IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 29, 2019, accepted September 9, 2019, date of publication September 26, 2019, date of current version October 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2943971

Efficient Verifiable Multi-Key Searchable Encryption in Cloud Computing

YAPING SU[1], JIANFENG WANG[1], YUNLING WANG[1], AND MEIXIA MIAO[2]
[1]State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an 710126, China
[2]National Engineering Laboratory for Wireless Security, Xi'an University of Posts and Telecommunications, Xi'an 710121, China

Corresponding authors: Jianfeng Wang (jfwang@xidian.edu.cn) and Meixia Miao (miaofeng415@163.com)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802202, in part by the National Natural Science Foundation of China under Grant 61702401, Grant 61572382, and Grant 61902315, in part by the Fundamental Research Funds for the Central Universities under Grant XJS17053, in part by the National Cryptography Development Fund under Grant MMJJ20180110, and in part by the China 111 Project under Grant B16037.

ABSTRACT The notion of Multi-Key Searchable Encryption (MKSE) enables data owners to outsource their data into a cloud server, while supporting fine-grained data sharing with the authorized users. Note that the traditional MKSE is vulnerable to data leakage. That is, the malicious data owner may collude with the server and recover the search queries of authorized users. Recently, Hamlin et al. (PKC'18) presented a new MKSE construction that can ensure data privacy between data owner and authorized users, where the share key is generated depending on data owner, authorized user and the specific document. However, their scheme cannot support verifiable search in the case of the malicious cloud server. In this paper, we propose a new verifiable MKSE (VMKSE) scheme by leveraging Garbled Bloom Filter, which can simultaneously support verifiability of search result and secure data sharing in multi-user setting. Compared to the state-of-the-art solution, the proposed scheme is superior in efficiency and verifiability. The experiment results demonstrate the efficiency of our scheme.

INDEX TERMS Searchable encryption, garbled bloom filter, verifiable search.

I. INTRODUCTION

Cloud computing enables client to enjoy high-quality data storage and computing services in a pay-as-you-go manner. With the popularity of cloud computing, a growing number of organizations and individuals prefer to outsource their massive data to the cloud server. Despite its tremendous benefits, outsourcing data to a remote server brings some security issues [1]–[3]. Very recently, it is reported that the famous social networking company Facebook exposed more than 540 million users' personal information, including users Facebook IDs, account names, and their activities [4].

A traditional encryption method can protect the confidentiality of the outsourced data. However, it brings difficulties to search over the encrypted data [5]. Searchable Encryption (SE), as a promising solution, has drawn great attention in both academic and industrial community, which allows a data owner to outsource his encrypted documents to a cloud server while retaining search ability. Specifically, the data owner encrypts his documents with his private key and generates a search index, then outsources them to the cloud server. Upon receiving a search token from a user, the server performs search over the search index and finds out the matched results. There is a rich research on both Searchable Symmetric Encryption (SSE) schemes [6]–[10] and Public-Key Encryption with Keyword Search (PEKS) schemes [11]–[13]. These SSE schemes can be adapted to a variety of scenarios in the cloud environment. Most of previous SSE schemes mainly focus on keyword search, including single-keyword search and multi-keyword search. These keyword search schemes are all designed in the single-user or multi-user setting.

In single-user SSE [8], [14]–[16], the data owner outsources the encrypted documents and later only allows himself to perform search. While in multi-user SSE [17]–[21], the data owner outsources the encrypted documents and selectively shares them with a group of authorized users. In this case, any authorized user needs provide valid search tokens to the server and get all the matched documents encrypted with different keys. Obviously, the number of

The associate editor coordinating the review of this manuscript and approving it for publication was Chien-Ming Chen.

search tokens are linearly related to the number of documents to search. This can lead to inefficient searching for a large number of documents.

In Multi-Key Searchable Encryption (MKSE) schemes, multiple data owners share documents encrypted with their own different data keys which allows any authorized user to provide the server with a search token of a single keyword, but still allows the server to search across these documents encrypted with different shared with them. The size of the search token is independent of the number of documents. PoPa *et al.* [37] presented the first MKSE scheme which generates a share key for each authorized data user. The share key is used to transform a query under a specific user's key into the one under the data owner's key. Thus the server can perform search in documents encrypted with different keys using the transformed query. However, Grubbs *et al.* [22] and Van Rompay *et al.* [23] later that the first MKSE scheme suffers from query leakage when the malicious data owner colludes with the cloud server. The reason is that the transformed query can be used to search over any outsourced documents. As a result, the data owner can collude with the cloud server to launch an offline attack against a given keyword to recover the user's query. Very recently, Hamlin *et al.* [24] proposed a new MKSE scheme which can prevent the above attack. The share key in this scheme is related to not only the keys of data owner and data user, but also the shared documents, which makes that the cloud server can only perform search over the shared documents instead of arbitrary one.

As to the scheme of [25], which can also prevent the above attack. More specifically, to resist the threat of the malicious data owner colluding with the cloud server, the scheme uses two non colluding servers. Note that each server has a portion of the information that would be held by a single server. One server is mainly responsible for transforming trapdoors, another server is responsible for storing encrypted indexes and searching for related indexes based on the transformed trapdoors. To achieve strong security, namely response unlinkability, Oblivious Transfer (OT) technique is used between the two servers in search protocol, but the rounds of interaction between the two servers are frequent. In addition, the implementation of the scheme is more complicated, and the computational cost between the two servers is large (since several logarithmic and exponential operations are involved). Notice that all of the aforementioned schemes are designed in an honest but curious server model, in which the server performs all search operations honestly and returns all search result correctly.

However, in practice, the cloud server is also likely to be malicious. That is, it may well return incorrect or incomplete search results. The reason comes from the software and hardware failure, or the server's selfish behavior, such as saving the bandwidth or computation resources. To fight against a malicious server, verifiable SSE has received considerable attention [26]–[32]. Chai and Gong [26] constructed the first verifiable SSE scheme based on the character tree. Kurosawa and Ohtaki [27] constructed a verifiable SSE scheme based on MAC against non-adaptive adversaries, which can verify the correctness of the search result. Sun *et al.* [29] designed a verifiable SSE scheme based on bilinear-map accumulator. However, it cannot provide a reasonable proof when the server returns an empty set. Similarly, Wang *et al.* [32] presented a verifiable SSE scheme based on the accumulator in 2018. Liu *et al.* [28] presented a verifiable SSE scheme based on the approach of key aggregation and the authenticated data structure Bloom Filter under the multi-owner setting, and their scheme can execute even if when the server returned an empty set. However, since the verification phase involves multiplication, exponentiation and pairing operations, the expensive operations lead to the verification inefficiency. Recently, Liu *et al.* [33] presented a multi-user verifiable SSE against non-adaptive adversaries, which allows multi-user to perform search and validate the integrity of search result. Unfortunately, the verification time is linear with the number of documents shared and the number of documents returned by the server.

To our best knowledge, how to construct an efficient verifiable MKSE (VMKSE) scheme which can simultaneously ensure verifiability of search result and support multi-user search remains a great challenge problem.

A. OUR CONTRIBUTION

In this paper, we propose a VMKSE scheme, the major contributions of our proposed scheme are summarized as follows:

- We present a new VMKSE scheme based on Garbled Bloom Filter, which can ensure verifiability of search result even if the server deliberately returns an empty set. The proposed scheme can support efficient verifiable search using only symmetric operation (i.e., XOR and hash operations).
- We formally define a security model and make a rigorous security and efficiency analysis for the proposed scheme, which can well achieve the desired security goals with a comparable computational cost. Besides, we provide an implementation on a real-world email dataset, the experiment results show that our scheme is highly efficient.

B. ORGANIZATION

The rest of this paper is organized as follows. Some necessary preliminaries applied in the proposed scheme are given in Sect II. The concrete construction is presented in Sect III. Section IV provides the detailed security and efficiency analysis. Section V describes the performance evaluation. Conclusion is described in Sect VI.

II. PRELIMINARIES

In this section, we firstly give some necessary notations (as TABLE 1 shows) and a brief revisit on Garbled Bloom Filter. Then, system and threat model is given, and security definition of the proposed scheme is presented.

TABLE 1. Notations.

Notations	Meaning
d	The plaintext document
W_d	The set of all keywords in document d
T	The ciphertext for each keyword w in W_d
w_i	The ith keyword in set W_d
W	The set of different keywords in all shared documents
$DB[w]$	The set of document-specific random identifiers containing the keyword w
m	The length of GBF
λ	Security parameter

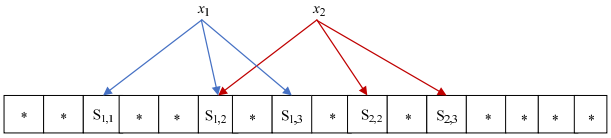


FIGURE 1. Add elements into a Garbled Bloom Filter.

A. GARBLED BLOOM FILTER

In 2013, Dong *et al.* [34] proposed a new variant of the Bloom Filter (BF, see [35] for details), called Garbled Bloom Filter (GBF). GBF is a data authentication structure which is used to check whether an element belongs to a set S. A GBF consists of m numbers of λ-bit strings and k independent hash functions $H = \{h_1, \ldots, h_k\}$, where $h_i : \{0,1\}^* \rightarrow [1, m]$, $1 \leq i \leq k$. Initially, all the locations of GBF are set to NULL. To insert an element x into the GBF, we determine k locations based on the k hash functions. The strings filled in the k locations satisfy that $GBF[h_1(x)] \oplus \ldots \oplus GBF[h_k(x)] = x$, where $GBF[j]$ denotes the strings at location j. After locating all the elements in S, each unoccupied location stores a random string.

Given an element y, the steps for checking whether it belongs the set S as follows. We first determine k locations based on the k hash functions. Then we XOR all the strings at the k locations and obtain a value y'. Finally we check whether the two values y and y' are equal. If yes, it indicates that $y \in S$. Otherwise, $y \notin S$.

A simple example is shown in FIGURE 1. In the initial phase, we set all the locations in the GBF to NULL (expressed as '*'). Then, we will insert elements x_1, x_2 into the GBF. First, we insert x_1 as 3 shares $S_{1,1}, S_{1,2}, S_{1,3}$ at 3 locations. Second, when inserting element x_2, we find that $GBF[5]$ has been occupied by the share $S_{1,2}$, so we reuse $S_{1,2}$ as a share of x_2. In this case, we set $x_2 = S_{1,2} \oplus S_{2,2} \oplus S_{2,3}$. After inserting all elements, we set a random λ-bit string to each unoccupied location.

In the following, we give two different probabilities of the GBF, collision probability and maximum false positive probability. The collision probability refers to the probability when $y \notin S$, but it hashes to the same locations for some $x \in S$. The collision probability can be expressed as: $Pr|(\oplus_{i=0}^{i=k-1} GBF[h_i(y)]) = x| \leq \epsilon$, where ϵ is the maximum false positive probability of BF (see [35] for details). Note that, a collision does not lead to false positive probability, but it reveals x. The false positive probability of a GBF is the probability when $y \notin S$ but the recovered strings equals to y accidentally. The false positive probability can be expressed as: $Pr|(\oplus_{i=0}^{i=k-1} GBF[h_i(y)]) = y| \leq 2^{-\lambda}$.

Very recently, Van Rompay and Önen [36] pointed out the proof of security of GBF in [34] is flawed and gave a new formal proof. In particular, in the security analysis of GBF, Van Rompay et al. ensure that an adversary obtains the same information from a random string in unoccupied location as from a fixed value. In our scheme, in order to meet certain functional requirements, we modify the original GBF to get a variant GBF, the security of the variant GBF is also based on the new proof in Sect.5.2 in [36].

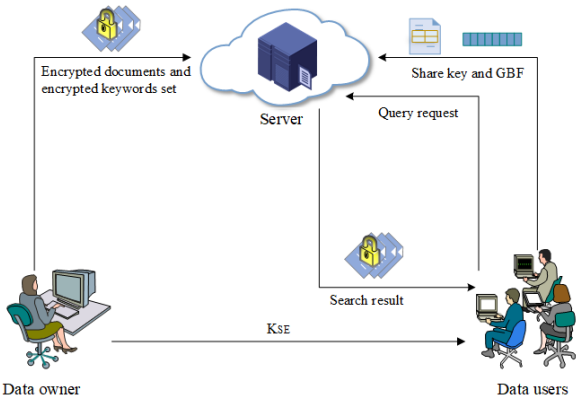


FIGURE 2. System for our model.

B. SYSTEM AND THREAT MODEL

In this paper, as shown in FIGURE 2, we consider three different entities: data owners, data users and the server.

- Data owners: The data owners have a set of documents and wish to share them with a group of authorized data users. It requires the data owners to encrypt the document and all keywords contained in that document, and then outsources encrypted document alongside with the encrypted set of keywords to a cloud server. However, the data owner is malicious, he may collude with the cloud server and try to recover the data user's query privacy.
- Data users: A data user wants to access the outsourced data on the server. To support efficient retrieval of the encrypted documents stored on the server, the data user generates the share key. To support verifiability of the search results by the cloud server, the data user generates the auxiliary information GBF. To search the encrypted documents for a keyword, data user uses his query key to generate his token and send it to the server. Notice that the data user performs the above operations honestly.
- Server: The server is mainly responsible for storing encrypted data, share key, the auxiliary information GBF and providing retrieval services for data users. Assume the server is malicious, it may return incomplete and/or incorrect search result deliberately.

C. DEFINING VERIFIABLE MULTI-KEY SEARCHABLE ENCRYPTION

We present the notion of VMKSE scheme in this section. An MKSE scheme permits data owners to share documents with authorized data users, and the data users can query those documents using their own query keys while protecting both data and query privacy. Furthermore, a VMKSE scheme allows data users to verify the integrity of search result. Precisely, a VMKSE scheme composes of five polynomial-time algorithms $\prod$ = (**Setup**, **Share**, **TokenGen**, **Search**, **Verify**).

Definition 1 (VMKSE): A tuple $\prod$ = (***Setup***, ***Share***, ***TokenGen***, ***Search***, ***Verify***) *of polynomial-time algorithms is a VMKSE scheme for a universal set* $\mathcal{U}$*, if the following holds.*

- $(K_{\mathrm{SE}}, T) \leftarrow$**Setup**$(1^{\lambda}, d)$: Data owner runs the algorithm to generate data key K_{SE}. The algorithm takes a security parameter 1^{λ} and the document d as input and outputs a symmetric data key K_{SE} and the encrypted set T. The data owner encrypts d with K_{SE} to generate the encrypted document C. T is obtained by encrypting each keyword in d. Then, the data owner uploads (C, T) to the server. If the data owner wishes to share the document d with an authorized user, he will send K_{SE} to the user.
- $(K_{\mathrm{PRF}}, \Delta = (\{r_1, \ldots, r_n\}, D), GBF) \leftarrow$**Share**$(\{K_{\mathrm{SE}_i}\}_{i=1}^{n}, \{T_i\}_{i=1}^{n}, GBF)$: Data user runs the algorithm to generate a query key K_{PRF}, a share key $\Delta = (\{r_1, \ldots, r_n\}, D)$ and an authenticated structure GBF. The algorithm takes as input the set of keys $\{K_{\mathrm{SE}_i}\}_{i=1}^{n}$ and the encrypted set $\{T_i\}_{i=1}^{n}$, and outputs the query key K_{PRF}, the share key $\Delta = (\{r_1, \ldots, r_n\}, D)$ and GBF. Each data user has a query key K_{PRF}, which is used to generate secure tokens. Finally, the data user uploads $(\Delta = (\{r_1, \ldots, r_n\}, D), GBF)$ to the server.
- $q \leftarrow$**TokenGen**(K_{PRF}, w): Data user runs the algorithm to generate a secure search token. The algorithm takes as input the key K_{PRF} and the keyword w, and outputs a token q of keyword w.
- $(proof, IdSet) \leftarrow$**Search**$(\Delta = (\{r_1, \ldots, r_n\}, D), q, \{T_i\}_{i=1}^{n}, GBF)$: The server runs the algorithm to search for matched documents. The algorithm takes as input the share key Δ, the token q, encrypted sets $\{T_i\}_{i=1}^{n}$ and GBF, and outputs the *proof* and the set of identifiers $IdSet$.
- True/False $\leftarrow$**Verify**$(q, proof, IdSet)$: Data user runs the algorithm to test whether the server is malicious. The algorithm takes as input a token q, *proof* and a set of identifiers $IdSet$. Finally, it outputs True if the test pass, otherwise outputs False.
- **Correctness:** A VMKSE scheme is correct if for every security parameter $\lambda \in \mathbb{N}$ and every keyword $w \in \mathcal{U}$: For $(K_{\mathrm{SE}}, T) \leftarrow$**Setup**$(1^{\lambda}, d)$, $(K_{\mathrm{PRF}}, \Delta = (\{r_1, \ldots, r_n\}, D), GBF) \leftarrow$**Share**$(\{K_{\mathrm{SE}_i}\}_{i=1}^{n}, \{T_i\}_{i=1}^{n})$, $q \leftarrow$**TokenGen**(K_{PRF}, w), $(proof, IdSet) \leftarrow$**Search**$(\Delta = (\{r_1, \ldots, r_n\}, D), q, \{T_i\}_{i=1}^{n}, GBF)$:

  $\Pr[\text{True/False} \leftarrow$**Verify**$(q, proof, IdSet)] \geq 1 - negl$

  Where True $\leftarrow$**Verify**$(q, proof, IdSet)$ if $IdSet$ is correct and complete for keyword w, otherwise False $\leftarrow$**Verify**$(q, proof, IdSet)$.
- **Security:** A VMKSE scheme is secure if for every PPT adversary $\mathcal{A}$, it has a negl(λ) advantage in the following security game with the challenger $\mathcal{C}$.
  - ***Step 1***: Adversary $\mathcal{A}$ sends the following information to the challenger $\mathcal{C}$:
    1) A set $U = \{1, \ldots, s\}$ represents the number of data users, a set $D = \{1, \ldots, t\}$ represents the number of data owners, the corrupted data owners is represented by a subset $D_c \subseteq D$.
    2) For each $i \in D_c$, n keys $\{K_{\mathrm{SE}_z}\}_{z=1}^{n}$. For each $i \in D$, n sets of keywords $\{W_{d_z}^0\}_{z=1}^{n} \subseteq \mathcal{U}$, n sets of keywords $\{W_{d_z}^1\}_{z=1}^{n} \subseteq \mathcal{U}$, where $\left|W_{d_z}^0\right| = \left|W_{d_z}^1\right|$ for $i \notin D_c$, and $W_{d_z}^0 = W_{d_z}^1$ for $i \in D_c$. Besides, a set of share edges E. Concretely, an edge $e(e \in E)$ is denoted by (j, i) for $i \in D$ and $j \in U$ which represents data owner i shares documents with data user j, it allows j to access to those documents.
    3) For each $j \in U$, n sequences of keywords$((w_{j,1,1}^0, \ldots, w_{j,1,k_j}^0), \ldots, (w_{j,n,1}^0, \ldots, w_{j,n,k_j}^0))$ for $k_j \in \mathbb{N}$, and another n sequences of keywords $((w_{j,1,1}^1, \ldots, w_{j,1,k_j}^1), \ldots, (w_{j,n,1}^1, \ldots, w_{j,n,k_j}^1))$ for $k_j \in \mathbb{N}$, there are two constraints: for each $i \in D$, if $(j, i) \in E$, for each $1 \leq z \leq n$, for each $1 \leq l \leq k_j$, $w_{j,z,l}^0 \in W_{d_z}^0$ if and only if $w_{j,z,l}^1 \in W_{d_z}^1$. For each $1 \leq z \leq n$, for each $1 \leq l < k \leq k_j$, $w_{j,z,l}^0 = w_{j,z,k}^0$ if and only if $w_{j,z,l}^1 = w_{j,z,k}^1$.
  - ***Step 2***: The challenger $\mathcal{C}$ performs the following operations:
    1) First, randomly chooses a bit $b \leftarrow \{0, 1\}$.
    2) Second, invokes n times $(K_{\mathrm{SE}}, T) \leftarrow$**Setup** $(1^{\lambda}, d)$: generates $\{K_{\mathrm{SE}_z}\}_{z=1}^{n}$ for every data owner $i \in D \setminus D_c$. For every data owner $i \in D$, generates $\{T_z\}_{z=1}^{n}$ of $\{W_{d_z}^b\}_{z=1}^{n}$.
    3) Third, invokes $(K_{\mathrm{PRF}_j}, \Delta_{j,i} = (\{r_1, \ldots, r_n\}, D), GBF_j) \leftarrow$**Share**$(\{K_{\mathrm{SE}_z}\}_{z=1}^{n}, \{T_z\}_{z=1}^{n})$: generates K_{PRF_j} for every data user $j \in U$, generates $\Delta_{j,i} = (\{r_1, \ldots, r_n\}, D)$ for each edge $(j, i) \in E$, generates GBF_j for every data user j.
    4) Fourth, invokes $q_{j,z,l} \leftarrow$**TokenGen**(K_{PRF_j}, w): generates a search token $q_{j,z,l}$ for every data user j, for keyword $w_{j,z,l}^b \in W_{d_z}^b$, where $1 \leq z \leq n$ and $1 \leq l \leq k_j$.
    5) Fifth, invokes $(proof_j, IdSet) \leftarrow$**Search**$(\Delta = (\{r_1, \ldots, r_n\}, D), q_{j,z,l}, \{T_z\}_{z=1}^{n}, GBF_j)$: generates the $proof_j$ and $IdSet$ for every data user j.
    6) Sixth, invokes True/False $\leftarrow$**Verify**$(q_{j,z,l}, proof_j, IdSet)$: if $IdSet$ is correct and complete, outputs True. Otherwise, outputs False.

7) Finally, for each $i \in D, (j,i) \in E, j \in U$, sends $\{T_z\}_{z=1}^{n}$, $\Delta_{j,i}$, $proof_j$, $(q_{j,z,1}, .., q_{j,z,k_j})$, $IdSet$ and True/False to adversary $\mathcal{A}$.

- ***Step 3*** : Adversary $\mathcal{A}$ outputs b' with $\mathtt{Adv}_{\mathcal{A}}(1^{\lambda}) = \frac{1}{2} - \Pr[b = b']$ advantage.

Discussion. Similar to previous works of searchable encryption, our definition also allows some information to leak. More specifically, because the adversary is limited by choosing $w_{j,z,l}^{0} \in W_{d_z}^{0}$ if and only if $w_{j,z,l}^{1} \in W_{d_z}^{1}$ for every $(j,i) \in E$, the scheme inevitably leaks access patterns, namely leaks which documents matched which query. Besides, we require that the repeated queries of adversary are located in the same locations of both n query sequences. Therefore, if a data user performs repeated queries, then the repeated queries are likely to leak to the server. Additionally, some ''benign leakage'' is leaked to the server, such as the number of matched documents, the number of queries of each data user.

III. VERIFIABLE MULTI-KEY SEARCHABLE ENCRYPTION SCHEME

We firstly introduce the high-level idea of our construction in this section. Then, we describe the proposed VMKSE scheme concretely.

A. HIGH-LEVEL IDEA

The problem of verifiable search in multi-user setting can be expressed as follows: The data owners store encrypted data onto a cloud server and selectively share it with some authorized users. To search for some specific documents, the authorized user sends a search token to the cloud server and retrieves the matched documents that authorized to him. We assume that the cloud server is not fully trusted, namely, the server may collude with data owners and even return a fragment of search results to response to a search query about an honest data user. Thus, in order to enable the data user to verify the integrity of the result, the cloud server should provide the corresponding proof along with the matched documents.

The main idea is that each authorized user generates an authenticated data structure GBF and stores all the document identifiers shared with him. That is, the authorized user firstly assigns a random value to each document, and then he generates an aggregated value for all the documents containing a specific keyword by XOR all the corresponding document-specific random values. Finally, all the aggregated values are inserted into GBF. Based on the self-checkability property of GBF, the authorized user can simultaneously achieve completeness and correctness of search result.

B. THE CONCRETE CONSTRUCTION

In this section, we describe the proposed VMKSE scheme in detail. The construction is based on a pseudorandom function F, a symmetric encryption scheme (**SE.KeyGen**, **SE.Enc**, **SE.Dec**), and an authenticated data structure GBF. The proposed scheme composes of five polynomial-time algorithms (**Setup**, **Share**, **TokenGen**, **Search**, **Verify**). Concretely, the details of the scheme are as follows:

- $(K_{\mathrm{SE}}, T) \leftarrow$**Setup**$(1^{\lambda}, d)$: For a document d, the data owner generates a key K_{SE} by the **SE.KeyGen** algorithm. On one hand, the key K_{SE} is used to encrypt the document d and generate the encrypted document C, $C \leftarrow$**SE.KeyGen**(K_{SE}, d). Each document d has its own data key. On the other hand, the data key K_{SE} is used to encrypt each keyword w in the document d, i.e. $t \leftarrow$**SE.Enc**(K_{SE}, w). We set each t in the set T. Finally, the data owner outsources (C, T) to the cloud server. If the data owner wants to share the document d with an authorized user, he will send K_{SE} to the user.
- $(K_{\mathrm{PRF}}, \Delta = (\{r_1, \ldots, r_n\}, D), GBF) \leftarrow$**Share**$(\{K_{\mathrm{SE}_i}\}_{i=1}^{n}, \{T_i\}_{i=1}^{n})$: When the user is shared with n documents, he downloads the corresponding encrypted keyword set $\{T_i\}_{i=1}^{n}$ and generates a share key $\Delta = (\{r_1, \ldots, r_n\}, D)$ for the n shared documents, and finally generates an authenticated structure GBF. More specifically, the user uses $\{K_{\mathrm{SE}_i}\}_{i=1}^{n}$ to decrypt $\{T_i\}_{i=1}^{n}$ and obtains the plaintext of keyword w in the shared document. After that, the user selects a new key K_{PRF}, then assigns a random identifier r for each shared document and then generates the share key $\Delta = (\{r_1, \ldots, r_n\}, D)$ for the shared documents. Finally, the data user generates a GBF for all the shared documents. The details are described in Algorithm 1. Here we need to mention that there are two kinds of differences to generate our GBF compared with the one proposed by [34]. The reason to make this variant is that we also need to provide a valid proof when the server returns an empty set. First, the locations in GBF are not determined by the elements in the set, but by another value (the keyword w in our scheme). Second, the unoccupied locations of the GBF in our scheme are set to 0^{λ}, instead of random λ-bit string. We note that the variant GBF has the same security as the original GBF, the detailed proof is described in [36].
- $q \leftarrow$**TokenGen**(K_{PRF}, w): When a data user is interested in a specific keyword w, he generates the token $q = F_{K_{\mathrm{PRF}}}(w)$ and sends it to the server.
- $(proof, IdSet) \leftarrow$**Search**$(\Delta = (\{r_1, \ldots, r_n\}, D), q, \{T_i\}_{i=1}^{n}, GBF)$: Upon receiving the token q, the server generates $tk_i' = F_q(r)$ and performs search in the hash table D. If tk_i' is found in D, the server adds the identifier r (each r identifies a encrypted set T, each T corresponds to a encrypted document C) into the result set $IdSet$. Finally, the server returns the $proof$ (the proof is the GBF uploaded to the server by the data user in the share algorithm) and $IdSet$ to the data user. The details are shown in Algorithm 2.
- True/False$\leftarrow$**Verify**$(q, proof, IdSet)$: To verify the integrity of search result returned by the server, the data user needs to check both the correctness and completeness. The output of this algorithm is True or False, which represents the honest or malicious of the server

Algorithm 1 Share $(\{K_{\mathrm{SE}_i}\}_{i=1}^n, \{T_i\}_{i=1}^n)$

Input: Data keys $\{K_{\mathrm{SE}_i}\}_{i=1}^n$, encrypted keywords set $\{T_i\}_{i=1}^n$
Output: A query key K_{PRF}, share key $\Delta = (\{r_1, \ldots, r_n\}, D)$ and GBF

$K_{\mathrm{PRF}} \xleftarrow{\$} \{0,1\}^\lambda$;
$D \leftarrow \phi$;
$DB \leftarrow emptymap$;
$W \leftarrow \phi$;
for $i = 0$ *to* $m-1$ **do**
  $GBF[i] \leftarrow NULL$;
end
for $i = 1$ *to* n **do**
  $r_i \xleftarrow{\$} \{0,1\}^\lambda$;
  $W_{d_i} \leftarrow$**SE.Dec**(K_{SE_i}, T_i);
  $W = W \cup W_{d_i}$;
  for *all* $w \in W_{d_i}$ **do**
    $k_w = F_{K_{\mathrm{PRF}}}(w)$; $tk_w = F_{k_w}(r_i)$;
    insert tk_w into hash table D;
  end
  $DB[w] = DB[w] \cup \{r_i\}$;
end
$H = (h_1, \ldots, h_k)$;
for *all* $w \in W$ **do**
  $R = 0^\lambda$;
  $emptyTag = -1$;
  $q = F_{K_{\mathrm{PRF}}}(w)$;
  for $r \in DB[w]$ **do**
    $R = R \oplus r$;
  end
  for $i = 1$ *to* k **do**
    $j = h_i(q)$;
    if $GBF[j] == NULL$ **then**
      if $emptyTag == -1$ **then**
        $emptyTag = j$;
      else
        $GBF[j] \xleftarrow{\$} \{0,1\}^\lambda$;
        $value = value \oplus GBF[j]$;
      end
    else
      $value = value \oplus GBF[j]$;
    end
  end
  $GBF[emptyTag] = value$;
end
for $i = 0$ *to* $m-1$ **do**
  if $GBF[i] == NULL$ **then**
    $GBF[i] \xleftarrow{\$} 0^\lambda$;
  end
end
return $(K_{\mathrm{PRF}}, \Delta = (\{r_1, \ldots, r_n\}, D), GBF)$;

respectively. Specifically, the algorithm is performed as the follows. The details are described in Algorithm 3.

Algorithm 2 Search $(\Delta = (\{r_1, \ldots, r_n\}, D), q, \{T_i\}_{i=1}^n, GBF)$

Input: Share key $\Delta = (\{r_1, \ldots, r_n\}, D)$, a token q, encrypted sets $\{T_i\}_{i=1}^n$ and GBF
Output: The *proof* and an identifiers set *IdSet*

$IdSet \leftarrow \phi$;
for $i = 1$ *to* n **do**
  $tk_i' = F_q(r_i)$;
  if tk_i' *in* D **then**
    $IdSet = IdSet \cup \{r_i\}$;
  end
end
//the GBF as the *proof*
return (*proof* , *IdSet*);

Algorithm 3 Verify $(q, proof, IdSet)$

Input: The token q, *proof* and *IdSet*
Output: True or False

Initializes $R = 0^\lambda$, $R' = 0^\lambda$;
$H = (h_1, \ldots, h_k)$;
if *IdSet*==$\emptyset$ **then**
  for $i = 1$ *to* k **do**
    $j = h_i(q)$;
    //each element of *proof* is $GBF[j]$
    if $GBF[j] != 0^\lambda$ **then**
      continue;
    else
      return True;
    end
  end
  return False;
else
  for *all* $r \in IdSet$ **do**
    $R' = R' \oplus r$;
  end
  for $i = 1$ *to* k **do**
    $j = h_i(q)$;
    $R = R \oplus GBF[j]$;
  end
  if $R' == R$ **then**
    return True;
  else
    return False;
  end
end

– Case 1: When *IdSet* is an empty set, it indicates that there are no matched documents for the token q. Concretely, the data user computes locations $j = h_i(q)$ for $1 \le i \le k$ in the *proof*. Then the data user checks whether there is an element at any of the k locations is 0^λ. If yes, the algorithm outputs True. That is, there is indeed no matched document. Otherwise, the output is False, which denotes that there are matched documents.

– Case 2: When *IdSet* is not an empty set, it indicates that there are some matched documents for the token q. The detailed steps for verification are as follows:
  * Step 1: The data user also computes k locations $j = h_i(q)$ for $1 \le i \le k$ in the *proof* returned by the server. Then the data user performs XOR operations over all the elements in the k locations and gets the value R.
  * Step 2: For all the elements in *IdSet* returned by the server, the data user also performs XOR operation and gets the value R'. Then the data owner checks whether the two values R and R' is equal. If yes, the output is True which indicates that the search result is correct and complete. Otherwise, the output is False which indicates that some results in *IdSet* are incorrect or incomplete.

Remark 1: BF is also a popular authenticated data structure. There are two reasons for us to adopt GBF as a verification tool instead of BF in our scheme. Firstly, BF cannot verify the correctness of the search result in our scheme. Fortunately, the XOR operation of GBF can further to verify the correctness of the search result. Secondly, the collision probability of GBF is smaller than BF.

IV. SECURITY AND EFFICIENCY ANALYSIS

In this section, we give the security analysis and efficiency analysis of the proposed VMKSE scheme.

A. SECURITY ANALYSIS

Theorem 1: A VMKSE scheme is secure against non-adaptive adversaries if for every PPT adversary $\mathcal{A}$ has a negl(λ) advantage with the challenger $\mathcal{C}$.

Proof: For every $i \in D$, adversary $\mathcal{A}$ chooses n sets $\{W^0_{d_z}\}^n_{z=1}$ and another n sets $\{W^1_{d_z}\}^n_{z=1}$ for data owner i, n sequences of keywords for queries $((w^0_{j,1,1}, \ldots, w^0_{j,1,k_j}), \ldots, (w^0_{j,n,1}, \ldots, w^0_{j,n,k_j}))$, namely $\{W^0_{j_z}\}^n_{z=1}$, and another n sequences of keywords for queries $((w^1_{j,1,1}, \ldots, w^1_{j,1,k_j}), \ldots, (w^1_{j,n,1}, \ldots, w^1_{j,n,k_j}))$, namely $\{W^1_{j_z}\}^n_{z=1}$ for data user $j \in U$. Let F^1 represent the pseudorandom function F invoked in share algorithm and TokenGen algorithm, it is used to generate a key k'_i for F^2 and the query q respectively, let F^2 denote the pseudorandom function F invoked in share algorithm to computer tk_w. Every encrypted set of corrupted data owners and symmetric data keys have the same distribution in both views, so we fix all these values into $view_0$, $view_1$, and all distributions. To prove adversary $\mathcal{A}$ can distinguish $view_0$ and $view_1$ with only a negl(λ) advantage, we use hybrid argument technique to define some necessary hybrids:

$\mathcal{H}^b_0$: Let $\mathcal{H}^b_0$ denote the hybrid distribution, which is obtained from $view_b$ using a random function $\mathcal{R}$ instead of F^1 (to generate the share key and queries). Here, we consider $\mathcal{R}$ has two inputs, an index of data user as the first input. Thus, $\mathcal{R}$ defines a series $\mathcal{R}_j$ of functions. That is, for all data users j and all $w_l \in \{W^b_{j_z}\}^n_{z=1}$, each query is represented as $q'_{j,l} = R(j, w_l)$ (as a mark for denoting queries in $\mathcal{H}^b_0$, but all queries in $view_b$ have no marks). In accordance with the pseudorandomness of F, we replace the calls of F^1 with a standard hybrid argument of one data user at a time, then we come to the conclusion: $view_b \approx \mathcal{H}^b_0$.

$\mathcal{H}^b_1$: $\mathcal{H}^b_1$ is the same as $\mathcal{H}^b_0$, except that F^2 is replaced by $\mathcal{R}$. (Here, the query $q'_{j,l}$ as the first input of $\mathcal{R}$, random identifier r as the second input of $\mathcal{R}$), so the generation of $\Delta_{j,i}$ are: For each $w_l \in W^b_{d_z} \cap W^b_{j_z}$, its corresponding $d_{i,j,l} = F^2_{q'_{j,l}}(r)$. For every $w_l \in W^b_{d_z} \setminus W^b_{j_z}$, choosing a random value as $d_{i,j,l}$, the constraint is: $d_{i,j,l} \notin \{F^2_{q'_{j,l}}(r), w_l \in W^b_{j_z}\}$.

$\mathcal{H}^{b,*}$: To prove $\mathcal{H}^b_0 \approx \mathcal{H}^b_1$, a new intermediate distribution $\mathcal{H}^{b,*}$ is defined. Through the intermediate distribution, we can show $\mathcal{H}^b_0 \approx \mathcal{H}^b_1$ as follows: Firstly, we can prove $\mathcal{H}^b_0 \approx \mathcal{H}^{b,*}$. More specially, in $\mathcal{H}^{b,*}$, for every data owner i, every data user j, and every keyword $w_l \in W^b_{d_z} \cap W^b_{j_z}$, the token $d_{i,j,l}$ in share key $\Delta_{j,i}$ is replaced by a random value and satisfies: $d_{i,j,l} \notin \{F^2_{q'_{j,l}}(r), w_l \in W^b_{j_z}\}$. According to the pseudorandomness of F, the tokens replaced by a hybrid argument one at a time, then we come to the conclusion: $\mathcal{H}^b_0 \approx \mathcal{H}^{b,*}$. Secondly, to show $\mathcal{H}^b_1 \approx \mathcal{H}^{b,*}$, some sub-hybrid distributions are defined. Notice that each distinct keyword queried by each data user represents the key for F^2 in all share keys related to a data user. Hence, in all sub-hybrid distributions, through replacing F^2 by the random function $\mathcal{R}$ to generate one query for a data user at a time. More specifically, the number of data users denoted by $t = |U|$, for every $j \in U$, let l_j indicate the number of different keywords in $W^b_{j_z}$. Then, for each $1 \le j \le s$, for each $0 \le l \le l_j$, we define the sub-hybrid distribution $\mathcal{H}^{b,j,l}$: $\mathcal{H}^{b,j,l}$ is obtained from $\mathcal{H}^{b,*}$ by generating all queries for the first $j-1$ data users, all the queries corresponding to all occurrences of the first l different keywords queried by data user j with a random function $\mathcal{R}$, and generates keyword tokens accordingly. Thus $\mathcal{H}^{b,1,0} = \mathcal{H}^{b,*}$, $\mathcal{H}^{b,t,l_t} = \mathcal{H}^b_1$. Obviously, to show that $\mathcal{H}^b_1 \approx \mathcal{H}^{b,*}$, so we need to prove $\mathcal{H}^{b,1,0} \approx \mathcal{H}^{b,t,l_t}$ as follows: For every $1 \le j \le s$, for every $0 \le l \le l_j$, according to the pseudorandomness of F^2, then $\mathcal{H}^{b,j,l} \approx \mathcal{H}^{b,j,l-1}$. Furthermore, for every $1 \le j \le s$, we have $\mathcal{H}^{b,j,0} = \mathcal{H}^{b,j-1,l_{j-1}}$. So we derive that $\mathcal{H}^{b,1,0} \approx \mathcal{H}^{b,t,l_t}$ and come to the conclusion: $\mathcal{H}^b_1 \approx \mathcal{H}^{b,*}$.

$\mathcal{H}^b_2$: $\mathcal{H}^b_2$ is the same as $\mathcal{H}^b_1$, except that for the generation of each index $j' = h_i(q)$ $(1 \le i \le k)$ of GBF_j in share algorithm, and the input q of each hash function h_i is computed by the random function $\mathcal{R}$. Concretely, for every $j \in U$, $i \in D$ and keyword $w_l \in W^b_{d_z}(1 \le z \le n)$, let $q_{i,j,l}$ denote the input of each hash function h_i in share algorithm, and $q_{i,j,l}$ is replaced with $\mathcal{R}$ for a data user at a time. According to the pseudorandomness of F^1, we come to the conclusion: $\mathcal{H}^b_1 \approx \mathcal{H}^b_2$.

$\mathcal{H}^b_3$: $\mathcal{H}^b_3$ is the same as $\mathcal{H}^b_2$, except that for all honest data owners $i \notin D_c$, each the encrypted set of $W^b_{d_z}(1 \le z \le n)$ is encrypted instead of $\overrightarrow{0}$ (in $\mathcal{H}^b_2$ encrypts the set $W^b_{d_z}$). $\mathcal{H}^b_2 \approx \mathcal{H}^b_3$ depends on the security of the symmetric encryption scheme by a standard hybrid argument, where the

TABLE 2. Performance comparison.

Reference	Data owner-server collision	Multi-key	Verifiability	Adaptive adversary	Multi-keyword search	Share(index) cost	TokenGen cost	Search cost	Verify cost
Scheme [24]	✓	✓	✗	✗	✗	$(2F + Dec)L.N$	F	$F + C.N$	-
Scheme [32]	✗	✗	✓	✓	✓	$(\|DB\| + 3\|W\| + 2)E$	$\|R_{w_1}\|(d-1)E$	$(\|R_{w_1}\|(d-1)+(d+1))E$	$2E + 4P + s(2E + 2P)$
Scheme [33]	✗	✗	✓	✗	✗	$(E+M+X+H)L.N$	$2P + E + M$	$(H + 2X + 2MOD)N$	$(2E + 2MOD + M + D)N$
Scheme [37]	✓	✓	✗	✓	✗	$(D + E + P)L.N$	E	$H + C.N$	-
Scheme [27]	✗	✗	✓	✗	✗	$(2\pi_K + 3H)L.N$	$\pi_K.N$	$\|DB[w]\|C.N$	$H.N$
Scheme [38]	✗	✗	✓	✓	✓	$(3F + Enc + 2E)\|DB[w]\|$	$(2F + M + E)d$	$\|W_c\|F + Dec + dE + M + P$	$\|R_{w_1}\|(F + dE + M))$
Our Scheme	✓	✓	✓	✗	✗	$(2F + Dec)L.N + (\|DB[w]\|X + kH)\|W\|$	F	$F + C.N$	$\|R\|X + kH$

encrypted set of $W^b_{d_z}$ is replaced with encrypted $\overrightarrow{0}$ one at a time. According to $\mathcal{H}^0_3 = \mathcal{H}^1_3$, then $view_0 \approx view_1$. Thus, the proposed VMKSE scheme is secure. □

B. COMPARISON

In this section, we compare some existing MKSE schemes and verifiable SSE schemes with the proposed scheme. Specifically, we compare our scheme with Hamlin et al.'s scheme [24], Wang et al.'s scheme [32], Liu et al.'s scheme [33], Popa et al.'s scheme [37], Kurosawa et al.'s scheme [27] and Miao et al.'s scheme [38].

Our scheme, Hamlin et al.'s scheme and Popa et al.'s scheme are designed in malicious data owner-server collision model. Wang et al.'s scheme, Miao et al.'s scheme, Liu et al.'s scheme and Kurosawa et al.'s scheme are designed in malicious server model. Only our scheme, Hamlin et al.'s scheme and Popa et al.'s scheme are designed in multi-key setting. Our scheme can be viewed as an extension from Hamlin et al.'s scheme to support verifiability of search result, all schemes except Hamlin et al.'s scheme and Popa et al.'s scheme are capable of verifying the search result returned by the server. Wang et al.'s scheme, Miao et al.'s scheme and Popa et al.'s scheme are secure against adaptive adversary. Wang et al.'s scheme and Miao et al.'s scheme support multi-keyword search, others only support a single-keyword search.

In our scheme, although it spends extra XOR operations and hash operations computational cost to generate GBF in the share phase, we note that the data user only needs to spend little extra computational overhead to generate the GBF once. It can be seen very intuitively that our scheme achieves verifiability of search result, which only spends some extra computational overhead (i,e. XOR and hash functions) compared to Hamlin et al.'s scheme. Therefore, it is efficient for the data user to verify the search result. However, the verification cost is proportional to the number of documents shared in Kurosawa et al.'s scheme and Liu et al.'s scheme. The exponentiation operation of verification phase in Wang et al.'s scheme and Liu et al.'s scheme leads to the verify phase inefficient.

We denote by N the number of shared documents, $|W|$ the number of different keywords from the n documents, L the number of keywords in each document, $|DB[w]|$ the number of documents containing the keyword w, d the number of multiple keywords to be queried at one time, s the number of identifiers selected for integrity verification, k the number of hash of GBF, $|R|$ the size of search result for keyword w, $|R_1|$ the size of search result for the least frequent keyword w_1, $|W_c|$ the number of documents containing w_1, D a division operation, E an exponentiation operation, M a multiplication operation, P a computation of pairing, Dec a decryption operation, C an equal comparison operation, π_K a pseudo-random permutation operation, MOD a modular operation, X a XOR operation, H a hash operation, F a PRF operation. The detailed performance comparison of the aforementioned seven schemes are shown in TABLE 2.

V. PERFORMANCE EVALUATION

We conduct an experimental evaluation of the three compared schemes(scheme [24], scheme [32] and our scheme) in this section. The experiments simulated with Java language and deployed with a LINUX machine on a regular desktop computer with 2.6 GHz Intel Core i7 CPU and 16GB RAM.

We simulate the experiments with a real-world dataset: Enron Email Dataset [39]. According to the statistics, the dataset contains a total of 517,401 plaintext documents with 1.32 GB. After processing all the plaintext email documents, the number of extracted distinct keywords is 94,650. Meanwhile, we use HMAC for PRF, AES in CBC mode to encrypt plaintext documents and corresponding keywords set. We fix the number of hash functions and each λ-bit string of GBF as $k = 13$ and $\lambda = 20$, respectively. According to [34], [40], to guarantee the requirement of collision probability $p_1 \leq 2^{-20}$ and false positive probability $p_2 \leq 2^{-20}$, The ratio of the length m of the GBF to the number n of elements to be inserted is fixed at 31.

FIGURE 3 shows the detailed evaluation of experimental results by comparing our scheme with scheme [24] and scheme [32]. Our experiments mainly focus on the storage required by each user to store the keys, the size of GBF and

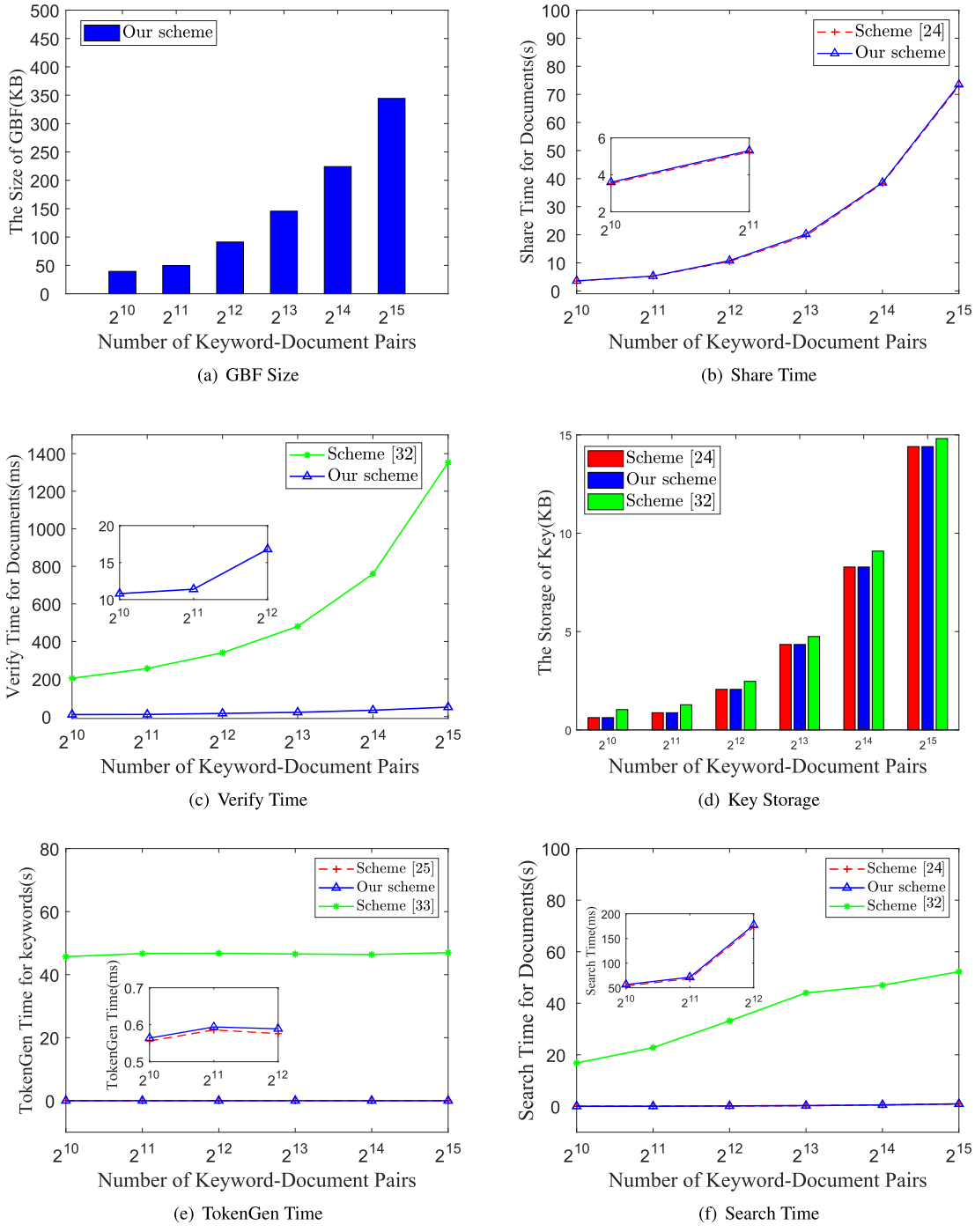


FIGURE 3. The performance comparison of the three schemes.

the time overhead of the share phase, the token generation phase, search phase and verify phase. The storage of keys in scheme [32] is slightly larger than both of our scheme and scheme [24]. We evaluate the size of GBF in our scheme. The size of GBF depends only on the shared documents. When the number of keyword-document pairs grows up to 2^{15}, the size of GBF is less than 400 KB. As FIGURE 3(b) shows, the two lines are nearly coincident for the reason that the generation time of GBF is too small to be negligible in share phase. As the number of shared documents increases, although it takes a certain amount of time to generate GBF, the extra computational overhead is relatively small compared to the share algorithm in scheme [24]. Compared with scheme [32], it is more efficient for a user to generate a token because of the TokenGen phase uses only one operation of a pseudorandom function in our scheme. Besides, our scheme is more efficient in searching for documents, but the search complexity depends on the documents shared with the user. The search

time in our scheme is slightly larger than the scheme [24] for the reason that the search algorithm needs to record the identifiers of documents. Obviously, it is more efficient for the user to verify the search result in our scheme. Even when the keyword-document pairs are 2^{15}, the data user needs only 50.2 ms to verify the search result, since the verify algorithm uses only symmetric operations (i.e. XOR and hash operations). The experimental results denote that our scheme can achieve security goals against malicious data owner-server collusion while maintaining a comparable performance.

VI. CONCLUSION

In this work, we focus on the problem of verifiable searchable encryption under a multi-user setting. A novel VMKSE scheme is proposed based on the primitive of GBF, which can support verifiability of search result even when both the data owner and the cloud server are malicious. Rigorous security analysis demonstrates that the proposed scheme can achieve the desired security goals. Furthermore, we implement our scheme for a real-world dataset, and the experiment results show that the proposed scheme can provide a comparable computation overhead. However, our scheme only supports a single keyword search. How to extend it to a new scheme that supports multi-keyword search is an interesting question.

REFERENCES

[1] D. Agrawal, A. El Abbadi, F. Emekci, and A. Metwally, "Database management as a service: Challenges and opportunities," in *Proc. IEEE 25th Int. Conf. Data Eng.* Mar./Apr. 2009, pp. 1709–1716.

[2] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms for secure outsourcing of modular exponentiations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2386–2396, Sep. 2014.

[3] 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.

[4] *Latest Facebook Data Breach Totals Over 540 Million Records Found Unsecured*, Accessed: Apr. 4, 2019. [Online]. Available: https://latesthackingnews.com/2019/04/04/latest-facebook-data-breach-totals-over-540-million-records-found-unsecured/

[5] J. K. Liu, M. H. Au, W. Susilo, K. Liang, R. Lu, and B. Srinivasan, "Secure sharing and searching for real-time video data in mobile cloud," *IEEE Netw.*, vol. 29, no. 2, pp. 46–50, Mar./Apr. 2015.

[6] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in *Proc. Annu. Int. Cryptol. Conf.*, Santa Barbara, CA, USA, 2007, pp. 535–552.

[7] 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.

[8] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. Conf. Comput. Commun. Secur.*, 2012, pp. 965–976.

[9] H. Li, D. Liu, Y. Dai, and T. H. Luan, "Engineering searchable encryption of mobile cloud networks: When QoE meets QoP," *IEEE Wireless Commun.*, vol. 22, no. 4, pp. 74–80, Aug. 2015.

[10] M. Nabil, A. Alsharif, A. Sherif, M. Mahmoud, and M. Younis, "Efficient multi-keyword ranked search over encrypted data for multi-data-owner settings," in *Proc. IEEE Int. Conf. Commun. (ICC)* Kansas City, MO, USA, May 2018, pp. 1–6.

[11] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, Interlaken, Switzerland, 2004, pp. 506–522.

[12] Y. H. Hwang and P. J. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Proc. Int. Conf. Pairing-Based Cryptogr.*, Tokyo, Japan, Jul. 2007, pp. 2–22.

[13] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack," *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2266–2277, Nov. 2013.

[14] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for Boolean queries," in *Proc. Annu. Cryptol. Conf.* Berlin, Germany: Springer, 2013, pp. 353–373.

[15] S. Faber, S. Jarecki, H. Krawczyk, Q. Nguyen, M. Rosu, and M. Steiner, "Rich queries on encrypted data: Beyond exact matches," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2015, pp. 123–145.

[16] P. Van Liesdonk, S. Sedghi, J. Doumen, P. Hartel, and W. Jonker, "Computationally efficient searchable symmetric encryption," in *Proc. Workshop Secure Data Manage.* Berlin, Germany: Springer, 2010, pp. 87–100.

[17] M. R. Asghar, G. Russello, B. Crispo, and M. Ion, "Supporting complex queries and access policies for multi-user encrypted databases," in *Proc. Workshop Cloud Comput. Secur. Workshop*, 2013, pp. 77–88.

[18] C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," *J. Comput. Secur.*, vol. 19, no. 3, pp. 367–397, 2011.

[19] Z. Liu, Z. Wang, X. Cheng, C. Jia, and K. Yuan, "Multi-user searchable encryption with coarser-grained access control in hybrid cloud," in *Proc. 4th Int. Conf. Emerg. Intell. Data Web Technol.*, Shaanxi, China, Sep. 2013, pp. 249–255.

[20] A. Kiayias, O. Oksuz, A. Russell, Q. Tang, and B. Wang, "Efficient encrypted keyword search for multi-user data sharing," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2016, pp. 173–195.

[21] Y. Wang, J. Wang, S.-F. Sun, J. K. Liu, W. Susilo, J. Baek, I. You, and X. Chen, "Towards multi-user searchable encryption supporting Boolean query and fast decryption," *J. Universal Comput. Sci.*, vol. 25, no. 3, pp. 222–244, 2019.

[22] P. Grubbs, R. McPherson, M. Naveed, T. Ristenpart, and V. Shmatikov, "Breaking Web applications built on top of encrypted data," in *Proc. SIGSAC Conf. Comput. Commun. Secur.*, Vienna, Austria, Oct. 2016, pp. 1353–1364.

[23] C. Van Rompay, R. Molva, and M. Önen, "A leakage-abuse attack against multi-user searchable encryption," *Proc. Privacy Enhancing Technol.*, vol. 2017, no. 3, pp. 168–178, 2017.

[24] A. Hamlin, A. Shelat, M. Weiss, and D. Wichs, "Multi-key searchable encryption, revisited," in *Proc. Int. Workshop Public Key Cryptogr.* Springer, 2018, pp. 95–124.

[25] C. Van Rompay, R. Molva, and M. Önen, "Secure and scalable multi-user searchable encryption," in *Proc. 6th Int. Workshop Secur. Cloud Comput.*, 2018, pp. 15–25.

[26] Q. Chai and G. Gong, "Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2012, pp. 917–922.

[27] K. Kurosawa and Y. Ohtaki, "UC-secure searchable symmetric encryption," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2012, pp. 285–298.

[28] Z. Liu, T. Li, P. Li, C. Jia, and J. Li, "Verifiable searchable encryption with aggregate keys for data sharing system," *Future Gener. Comput. Syst.*, vol. 78, pp. 778–788, Jan. 2018.

[29] W. Sun, X. Liu, W. Lou, Y. T. Hou, and H. Li, "Catch you if you lie to me: Efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr./May 2015, pp. 2110–2118.

[30] 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.

[31] J. Wang, X. Chen, X. Huang, I. You, and Y. Xiang, "Verifiable auditing for outsourced database in cloud computing," *IEEE Trans. Comput.*, vol. 64, no. 11, pp. 3293–3303, Nov. 2015.

[32] J. Wang, X. Chen, S. F. Sun, J. K. Liu, H. A. Man, and Z. H. Zhan, "Towards efficient verifiable conjunctive keyword search for large encrypted database," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2018, pp. 83–100.

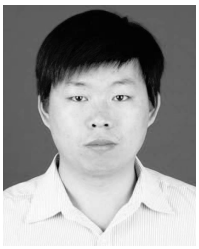
[33] X. Liu, G. Yang, Y. Mu, and R. Deng, "Multi-user verifiable searchable symmetric encryption for cloud storage," *IEEE Trans. Dependable Secure Comput.*, to be published.

[34] C. Dong, L. Chen, and Z. Wen, "When private set intersection meets big data: An efficient and scalable protocol," in *Proc. SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 789–800.

[35] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[36] C. Van Rompay and M. Önen, "Breaking and fixing the security proof of garbled bloom filters," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*. Cham, Switzerland: Springer, 2018, pp. 263–277.
[37] R. A. Popa and N. Zeldovich, "Multi-key searchable encryption," Cornell Univ., Ithaca, NY, USA, Tech. Rep. 2013/508, 2013.
[38] M. Miao, J. Wang, S. Wen, and J. Ma, "Publicly verifiable database scheme with efficient keyword search," *Inf. Sci.*, vol. 475, pp. 18–28, Feb. 2019.
[39] *Enron Email Dataset*. Accessed: Feb. 4, 2019. [Online]. Available: https://www.cs.cmu.edu/ enron/
[40] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Math.*, vol. 1, no. 4, pp. 485–509, 2004.

YAPING SU received the bachelor's degree from the School of Computer Science and Engineering, Northwest Normal University, China, in 2017. She is currently pursuing the master's degree with the School of Cyber Engineering, Xidian University. Her research interests include searchable encryption and cloud security.



JIANFENG WANG received the M.S. degree in mathematics, and the Ph.D. degree in cryptography from Xidian University, in 2016. He is currently with Xidian University. He visited the Swinburne University of Technology, Australia, from December 2017 to December 2018. His research interests include applied cryptography, cloud security, and searchable encryption.

YUNLING WANG received the master's degree in electronics and communication engineering from Xidian University, China, in 2015, where she is currently pursuing the Ph.D. degree in cryptography. During her Ph.D. study, she studied as a Joint Ph.D. Student with the Faculty of Information Technology, Monash University, Australia, for one year. Her research interests include searchable encryption and cloud computing.

MEIXIA MIAO received the M.S. degree in business administration from Xidian University, China, in 2013, and the Ph.D. degree from the School of Computer Science and Technology, Xidian University, in 2018. She is currently with the Xi'an University of Posts and Telecommunications. Her research interests include cloud computing and data security.