subset of the mapped document set, then the query keyword must belong to the intersection set $KeyInter(P')$ of corresponding plaintext documents P'.

According to the above definitions, the document mapping algorithm against the *LM*2 leakage model (such as [3] and [22] schemes) is shown in Algorithm 3. When a query result $R(q)$ of a search trapdoor q contains an encrypted document set E' that is a subset of the mapped document set, we can then filter the candidate encrypted document set S, according to the relationship between the keyword intersection set $KeyInter(P')$ of corresponding plaintext documents P' and the keyword vector V_d of the not-mapped plaintext document d.

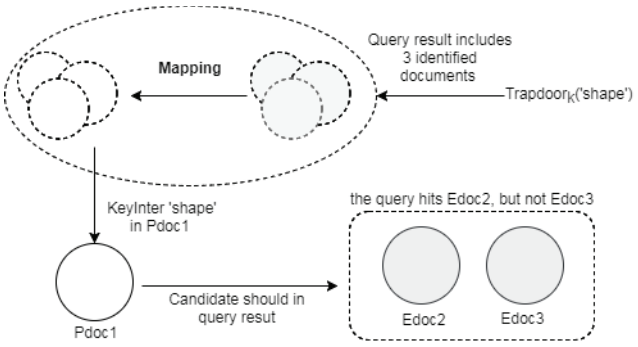

FIGURE 3. A document mapping example in the *LM*2 leakage model. According to built mappings, *Pdoc*1 is mapped to *Edoc*2.

An example of document mapping attack against the *LM*2 leakage model is shown in Fig. 3. First, when a query for "shape" hits three mapped encrypted documents, then the

Algorithm 3 Document Mapping Algorithm on $LM2$

```
input : all encrypted document set e, plaintext
        document set p;
output: mapping set between p and e;
1  initialize the base mapping set M;
2  while size of M is increasing do
3      for each query q do
4          for each not-mapped plaintext document
           d ∈ p − M do
5              set candidate encrypted document set S = {s
               : leaked keyword count of s equals the
               keyword count of d};
6              if E' = R(q) ∩ Set(E) ≠ ∅ then
7                  if KeyInter(P')∩ keyword set V_d of
                   d = ∅ then
8                      S = S − R(q);
9                  else if KeyInter(P') ∈ V_d then
10                     S = S ∩ R(q);
11             else if R(q) ∩ Set(E) = ∅ then
12                 if V_d ∈ KeyUni(Set(P)) then
13                     S = S − R(q);
14         if one document s remains in S then
15             add (d, s) to M;
16 return the mapping set M;
```

keyword intersection including the query keyword can be obtained from the corresponding plaintext documents. Second, we have two candidate encrypted documents $Edoc2$ and $Edoc3$ for $Pdoc1$. If the keyword intersection is a subset of the keyword set of $Pdoc1$, $Pdoc1$'s candidate should belong to the query result. As $Edoc3$ is not in the query result, we can determine the mapping between $Edoc2$ and $Pdoc1$.

Based on the document mapping algorithm against $LM2$, we delete the encrypted documents (or IDs) in the obtained query results that do not belong to the mapped document set. Then, we can build the mappings between the query trapdoors that hit more than one mapped encrypted document and keywords in the mapped plaintext documents according to keyword frequency information and keyword co-occurrence frequency information, as shown in Algorithm 2.

C. ATTACK AGAINST LM1

From the $LM1$ leakage model (such as [4] and [23] schemes), we can learn nothing except the sizes of the documents and the index prior to search. We can only use the leaked information from the query process to perform the document mappings. Actually, if we intercept a set of queries Q for enough time, it has a good chance to count the most high-frequency keywords. Similarly, we can count the most frequent keywords in known plaintext documents. Based on the query results and known documents, we can make an initial guess that we map some encrypted documents to corresponding plaintext documents as base mappings, map the remaining encrypted documents to a plaintext document as candidates, and then run the remainder of Algorithm 3. For example, we can map the most frequently occurring encrypted documents in a series of query results to the plaintext documents containing the most high-frequency keywords, or we can count the least frequent keywords in the known plaintext documents. If the keywords are queried, and the query results contain the same number of documents, then we can build the mappings accordingly. In addition, we can also use the document size information to build document mappings. If the initial guess is wrong, and Algorithm 3 detects an inconsistency, we will try other mappings.

Remarks: Our inference attacks against index-base SE schemes with partial knowledge can also be applied to dynamic SE schemes, in which a user may add or delete some documents, because our methods first build the document mappings between known documents and encrypted documents to remove interference information (not-mapped documents). Similarly, the padding method that adds sufficient bogus documents (or IDs) to the query results has no effect on our document mapping attacks, because we first build base mappings using keyword number information of a document, then build other mappings with the help of query results and the base mappings. After the document mapping attack, we can remove the interference information (not-mapped documents) from the query results and known documents.

D. ATTACK TOKEN-BASED SE WITHOUT QUERY INFORMATION

Prior to searching, the $LM4$ leakage model (such as [6], [7], and [12]–[14] schemes) exposes the information of $Num(D_i)$ and $Sim(D_i, D_j)$. We can build the base mapping set according to Definition 1. If multiple encrypted documents have the same number of tokens as the number of keywords in a plaintext document, we can filter the candidate encrypted documents by comparing the document similarity with the help of base mappings, as shown in Algorithm 4. At line 2, we build a known documents similarity matrix C_p, where $C_p[i, j]$ denotes the number of the same keywords in two documents D_i and D_j, and an encrypted documents similarity matrix C_e where $C_e[i, j]$ denotes the number of common tokens in encrypted documents D_i and D_j.

After document mappings, we cannot directly map the tokens in the mapped encrypted documents to the keywords in the corresponding plaintext documents by order, as the keyword order information of a document is hidden. To address this issue, based on Algorithm 2, we can build the mappings between the tokens in the mapped encrypted documents and the keywords in the corresponding plaintext documents, referring to the keyword/token occurrence pattern and the keyword/token co-occurrence pattern. In this case, we do not need to count the query information, as all search tokens and their distributions are exposed, once encrypted data is uploaded. In the token-based SE scheme, the contents of other

Algorithm 4 Document Mapping for Token-Based SE

input : all encrypted document set e, plaintext document set p;
output: mapping set between p and e;
1 initialize the base mapping set M;
2 compute the similarity matrix C_p for p and the similarity matrix C_e for e;
3 **while** *size of* M *is increasing* **do**
4 **for** *each not-mapped plain document* $d \in p - M$ **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* $(d', s') \in M$ **do**
8 **if** $C_p[d, d'] \neq C_e[s, s']$ **then**
9 remove s from S;
10 **if** *one encrypted document* s *remains in* S **then**
11 add (d, s) to M
12 return the mapping set M;

not-mapped encrypted documents can be learned if it contains the mapped tokens.

According to Algorithm 2, we first build the base mapping set Q between the tokens contained in a unique number of mapped encrypted documents and the keywords in the same number of mapped plaintext documents. At line 2, we build a 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. Similarly, a keyword co-occurrence count matrix K' can also be built, where $K'[i, j]$ represents the number of documents in which keyword w_i and keyword w_j both appear. We then run the reminder of Algorithm 2 to invert the tokens.

In the attacks against the $LM5$ leakage model (such as [12] and [13] schemes), we first build the mappings between the known documents and the encrypted documents according to Algorithm 4. In this way, the contents of mapped encrypted documents can be learned by viewing the corresponding plaintext documents. After that, the tokens in the mapped documents can be mapped to the keywords in corresponding plaintext documents directly, according to the keyword order information.

Remarks: In a token-based SE scheme, if a user pads a document with randomly selected irrelevant keywords, the document mapping attacks according to the number of keywords information of a document may be influenced to a certain extent. However, we can make an initial guess to map some encrypted documents to corresponding plaintext documents as a base mapping set, then run Algorithm 4 using document similarity information. If the guess is wrong, the remainder procedures would detect inconsistency, and then we will try another candidate.

V. EVALUATION

In this section, we investigated the vulnerability of different leakage models with simulated inference attacks, using a publicly available email dataset [24]. We conducted experiments to validate the effectiveness of the document mapping attacks and the improved success rate of query recovery attacks. The testing virtual machine has an Intel 2.5 GHz dual-core with 8 GB memory.

The efficiency of the inference attack is related to the total number of encrypted documents, the number of known documents, the number of keywords in each document, and the distribution of keywords in all documents. In general, the attacks against $LM4$ and $LM5$ depend on the number of keywords in the document and the keyword distribution of all documents. After the similarity matrix is set up, it takes very little time to complete the attacks. In the above experiments, even the number of known documents is very large (such as 80%), the running time of an attack is only a few minutes. In the attacks against $LM2$ and $LM3$, the similarity matrix does not need to be calculated. However, the attacks need to do a lot of union or intersection calculation of the keywords in the documents. Therefore, the running efficiency is relatively slow. When there are many known documents (such as 80%), the running time of an attack may take a few hours. Here we omit the attacks against $LM1$, since there are no fixed attack algorithms that are easy to implement for $LM1$.

A. EXPERIMENTAL SETUP

We used the Enron [24] email dataset as our test data. We chose sent-emails from the "_sent_mail" folder of 78 employees, resulting in a total of 30,109 messages.

An email is considered as one document. We extracted searchable keywords from the dataset as follows. We strip the first few lines of each email off in a preprocessing step, because these lines usually contain auxiliary information about the email, such as senders, receivers, and timestamps. We stemmed the words in each email using the standard Porter stemming algorithm [25]; then removed stop-words [26] as well as duplicate keywords. There are 49,982 distinct keywords in the selected 30,109 messages.

Given a set of n emails, the keyword extraction process produces a set of distinct keywords for each email, resulting in n keyword sets. Assume there are M distinct keywords in a document set, we establish a fixed-size keyword vocabulary from the keywords by taking the most frequent m keywords. We always choose the most frequent m keywords as the keyword universe; unless noted otherwise.

In our inference attacks, the adversary only knows a subset of documents. The leaked messages of different users are expected to vary significantly. So it is hard to adopt a methodology to decide which emails are more likely to be known by an adversary. Without loss of generality, we randomly select a subset of emails as the known plaintext documents for each setting.

B. EFFECTIVENESS OF DOCUMENT MAPPING ATTACK

To achieve a high success rate in query recovery process with partial knowledge, we first perform document mapping attacks to find the correspondence of known documents to the encrypted documents (or encrypted document *ID*s) revealed by the SE schemes. On the basis of mapped documents, we perform query recovery attacks to find the mappings between known keywords and search trapdoors (or tokens).

In the document mapping attacks against different leakage models, we assume an adversary has partial knowledge about a victim's documents. There are 30109 emails named with different index numbers in our dataset. In each setting, we choose a proportion of the emails as partially known documents. Table 2 shows we selected the "total" number of documents in different subsets of the dataset, and the "base" represents the number of documents which can be directly mapped between known documents and all encrypted documents according to Definition 1.

TABLE 2. Numbers of documents in different subsets of documents.

Proportion	5%	10%	20%	40%	60%	80%
total	1505	3011	6022	12044	18065	22179
base	9	15	32	54	85	99

The experimental results of the document mapping attacks with the different subsets of all documents are shown in Fig. 4. The x-axis represents the percentage of the known documents, and the y-axis represents the percentage of documents that have been mapped in the known documents.

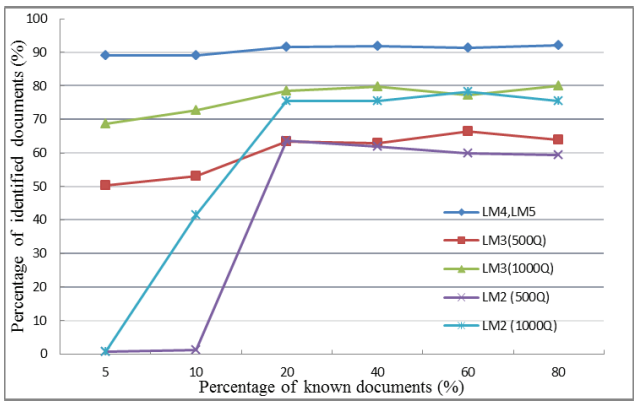

FIGURE 4. Document mapping results in different leakage models. "500Q" means the query keywords randomly selected from the first 2500 most frequent keywords, and "1000Q" means the keywords are selected likewise.

In the attacks against the token-based SE leakage models, the top (blue) line with "$LM4, LM5$" mark represents the percentage of mapped documents in the known documents, without the help of queries. In the attacks against the $LM3$ leakage model, the red curve with "$LM3(500Q)$" mark represents the percentage of mapped documents after 500 queries; the green curve with "$LM3(1000Q)$" mark represents the percentage of mapped documents after 1000 queries. We simulated the query keywords that are initiated by a user, which are invisible to the adversary. In our experiments, they are randomly selected from the top 2500 most frequent keywords in all the dataset. Similarly, in the attacks against the $LM2$ leakage model, the purple curve with "$LM2(500Q)$" mark represents the percentage of mapped documents after 500 queries, and the cyan curve with "$LM2(1000Q)$" mark means the percentage of mapped documents after 1000 queries.

From Fig. 4, we can see that the more information leaked from the leakage modes, the higher the document mapping rate will be. For example, in the $LM3$ and $LM2$ leakage models, the more queries we have, the more information that we can use to map known documents with corresponding encrypted versions. The experimental results show that the attacks work well, even if just a small fraction of messages are known to the adversary. Although only a few documents are mapped in the base mappings, we can map a large proportion of known documents to their encrypted versions. However, the attacks are dependent on that at least one known document is initially mapped in the base mappings. We can resolve this by making an initial guess that we map a known document to one of its candidate encrypted document set, and then run the remainder of the attack algorithms. If the guess is wrong, the remainder procedures will detect inconsistency, and then we will try another candidate. Note that the results of the attacks are dependent on the randomly selected subsets of the document set; however, we have repeated the attacks in the same setting with different selected documents many times, and the results are consistent.

C. INFERENCE ATTACKS ON MAPPED DOCUMENTS

In this section, we present the experimental results of the inference attacks on the proposed leakage models based on the mapped plaintext document set and their encrypted versions.

As shown in Fig. 4, there are different numbers of mapped documents in different leakage models. In the inference attacks against the $LM4$ leakage model that sorted all the tokens in a document, we use Algorithm 2 to guess the keywords of the tokens. We first build the initial base mapping collection between keywords and tokens referring to keyword frequency information, then infer other tokens according to keyword co-occurrence patterns. As shown in Fig. 5, we select varying number of the most frequent keywords, from 1000 to 7000, in different mapped known documents as the keyword universe, and try to find their corresponding tokens in the mapped encrypted versions. The results show that the accuracy rate decreased lightly as more tokens are considered in the same setting. However, when the size of keyword vocabulary is sufficiently large, we can invert a large proportion of corresponding tokens. In summary, we can recover almost all tokens if sufficient encrypted documents are mapped. In contrast, without the proposed document

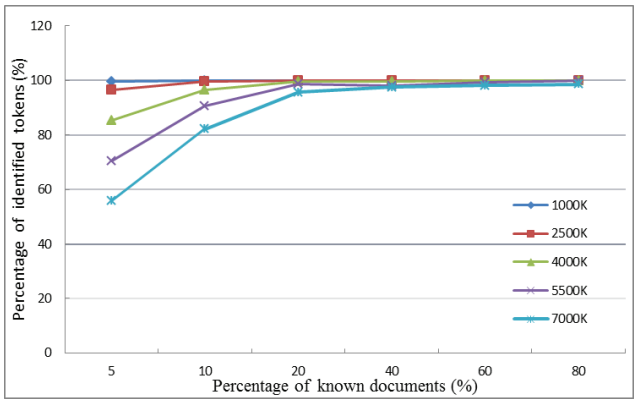

FIGURE 5. Token recovery results of varying number of the most frequent keywords from 1000 to 7000 in different mapped document sets.

mapping attack, the query recovery rates in the settings of different proportions of known documents are all close to zero.

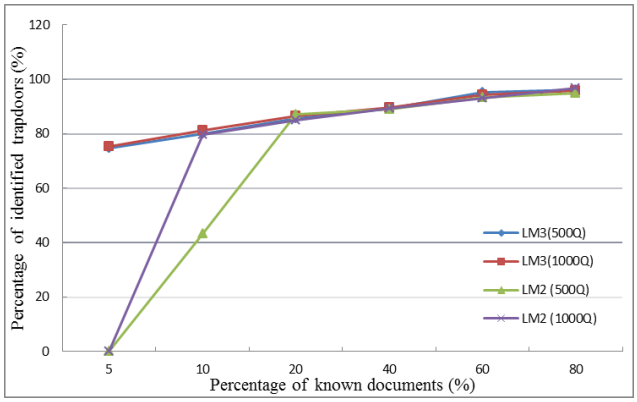

FIGURE 6. Invert trapdoors for the *LM*3 and *LM*2 leakage models. "500*Q*" and "1000*Q*" mean the query keywords selected in the document mapping process.

Fig. 6 shows the accuracy rates of query recovery in the attacks against the *LM*3 and *LM*2 leakage models. Based on the queried trapdoors, we selected the top 2500 most frequent keywords from the mapped known documents as the keyword universe. Given the trapdoors of the selected query keywords and their query results, we try to find their corresponding keywords in the keyword universe. In the query results, we first delete the returned documents (or *ID*s) that do not belong to our mapped document set, then use Algorithm 2 to build the mappings between keywords in the keyword universe and queried trapdoors. The experimental results show that we can invert most queries in the *LM*3 leakage model. However, if we have a small number of mapped documents as in the setting of knowing 5% of all the documents, the query recovery rate is low in the *LM*2 leakage model. The not-mapped query keywords may not be contained or occur infrequently in the mapped document set, or cannot be inferred according to the information of keyword frequency and keyword co-occurrence frequency. On the other hand, the success rates of query recovery attacks in the settings of different proportional known documents are all close to zero without the help of the document mapping attack, as IKK [15] and CGPR [17] did.

VI. RELATED WORK

IKK [15] attack first constructed a trapdoor co-occurrence matrix C_{td}, in which each member represents the number of documents that every two trapdoors both hit. If the adversary has the complete knowledge of the indexed documents, a keyword co-occurrence matrix C_k can also be constructed. Then it used simulated annealing to find the best match of C_{td} to C_k, so as to invert the query trapdoors. CGPR [17] presented a simpler count attack scheme without involving optimization problems. It counts keyword occurrence patterns and keyword co-occurrence patterns from known documents, and trapdoor occurrence patterns and trapdoor co-occurrence patterns from query results, to build mappings between keywords and trapdoors. The Shadow Nemesis [18] launched inference attacks on token based SE. It creates two co-occurrence matrix graphs G and H, based on known information and target documents, respectively, then uses the well-known WGM optimization problem to find the meanings of tokens. However, all the above passive attacks require almost complete document knowledge to achieve a high recovery accuracy for a certain size of keyword vocabulary. Recent works [27], [28] built attacks against a multi-user SE scheme [21] using leaked information. In [27] an adversary needs to collude with some users to invert a query, and [28] mainly leverages implementation or design issues of [21], or requires an active attacker.

CGPR [17] and ZKP [29] used active attacks to guess the information of queries. An adversary needs to inject multiple known documents into a target document set; then the adversary can observe the injected documents contained in query results to infer the queries. The number of injected documents is dependent on the keyword vocabulary size.

CGPR [17] proposed the attacks with partial knowledge, and designed active schemes. ZKP [29] also designed advanced active attacks with partial knowledge. Giraud, *et al.* [20] presented effective passive attacks with partial knowledge against token-based SE. However, it is not applied to index-based SE schemes, while our attacks do.

VII. CONCLUSION

In this paper, we defined five leakage models of common SE schemes, then presented our document mapping attacks and query recovery attacks accordingly. We showed that our inference attacks are effective even when only a small fraction of documents is known to an adversary. Our document mapping attacks rely on the number of keywords information in a document, so padding a document with randomly selected irrelevant keywords can defend the attacks to a certain extent. However, we can make an initial guess to map some encrypted documents to corresponding plaintext documents, and then run the remainder of the algorithms. This will be studied in our future work. In addition, some

advanced SE functions may leak extra information, which can be used to construct simple yet powerful attacks. For example, a dynamic SE scheme [30] leaks the structure of the added documents in update process, including the keyword number information of a document and document similarity information.

REFERENCES

[1] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious RAMs," *J. ACM*, vol. 43, no. 3, pp. 431–473, 1996.
[2] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy (S&P)*, May 2000, pp. 44–55.
[3] E.-J. Goh, "Secure indexes," in *Proc. Int. Assoc. Cryptol. Res. Cryptol. ePrint Arch.*, 2003, p. 216.
[4] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *J. Comput. Secur.*, vol. 19, no. 5, pp. 895–934, 2011.
[5] D. Boneh *et al.*, "Public key encryption with keyword search," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, Berlin, Germany, 2004, pp. 506–522.
[6] W. He, D. Akhawe, S. Jain, E. Shi, and D. Song, "ShadowCrypt: Encrypted Web applications for everyone," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 1028–1039.
[7] B. Lau *et al.*, "Mimesis aegis: A mimicry privacy shield—A system's approach to data privacy on public cloud," in *Proc. 23rd USENIX Secur. Symp. (USENIX Security)*. 2014, pp. 33–48.
[8] G. S. Poh, J.-J. Chin, W.-C. Yau, K.-K. R. Choo, and M. S. Mohamad, "Searchable Symmetric Encryption: Designs and Challenges," *ACM Comput. Surv.*, vol. 50, no. 3, pp. 40:1–40:37, 2017.
[9] M. Ma *et al.*, "Certificateless searchable public key encryption scheme for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 2, pp. 759–767, 2018. [Online]. Available: http://dx.doi.org/10.1109/TII.2017.2703922
[10] L. Wu, B. Chen, K.-K. R. Choo, and D. He, "Efficient and secure searchable encryption protocol for cloud-based Internet of Things," *J. Parallel Distrib. Comput.*, vol. 111, pp. 152–161, Jan. 2008.
[11] C. Bösch, P. Hartel, W. Jonker, and A. Peter, "A survey of provably secure searchable encryption," *ACM Comput. Surv.*, vol. 47, no. 2, 2015, Art. no. 18.
[12] *Skyhigh Security Cloud*. Accessed: 2017. [Online]. Available: https://www.skyhighnetworks.com/
[13] *The CipherCloud Platform-Actionable CASB*. Accessed: 2017. [Online]. Available: https://www.ciphercloud.com/
[14] *Virtru-Protection That Travels With Your Data*. Accessed: 2017. [Online]. Available: http://www.virtru.com
[15] M. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation," in *Proc. NDSS*, vol. 20. 2012, p. 12.
[16] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–679, May 1983.
[17] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 668–679.
[18] D. Pouliot and C. V. Wright, "The shadow nemesis: inference attacks on efficiently deployable, efficiently searchable encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1341–1352.
[19] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 18, no. 3, pp. 265–298, 2004.
[20] M. Giraud *et al.*, "Practical passive leakage-abuse attacks against symmetric searchable encryption," in *Proc. IACR Cryptol. ePrint Arch.*, 2017, pp. 1–16.
[21] R. A. Popa *et al.*, "Building Web applications on top of encrypted data using Mylar," in *Proc. 11th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2014, pp. 157–172.
[22] R. A. Popa *et al.*, "CryptDB: Protecting confidentiality with encrypted query processing," in *Proc. 23rd ACM Symp. Oper. Syst. Principles*, 2011, pp. 85–100.
[23] D. Cash *et al.*, "Dynamic searchable encryption in very-large databases: Data structures and implementation," in *Proc. NDSS*, vol. 14. 2014, pp. 23–26.
[24] *Enron Email Dataset*. Accessed: May 13, 2015. [Online]. Available: https://www.cs.cmu.edu/~./enron/
[25] M. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.
[26] *Common-English-Words*. Accessed: 2017. [Online]. Available: http://www.textfixer.com/tutorials/common-english-words.txt/
[27] C. Van Rompay, R. Molva, and M. Önen, "A leakage-abuse attack against multi-user searchable encryption," in *Proc. Privacy Enhancing Technol.*, vol. 3. 2017, pp. 164–174.
[28] P. Grubbs, R. McPherson, M. Naveed, T. Ristenpart, and V. Shmatikov, "Breaking Web applications built on top of encrypted data," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1353–1364.
[29] Y. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," in *Proc. USENIX Secur. Symp.*, 2016, pp. 707–720.
[30] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 965–976.

GUOFENG WANG was born in Jining, Shandong, China, in 1988. He received the M.S. degree in computer science and technology from the Beijing University of Posts and Telecommunications, Beijing, in 2014, where he is currently pursuing the Ph.D. degree in information security.

His research interests include the cloud computing, searchable encryption, and data security.

CHUANYI LIU was born in Leshan, Sichuan, China, in 1982. He received the M.S. and Ph.D. degrees in computer science and technology from Tsinghua University, Beijing, in 2010. From 2007 to 2008, he was a Visiting Scholar in computer and information science from the University of Minnesota.

His research interests include the cloud computing and cloud security, mass storage systems, data protection, and data security.

YINGFEI DONG received the B.S and M.S. degrees in computer science from the Harbin Institute of Technology, in 1989 and 1992, respectively, the Ph.D. degree in engineering from Tsinghua University in 1995, and the Ph.D. degree in computer and information science from the University of Minnesota in 2003. He joined the Department of Electrical and Computer Engineering, College of Engineering, University of Hawaii, as an Assistant Professor, in 2003.

His main research interests are in the areas of computer networking, networking security, multimedia content delivery, Internet services, distributed systems, advanced computer architecture, and user privacy.

KIM-KWANG RAYMOND CHOO (SM'15) received the Ph.D. degree in information security from the Queensland University of Technology, Australia, in 2006. He currently holds the Cloud Technology Endowed Professorship with The University of Texas at San Antonio. He is also a Fellow of the Australian Computer Society. He serves on the Editorial Board of *Computers & Electrical Engineering*, *Cluster Computing*, *Digital Investigation*, the IEEE Access, the IEEE Cloud Computing, the IEEE *Communications Magazine*, *Future Generation Computer Systems*, the *Journal of Network and Computer Applications*, *PLOS One*, and *Soft Computing*. He also serves as the Special Issue Guest Editor of the IEEE Cloud Computing in 2015, *ACM Transactions on Internet Technology* in 2016, *Digital Investigation* in 2016, the IEEE Network in 2016, *Pervasive and Mobile Computing*in 2016, *Future Generation Computer Systems* in 2016 and 2018, *ACM Transactions on Embedded Computing Systems* in 2017, *Computers & Electrical Engineering* in 2017, the IEEE Transactions on Cloud Computing in 2017, the IEEE Transactions on Dependable and Secure Computing in 2017, the *Journal of Computer and System Sciences* in 2017, *Multimedia Tools and Applications* in 2017, *Personal and Ubiquitous Computing* in 2017, and *Wireless Personal Communications* in 2017. In 2016, he was named the Cybersecurity Educator of the Year-APAC (Cybersecurity Excellence Awards are produced in cooperation with the Information Security Community on LinkedIn) and in 2015 he and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen-Nuremberg. He was a recipient of ESORICS 2015 Best Paper Award, 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, Fulbright Scholarship in 2009, 2008 Australia Day Achievement Medallion, and British Computer Society's Wilkes Award in 2008.

PEIYI HAN was born in Xiaoyi, Shanxi, China, in 1992. He received the M.S. degree in computer science and technology from the Beijing University of Posts and Telecommunications, Beijing, in 2016, where he is currently pursuing the Ph.D. degree in information security.

His research interests include the cloud computing, searchable encryption, and data security.

HEZHONG PAN was born in Benxi, Liaoning, China, in 1991. He received the M.S. degree in computer science and technology from the Beijing University of Posts and Telecommunications, Beijing, in 2015, where he is currently pursuing the Ph.D. degree in information security.

His research interests include the cloud computing, searchable encryption, and data security.

BINXING FANG was born in Wanning, Jiangxi, China, in 1960. He received the M.S. degree in computer science and technology from Tsinghua University, Beijing, in 1984, and the Ph.D. degree in computer science and technology from the Harbin Institute of Technology, Harbin, in 1989. He is currently a member of the Chinese Academy of Engineering.

His current research interests include the computer network, information and network security, and content security.