11-2018

VPSearch: Achieving verifiability for privacy-preserving multi-keyword search over encrypted cloud data

Zhiguo WAN

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

DOI: https://doi.org/10.1109/TDSC.2016.2635128

Citation

WAN, Zhiguo and DENG, Robert H.. VPSearch: Achieving verifiability for privacy-preserving multi-keyword search over encrypted cloud data. (2018). *IEEE Transactions on Dependable and Secure Computing*. 15, (6), 1083-1095. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/4212

Published in IEEE Transactions on Dependable and Secure Computing, 2018 November, Volume 15, Issue 6, Pages 1083-1095
https://doi.org/10.1109/TDSC.2016.2635128

VPSearch: Achieving Verifiability for Privacy-Preserving Multi-Keyword Search over Encrypted Cloud Data

Zhiguo Wan and Robert H. Deng, *Fellow, IEEE*

Abstract—Although cloud computing offers elastic computation and storage resources, it poses challenges on verifiability of computations and data privacy. In this work we investigate verifiability for privacy-preserving multi-keyword search over outsourced documents. As the cloud server may return incorrect results due to system faults or incentive to reduce computation cost, it is critical to offer verifiability of search results and privacy protection for outsourced data at the same time. To fulfill these requirements, we design a Verifiable Privacy-preserving keyword Search scheme, called VPSearch, by integrating an adapted homomorphic MAC technique with a privacy-preserving multi-keyword search scheme. The proposed scheme enables the client to verify search results efficiently without storing a local copy of the outsourced data. We also propose a random challenge technique with ordering for verifying top-k search results, which can detect incorrect top-k results with probability close to 1. We provide detailed analysis on security, verifiability, privacy, and efficiency of the proposed scheme. Finally, we implement VPSearch using Matlab and evaluate its performance over three UCI bag-of-words data sets. Experiment results show that authentication tag generation incurs about 3 percent overhead only and a search query over 300,000 documents takes about 0.98 seconds on a laptop. To verify 300,000 similarity scores for one query, VPSearch costs only 0.29 seconds.

Index Terms—Cloud computing, verifiability, keyword search, privacy

1 Introduction

Cloud computing revolutionizes the computation model by offering elastic computation and storage resources. A client can outsource his data to the cloud server and later access the data with other devices from anywhere. The client can further ask the cloud server to perform some computation over his data on his behalf. These advantages lead to great success of cloud computing, but they also raise security and privacy concerns with respect to the client's data [1]. The concerns come from the fact that the service providers can not be fully trusted. To protect data privacy, data should be encrypted before being outsourced to the cloud. However, this is nontrivial as the encryption scheme need to support computations over encrypted data. Although fully homomorphic encryption provides a seemingly perfect solution for supporting arbitrary computations over encrypted data, its prohibitive computation cost makes it impractical currently.

One of the fundamental and most frequent data operations is keyword search. How to perform efficient and versatile search operations on encrypted cloud data has been an important research direction since the seminal work of Song et al. [2]. More recent works have achieved keyword privacy for keyword search over encrypted data, i.e., the keywords in queries are protected from the cloud server. Solutions for both single-keyword search [2], [3], [4] and multi-keyword search [5], [6], [7], [8], [9], [10], [11], [12], [13], [14] have been proposed in the literature. Unfortunately, all these schemes assume the cloud server will follow the designated protocols honestly, i.e., the honest-but-curious model, while whether the computation result is authentic is not considered in those proposals. However, as the cloud may return false results to the client either due to system faults or incentive to reduce computation cost, the keyword search results cannot be fully trusted, especially for very critical computation results. Thus, it is important for the client to be able to verify the keyword search results over encrypted cloud data, but this has not been studied sufficiently.

Verifiability of delegated computation over outsourced data is an important research problem for cloud computing. This problem has attracted considerable research interests, and many solutions have been proposed, including verifiable computation [15], [16], [17], [18], [19], [20], [21], homomorphic MACs/signatures [22], [23], [24], [25] as well as other proposals. However, most of these proposals suffer from various limitations. Verifiable computation schemes aim to minimize computation effort of the client, but not storage or communication cost. They require the client and the server to interactively authenticate the computation result. Moreover, the client needs to have a copy of the outsourced data, and the data over which computation is verified cannot be changed in the future.

Homomorphic message authenticator schemes [22], [23], which were proposed recently, avoid a local copy of the outsourced data on the client side. Homomorphic message

- *Z. Wan is with the School of Computer Science and Technology, Shandong University, Jinan 250101, China. E-mail: wanzhiguo@sdu.edu.cn.*
- *R.H. Deng is with the School of Information Systems, Singapore Management University, Singapore 188065. E-mail: robertdeng@smu.edu.sg.*

Manuscript received 6 Aug. 2016; revised 25 Oct. 2016; accepted 27 Nov. 2016. Date of publication 2 Dec. 2016; date of current version 9 Nov. 2018.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TDSC.2016.2635128

authenticator schemes allow a client to authenticate a collection of data $m_1, \ldots, m_n$ with his secret key sk, and later to authenticate the computation result $m = \mathcal{P}(m_1, \ldots, m_n)$ of a running program $\mathcal{P}$ over the data. The authentication tag σ that certifies m can be produced without knowing sk. Consequently, anyone who knows sk can certify the computation result m of the outsourced data with a short authentication tag σ. A recent homomorphic MAC scheme proposed in [24] improves the verification efficiency, but the same as all aforementioned proposals, it does *not* consider the problem over encrypted data. To achieve verifiable computation over encrypted cloud data, Fiore et al. [26] recently proposed a general scheme based on homomorphic MACs. Nevertheless, their homomorphic MAC scheme consider integer computation only and do not support computations over real numbers.

Our Contribution. In this paper, we investigate the problem of achieving verifiability for privacy-preserving multi-keyword search over encrypted cloud data. Different from the honest-but-curious model used in existing privacy-preserving keyword search schemes, we assume a partially honest model, in which the cloud server may return wrong results due to system faults or incentive to reduce computation cost.

To this end, we adapt the efficient homomorphic MAC scheme from [23] to support privacy-preserving multi-keyword search. The original homomorphic MAC is designed for computations over plaintexts in finite fields only, while privacy-preserving multi-keyword search schemes deal with encrypted real numbers. Moreover, because the homomorphic MAC scheme in [23] uses modulo operations, one cannot compare or sort the computation results. As a result, it is necessary to adapt the homomorphic MAC to support privacy-preserving multi-keyword search schemes.

We choose to work on a privacy-preserving multi-keyword ranked searchable encryption (MRSE) scheme in [11], which uses matrix multiplication to encrypt data index and inner product to compute similarity scores. By applying our adapted homomorphic MAC, our proposed scheme is able to achieve verifiable privacy-preserving multi-keyword search over encrypted cloud data. To our best knowledge, our solution is the first to use the homomorphic MAC to achieve both verifiability and privacy for multi-keyword search. VPSearch is highly efficient as it uses only a one-way function (pseudorandom function) for security, making it scalable to large databases. Since top-k retrieval for search results is a commonly used operation, we further propose a random challenge technique to verify top-k multi-keyword search results. This technique enables the client to detect wrong top-k results with probability close to 1.

We implement VPSearch using Matlab and evaluate its performance over three UCI bag-of-words data sets [27]. Compared with the privacy-preserving keyword search scheme without verifiability, VPSearch incurs only about 3 percent overhead for authentication tag generation. For keyword search, VPSearch can finish a query over 300,000 documents in 0.98 seconds. Parallel execution of keyword search over multiple servers can be used to further speed up this operation.

The contributions of our work can be summarized as follows:

- We design an efficient, verifiable and privacy-preserving multi-keyword ranked searchable encryption scheme for outsourced cloud data under the partially honest cloud server model. It is realized by integrating an adapted homomorphic MAC technique with a privacy-preserving multi-keyword search scheme. The proposed scheme is very efficient as it relies on only one-way function for security.
- We also provide the random challenge technique to verify top-k search results for a given query. With this solution, the client can be sure that the top-k results are authentic for probability close to 1.
- We provide detailed analysis on security, privacy, verifiability and efficiency of VPSearch. Specifically, the underlying homomorphic MAC scheme used in VPSearch can be proved to be secure in the same way as [23].
- We implement VPSearch using Matlab and evaluate its performance over three UCI data sets. Experiment results on a laptop show that VPSearch is very efficient on authentication tag generation and keyword search operations.

More importantly, although we only describe how to implement verifiability for a specific MRSE scheme [11], our solution can be extended to other MRSE schemes adopting the similar models, e.g., [12], [13], [28].

Paper Organization. We review related work in the next section, then some preliminaries on homomorphic MAC are provided in Section 3. System model, threat model and design goals of our proposed scheme are presented in Section 4. Detailed description of our scheme is given in Section 5, followed by discussion and security analysis. Then we present performance evaluation of our scheme and conclude the paper after that.

2 Related Work

2.1 Verifiable Computation

Our work is closely related to verifiable computation, which has been an active area in the past few years, and many schemes with different design goals have been proposed in the literature. Verifiable computation enables a client to delegate a computationally heavy task to a server with strong computation capability, and then verify correctness of the computation result.

Existing works can be categorized into *generic protocols* [15], [16], [17], [18] which allow arbitrary computations and *ad hoc protocols* which deal with specific classes of computation [19], [20], [21]. Generic protocols for verifiable computation are computationally more expensive than ad hoc ones which aim at efficient computation for specific classes of functions, e.g., high-degree functions [21].

Most verifiable computation schemes aim at reducing computation cost of the delegator, while storage cost of the delegator is not considered, i.e., the delegator needs a copy of the data for verification. Our work enjoys the advantage over verifiable computation in that the client does not need to maintain a local copy of the outsourced data. This advantage is becoming increasingly important for big data applications which compute over huge volume of data. Another advantage of our solution over verifiable computation is that we deal with encrypted data while most existing schemes consider plaintext data only.

TABLE 1
UCI Bag-of-Words Data Sets

Data set Name	Short Name	# Documents	# Keywords
NIPS full papers	NIPS	1,500	12,419
Enron Emails	Enron	39,861	28,102
NYTimes news articles	NYTimes	300,000	102,660

2.2 Homomorphic Signatures and MACs

Homomorphic message authentication codes and signatures with restricted homomorphism have been investigated for securing network coding since Johnson et al.'s work [29]. Most works since then focus on linear functions until Boneh and Freeman [25] proposed a homomorphic signature scheme for bounded constant degree polynomials. After that, homomorphic signatures and MACs are used to verify authenticity of computation results.

Gennaro and Wichs [22] constructed the first fully homomorphic message authenticator based on fully homomorphic encryption. Though it has great theoretic significance, the scheme suffers from significant computation overhead due to fully homomorphic encryption. Following this work, Catalano and Fiore [23] proposed a more computation-efficient homomorphic MAC for arithmetic circuits. Backs et al. [24] utilized the amortized closed-form efficiency technique [21] to reduce verification overhead. However, none of these solutions consider verification of computation over encrypted data, so privacy of the outsourced data is not preserved.

Although a straightforward enhancement is to apply homomorphic encryption on client data before outsourcing [26], it has an obvious drawback on computation overhead. Libert et al.'s scheme [30] deals with evaluation of linear combination over encrypted data, which is not applicable under our setting. Homomorphic authenticated encryption has been used by [31], [32] to verify computations over encrypted data. Different from existing schemes on verifiable computation over encrypted data, our work achieves efficient and verifiable keyword search by using the more efficient homomorphic MAC with the privacy-preserving keyword search scheme from [11]. The reader is referred to [24] for a more complete discussion.

2.3 Privacy-Preserving Keyword Search over Encrypted Data

Song et al. [2] initiated the research on keyword search over encrypted data, and proposed a scheme enabling single keyword search over an encrypted document. The basic idea is to build an encrypted searchable index while hiding the content of the corresponding document. Noticing the importance of keyword privacy, researchers proposed enhanced schemes capable of protecting keywords [3], [4]. Unfortunately, these schemes can only process single keyword search, so they obviously lack flexibility and expressiveness.

Multi-keyword search over encrypted data has been realized in the public key setting with conjunctive keyword search [5], [6], [7]. Some works like [8] employ predicate encryption technique to support conjunctive and disjunctive search. These schemes have several limitations due to the underlying cryptographic techniques. On one hand, they incur significant computation overhead due to expensive computations like bilinear mapping; on the other hand, conjunctive and disjunctive search fall short of search expressiveness.

Cao et al. [10], [11] designed a privacy-preserving multi-keyword ranked search based on "Coordinate matching". More recently, Xia et al. [28] further improve keyword search efficiency using a tree-based index. This tree-based index is combined with the vector model and the TF×IDF model. Fu et al. [12] considered the user search experience problem, and proposed a personalized multi-keyword searchable encryption scheme using a user's search history to build an interest model. This scheme can improve user search experience while preserving user privacy. Fu et al. [13] studied multi-keyword fuzzy ranked searchable encryption problem, and proposed an efficient solution using keyword transformation and word stemming. This scheme can deal with several types of keyword spelling errors. With appropriate modification, our solution can also be applied on the schemes in [12], [13], [28], [33] similarly.

All the above privacy-preserving keyword search schemes assume an honest-but-curious model, in which the cloud server strictly follows the designated protocol while being curious about privacy of the clients. These schemes did not consider verifiability of keyword search results, but it is possible the returned results may be incorrect due to system faults or incentive to reduce computation cost.

Sun et al. [34] employs multi-dimensional-B (MDB) tree to implement multi-keyword search and search result verifiability. Although this approach achieves efficiency and verifiability, it does not support dynamic operations for file addition and deletion. Whenever a new file is outsourced to the cloud, the data owner needs to generate a new signature for verification. Cheng et al.'s work [35] is based on indistinguishability obfuscation technique. Indistinguishability obfuscation may incur high computation overhead. In contrast, our work achieves both keyword privacy and verifiability for searchable encryption at modest cost.

3 Preliminaries

We review the definition of the labelled program in [22] and present our homomorphic message authenticator scheme for real numbers adapted from [23].

1) Labelled Programs: The labelled program is composed of a function[1] f and n input variables with each input assigned a label $\mathsf{L}_i (i = 1, 2, \ldots, n)$. L_i is a unique string to "label" the variable of the function f. For example, suppose a company outsources its employee payroll database D to the cloud, and then asks the cloud server to compute f as the average salary of all its employees. Then L_i can be constructed as ("the ith employee's salary") where i is the index of an employee. Then the salary amount m_i can be authenticated with respect to the label L_i, which essentially binds the data m with the corresponding label.

2) Our Homomorphic MAC for Real Numbers: As the homomorphic MAC in [23] only supports computations over finite fields, we propose an adapted homomorphic MAC for real numbers, which is to be used in our proposed scheme.

[1] More accurately, an arithmetic circuit that defines a polynomial function.

The underlying idea for the homomorphic MAC is simple but effective. Each message m_i is encoded as a degree-1 polynomial $y(x)$ such that $y(0) = m_i$ and $y(\alpha) = r_i$ where α, r_i are secrets known only to the verifier. That is, $y(x) = m_i + (r_i - m_i) \cdot x/\alpha$. As elementary arithmetic operations over polynomials are homomorphic, the cloud server can compute a function f, which involves only elementary arithmetic operations, over these degree-1 polynomials to obtain a polynomial $g(x)$. Then one can verify $f(m_1, \dots, m_n) = g(0)$ if $f(r_1, \dots, r_n) = g(\alpha)$.

Our modification to the homomorphic MAC scheme is to treat all messages, e.g., m_i and r_i, as real numbers encoded by a format like the double-precision floating point format defined in IEEE 754 standard. Then the new MAC scheme can deal with real numbers, while the homomorphic property of the MAC scheme is preserved. The details of our homomorphic MAC scheme $\mathsf{RealHomMAC}$ for real numbers is as follows:

- $\mathsf{KeyGen}(1^\lambda)$: This algorithm generates a secret key sk from the security parameter λ. Specifically, it chooses a seed K of a pseudo-random function $F_K : \{0, 1\}^* \rightarrow \mathbb{R}_\lambda$, where $\mathbb{R}_\lambda$ is the set of real numbers that is encode with λ bits. The algorithm also selects a random value $\alpha \in \mathbb{R}_\lambda$, and the secret key is $\mathsf{sk} = (K, \alpha)$.
- $\mathsf{Auth}(\mathsf{sk}, \mathsf{L}, m)$: This algorithm takes as inputs the secret key sk, a label L and a message $m \in \mathbb{R}_\lambda$, generates the authentication tag σ. Specifically, the algorithm computes $r_\mathsf{L} = F_K(\mathsf{L})$, sets $y_0 = m$ and $y_1 = (r_\mathsf{L} - m)/\alpha$, and outputs $\sigma = (y_0, y_1)$ as the authentication tag for m.
- $\mathsf{Eval}(f, \vec{\sigma})$: This algorithm performs evaluation of f on a vector of tags $\vec{\sigma} = (\sigma_1, \dots, \sigma_n)$, and outputs the resulting tag σ. f can be viewed as a circuit composed of addition and multiplication gates.

  The algorithm processes each gate of the circuit f as follows. At each gate f_g, given two tags $\sigma^{(1)}, \sigma^{(2)}$, it runs the algorithm $\sigma \leftarrow \mathsf{GateEval}(f_g, \sigma^{(1)}, \sigma^{(2)})$ as described below to derive a new tag σ. The new tag is in turn passed on as input to the next gate in the circuit f. The tag σ obtained at the last gate of the circuit f is the final output. The subroutine $\mathsf{GateEval}$ is defined as follows:

  - $\mathsf{GateEval}(f_g, \sigma^{(1)}, \sigma^{(2)})$. Let $\sigma^{(i)} = \vec{y}^{(i)} = (y_0^{(i)}, \dots, y_{d_i}^{(i)})$ for $i = 1, 2$ and $d_i \geq 1$.
    1) $f_g = +$: The algorithm computes the coefficients $(y_0, \dots, y_d)$ of the polynomial $y(x) = y^{(1)}(x) + y^{(2)}(x)$ where $d = \max(d_1, d_2)$. This can be done by simply adding the two vectors of coefficients, $\vec{y} = \vec{y}^{(1)} + \vec{y}^{(2)}$.
    2) $f_g = \times$: The algorithm computes the coefficients $(y_0, \dots, y_d)$ of the polynomial $y(x) = y^{(1)}(x) * y^{(2)}(x)$ using the convolution operator $*$, where $d = d_1 \cdot d_2$. That is, $y_k = \sum_{i=0}^{k} y_i^{(1)} \cdot y_{k-i}^{(2)} \; \forall k = 0, \dots, d$.

Finally the algorithm returns $\sigma = (y_0, \dots, y_d)$, the vector of coefficients of a d-degree polynomial.

- $\mathsf{Ver}(\mathsf{sk}, \mathcal{P}, m, \sigma)$: This algorithm verifies that m is the correct computation result of a labeled program $\mathcal{P} = (f, \mathsf{L}_1, \dots, \mathsf{L}_n)$ with the secret key sk and a tag $\sigma = (y_0, \dots, y_d)$.

  Specifically, this algorithm computes $f_0 = f(r_{\mathsf{L}_1}, \dots, r_{\mathsf{L}_n})$ where $r_{\mathsf{L}_i} = F_K(\mathsf{L}_i)$. Finally, it verifies the result by checking the following two equations:

  $$m \ = y_0 \tag{1}$$

  $$f_0 = \sum_{k=0}^{d} y_k \alpha^k. \tag{2}$$

  It accepts the result if both equations hold, and rejects otherwise.

4 Problem Formulation

4.1 System Model

Our scheme involves two different players: the client $\mathcal{C}$ who outsources her encrypted documents to the cloud, and a cloud server $\mathcal{S}$ that provides data storage and keyword search service to the clients.

To enable keyword search over encrypted documents, $\mathcal{C}$ constructs an encrypted searchable index for the encrypted documents, generates authentication tags for the index. Then she outsources the authenticated index and the encrypted documents to the cloud. When $\mathcal{C}$ wants to search the documents for some keywords, she prepares an authenticated search trapdoor for the cloud. On receiving the search trapdoor from $\mathcal{C}$, the cloud server performs search using the trapdoor and return the result to $\mathcal{C}$.

The cloud server should be able to order the search results according to a given ranking criteria, so as to enable the client $\mathcal{C}$ to get the most relevant documents with respect to the query. To reduce unnecessary communication, $\mathcal{C}$ can ask the cloud server to returns the k most relevant documents only, which is called top-k queries.

4.2 Threat Model

In contrast to the honest-but-curious cloud server assumption in most privacy-preserving keyword search schemes, we assume a partially honest model, in which the cloud server may return wrong results due to system faults or in order to reduce computation cost. More specifically, the cloud may finish only a part of the delegated computation task to reduce cost, and return the wrong result to the client [34]. However, to gain trust from the clients, the cloud server will not misbehave when the misbehavior detection probability is non-negligible.

The client $\mathcal{C}$, who takes both roles of data owner and data user, is assumed to be always honest in the entire process. That is, the client honestly encrypts outsourced documents, builds encrypted the searchable index, generates authentication tags, and composes search trapdoors.

As analyzed in existing privacy-preserving keyword search schemes, there are two types of knowledge held by the cloud server: 1) *known ciphertext*: the cloud server only knows the encrypted documents and the searchable index; 2) *known background knowledge*: the cloud server also has some background knowledge, e.g., statistics of the outsourced documents or relationship between different

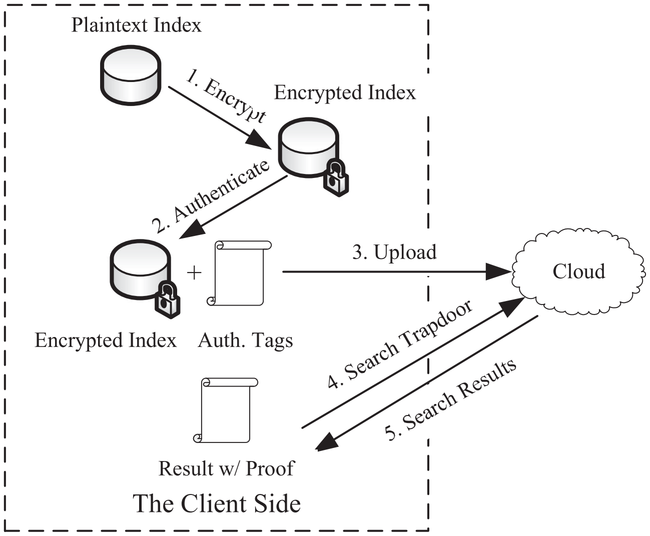


Fig. 1. Overview of our verifiable privacy-preserving multi-keyword search scheme.

trapdoors. In this work, we consider the case that the cloud server can combine its knowledge on ciphertext and background knowledge to deduce privacy about documents or keywords in the queries.

4.3 Design Goals

Our scheme is designed to achieve the following goals:

- *Verifiability.* The cloud server is able to produce a proof certifying correctness of the delegated multi-keyword search results, and the client is able to verify the results using the proof returned by the server *without* storing the outsourced data locally.
- *Efficiency.* The client is able to produce authentication tags for outsourced data and verify the search results efficiently. On the other hand, the server should be able to accomplish keyword search and produce authentication tags for the search results efficiently.
- *Keyword Privacy.* The cloud server should not be able to learn keywords in trapdoors or authentication tags generated by the client. Moreover, even if the server gains some background knowledge about the outsourced documents, it should not be able to infer any information about the queried keywords with such knowledge.
- *Data Privacy.* The server should not be able to deduce any information about the outsourced data, including the documents and the index.
- *Unbounded Storage.* The client should be able to continuously outsource any amount of data to the cloud.

5 Verifiable Privacy-Preserving Multi-Keyword Search over Encrypted Data

In this section we present VPSearch, the Verifiable Privacy-preserving multi-keyword Search scheme over encrypted cloud data. We first offer a high-level description of VPSearch before we present its details. After that, we describe a technique for VPSearch to support verifiability for top-k search results.

5.1 Overview of Our Scheme

Our scheme builds on the privacy-preserving multi-keyword search scheme in [11], which is integrated with our homomorphic MAC RealHomMAC to achieve both verifiability and privacy. The keyword search scheme is chosen for two reasons: (i) it supports flexible and expressive multi-keyword searches; (ii) its keyword search operation is actually inner product, which is fully supported by our homomorphic MAC.

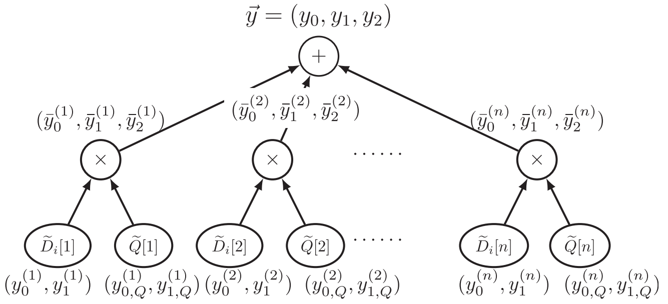
An overview of our schemes is illustrated in Fig. 1. The client first encrypts the plaintext index, then the encrypted index is authenticated with our homomorphic MAC technique, which produces authentication tags for the encrypted index. Next, the index and authentication tags are uploaded to the cloud. Then the client can generate a search trapdoor, and uses our homomorphic MAC technique to authenticate the trapdoor. With the authenticated trapdoor, the cloud server can homomorphically execute the search function over the authentication tags to derive the result with a proof, which can certify the search result.

Suppose the document set $\mathcal{D}$ to be outsourced contains N documents. For the ith document, its subindex D_i is constructed as an n-dimensional bit vector. $D_i[j]$ is set to 1 if this document contains the jth keyword in a given dictionary; otherwise, it is set as 0. Similarly, the query Q is also an n-dimensional bit vector, with the bits corresponding to the keywords interested by the client set to 1. Then the subindex D_i and the query Q are encrypted using matrix multiplication, i.e., $\widetilde{D}_i = M^T D_i$ and $\widetilde{Q} = M^{-1}Q$ where M is a real secret matrix. The similarity score can still be obtained via inner product of the encrypted index and query, i.e., $(M^T D_i)^T M^{-1}Q = D_i^T Q$. To mask the real similarity scores for stronger privacy, the document index and the query vector can be extended with random numbers as in [11]. Specifically, a document index is extended to $\vec{D}_i = (D_i, \epsilon_i, 1)$, while the query vector is extended to $\vec{Q} = (rQ, r, t)$, where ϵ_i, r, t are random numbers and t is relatively small. At last, the final similarity score computed by the cloud server will be $r(D_i \cdot Q + \epsilon_i) + t$. Although the similarity scores are randomized with the random numbers, the cloud server is still able to rank the results without knowing the real scores.

We apply our homomorphic MAC technique on the encrypted index $\widetilde{D}_i$ and query $\widetilde{Q}$ (trapdoor) to generate authentication tags for them, then the cloud server homomorphically executes the multi-keyword search function (inner product) over the authentication tags to obtain the results. Since our homomorphic MAC technique is adapted to support operations over real numbers, it supports keyword search operations perfectly. Using a one-way function, our homomorphic MAC technique is highly efficient in computation.

For each value in the encrypted document subindex and query, authentication tags are produced as follows. VPSearch authenticates each item (a real number) in the encrypted index $\widetilde{D}_i$ and query $\widetilde{Q}$ with two coefficients of a degree-1 polynomial. Verifiability of keyword search results is achieved by homomorphically evaluating the search function over the authentication tags. The cloud server performs the multi-keyword search function as showed in Fig. 2, i.e., calculating $\widetilde{D}_i^T \widetilde{Q}$.

The labeling approach in our homomorphic MAC can effectively reduce storage space compared to arbitrarily



$\vec{y} = (y_0, y_1, y_2)$
$+$
$(\bar{y}_0^{(1)}, \bar{y}_1^{(1)}, \bar{y}_2^{(1)})$
$(\bar{y}_0^{(2)}, \bar{y}_1^{(2)}, \bar{y}_2^{(2)})$
$(\bar{y}_0^{(n)}, \bar{y}_1^{(n)}, \bar{y}_2^{(n)})$
$\times$
......
$\widetilde{D}_i[1]$
$\widetilde{Q}[1]$
$\widetilde{D}_i[2]$
$\widetilde{Q}[2]$
$\widetilde{D}_i[n]$
$\widetilde{Q}[n]$
$(y_0^{(1)}, y_1^{(1)})$
$(y_{0,Q}^{(1)}, y_{1,Q}^{(1)})$
$(y_0^{(2)}, y_1^{(2)})$
$(y_{0,Q}^{(2)}, y_{1,Q}^{(2)})$
$(y_0^{(n)}, y_1^{(n)})$
$(y_{0,Q}^{(n)}, y_{1,Q}^{(n)})$

Fig. 2. The circuit structure of the keyword search (inner product) function f and the homomorphic MACs for the document index $\widetilde{D}_i$ and the query $\widetilde{Q}$. $\widetilde{D}_i[j]$ or $\widetilde{Q}[j]$ is authenticated with an authentication tag $\sigma_j^D = (y_0^{(j)}, y_1^{(j)})$ or $\sigma_j^Q = (y_{0,Q}^{(j)}, y_{1,Q}^{(j)})$, the coefficients of a degree-1 polynomial. The keyword search function f outputs a new authentication tag $\sigma = \vec{y} = (y_0, y_1, y_2)$. n is the length of the document index.

labeling large amount of data. Therefore, the client needs to compute the value $f(\mathsf{F}_K(\mathsf{L}_1), \ldots, \mathsf{F}_K(\mathsf{L}_n))$ for verification for each file. As the function f is lightweight, the verification can be performed efficiently.

5.2 VPSearch: Verifiable Privacy-Preserving Multi-Keyword Search Based on Homomorphic MAC

We present the verifiable privacy-preserving multi-keyword search scheme based on one-way functions, which is referred to as VPSearch. The homomorphic MAC technique used in VPSearch can authenticate computations over real numbers, and its security relies on existence of one-way functions. Without any public key operation, this scheme enjoys great computational efficiency.

VPSearch consists the following six algorithms: Setup, Index, Trapdoor, Auth, KeywordSearch and Verify.

- Setup$(1^\lambda, n)$. This algorithm takes as inputs a security parameter λ and the dictionary size n. n is the number of keywords in a given dictionary (in the next algorithm), and it also represents the length of the document index.

  Given the security parameter λ, $\mathcal{C}$ invokes KeyGen(1^λ) of RealHomMAC to obtain two secrets (K, α) as the secret key. Then $\mathcal{C}$ selects two $(n+2) \times (n+2)$ invertible matrix M_1 and M_2 in $\mathbb{R}_\lambda^{(n+2)\times(n+2)}$. $\mathcal{C}$ also chooses an $(n+2)$ bit random vector S. The bit vector S is used to randomly split the document sub-indices for privacy as [11]. The secret key held by $\mathcal{C}$ is $\mathsf{sk} = (K, \alpha, S, M_1, M_2)$.
- Index$(\mathcal{D}, \mathsf{sk}, \mathsf{dict})$. This algorithm takes as inputs the document set containing N plaintext documents, the secret key sk, and a dictionary dict of size n, and outputs the encrypted document index for outsourcing to the cloud. Its goal is to protect the document index from the cloud server. It processes the document index as follows:

  For the ith document in $\mathcal{D}$, its subindex D_i is first constructed as a bit vector. $D_i[j]$ is set as 1 if this document contains the jth keyword w_j in dict; otherwise, it is set as 0. Then D_i is extended to be $\vec{D}_i = (D_i, \epsilon_i, 1)$ where ϵ_i is a random number conforming to normal distribution.

  Next, $\vec{D}_i$ is split into two vectors $\vec{D}_i'$ and $\vec{D}_i''$ according to the following rule (Split in Fig. 3): if $S[j] = 0$, $\vec{D}_i'[j] = \vec{D}_i''[j] = \vec{D}_i[j]$; otherwise, $\vec{D}_i'[j] + \vec{D}_i''[j] = \vec{D}_i[j]$ (i.e., random split of $\vec{D}_i[j]$).

  At last, $\mathcal{C}$ encrypts the subindex $(\vec{D}_i', \vec{D}_i'')$ using the secret matrices M_1, M_2, yielding $\widetilde{D}_i = (M_1^T \vec{D}_i', M_2^T \vec{D}_i'')$ (Encrypt in Fig. 3). The encrypted document index, containing all sub-indices $\widetilde{D}_i$, along with the encrypted document will be uploaded to the cloud along with some authentication tags generated by the Auth algorithm.
- Trapdoor(Q, sk). This algorithm takes as inputs a search query in plaintext and the secret key sk, and outputs the encrypted query as a trapdoor. Its goal is to protect the keyword information in the query from the cloud server. It processes the keyword query as follows:

  The search query is constructed as a bit vector, with $Q[j]$ set to 1 if the client is interested in keyword w_j in the dictionary or 0 otherwise. Then $\mathcal{C}$ generates

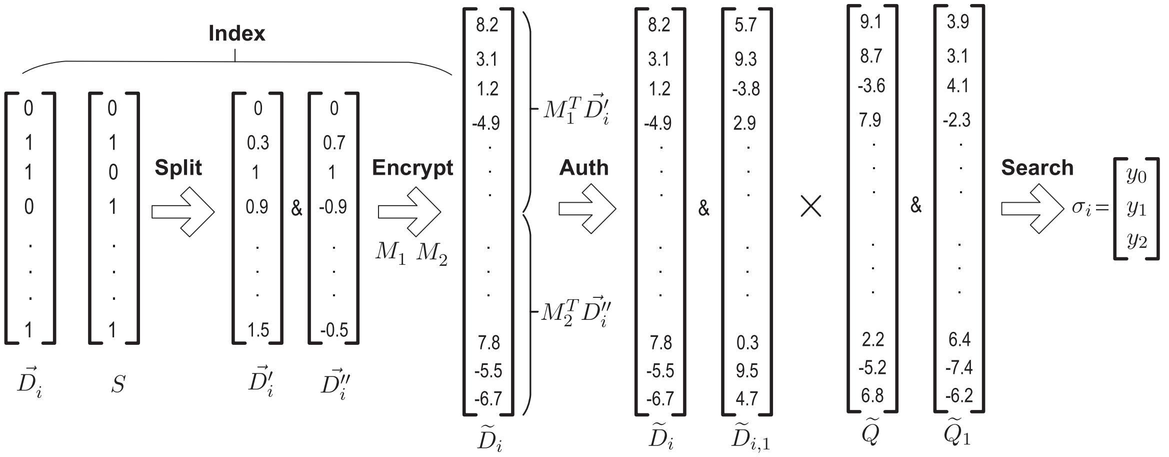


Fig. 3. Illustration of the VPSearch Scheme: an extended document index $\vec{D}_i$ is split into two vectors $\vec{D}_i'$ and $\vec{D}_i''$ according to the secret S; then they are encrypted by multiplying with secret matrices M_1, M_2; the resulting vector is authenticated using our homomorphic MAC, yielding $\widetilde{D}_i$ and $\widetilde{D}_{i,1}$; the authentication tags of the document index and the query are multiplied to obtain the final result $\sigma_i = (y_0, y_1, y_2)$, which is used by the client to verify the search result.

two random numbers $r, t \in \mathbb{R}_\lambda$, and extends Q to $\vec{Q} = (rQ, r, t)$. Next, $\mathcal{C}$ split $\vec{Q}$ according to the following rule: if $S[j] = 0$, $\vec{Q}'[j] + \vec{Q}''[j] = \vec{Q}[j]$ (i.e., random split of $\vec{Q}[j]$); otherwise, $\vec{Q}'[j] = \vec{Q}''[j] = \vec{Q}[j]$.

Finally, $\mathcal{C}$ generates the keyword search trapdoor as $\widetilde{Q} = (M_1^{-1}\vec{Q}', M_2^{-1}\vec{Q}'')$.

- $\mathsf{Auth}(\mathsf{sk}, \mathsf{L}, \widetilde{D}_i \text{ or } \widetilde{Q})$. This algorithm produces authentication tags for each item in the encrypted subindex $\widetilde{D}_i$ or the search trapdoor $\widetilde{Q}$. Each item in the encrypted subindex $\widetilde{D}_i$ and $\widetilde{Q}$ is labeled by a string L. For instance, the jth item in $\widetilde{D}_i$ is labeled $\mathsf{L}_{\widetilde{D}_i, j}$ = "The jth item in the index for the ith document". The jth item in the trapdoor is labeled as $\mathsf{L}_{\widetilde{Q}, j}$ = "The jth item in trapdoor $\widetilde{Q}$".

  To authenticate a message $m \in \mathbb{R}_\lambda$ with label $\mathsf{L} \in \{0,1\}^*$, $\mathcal{C}$ invokes $\mathsf{RealHomMAC.Auth}(\mathsf{sk}, \mathsf{L}, m)$ to output a tag σ. Specifically, the algorithm computes $r_\mathsf{L} = F_K(\mathsf{L})$, sets $y_0 = m$, $y_1 = (r_\mathsf{L} - m)/\alpha$, and outputs an authentication tag (y_0, y_1). Note y_0 is just the original message m.

  Therefore, for the encrypted subindex $\widetilde{D}_i$, the authentication tag is $\sigma_{D_i} = (\widetilde{D}_i, \widetilde{D}_{i,1})$; while for the trapdoor $\widetilde{Q}$ it will be $\sigma_Q = (\widetilde{Q}, \widetilde{Q}_1)$. The first item in the authentication tag corresponds to $y_0 = m$, the message to be authenticated, and the second item corresponds to y_1.

  Since authentication tags are generated over the extended index, including not only the original index, but also additional random numbers (i.e., ϵ, r and t) and dummy keywords, they can be used to verify search results with errors due to random numbers and dummy keywords.
- $\mathsf{KeywordSearch}(f, \vec{\sigma}_D, \sigma_Q)$. This algorithm takes as inputs the keyword search function f, the authentication tags of the document index $\vec{\sigma}_D$ and the search trapdoor σ_Q. $\vec{\sigma}_D$ is parsed as $\{\sigma_{D_1}, \sigma_{D_2}, \ldots, \sigma_{D_N}\}$. Then for each subindex D_i it performs keyword search by invoking $\mathsf{RealHomMAC.Eval}(f, \vec{\sigma})$ where $\vec{\sigma} = \{\sigma_{D_i}, \sigma_Q\}$. Essentially, it homomorphically evaluates the search function f over σ_{D_i} and σ_Q, and outputs coefficients (also seen as an authentication tag) $\sigma_i = (y_0, y_1, y_2)$.

  Among the coefficients returned by this algorithm, y_0 is the actual similarity score. According to the keyword search function showed in Fig. 2, the three coefficients are calculated as follows:

$$y_0 = \widetilde{D}_i \cdot \widetilde{Q} \tag{3}$$

$$= (M_1^T \vec{D}_i', M_2^T \vec{D}_i'') \cdot (M_1^{-1}\vec{Q}', M_2^{-1}\vec{Q}'') \tag{4}$$

$$= r(D_i \cdot Q + \epsilon_i) + t \tag{5}$$

$$y_1 = \widetilde{D}_i \cdot \widetilde{Q}_1 + \widetilde{D}_{i,1} \cdot \widetilde{Q} \tag{6}$$

$$y_2 = \widetilde{D}_{i,1} \cdot \widetilde{Q}_1. \tag{7}$$

  Accordingly, the polynomial corresponding to σ_i evaluates to $f(r_{\mathsf{L}_1}, \ldots, r_{\mathsf{L}_{2n}})$ at the secret point α, i.e., $y_0 + y_1\alpha + y_2\alpha^2$. Note here keyword search f has $2n$ inputs as illustrated in Fig. 2. $\mathsf{L}_1, \mathsf{L}_2, \ldots, \mathsf{L}_n$ are labels for the document subindex, and $\mathsf{L}_{n+1}, \mathsf{L}_{n+2}, \ldots, \mathsf{L}_{2n}$ are for the search trapdoor.
- $\mathsf{Verify}(\mathsf{sk}, f, \vec{\mathsf{L}}, m, \sigma)$.$\mathcal{C}$ parses $\vec{\mathsf{L}}$ as $(\mathsf{L}_1, \ldots, \mathsf{L}_{2n})$ and σ as (y_0, y_1, y_2). Then it sets $\mathcal{P} = (f, \mathsf{L}_1, \ldots, \mathsf{L}_{2n})$, and invokes $\mathsf{RealHomMAC.Ver}(\mathsf{sk}, \mathcal{P}, m, \sigma)$ to obtain the result. Specifically, the algorithm computes $f_0 = f(r_{\mathsf{L}_1}, \ldots, r_{\mathsf{L}_{2n}})$ where $r_{\mathsf{L}_i} = F_K(\mathsf{L}_i)$. Finally, $\mathcal{C}$ verifies the result by checking the following equation:

$$m = y_0 \tag{8}$$

$$f_0 = \sum_{k=0}^{2} y_k \alpha^k. \tag{9}$$

  According to the definition of our homomorphic MAC given in Section III, $f_0 = f(r_{\mathsf{L}_1}, \ldots, r_{\mathsf{L}_{2n}})$ is actually the search function f computed over the pseudorandom numbers generated from labels, so it should be equal to the result evaluated at point α for the polynomial defined by (y_0, y_1, y_2). If both equations are satisfied, $\mathcal{C}$ accepts y_0 as the search result; otherwise $\mathcal{C}$ rejects the result.

Fig. 3 illustrates the proposed scheme with a simple example. Besides Setup is omitted in the figure, $\mathsf{Trapdoor}$ is not showed as it is similar to Auth. In this example, an extended document index $\vec{D}_i$ is split into two vectors $\vec{D}_i'$ and $\vec{D}_i''$ according to the secret S. Then they are encrypted by multiplying with secret matrices M_1, M_2, and the resulting vector is authenticated using our homomorphic MAC, yielding $\widetilde{D}_i$ and $\widetilde{D}_{i,1}$. Here $\widetilde{D}_i$ and $\widetilde{D}_{i,1}$ can be viewed as coefficients of a degree-1 polynomial, and the polynomial evaluates to $\widetilde{D}_i$ at point 0 and $\widetilde{D}_{i,1}$ at point α.

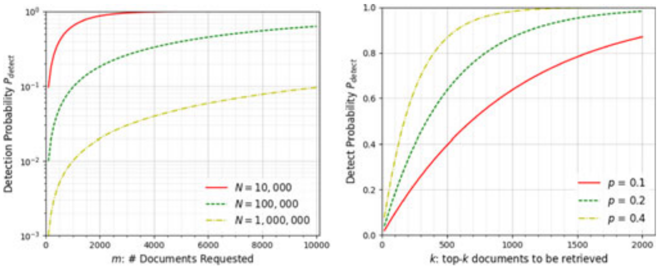
The authentication tags of the document index and the query are multiplied to obtain the final result $\sigma_i = (y_0, y_1, y_2)$. The $\mathsf{KeywordSearch}$ is basically a multiplication of two polynomials defined by the authenticated tags of $\widetilde{D}_i$ and $\widetilde{Q}$.

5.3 Support for Top-k Verification

Usually the client is only interested in receiving k most relevant results for his search query so as to save communication cost. However, the scheme described above achieves verifiability for every single similarity score, not for top-k search results. The cloud server may return incorrect top-k results either because it wants to save energy or the cloud computing system goes faulty. For instance, the cloud server only searches a part of the document set and returns the top-k search results.

It is not easy for the client to discover that since the client does not know all similarity scores computed by the cloud server. To solve this problem, we propose the random challenge technique to achieve verifiability for VPSearch.

Random Challenge. A natural way to achieve verifiability for top-k results is for the client to verify similarity scores of some documents randomly selected by the client. Only if all



(a) P_{detect} for $p = 0.1$ and $k = 100$ (b) P_{detect} for $m = 1000$ and $N = 100,000$

Fig. 4. P_{detect} as a function of the number of randomly selected documents m, N, p and k.

these documents have smaller similarity scores than those of the top-k results returned by the cloud server, can the client be sure that the returned k results are truly top-k for some probability.

Suppose the client $\mathcal{C}$ has outsourced N documents to the cloud, and the client expects to receive top-k results with the highest scores with respect to a query Q for some $k \ll N$. $\mathcal{C}$ randomly chooses m documents and requests for their similarity scores from the cloud server $\mathcal{S}$. Thus the probability of $\mathcal{C}$ detecting a fraction p of top-k results are not returned by $\mathcal{S}$ can be computed as follows:

$$P_{\text{detect}} = 1 - \left(1 - \frac{pk}{N-k}\right)^m. \tag{10}$$

When $pk \ll N - k$, the detection probability is close to $\frac{pkm}{N}$. Given k, N and p, the client can control trade-off between the detection probability P_{detect} and communication cost $O(m)$. With Fig. 4a we show the relationship between the probability of detection P_{detect} and the number of randomly selected documents m for $k = 100$ and $p = 0.1$. For instance, given $N = 10{,}000, k = 100$ and $p = 0.1$, one can choose $m = 100$ resulting in $P_{\text{detect}} = 9.6$ percent; or one chooses $m = 1{,}000$ so that $P_{\text{detect}} = 63.6$ percent.

Fig. 4b shows the relationship between the probability of detection P_{detect} and k of top-k retrieval for $N = 100{,}000$ and $m = 1{,}000$. It can be seen that the detection probability increases as the number of returned results k increases. It is highly possible that misbehaviour of the cloud server is undetected for small k.

Random Challenge with Ordering. As the above detection probability of incorrect top-k results is too low for large N (though it can detect incorrect top-k results returned by a malicious cloud server), we propose the following enhancement called random challenge with ordering.

First, after the cloud server has searched all documents for the client, the cloud server sorts the resulted similarity scores in descending order. The sorting operation will incur an overhead of complexity $\mathcal{O}(N \log N)$. Then the sorted list, along with the top-k results, is returned to the client.

Next, the client challenges the cloud server with m randomly selected documents, and the cloud server returns the similarity scores with authentication tags to the client.

Finally, the client verifies: 1) the similarity scores of the m documents are correct; 2) all the m similarity scores are less than that of the top-k results; 3) the ordering of the m similarity scores is correct. If all verifications are successful, the client accepts the top-k results.

Suppose the cloud server calculates similarity scores for only a fraction q of the N outsourced documents. For the remaining $(1-q)N$ documents, the cloud server decides not to compute their similarity scores. Then the cloud server needs to compose a sorted list of documents in descending order with only qN similarity scores. Without knowing other $(1-q)N$ similarity scores, the cloud server can only return a random permutation of the $(1-q)N$ documents. When the client challenges the cloud server for similarity scores of m random documents, the client would detect that the top-k results are incorrect with the following probability:

$$P_{\text{detect}} = 1 - \frac{1}{P_{(1-q)m}^{(1-q)m} \cdot C_{qm}^{m}} = 1 - \frac{(qm)!}{m!},$$

where P_n^m denotes permutation and C_n^m denotes combination.

Note the above probability is independent of N and k. When $m = 20$ and $q = 0.9$, meaning that the cloud server has honestly searched 90 percent documents, the detection probability is $P_{\text{detect}} = 99.7$ percent. So it is a significant improvement over the random challenge approach without ordering. If a partially honest cloud server has calculated all similarity scores, it will returns the correct top-k results to the client as this is in its best interest.

6 Analysis and Discussion

6.1 Security of RealHomMAC

VPSearch employs the homomorphic MAC technique RealHomMAC adapted from HomMAC in [23], thus it has similar features and security properties, namely authentication correctness, evaluation correctness, succinctness and authenticator security.

Authentication correctness means that the authentication tag output by Auth correctly authenticates the data m under the corresponding label. Evaluation correctness means that if $\vec{\sigma}$ authenticates data $m_1, \ldots, m_n$, then σ authenticates the the output of $f(m_1, \ldots, m_n)$. Succinctness means the size of tag σ is bounded by a polynomial in the security parameter λ, and is independent from the number of inputs or the size of circuit f. Authenticator security means the homomorphic MAC scheme is secure against forgery.

First of all, RealHomMAC satisfies authentication correctness and evaluation correctness like HomMAC. This can be readily proved in the same way as in [23]. Second, RealHomMAC also satisfies succinctness as the size of the resulting authentication tag is independent of the number of inputs of the evaluated function.

Security of RealHomMAC can be argued intuitively. As described in Section 3, the cloud server knows only coefficients of a set of polynomials, i.e., $(m_i, (r_i - m_i)/\alpha)$ where r_i and α are unknown to the cloud server. Hence, there are $n+1$ unknowns while the cloud server can only establish an equation system of n linear equations. So it is impossible for the cloud server to solve the linear equation system if r_i is generated from a pseudorandom function and α is a random secret.

The main difference between our homomorphic MAC technique and HomMAC is that RealHomMAC deals with real numbers while HomMAC deals with integers in finite fields. Therefore, our MAC scheme is expected to have similar security property as HomMAC.

However, RealHomMAC has overflow problems which can have impact on its security. Suppose we adopt the double precision floating number format defined in IEEE 754 standard. A double-precision floating number of 64-bit size ranges from $-(2-2^{-52}) \cdot 2^{1023}$ to $(2-2^{-52}) \cdot 2^{1023}$, with 11 bits for the exponent and 52 bits for the fraction. To prevent overflow problems in VPSearch, we can restrict the largest values in the matrices and vectors. According to VPSearch, the largest possible value comes from y_1 in Equation (6). For dictionary size $n = 100{,}000$ and the largest double-precision floating number allowed M, $y_1 \leq 4n^2M^4 = 4 \cdot 10^{10} \cdot (M)^4 \leq 2^{1023}$. Solving this inequation, we get $M \leq 2^{246}$. So the exponent part of the floating number should be less than 246, i.e., only 8 bits (1 as the exponent sign) instead of 11 bits can be freely used for the exponent. That is, for l-bit floating numbers, the actual security of VPSearch is $\lambda = l - 3$ bits.

Formally, security of RealHomMAC can be obtained from the following theorem:

Theorem 6.1. RealHomMAC *is a secure homomorphic MAC scheme that is secure against forgeries if $F_K()$ is a pseudorandom function.*

The above theorem can be rigorously proved in the same way as in [23], and it is omitted to avoid repetition.

6.2 Verifiability

Verifiability achieved by VPSearch is the key to motivate a partially honest cloud server to act honestly. For a partially honest cloud service provider, it tries its best to reduce energy cost while maximizing revenue. If there is no verification mechanism in the keyword search algorithm, then the cloud server might simply return a wrong result without doing any computation.

Verifiability of VPSearch directly follows from the evaluation correctness and authenticator security property of VPSearch. With evaluation correctness, VPSearch guarantees that the client can correctly verify the authentication tag produced by the cloud server. With the authenticator security property, VPSearch ensures that the client would not accept any forged authentication tag produced by a malicious cloud server.

Top-k Verifiability. With the random challenge technique with ordering, the detection probability can reach close to 1 for just tens of challenges, independent of the number of documents and k. However, we emphasize that a malicious cloud server, i.e., not partially honest, can still manage to cheat in answering top-k queries.

6.3 Efficiency

In the following analysis, n denotes the dictionary size and N is the number of documents. We analyze complexity of each algorithm in VPSearch as follows:

Setup. The main computation cost in this algorithm is inversion of two matrices, whose complexity is $\mathcal{O}(n^3)$. However, this step is one-time and can be pre-computed, so it will not have impact on runtime performance of VPSearch.

Index. The complexity of constructing a plaintext index for a document set depends on the underlying data set, so we cannot give an analytic complexity for it. For index encryption, its complexity is $\mathcal{O}(n^2N)$. This operation is also a one-time computation, and it will not have impact on runtime performance of VPSearch.

Trapdoor. Trapdoor is generated by multiplying with M_1^{-1} and M_2^{-1}, so the complexity of constructing a trapdoor for keyword search is $\mathcal{O}(n^2)$ for each query.

Auth. The cost of generating authentication tags for a document index is linear to the dictionary size and the number of documents, i.e., $\mathcal{O}(nN)$. This computation is one-time, and will not have impact on performance of VPSearch during keyword search.

To generate authentication tags for trapdoors, one needs to compute coefficients of a degree-1 polynomial for each item in each trapdoor. This computation cost is $\mathcal{O}(n)$, independent of the dictionary size.

KeywordSearch. This algorithm involves inner product of a search trapdoor and each document subindex, so its complexity is $\mathcal{O}(nN)$ for each query.

Verify. This algorithm is run by the client, and its complexity is $\mathcal{O}(nN)$ to verify search results of one query. For verification of top-k results, the random challenge technique with ordering requires the cloud server to order all search results, so its complexity is $\mathcal{O}(N \log N)$.

With regard to storage cost, the encrypted document index takes up $8nN$ bytes if we adopt 64-bit double-precision floating number format. The authentication tags takes up another $8nN$ bytes. That is, the storage required by VPSearch is twice as that of Cao et al.'s scheme.

6.4 Data/Keyword Privacy

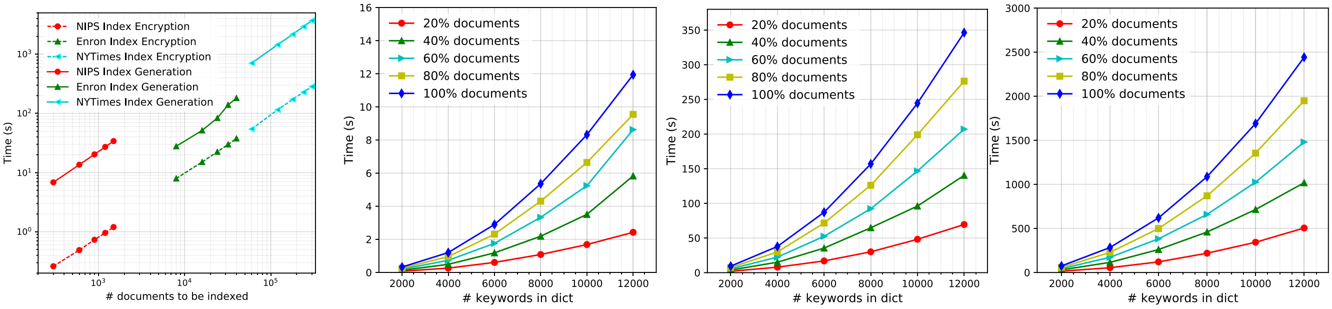
VPSearch inherits the privacy-preserving feature of Cao et al.'s scheme in [11], which has shown that this scheme can achieve data privacy, index privacy, keyword privacy and trapdoor unlinkability under the *known ciphertext* model. Under the *known background knowledge* model, simple modification can be applied to make it secure.

More specifically, it uses secret matrix multiplication to encrypt the index and the queries. Without the secret matrix M_1 and M_2, the cloud server has no way to figure out the keywords queried by the client or the index associated with a file. The similarity score is masked with secret random number r and t, which ensures privacy of the search results. As this is not the focus of our work, we refer the readers to [11] for more detailed discussion.

6.5 Unbounded Storage

Because each document subindex is independent of others, there is no restriction on the number of documents outsourced to the cloud. As a result, the client can continuously upload documents onto the cloud, and performs search over all documents later.

However, because the index is encrypted by matrix multiplication, the encryption matrices should also be extended if a new keyword is added. So if a new keyword is added, the whole index should be rebuilt. As a result, VPSearch as well as the underlying MRSE scheme does not support keyword dynamics. For the same reason, multi-keyword ranked searchable encryption schemes proposed in [12], [13], [28] do not support keyword dynamics either.



(a) Plaintext Index Generation and Index Encryption

(b) Index Encryption for NIPS

(c) Index Encryption for Enron

(d) Index Encryption for NYTimes

Fig. 5. Time cost for index construction.

7 Performance Evaluation

To demonstrate performance of the proposed scheme VPSearch, we have implemented a prototype of VPSearch using Matlab. Then a set of experiments are conducted to evaluate its performance. All experiments are conducted on a laptop equipped with an Intel Core i7-6560U CPU (2-core 2.2 GHz) and 16 GB RAM running 64-bit Ubuntu 16.04. We use 64-bit Matlab R2015b for Linux to implement VPSearch and python to process the document indices.

We evaluate VPSearch's performance on three UCI bag-of-words data sets:

We pad the Enron data set to 40,000 documents and remove meaningless keywords of the NYTimes data set. These three data sets are chosen as they have different document numbers and the performance over them can provide insights of our proposed scheme. We conduct comprehensive experiments over dictionaries with different sizes and different number of documents from these data sets.

As VPSearch is designed by integrating the homomorphic MAC technique with Cao et al.'s scheme in [11], we compare the its performance with that of VPSearch.

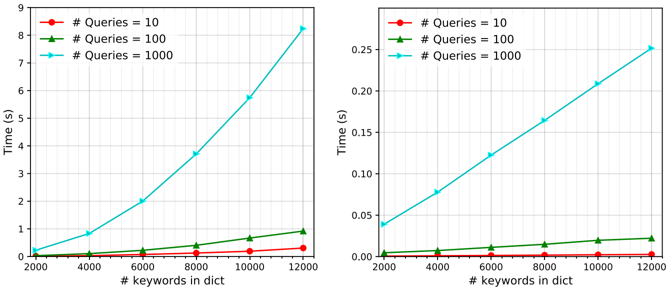
7.1 Index Construction

The index construction is the most time-consuming step in VPSearch, but note that this step is executed only once.

Fig. 5 shows the time cost for index construction. Fig. 5a illustrates the time for index construction when the dictionary size is 4,000. The solid lines in Fig. 5a represent the time cost of plaintext index generation. The plaintext index generation is done by scanning the bag-of-words format files with a python program. To save storage space and speed up processing time, we use bit array to construct the indices. The dashed lines in Fig. 5a represent the time for index encryption. For each data set, we obtain the computation time for 20, 40, 60, 80 and 100 percent of the documents in the data set.

It can be seen that the time cost for either plaintext index generation or index encryption is linear to the number of documents when the dictionary size is fixed. When the dictionary size is 4,000, the time to construct plaintext document index is around 34 seconds for 1,500 documents (NIPS), 181 seconds for 40,000 documents (Enron), and 3,693 seconds for 300,000 documents (NYTimes); the time to encrypt document index is around 1.2 seconds for 1,500 documents (NIPS), 40 seconds for 40,000 documents (Enron), and 300 seconds for 300,000 documents (NYTimes).

Figs. 5b, 5c, and 5d show the time cost for index encryption for all three data sets respectively. The time cost is approximately linear to the dictionary size, i.e., the number of keywords in the dictionary.[2] Although it takes about 2,500 seconds to encrypt the entire document index for NYTimes, this time cost is acceptable for a one-time operation. Note that the index generation/encryption of VPSearch is essentially the same as that of Cao et al.'s scheme [11].



(a) Trapdoor Generation

(b) Trapdoor Authentication

Fig. 6. Time cost for trapdoor generation and authentication.

7.2 Trapdoor Generation and Authentication

Fig. 6 illustrates the time cost of trapdoor generation and authentication respectively.

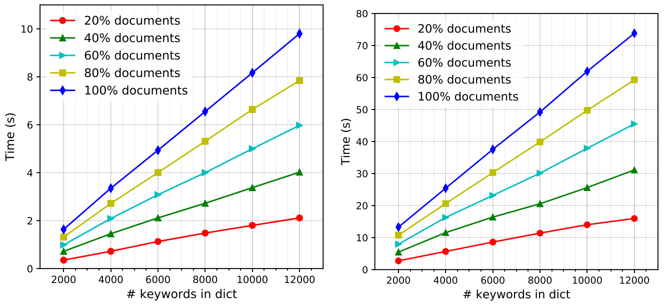
According to Fig. 6a, the time cost of trapdoor generation exhibits quadratic growth with respect to dictionary size. It costs only 8.24 seconds to generate 1,000 trapdoors for dictionary size 12,000. Fig. 6b shows that the trapdoor authentication time is almost strictly linear to dictionary size and the number of queries. It costs only 0.252 seconds to authenticate 1,000 trapdoors for the dictionary with 12,000 keywords, which is about 30 times faster than trapdoor generation. VPSearch and Cao et al.'s scheme have the same trapdoor generation operation, so the above results mean that trapdoor authentication incurs about 3 percent overhead on top of Cao et al.'s scheme.

7.3 Authenticating Data Set Index

Fig. 7 shows the time cost for generating authentication tags for the document indices of Enron and NYTimes.

From Figs. 7a and 7b it can be seen that the time cost is roughly linear to the size of dictionary and the number of

[2] In our experiments, dictionary size ranges from 2,000 to 12,000. For larger dictionary sizes, the computation cost is too much for our laptop.



(a) Authenticating the Index of Enron (b) Authenticating the Index of NYTimes

Fig. 7. Time cost for authenticating document indices.

documents to be authenticated. VPSearch takes 9.80 seconds and 73.81 seconds to generate authentication tags for all documents in Enron (40,000 documents) and NYTimes (300,000 documents), respectively. The time cost of authenticating index is less than 3 percent of that of index encryption operation (cf. Figs. 5c and 5d), which means index authentication of VPSearch adds only 3 percent overhead compared to Cao et al.'s scheme.

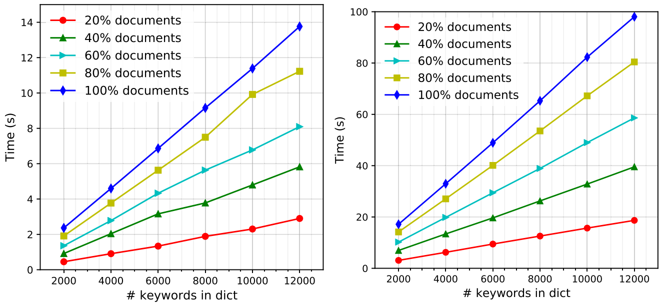
7.4 Keyword Search

Fig. 8 shows the time cost for keyword search for 100 randomly generated queries over Enron and NYTimes. In accordance with our analysis, the cost of keyword search is linear to the dictionary size and the number of documents to be searched. As showed in Figs. 8a and 8b, it costs 13.76 seconds to finish 100 searches over all 40,000 documents in Enron, and 98 seconds over all documents in NYTimes.

The keyword search complexity of VPSearch is four times as that of Cao et al.'s scheme [11], since VPSearch operations on two coefficients of degree-1 polynomials while Cao et al.'s scheme works directly on real numbers. Note that the keyword search operation can be executed on multiple servers in parallel to reduce time.

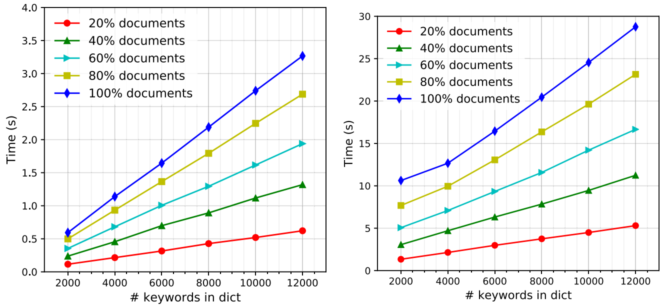
7.5 Verification

In Fig. 9 we show the verification cost for Enron and NYTimes for 100 randomly generated queries. In accordance with our analysis, the verification cost is roughly linear to the size of dictionary and the number of results (searched documents). As showed in Figs. 9a and 9b, it costs 3.27 seconds to verify results of 100 query for all 40,000 documents in Enron, and 28.75 seconds to verify results of 100 query for all documents in NYTimes. Note that for each query, there are N similarity scores to be verified where N is the number of documents.



(a) Keyword Search over Enron (b) Keyword Search over NYTimes

Fig. 8. Time cost for 100 keyword search queries).



(a) Search Verification for Enron (b) Search Verification for NYTimes

Fig. 9. Time cost for verification of 100 queries.

Normally, the client only requests top-k results, so the verification cost will be significantly less than our results.

7.6 Runtime Performance

We compare the runtime performance of VPSearch with that of Cao et al.'s scheme in [11] for 100 queries over the entire Enron data set in Fig. 10, which includes time costs of trapdoor generation, trapdoor authentication, keyword search and verification. Time costs of index construction and index authentication are excluded as these two operations can be pre-computed.

Fig. 10 shows that the performance of VPSearch at runtime is roughly three times slower than that of Cao et al.'s scheme. The main cost comes from keyword search, which can be executed on multiple servers in parallel to reduce time. Nevertheless, VPSearch is still efficient as the total time cost is less than 18 seconds for 100 queries over the entire Enron.

8 Conclusion and Future Work

In this paper we have proposed the first verifiable privacy-preserving multi-keyword search scheme for cloud computing. Our scheme is implemented by applying an efficient

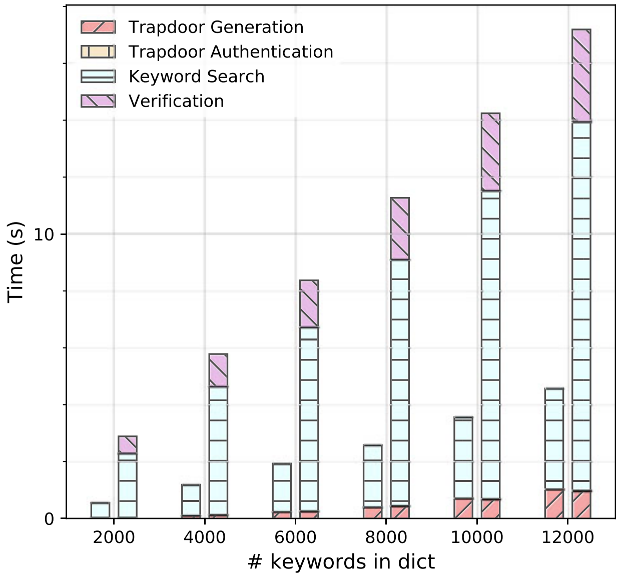


Fig. 10. Runtime performance of VPSearch and comparison with Cao et al.'s scheme [11] for 100 queries over the entire Enron. The left bars are for Cao et al.'s scheme, and the right bars are for VPSearch.

homomorphic MAC scheme on a privacy-preserving multi-keyword scheme, and we have made necessary modifications to the homomorphic MAC scheme so that it supports privacy-preserving multi-keyword search. We also provide a verification technique called random challenge with ordering for top-k search results. We have analyzed security of our scheme and showed that it fulfills the requirement of verifiability, efficiency, data/keyword privacy, and unbounded storage.

However, VPSearch is not efficient for large-scale databases since the cloud needs to search through the whole database, which is very inefficient. Our future work in this line will be enhancements for efficient verification for large-scale outsourced data.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 61370027.

References

[1] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, Jan./Feb. 2012.

[2] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, 2000, pp. 44–55.

[3] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proc. 3rd Int. Conf. Appl. Cryptography Netw. Secur.*, 2005, pp. 391–421.

[4] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1467–1479, Aug. 2012.

[5] P. Golle, J. Staddon, and B. R. Waters, "Secure conjunctive keyword search over encrypted data," in *Proc. 2nd Int. Conf. Appl. Cryptography Netw. Secur.*, 2004, pp. 31–45.

[6] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. 4th Conf. Theory Cryptography*, 2007, pp. 535–554.

[7] Y. Hwang and P. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Proc. 1st Int. Conf. Pairing-Based Cryptography*, 2007, pp. 2–22.

[8] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," in *Proc. 4th Conf. Theory Cryptography*, 2009, pp. 457–473.

[9] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2009, pp. 139–152.

[10] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proc. IEEE INFOCOM*, 2011, pp. 829–837.

[11] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.

[12] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2546–2559, Sep. 2016.

[13] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 12, pp. 2706–2716, Dec. 2016.

[14] Z. Fu, X. Sun, S. Ji, and G. Xie, "Towards efficient content-aware search over encrypted outsourced data in cloud," in *Proc. IEEE INFOCOM*, 2016, pp. 1–9.

[15] K.-M. Chung, Y. Kalai, and S. P. Vadhan, "Improved delegation of computation using fully homomorphic encryption," in *Proc. 30th Annu. Conf. Advances Cryptology*, 2010, pp. 483–501.

[16] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proc. 30th Annu. Conf. Advances Cryptology*, 2010, pp. 465–482.

[17] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *Proc. IEEE Symp. Secur. Privacy*, 2013, pp. 238–252.

[18] B. Parno, M. Raykova, and V. Vaikuntanathan, "How to delegate and verify in public: Verifiable computation from attribute-based encryption," in *Proc. 9th Int. Conf. Theory Cryptography*, 2012, pp. 422–439.

[19] D. Fiore and R. Gennaro, "Publicly verifiable delegation of large polynomials and matrix computations with applications," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 501–512.

[20] S. Setty, R. McPherson, A. Blumberg, and M. Walfish, "Making argument systems for outsourced computation practical (sometimes)," in *Proc. 19th Annu. Netw. Distrib. Syst. Secur. Symp.*, 2012.

[21] S. Benabbas, R. Gennaro, and Y. Vahlis, "Verifiable delegation of computation over large datasets," in *Proc. 31st Annu. Conf. Advances Cryptology*, 2011, pp. 111–131.

[22] R. Gennaro and D. Wichs, "Fully homomorphic message authenticators," in *Proc. 19th Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2013, pp. 301–320.

[23] D. Catalano and D. Fiore, "Practical homomorphic MACs for arithmetic circuits," in *Proc. 17th Int. Conf. Practice Theory Public-Key Cryptography*, 2013, pp. 336–352.

[24] M. Backes, D. Fiore, and R. M. Reischuk, "Verifiable delegation of computation on outsourced data," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2013, pp. 863–874.

[25] D. Boneh and D. M. Freeman, "Homomorphic signatures for polynomial functions," in *Proc. 30th Annu. Int. Conf. Theory Appl. Cryptographic Techn.: Advances Cryptology*, 2011, pp. 149–168.

[26] D. Fiore, R. Gennaro, and V. Pastro, "Efficiently verifiable computation on encrypted data," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 844–855.

[27] UCI Machine Learning Repository, "Bag of words data set," (2008). [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Bag+of+Words

[28] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 340–352, Feb. 2016.

[29] R. Johnson, D. Molnar, D. X. Song, and D. Wagner, "Homomorphic signature schemes," in *Proc. Cryptographer's Track RSA Conf. Topics Cryptology*, 2002, pp. 244–262.

[30] B. Libert, T. Peters, M. Joye, and M. Yung, "Linearly homomorphic structure-preserving signatures and their applications," in *Proc. of 33rd Annu. Cryptology Conf.*, 2013, pp. 289–307.

[31] D. Catalano, A. Marcedone, and O. Puglisi, "Authenticating computation on groups: New homomorphic primitives and applications," in *Proc. 20th Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2013, pp. 193–212.

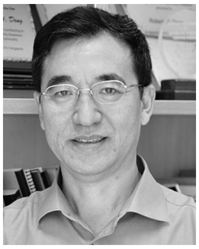
[32] C. Joo and A. Yun, "Homomorphic authenticated encryption secure against chosen-ciphertext attack," in *Proc. 20th Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2013, pp. 173–192.

[33] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, "Achieving efficient cloud search services: Multi-keyword ranked search over encrypted cloud data supporting parallel computing," *IEICE Trans. Commun.*, vol. E-98.B, no. 1, pp. 190–200, Jan. 2015.

[34] W. Sun, et al., "Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 3025–3035, Nov. 2014.

[35] R. Cheng, J. Yan, C. Guan, F. Zhang, and K. Ren, "Verifiable searchable symmetric encryption from indistinguishability obfuscation," in *Proc. 10th ACM Symp. Inf. Comput. Commun. Secur.*, 2015, pp. 621–626.

Zhiguo Wan received the BS degree in computer science from Tsinghua University, in 2002, and the PhD degree from the School of Computing, National University of Singapore, in 2007. He is an associate professor in the School of Computer Science and Technology, Shandong University, Jinan, China. His main research interests include applied cryptography, privacy enhancing techniques, and system security. He was a faculty member in the School of Software, Tsinghua University, and a postdoc with K.U. Leuven, Belgium, during 2006 to 2008. He has served in program committees in dozens of international conferences.



Robert H. Deng is a professor of information systems, director of Secure Mobile Center, and dean of Postgraduate Research Programme, Singapore Management University. His research interests include data security and privacy, multimedia security, network, and system security. He has served/is serving on the editorial boards of many international journals in security, such as the *IEEE Transactions on Information Forensics and Security*, the *IEEE Transactions on Dependable and Secure Computing*, and the *IEEE Security & Privacy*. He is the chair of the Steering Committee of the ACM Asia Conference on Computer and Communications Security (ASIACCS). He received the University Outstanding Researcher Award from the National University of Singapore in 1999 and the Lee Kuan Yew Fellow for Research Excellence from the Singapore Management University in 2006. He was named Community Service Star and Showcased Senior Information Security Professional by (ISC)^2 under its Asia-Pacific Information Security Leadership Achievements program in 2010. He is a fellow of the IEEE.