Authenticating Top-k Results of Secure Multi-keyword Search in Cloud Computing

Xiaojun Xiao[1], Yaping Lin[1(✉)], Wei Zhang[1], Xin Yao[1], and Qi Gu[2]

[1] College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China
{S1324W1015,yplin,zhangweidoc,xinyao}@hnu.edu.cn
[2] Google Inc., California, USA
qig@google.com

Abstract. Cloud computing brings abundant benefits to our lives nowadays, including easy data access, flexible management, and cost saving. However, due to the concern for privacy, most of us are reluctant to use it. To protect privacy while making full use of cloud data, secure keyword search is proposed and attracts many researchers' interests. However, all of the previous researches are based on a weak threat model, i.e., they all assume the cloud to be "curious but honest". Different from the previous works, in this paper, we consider a more challenging model where the cloud server would probably be compromised. To achieve a privacy preserving and personalized multi-keyword search, we first formulate different users' preference with a preference vector, and then adopt the secure k nearest neighbor (KNN) technique to find the most relevant files corresponding to the personalized search request. To verify the dynamic top-k search results, we design a novel Multi-Attribute Authentication Tree (MAAT). In particular, we propose an optimization scheme to reduce the size of verification objects so that the communication cost between the cloud and data users is tunable. Finally, by doing extensive experiments, we confirm that our proposed schemes can work efficiently.

Keywords: Cloud computing · Privacy preserving · Personalized multi-keyword search · Multi-Attribute Authentication Tree (MAAT) · Optimization

1 Introduction

Cloud computing brings abundant benefits to our lives nowadays, including easy data access, flexible management, and cost saving. It becomes critically important for data owners to outsource their data to the public cloud server while allowing data users to retrieve them [1].

However, most of us are reluctant to use it. One of the most important reasons is the concern for privacy. Data encryption would be an alternative way to reduce the data leakage. However, data encryption obviously prevents the plain-text based keyword search techniques. A trivial solution is downloading

B. Thuraisingham et al. (Eds.): SecureComm 2015, LNICST 164, pp. 212–229, 2015.
DOI: 10.1007/978-3-319-28865-9_12

all the encrypted data and decrypting them locally. But this is also impractical because of the huge amount of communication cost. Therefore devising a secure keyword search protocol is imperative.

Secure keyword search over encrypted cloud data has attracted several researchers' interests recently. Song et al. [2] first propose the notion of searchable encryption, which is further developed by [3], [7]. However, extending these researches to large scale cloud data will bring heavy computation and storage overhead. Wang et al. [8] first consider the secure keyword search over encrypted cloud data, which is followed by [9], [10], [11], [12]. These researchers not only enrich the search capabilities, but also reduce the computation and storage cost.

However, all these schemes are based on the ideal assumption that the cloud server is "curious but honest". Unfortunately, in practical applications, the cloud server may behave dishonestly with a lot of motivations, which mainly include:

- The cloud server may return forged search results. For example, an advertisement may be ranked higher than his competitors since the cloud server provider may earn profits from that advertising company.
- The cloud server may return incomplete search results in peak hours to avoid suffering from performance bottlenecks.

Therefore, enabling authorized data users to authenticate the search results would be significant. Additionally, a user-friendly system should enable data users to achieve a personalized multi-keyword search. To verify the search results, conventional solutions (including linked signature chaining [13] and the Merkle hash tree [14]) need the data owners to pre-know the order of search results. However, to enable personalized keyword search, search results have to be computed on the cloud server according to different data users' preferences, where data owners cannot pre-know the order of search results. An example is illustrated in Fig. 1. As we can see, data owner has four files (F_1, F_2, F_3, F_4), each file is attached with a file vector (each attribute in a file vector is a relevance score between a keyword and a file). Given different search vectors (Q_1, Q_2, Q_3, Q_4) (the order of search results is ranked by the inner product of the search vector and the file vectors), the order of search results are totally different.

In this paper, we consider a more challenging model where the cloud server would probably be compromised. A compromised cloud server would not only reveal sensitive data but also return forged or incomplete search results. To achieve a privacy-preserving personalized multi-keyword search, we first formulate different users' preference into a preference vector, and then adopt the secure k nearest neighbor (KNN) technique to find the most relevant files corresponding to the personalized search request. To preserve the relevance scores between keywords and files, we use an order and privacy preserving function. Additionally, we propose a novel Multi-Attribute Authentication Tree (MAAT) to authenticate the dynamic top-k search results. In particular, to reduce the size of verification objects, we propose an optimization scheme so that the communication cost between the cloud and data users is tunable. Finally, we conduct extensive experiments on real-world datasets which confirms that our proposed schemes work efficiently.

File Vectors	Search Vectors	Order of Results
F_1: (0.82,0.63,0.28)	Q_1: (0.1, 0.2, 0.7)	F_4, F_3, F_2, F_1
F_2: (0.92,0.54,0.43)	Q_2: (0.2, 0.3, 0.5)	F_4, F_2, F_3, F_1
F_3: (0.52,0.45,0.62)	Q_3: (0.5, 0.4, 0.1)	F_2, F_1, F_3, F_4
F_4: (0.25,0.62,0.68)	Q_4: (0.4, 0.6, 0.0)	F_1, F_2, F_3, F_4

Fig. 1. An example of dynamic order of search results corresponding to different search vectors

The main contributions of this paper are as follows:

- We consider a more challenging threat model where the cloud server would behave dishonestly. Based on this model, we solve the privacy preserving personalized multi-keyword search and dynamic top-k search results authentication.
- We propose a novel Multi-Attribute Authentication Tree (MAAT) to authenticate the dynamic top-k search results.
- We propose an optimization scheme to reduce the size of verification objects so that the communication cost between the cloud and data users is tunable.
- We analyze security properties and conduct extensive performance experiments for our proposed schemes.

The rest of this paper is organized as follows. Section 2 reviews the related works. Section 3 formulates the problem and introduces notations used in later discussions. Section 4 describes the secure search schemes and Section 5 introduces the authentication schemes. In Section 6, we introduce how to optimize the parameters. In Section 7 and 8, we presents security analysis and performance evaluation of our proposed schemes respectively. In Section 9, we conclude the paper.

2 Related Work

2.1 Traditional Searchable Encryption

Encrypted data search has been studied extensively in the literature. Song et al. [2] first defined the conception of searching on encrypted, proposed the cryptographic schemes for the problem of searching on encrypted data, and proved the security of their scheme. Goh et al. [3] defined a secure index to accelerate the search operation. Chang et al. [7] proposed a privacy preserving keyword search scheme, which not only enables data user to perform a keyword search over encrypted data, but also prevent from leaking the data privacy. The researches [4], [5], [6] further enhanced the search capabilities. But most of these works only support the search of single or boolean keyword, extending these techniques to large scale cloud data will bring heavy computation and storage overhead.

2.2 Secure Keyword Search in Cloud Computing

Secure keyword search in cloud computing has attracted many interests. Wang et al. [8] first defined the problem of secure ranked keyword search over encrypted cloud data. Cao et al. [9], Xu et al. [12] and Wen et al. [15] proposed to address the privacy preserving multi-keyword search over encrypted cloud data. To accelerate the search process, Hore et al. [11] proposed to adopt a set of colors to encode the presence of the keywords and create a search index. To enrich search functionality, Li et al. [10] proposed fuzzy keyword search over encrypted cloud data, respectively. To support multiple data owners to search over large scale cloud data, Sun et al. [16] proposed secure attribute-based keyword search schemes. Zhang et al. proposed to ensure secure ranked multi-keyword search to support multiple data owners in [17], [18], [19], and achieve secure distributed keyword search in geo-distributed clouds in [20], respectively.

However, all these schemes assume the cloud server to be "curious but honest". Different from these schemes, in this paper, we assume the cloud server would be compromised, under this assumption, we propose to securely authenticate the dynamic top-k search results.

2.3 Authenticating the Search Results

Methods used in authentication can be classified into two categories: the linked signature chaining, and the Merkle hash tree.

The linked signature chaining schemes [13], [21], require to pre-know the order of search result, so that the data owner can obtain an ordered link, and sign for the consecutive data in the link, which forms the linked signature chaining. Consequently, any data forging or deletion will be easily discovered once the signature chaining is incomplete. However, as illustrated in [22], the linked signature chaining will lead to very high computational cost, storage overhead, and user-side verification cost.

The Merkle hash tree proposed in [14], [23], [24] is proposed to verify the integrity of a very large data set. The merkle hash tree also require to pre-know the order of search results. The data owner constructs the merkle hash tree and signs for the root. Data users re-construct the merkle hash tree, and compare the computed root with the returned root. Therefore, any data forging or deletion will lead to the inconsistency of the comparison. However, as illustrated in Fig. 1, for the personalized keyword search, data owners cannot know the order of search results in advance, we cannot use the existed authentication method here. In this paper, we propose to construct a novel Multi-Attribute Authentication Tree (MAAT) to authenticate the dynamic top-k search results.

3 Problem Formulation

3.1 System Model

There are three entities involved in our system model, as illustrated in Fig. 2, they are data owner, cloud server and data users. The data owner has a collection of files $\mathcal{F}$. To enable search operation on these files which will be encrypted,

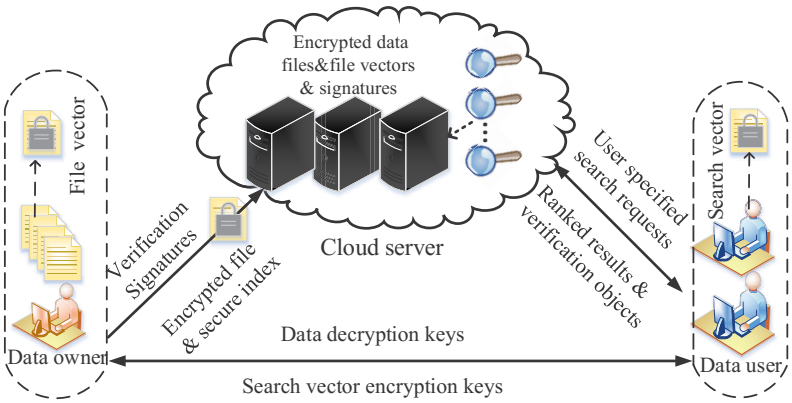


Fig. 2. Architecture of secure keyword search in cloud computing

the data owner performs some operations in advance which includes extracting a keyword set $\mathcal{W}$ from $\mathcal{F}$, computing relevance scores between keywords and files, constructing and signing a multi-attribute authentication tree. Then the data owner outsources all the encrypted data files, file vectors and signatures to the cloud server. Once an authorized data user wants to perform a secure keyword search over these encrypted files based on his preference, he first generates his trapdoor $\widetilde{Q}$ (encrypted query vector) and submits it to the cloud server. Upon receiving the trapdoor $\widetilde{Q}$, the cloud server first searches over the encrypted file vectors stored on it, then it returns the top-k relevant data files and corresponding verification objects. The authorized data user further verifies the integrity of returned search results. If the search results pass the verification, data user decrypts and obtains satisfied data files. Otherwise the search results are considered as contaminated and abandoned.

3.2 Threat Model

In our threat model, both data owner and authorized data users are trusted, however, different from previous works [8], [9], [12], the cloud server is not trusted and would be compromised, which is more challenging and takes a firm step towards practical application. Specifically, the cloud server not only aims at revealing the contents of encrypted files, keywords and relevance scores, but also tends to return forged or incomplete data. Note that how to authorize a data user is out of the scope of this paper, an outstanding example can be found in [25].

3.3 Design Goals

Our system design should simultaneously satisfy security and performance goals illustrated as follows:

- **Ranked multi-keyword search:** The proposed scheme should enable personalized and ranked multi-keyword search. Specifically, a data user constructs the personalized search vector, and submits its encryption to the cloud server. The cloud server returns the most relevant top-k results based on the personalized search vector.
- **Privacy preserving:** The proposed scheme should prevent the cloud server from learning the actual data of encrypted files, indexes, and signatures.
- **Authenticating the integrity of result:** When the cloud server behaves dishonestly, i.e., cloud server returns forged or incomplete search results, data user can discover the misbehavior.
- **Efficiency:** All the above goals should be achieved with low computation and communication overhead.

3.4 Notations

- $\mathcal{F}$: the plaintext file collection.
- $\mathcal{C}$: the ciphertext file collection of $\mathcal{F}$.
- $\mathcal{W}$: the keyword dictionary.
- P: each file vector P_i corresponds to the file F_i.
- $\widetilde{P}$: the encrypted file vectors of P.
- $\widehat{P}$: the encoded file vectors of P.
- Q: the search vectors issued by data users.
- $\widetilde{Q}$: the encrypted search vectors of Q.
- H: a one-way hash function.

4 Privacy-Preserving and User-Specified Ranked Multi-keyword Search

Since different data users may have different personal preferences. Additionally, huge amount of files are stored on cloud servers, we cannot simply return indifferential files to data users for two reasons. First, returning all satisfied files would cause tremendous communication overhead for the whole system. Second, data users would only concern top-k relevant files corresponding to their queries. So our scheme should also achieve ranked multi-keyword search.

Motivated by the secure k-nearest neighbor scheme proposed in [9] and [26], we use the inner product of a file vector P and a search vector Q, i.e., $P \cdot Q$, to quantitatively evaluate the similarity of a file and a query. A file corresponding to a higher value of the inner product will have higher probability to be returned. The file vector is assembled according to the following principle: the ith data item in the jth file vector is the relevance score between the ith keyword in the keyword set $\mathcal{W}$ and the jth file in the file set. Meanwhile, the search vector is formalized according to user's preference. For example, data user wants to search the ith and i'th keyword in the keyword set $\mathcal{W}$, since he thinks the ith keyword is more important than the i'th keyword, he gives each keyword a weight, say

0.8 and 0.2. Then the ith data item in the search vector is assembled with 0.8, and the i'th data item is assembled with 0.2.

Given a file vector P_i and a search vector Q_j, we use the encryption method proposed in [9] to encrypt them. Specifically, the data owner uses three secret keys to encrypt them, i.e., a vector split indicator S, two invertible matrixes M_1 and M_2. The encryption process is divided into two phases. First, data owner splits P_i into $P_{i'}, P_{i''}$ and Q_j into $Q_{j'}, Q_{j''}$ as follows, if the kth bit of S is 0, then $P_{i'}$ and $P_{i''}$ are set the same as P_i, while $Q_{j'}$ and $Q_{j''}$ are randomly set so that their sum are equal to Q_j. If the kth bit of S is 1, then $P_{i'}$ and $P_{i''}$ are randomly set so that their sum are equal to P_i while $Q_{j'}$ and $Q_{j''}$ are set the same as Q_j. Second, data owner encrypts $\{P_{i'}, P_{i''}\}$ as $\widetilde{P}_i = \{M_1^T \cdot P_{i'}, M_2^T \cdot P_{i''}\}$, and $\{Q_{j'}, Q_{j''}\}$ as $\widetilde{Q}_j = \{M_1^{-1} \cdot Q_{j'}, M_2^{-1} \cdot Q_{j''}\}$. Therefore,

$$\widetilde{P}_i \cdot \widetilde{Q}_j = \{M_1^T \cdot P_{i'}, M_2^T \cdot P_{i''}\} \cdot \{M_1^{-1} \cdot Q_{j'}, M_2^{-1} \cdot Q_{j''}\} = P_i \cdot Q_j \quad (1)$$

Finally, the cloud server returns the top-k relevant search results to the data user according to the rank of $\widetilde{P}_i \cdot \widetilde{Q}_j$. For more rigorous security requirement, we can use the techniques proposed in [9].

5 Dynamic Top-k Results Authentication

In the aforementioned section, we introduce how to achieve privacy preserving and personalized ranked multi-keyword search in cloud computing. When the cloud server behaves dishonestly, we need to verify whether there are false search results corresponding to different users' preference. In this section, we first introduce the privacy preserving function [27], which will be used to protect the privacy of relevance scores between keywords and files. Then we elaborate on how to construct our proposed Multi-Attribute Authentication Tree (MAAT). Finally, we describe how to authenticate the integrity of the dynamic top-k search results with the proposed MAAT.

5.1 Privacy Preserving Function

The privacy preserving function $F(x)$ is composed of a data processing part $f(x)$ and a disturbing part r_f. The data processing part preserves the order of x while the disturbing part r_f prevents cloud server from revealing $F(x)$. Therefore, $F(x) = f(x) + r_f$ and the $f(x)$ is defined as follows:

$$f(x) = \sum_{0 \le j \le \tau} A_j \cdot m^2(x, j) \quad (2)$$

where τ denotes the degree of $f(x)$ and A_j denotes the coefficients of $m^2(x, j)$.

The $m(x, j)$ is defined as follows: 1) $j = 0$, $m(x, j) = 1$; 2) $j = 1$, $m(x, j) = x + 1$; 3) $j > 1$, $m(x, j) = \lfloor (m(x, j-1) + \alpha) \cdot (1 + \lambda \cdot x) \rfloor$, where α and λ are two constant numbers.

$\forall x_1 \geq x_2$, where x_1 and x_2 are positive integer numbers,

$$\begin{aligned} & f(x_1) - f(x_2) \\ &= \sum_{0 \leq j \leq \tau} A_j \cdot \left(m^2(x_1, j) - m^2(x_2, j)\right) \\ &= \sum_{0 \leq j \leq \tau} A_j \cdot (m(x_1, j) + m(x_2, j)) \\ &\quad \cdot (m(x_1, j) - m(x_2, j)) \\ &\geq \sum_{0 \leq j \leq \tau} A_j \cdot \lambda \cdot (x_1 - x_2) \cdot m(x_2, j-1) \\ &\quad \cdot (m(x_1, j) + m(x_2, j)) \\ &\geq \lambda \cdot \alpha^2 \cdot \sum_{0 \leq j \leq \tau} A_j \end{aligned} \tag{3}$$

Obviously, $\forall x_1 > x_2$, we have $f(x_1) > f(x_2)$. Let ϵ be a system parameter such that $2^\epsilon \leq \lambda \cdot \alpha^2 \cdot \sum_{0 \leq j \leq \tau} A_j$, then the disturbing part r_f of $F(x)$ is set to $2^\epsilon - 1$.

5.2 Multi-Attribute Authentication Tree

Definition 1. *If each element (relevance score) in a file vector P_i is not smaller than that in P_j, i.e, $\forall k \in [1, n], P_{i,k} \geq P_{j,k}$, and at least one element in P_i is greater than that in P_j, i.e., $\exists k \in [1, n], P_{i,k} > P_{j,k}$. Then we define P_i dominates P_j, and P_j is dominated by P_i.*

Definition 2. *If the first k ($k = 0, 1, \cdots, n-1$) elements in P_i are equal to that in P_j (i.e., $P_{i,0} = P_{j,0}, P_{i,1} = P_{j,1}, \cdots, P_{i,k} = P_{j,k}$), for the (k+1)th element, if $P_{i,k+1} > P_{j,k+1}$, then we define $P_i > P_j$.*

Algorithm 1 illustrates the process of constructing MAAT, which is composed of two phases, i.e., generating the framework of MAAT, and aggregating the hash value of MAAT. The first phase is divided into three steps described as follows: first of all, sorting all encoded vectors in descending order according to the comparison method defined in Definition 2. Second, initializing the root of MAAT, i.e., the value of each item in the vector is a pre-defined maximum number. Finally, inserting the sorted vectors into the MAAT one by one. Specifically, given $\widehat{P}_i$, the algorithm inserts $\widehat{P}_i$ as follows: each time the algorithm traverses from the root node, if the visited node $\widehat{P}_j$ dominates $\widehat{P}_i$, then the algorithm sets $\widehat{P}_i$'s parent node to be $\widehat{P}_j$, if $\widehat{P}_j$ has no child node, the algorithm finishes inserting $\widehat{P}_i$. If $\widehat{P}_j$ has child node, the algorithm visits $\widehat{P}_i$'s whole child nodes, if no child nodes of $\widehat{P}_j$ dominate $\widehat{P}_i$, the algorithm finishes inserting $\widehat{P}_i$, otherwise, if $\widehat{P}_j$'s child node $\widehat{P}_c$ dominates $\widehat{P}_i$, the algorithm sets $\widehat{P}_i$'s parent node to be $\widehat{P}_c$, and conducts the insertion recursively. The MAAT framework is constructed when all encoded vectors are inserted. The second phase is aggregating the hash value of MAAT from the leaf nodes to the root node. Specifically, for a leaf node, the algorithm only computes its hash value. For a non-leaf node, the algorithm computes its hash value, conducts exclusive or operation on the hash value of all its children, and combines them as its hash values.

Algorithm 1.. The MAAT construction algorithm

Input:

Encoded file vectors $\widehat{P}$ and ciphertext of file vectors $\widetilde{P}$

Output:

MAAT

Step1: generate the framework of MAAT

Sort $\widehat{P}$ on their first encoded attribute.
root=$\{\widehat{P}_{max}\}$ //initialize the MAAT
for i=1 to m **do**
  $\widehat{P}_t = root$
  if $\widehat{P}_t$ has no child **then**
    add $\widehat{P}_i$ to the child set of $\widehat{P}_t$
    continue
  else
    for each child $\widehat{P}_j$ of $\widehat{P}_t$ **do**
      if $\widehat{P}_i$ is dominated by $\widehat{P}_j$ **then**
        $\widehat{P}_t = \widehat{P}_j$
        goto step 5
  add $\widehat{P}_i$ to the child set of $\widehat{P}_t$
for each node $\widehat{P}_i$ in the MAAT **do**
  add $\widetilde{P}_i$ to $\widehat{P}_i$

Step2: aggregate the hash value of MAAT

for each node in the MAAT **do**
  if node $\widehat{P}_i$ is a leaf node **then**
    $\widehat{P}_i$ submits $hash(\widehat{P}_i||\widetilde{P}_i)$ to its parent node
  else
    if $\widehat{P}_i$ has only one child **then**
      $\widehat{P}_i$ set the received value as H_i and submits $hash(\widehat{P}_i||\widetilde{P}_i||H_i)$ to its parent node.
    else
      $\widehat{P}_i$ first aggregates the hash value to H_i by doing XOR operation on received data from different child nodes and submits $hash(\widehat{P}_i||\widetilde{P}_i||H_i)$ to its parent node
return root

Now we give an example of constructing MAAT in Fig. 3, there are 10 encoded vectors, each vector includes four attributes. First, the algorithm sorts the 10 encoded vectors and gets $\{\widehat{P}_2, \widehat{P}_1, \widehat{P}_5, \widehat{P}_4, \widehat{P}_9, \widehat{P}_3, \widehat{P}_8, \widehat{P}_7, \widehat{P}_6\}$. Then the algorithm initializes the root of MAAT to be $\widehat{P}_{max}$, where the value of each attribute in $\widehat{P}_{max}$ is set to be maximal. Further, $\{\widehat{P}_2, \widehat{P}_1, \widehat{P}_5, \widehat{P}_4, \widehat{P}_9, \widehat{P}_3, \widehat{P}_8, \widehat{P}_7, \widehat{P}_6\}$ are inserted into MAAT subsequently. Finally, the algorithm computes the hash value. Specifically, $H_3 = hash(\widehat{P}_8||\widetilde{P}_8)$, $H_1 = hash(\widehat{P}_3||\widetilde{P}_3||H_3)$, and $H_2 = hash(\widehat{P}_1||\widetilde{P}_1||H_1) \oplus hash(\widehat{P}_9||\widetilde{P}_9)$.

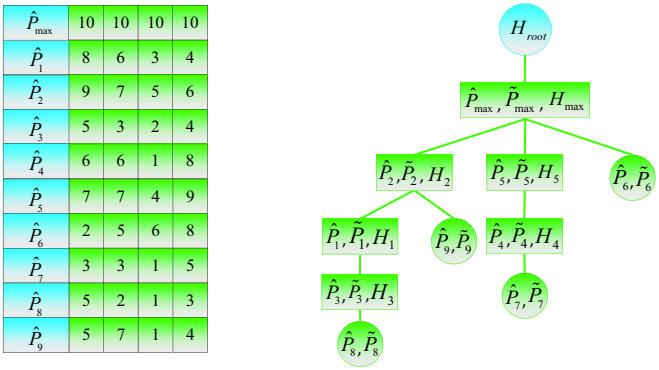


Fig. 3. An example of constructing MAAT

5.3 Authenticating Integrity of the Dynamic Top-k Search Results

In this subsection, we will introduce how to verify the integrity of ranked top-k search results based on MAAT. To enable the authorized data users to verify the dynamic top-k search results, the data owner processes his data as follows: first, the data owner extracts the file vectors from all of his files. Then, he encodes these file vectors with the privacy and order preserving function. Further, the data owner constructs the MAAT with the encoded file vectors. Finally, the data owner outsources encrypted vectors, encoded vectors, $sign(H_{root})$ (signature of the root of MAAT), and encrypted files to the cloud server. Once the cloud server finds the encrypted search results $\{C_{i1}, C_{i2}, \cdots, C_{im}\}$, it further prepares the authentication data with the following steps: first of all, the cloud server adds the nodes corresponding to $\{C_{i1}, C_{i2}, \cdots, C_{im}\}$ in MAAT to a node set S. Then it adds all the ancestors of these nodes to S. Further, it finds all the sibling nodes of nodes in S and adds them to S. Finally, together with $sign(H_{root})$, the encoded vector and encrypted vector corresponding to nodes in S are returned as authentication data. For example, given the search results $\{F_1, F_2\}$, the corresponding authentication data would be $\{\widehat{P}_1, \widehat{P}_2, \widehat{P}_5, \widehat{P}_6, \widehat{P}_9, \widetilde{P}_1, \widetilde{P}_2, \widetilde{P}_5, \widetilde{P}_6, \widetilde{P}_9, H_1, H_5, sign(H_{root})\}$. When the data user attains the returned result and authentication data, he verifies the results with the following steps: first of all, he reconstructs the MAAT with the corresponding encoded vectors. Then, he checks whether the computed root of MAAT is equal to H_{root}. If they are not equal, the results are contaminated and discarded. Finally, the data user checks whether the results are the most relative top-k files with the help of the decrypted file vectors. If any false results are detected during the process, the results are regarded as false and discarded.

6 MAAT Optimization

MAAT can achieve privacy preserving and dynamic top-k search results verification. However, when the keyword set is large, many encoded file vectors

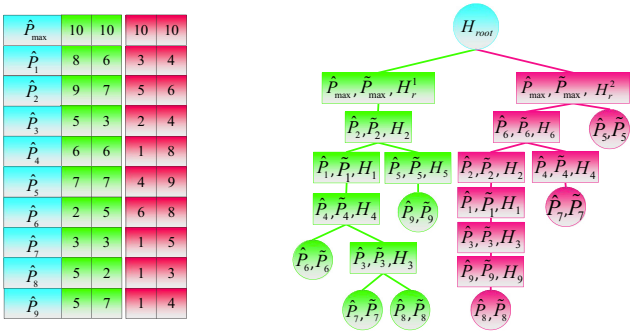


Fig. 4. An example of constructing optimized MAAT

would not be dominated by others. Consequently, the cloud server has to return numerous verification data for search results verification, which is obviously inapplicable. In the following subsection, we first introduce the scheme of optimizing the MAAT. Then we analyze the trade-off value between privacy and communication cost.

6.1 Optimizing Method

As we know, in real applications, when we want to perform a search, we often issue a few keywords. Therefore we can specify the cloud server to return verification data for these keywords. This can reduce a large number of verification data. Though telling cloud server which keywords we want to verify will bring the threat of privacy revealing, we can issue some dummy keywords to obfuscate the cloud server. The optimizing process is described as follows: first, we split each encoded vector into T encoded sub-vectors. Then we use these encoded sub-vectors to construct sub-MAAT. Finally, we combine the T sub-MAATs and get the optimized MAAT. Fig. 4 shows the optimized MAAT of the one in Fig. 3. As we can see, when $\{F_1, F_2\}$ are the search results, and the data user specifies to verify the first two attributes, the corresponding authentication data would be $\{\widehat{P}_1, \widehat{P}_2, \widehat{P}_5, \widetilde{P}_1, \widetilde{P}_2, \widetilde{P}_5, H_1, H_5, H_r^2, sign(H_{root})\}$. As we can see, compared with the former verification cost, the optimized one will obviously reduce verification cost.

6.2 Trade-off Between Privacy and Communication Cost

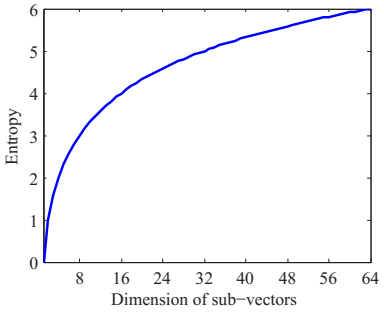
From the above discussion, when the number of items in each file vector is very large, the privacy is well preserved, while the communication cost spent on verification would be very large. On the other hand, when we split the file vector into very small sub-vectors, i.e., the number of items in each sub-vector is small, the communication cost would be reduced, while the privacy preservation would be weakened. Therefore, we need to find a trade-off between privacy and communication cost.

Recall that, to obfuscate the cloud server of which keywords are actually verified, we propose to add some dummy keywords in the specified keyword set. In this paper, we use entropy to evaluate the uncertainty of determining data user's verified keywords from all the candidate keywords. Without loss of generality, we define p_i to be the probability that a keyword is specified to be verified. For a sub-vector with d elements (keywords), the entropy of identifying an individual element in the sub-vector is defined as

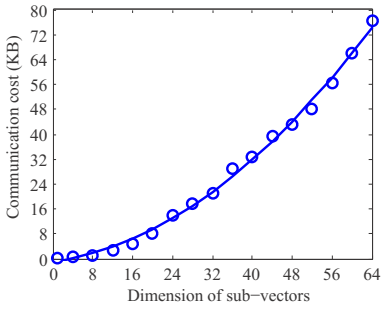
$$H(d) = -\sum_{i=1}^{d} p_i \cdot log_2 p_i \tag{4}$$

Obviously, when all the keywords in the sub-vector shares the same probability to be verified, i.e., $p_i = 1/d$, the maximum entropy is achieved, that is $H(d) = log_2 d$. When the dimension (number of elements in the sub-vector) of sub-vector is D, i.e., $d = D$, we get the maximum entropy $log_2 D$.

Now we investigate the relationship between the dimension of sub-vectors and the communication cost of verification. To get this relationship, we conduct experiment on a real data set [28], and get the empirical result. We set the size of keyword set to be 64, and $k = 10$. Fig. 5(b) illustrates the relationship between the dimension of sub-vectors and the communication cost of verification. The corresponding fitting equation is $y = 0.015 \cdot d^2 + 0.219 \cdot d - 0.203$.



(a) Entropy with dimension of sub-vectors



(b) Communication cost with dimension of sub-vectors

Fig. 5. Entropy and communication cost with dimension of sub-vectors

As we can see from Fig. 5(a) and Fig. 5(b), the larger the dimension of sub-vectors is, the higher entropy we get, while the more communication cost is also caused. Therefore, we need to find an optimal dimension(number of elements in the sub-vector) of the sub-vector, so that we can maximize the entropy while minimize the communication cost. The key idea is described as follows: first of all, to allow consistent computation, we convert the data range of both entropy and communication cost to the same range, say, [0,1]. Second, we define the difference

between entropy and communication cost as the optimization objective. Finally, we find the optimal dimension of sub-vector where the value of the difference is maximized. For example, we denote the relationship between the entropy and the dimension of sub-vector as $y_1 = log_2 x$, and the relationship between the communication cost and the dimension of sub-vector as $y_2 = 0.015 \cdot x^2 + 0.219 \cdot x - 0.203$. First of all, we encode them in the same range [0,1], therefore, we get $y_1' = log_2 x/6$ and $y_2' = 1.994 \times 10^{-4} \cdot x^2 + 0.003 \cdot x - 0.002$. Then we define the optimization objective as: $f(x) = y_1' - y_2' = log_2 x/6 - 1.994 \times 10^{-4} \cdot x^2 - 0.003 \cdot x + 0.002$. Finally, we compute the optimal value of x. Obviously, when $x \in (0, 22]$, $f(x)$ keeps increasing, when $x \in (22, 64]$, $f(x)$ keeps decreasing. Therefore, we can easily conclude that when $x = 22$, $f(x)$ gets the maximum data, i.e., the optimal dimension of the sub-vector is x=22. To make the dimension (length) of the original file vector divisible by the dimension of sub-vector 22, we can pad 2 dummy attributes into the original file vector.

7 Security Analysis

In this section, we analyze the security of our proposed scheme from the following two aspects.

7.1 Privacy Preserving and User Specified Ranked Multi-keyword Search

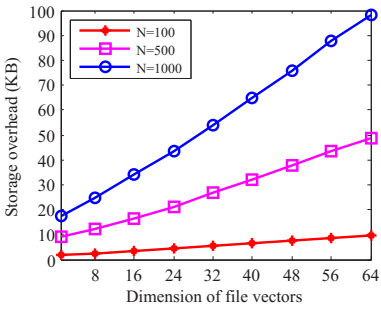
In our scheme, we use the inner product on the file vector P and the search vector Q, i.e., $P \cdot Q$, to quantitatively evaluate the similarity between a file and a query. Since the vector encryption method has been proved to be secure in the known ciphertext model in [26], the privacy of both P and Q are well protected if the secret key $\{S, M_1, M_2\}$ are kept secret.

7.2 Authenticating Dynamic Top-k Results

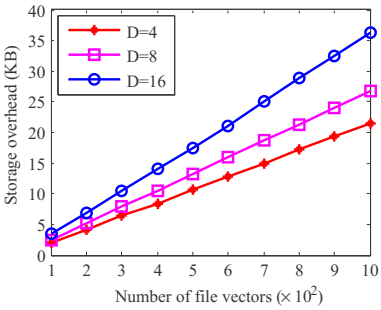
For search results verification, the cloud server only operates on random cipher-text, and returns the encoded vector, encrypted vector, hash value and $sign(H_{root})$. The security of privacy and order preserving function is proved in [27], therefore, the encoded vector is secure. The security of encrypted vector is proved [26]. Additionally, we adopt the RSA to get the signature, whose security is also guaranteed. Therefore, the security of the verification scheme is assured.

8 Performance Evaluation

In this section, we demonstrate a thorough evaluation on the storage overhead, communication cost, and time cost of our proposed schemes.



(a) Storage overhead with different dimension of file vectors



(b) Storage overhead with different number of file vectors

Fig. 6. Storage overhead of dimension of file vectors and number of file vectors

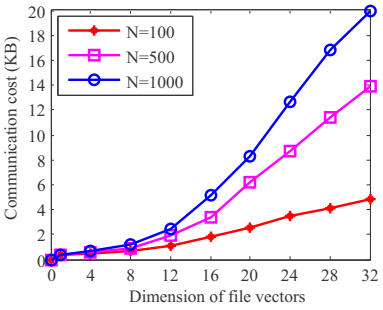
8.1 Experiment Settings

We conducted a performance evaluation on a real data set, i.e., US Census Data (1990) Data Set [28]. The data set has 2458285 census instances, where each data has 68 attributes. The data value of each attribute changes between 0 and 225. Our experiment is implemented with C++ on a PC with 3.40GHz Intel Core CPU and 4GB memory. We use RSA to sign the root node of MAAT with a 1024-bit key, and set the size of the hash digest to be 16 Bytes. Additionally, since the max attribute value is 225, we use 8 bits to represent each attribute. The performance of our scheme is evaluated regarding the effectiveness and efficiency of our proposed MAAT, including the storage overhead, the communication cost, and the construction time.

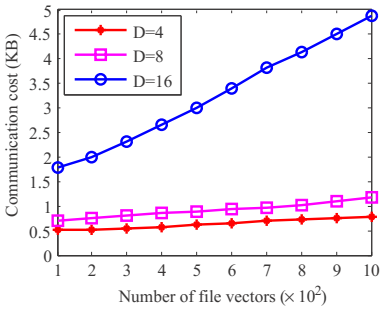
8.2 Experiment Results

Storage Overhead. Fig. 6(a) demonstrates the relationship between storage overhead and dimension of file vectors. As we can see, the storage overhead increases linearly with the dimension of file vectors increases. Additionally, the more file vectors we involve, the higher storage overhead is caused. The fundamental reason is that, the larger the dimension of file vectors is, the more storage overhead we spend to store the additional dimensions (attributes). Fig. 6(b) describes the relationship between the storage overhead and the number of file vectors. As we can see, the storage overhead also increases linearly and slowly with the number of file vectors. When D=16, and the number of file vectors changes from 100 to 1000, the storage overhead increases from 0.5 KB to 35 KB, which is acceptable.

Communication Cost. In our scheme, since the communication cost between the data owner and the cloud server is nearly the same with the storage overhead of the cloud server, we do not consider it here. Instead, we only consider the



(a) Communication cost with different dimension of file vectors(k = 10)



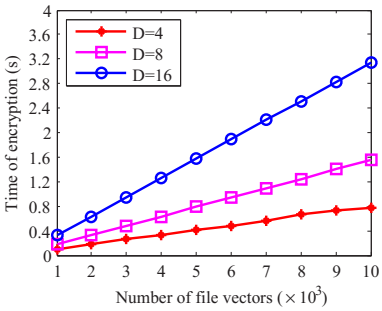
(b) Communication cost with different number of file vectors (k =10)

Fig. 7. Communication cost with number of file vectors and dimension of file vectors

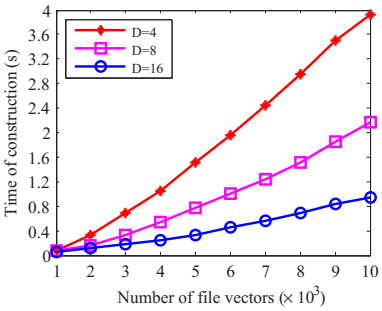
communication cost between the cloud server and the data users. When the data users submit the query vector to the cloud server, the cloud server would not only return top-k search results, but also return verification data. Since different search requests will contribute to different size of verification data, we show the average communication cost here.

Fig. 7(a) demonstrates the relationship between the communication cost and dimension of file vectors. As we can see, when the dimension of file vectors increases from 1 to 12, the communication cost increases slowly. When k=10, N=1000, and the dimension of file vectors increases from 0 to 32, the communication cost increases from 0 KB to about 20 KB. Fig. 7(b) shows that the communication cost increases linearly with the number of file vectors. As we can see, when the dimension D=4 and D=8, their communication cost is relatively small. However, when the dimension is more than 16, the communication cost increases rapidly with the number of file vectors. As we can see, when the dimension of file vectors is 16, and the number of file vectors changes from 100 to 1000, the communication cost increases from 1.75 KB to 5 KB. In our optimized MAAT, we propose that splitting the large vector into small sub-vectors will help reduce, and control the communication cost, which is proved by the experiment.

Time Cost. In our scheme, we mainly consider the time cost caused by constructing MAAT and encrypting file vectors. Fig. 8(a) demonstrates that, the encryption time increases linearly with the number of file vectors. As we can see, when the dimension of file vectors is 16, and the number of file vectors increases from 1000 to 10000, the encryption time increases from 0.3s to 3.2s. Fig. 8(b) shows the time cost of constructing MAAT with different number of file vectors. The time cost increases linearly with the number of file vectors. As shown in Fig. 8(b), when the dimension of file vectors is 4, and the number of file vectors



(a) Time cost of encryption with different number of file vectors



(b) Time cost of constructing MAAT with different number of files vectors

Fig. 8. Time of construction and encryption with different number of file vectors

increases from 1000 to 10000, the time spent on constructing MAAT increases from 0.1s to 4s, which is acceptable.

9 Conclusion

In this paper, for the first time, we consider a challenging security model where the cloud server would probably behave dishonestly. We first formalize different users' preferences and adopt the secure k nearest neighbor techniques to achieve privacy preserving personalized multi-keyword search. Then we use the order and privacy preserving function to preserve the relevance scores between keywords and files. Further, we propose a novel Multi-Attribute Authentication Tree (MAAT) to authenticate the dynamic top-k search results. In particular, we propose to optimize the MAAT, and compute the optimal parameter value to trade off the privacy and communication cost. Finally, we conduct extensive experiments on real-world datasets to confirm the efficacy and efficiency of our proposed schemes.

Acknowledgements. This work is supported in part by the National Natural Science Foundation of China (Project No. 61173038, 61472125).

References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, V.: A View of Cloud Computing. Communications of the ACM **53**(4), 50–58 (2010)
2. Song, D., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of the IEEE International Symposium on Security and Privacy (S&P00), pp. 44–55. IEEE, Nagoya (2000)

3. Goh, E.J.: Secure indexes. In: IACR Cryptology ePrint Archive, 216 (2003)
4. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
5. Golle, P., Staddon, J., Waters, B.: Secure conjunctive keyword search over encrypted data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004)
6. Ballard, L., Kamara, S., Monrose, F.: Achieving efficient conjunctive keyword searches over encrypted data. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 414–426. Springer, Heidelberg (2005)
7. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005)
8. Wang, C., Cao, N., Li, J., Ren, K., Lou, W.: Secure ranked keyword search over encrypted cloud data. In: The 30th International Conference on Distributed Computing Systems, pp. 253–262. IEEE, Genoa (2010)
9. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. In: Proceedings of the 30th IEEE International Conference on Computer Communications, INFOCOM 2011, pp. 829–837. IEEE, Shanghai (2011)
10. Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., Lou, W.: Fuzzy keyword search over encrypted data in cloud computing. In: Proceedings of the 29th IEEE International Conference on Computer Communications, INFOCOM 2010, pp. 1–5. IEEE, San Diego (2010)
11. Hore, B., Chang, E.-C., Diallo, M.H., Mehrotra, S.: Indexing encrypted documents for supporting efficient keyword search. In: Jonker, W., Petković, M. (eds.) SDM 2012. LNCS, vol. 7482, pp. 93–110. Springer, Heidelberg (2012)
12. Xu, Z., Kang, W., Li, R., Yow, K., Xu, C.Z.: Efficient multi-keyword ranked query on encrypted data in the cloud. In: The 18th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2012), pp. 244–251. IEEE, Singapore (2012)
13. Pang, H., Jain, A., Ramamritham, K., Tan, K.L.: Verifying completeness of relational query results in data publishing. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp. 407–418. ACM, New York (2005)
14. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)
15. Sun, W., Wang, B., Cao, N., Li, M., Lou, W., Hou, Y.T., Li, H.: Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. In: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, pp. 71–82. ACM, Hangzhou (2013)
16. Sun, W., Yu, S., Lou, W., Hou, Y.T., Li, H.: Protecting your right: attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. In: Proceedings of the 33nd IEEE International Conference on Computer Communications, INFOCOM 2014, pp. 226–234. IEEE, Toronto (2014)
17. Zhang, W., Xiao, S., Lin, Y., Zhou, T., Zhou, S.: Secure ranked multi-keyword search for multiple data owners in cloud computing. In: Proceedings of 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2014), pp. 276–286. IEEE, Atlanta (2014)

18. Zhang, W., Lin, Y., Xiao, S., Wu, J., Zhou, S.: Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing. In: The IEEE Transactions on Computers. IEEE (2015)
19. Zhang, W., Lin, Y.: Catch you if you misbehave: ranked keyword search results verification in cloud computing. In: IEEE Transactions on Cloud Computing. IEEE (2015)
20. Zhang, W., Lin, Y., Xiao, S., Liu, Q., Zhou, T.: Secure distributed keyword search in multiple clouds. In: 2014 IEEE 22nd International Symposium on Quality of Service (IWQoS), pp. 370–379. IEEE, Hongkong (2014)
21. Narasimha, M., Tsudik, G.: DSAC: integrity for outsourced databases with signature aggregation and chaining. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, pp. 235–236. ACM (2005)
22. Pang, H., Mouratidis, K.: Authenticating the query results of text search engines. Proceedings of the VLDB Endowment **1**(1), 126–137 (2008). ACM
23. Li, F., Hadjieleftheriou, M., Kollios, G., Reyzin, L.: Dynamic authenticated index structures for outsourced databases. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, pp. 121–132. ACM, Chicago (2006)
24. Sun, W., Wang, B., Cao, N., Li, M., Lou, W., Hou, Y.T., Li, H.: Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. IEEE Transactions on Parallel and Distributed Systems **52**(11), 3025–3035 (2013). IEEE
25. Jung, T., Li, X.Y., Wan, Z., Wan, M.: Privacy preserving cloud data access with multi-authorities. In: Proceedings of the 32nd IEEE International Conference on Computer Communications, INFOCOM 2013, pp. 2625–2633. IEEE, Turin (2013)
26. Wong, W.K., Cheung, D.W.L., Kao, B., Mamoulis, N.: Secure knn computation on encrypted databases. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, pp. 139–152. ACM, Paris (2009)
27. Yi, Y., Li, R., Chen, F., Liu, A.X., Lin, Y.: A digital watermarking approach to secure and precise range query processing in sensor networks. In: Proceedings of the 32nd IEEE International Conference on Computer Communications, INFOCOM 2013, pp. 1950–1958. IEEE, Turin (2013)
28. US Census Data (1990) Data Set. https://archive.ics.uci.edu/ml/datasets/US+Census+Data+ (1990)