Volume-Hiding Range Searchable Symmetric Encryption for Large-Scale Datasets

Feng Liu, Kaiping Xue, *Senior Member, IEEE*, Jinjiang Yang, Jing Zhang, Zixuan Huang, Jian Li, *Senior Member, IEEE*, and David S.L. Wei, *Senior Member, IEEE*

***Abstract*—Searchable Symmetric Encryption (SSE) is a valuable cryptographic tool that allows a client to retrieve its outsourced data from an untrusted server via keyword search. Initially, SSE research primarily focused on the efficiency-security trade-off. However, in recent years, attention has shifted towards range queries instead of exact keyword searches, resulting in significant developments in the SSE field. Despite the advancements in SSE schemes supporting range queries, many are susceptible to leakage-abuse attacks due to volumetric profile leakage. Although several schemes exist to prevent volume leakage, these solutions prove inefficient when dealing with large-scale datasets. In this article, we highlight the efficiency-security trade-off for range queries in SSE. Subsequently, we propose a volume-hiding range SSE scheme that ensures efficient operations on extensive datasets. Leveraging the order-weighted inverted index and bitmap structure, our scheme achieves high search efficiency while maintaining the confidentiality of the volumetric profile. To facilitate searching within large-scale datasets, we introduce a partitioning strategy that divides a broad range into disjoint partitions and stores the information in a local binary tree. Through an analysis of the leakage function, we demonstrate the security of our proposed scheme within the ideal/real model simulation paradigm. Our experimental results further validate the practicality of our scheme with real-life large-scale datasets.**

***Index Terms*—Searchable symmetric encryption, range query, volume-pattern leakage, large-scale datasets.**

I. Introduction

In the past decade, driven by the rapid advancements in cloud computing and network technologies, an increasing number of individuals and organizations have shown a willingness to outsource their data to public cloud services. These cloud services offer a stable storage environment, enabling clients to access their outsourced data conveniently over the Internet, anytime and from anywhere. Given the data's relocation to the cloud, a primary concern for clients revolves around data privacy, especially when the outsourced data includes sensitive information. A straightforward approach to counter data leakage is to encrypt all data before outsourcing. However, traditional symmetric encryption methods like AES come at the cost of data searchability. To surmount this challenge, a cryptographic concept called Searchable Symmetric Encryption (SSE) was introduced [1], [2]. SSE enables clients to privatize their data during outsourcing while maintaining the capability to search encrypted data. This dual functionality aims to address the trade-off between security and searchability.

A general SSE scheme is designed for keyword-based search over encrypted documents. Since then, a considerable amount of research has emerged on this subject, raising issues related to rich queries, such as Boolean queries [3], [4], [5], fuzzy keyword queries [6], [7], [8], and range queries [9], [10], [11], [12], [13], [14]. In this article, we mainly focus on the range queries in SSE. Following the typical range SSE schemes [10], [12], a common approach is to transform a range query into a keyword-based search. More specifically, the scheme by Demertzis et al. [10] applies *range covering techniques* to reduce a range query to a multi-keyword search. Another scheme by Wang et al. [12] further reduces a range query to a two-keyword search by designing an *order-weighted inverted index*.

Regardless of keyword-based search or range query, to assure high search efficiency, an SSE scheme needs to construct a specific index, such as an inverted index, and normally leaks some information called *leakage profiles* [15], [16], [17]. The leakage profiles can be categorized into three types: *search-pattern leakage* (whether two search tokens are generated from the same keyword), *access-pattern leakage* (which identifiers are associated with the search token), and *volume-pattern leakage* (the size of identifiers for the search token). In the early stages of research, it was considered acceptable to leak these patterns. However, recent studies [17], [18], [19], [20], [21], [22] have shown that the server can reconstruct the database by relying solely on volume-pattern leakage, especially in the case of range queries. In other words, compared to general SSE (which only supports keyword search), range SSE is more vulnerable to volume-leakage attacks. This is mainly due to the fact that volume information can reflect the size of the queried

Manuscript received 13 December 2022; revised 21 August 2023; accepted 12 November 2023. Date of publication 21 November 2023; date of current version 11 July 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 61972371, in part by the Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant Y202093, and in part by Anhui Province Key Technologies Research & Development Program under Grant 2022a05020050. *(Corresponding author: Kaiping Xue.)*

Feng Liu, Kaiping Xue, Jinjiang Yang, and Jian Li are with the School of Cyber Science and Technology, University of Science and Technology of China, Hefei 230027, China (e-mail: lf2020@mail.ustc.edu.cn; kpxue@ustc.edu.cn; xiaohjy@mail.ustc.edu.cn; lijian9@ustc.edu.cn).

Jing Zhang is with the Science Island Branch of Graduate School, University of Science and Technology of China, Hefei 230031, China, and also with the Institute of Space Integrated Ground Network, Hefei 230088, China (e-mail: zhangjing2021@mail.ustc.edu.cn).

Zixuan Huang is with the Institute of Space Integrated Ground Network, Hefei 230088, China (e-mail: 420650892@qq.com).

David S.L. Wei is with the Department of Computer and Information Science, Fordham University, Bronx, NY 10458 USA (e-mail: wei@cis.fordham.edu).

Digital Object Identifier 10.1109/TDSC.2023.3335304

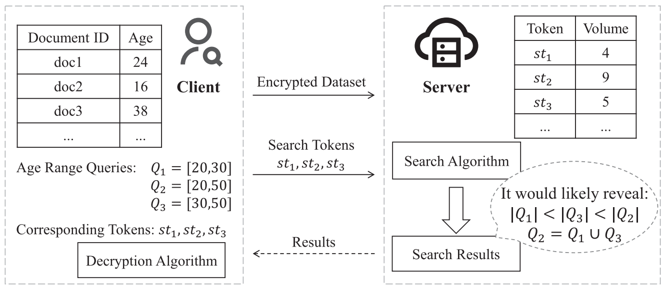


Fig. 1. Example of volume leakage in range SSE.

range to some extent, and the volume for a range can be regarded as the sum of volume for its subranges.

As a concrete example in Fig. 1, three range queries Q_1, Q_2 and Q_3 are issued by the client. After executing the search algorithm, the server can observe the number of records (i.e., volume pattern) corresponding to each range query. According to the volume patterns, the server can infer that the relations among these ranges' sizes may be $|Q_1| < |Q_3| < |Q_2|$. Moreover, because the volume corresponding to Q_2 is equal to the sum volume corresponding to Q_1 and Q_3, the server can further infer that the relations among these ranges may be $Q_2 = Q_1 \cup Q_3$. Once the server observes enough volume patterns, it can attempt to reconstruct the whole dataset by leveraging existing attack algorithms [17], [18], [19], [20].

Recently, the range SSE scheme by Zuo et al. [13] unintentionally prevent volume-pattern leakage by adopting the bitmap index structure. But Wang and Chow [14] argued that the bitmap index structure has inherent limitations when applying it to a large-scale dataset. Besides, according to the research [10], the Best Range Cover (BRC) technique employed in [13] would lead to extra leakage. In fact, how to support large-scale datasets is another important issue in range SSE. No matter the *order-weighted inverted index* or the *range covering technique*, there always exists at least one keyword which contains the whole records. This would result in low efficiency on index generating or a high burden on storage. A trivial approach to support large-scale datasets is taking each value as a keyword instead of adopting particular transforming techniques. But this method would cause significant inefficiency in search due to a huge number of search tokens (cf., Range SSE-I [12]). All these factors thus lead to the following question: *Can we design a range SSE to prevent volume-leakage attacks while keeping high security and search efficiency on large-scale datasets?*

In this article, we design a novel inverted index by utilizing the advantages of the order-weighted inverted index [12] and bitmap index [13]. Inspired by the idea of "searching locally" in [23], we construct a local search tree for retrieving document identifiers locally to prevent extra leakage. For supporting large-scale datasets, we adopt the partitioning strategy to split the whole range into disjoint subranges. In this way, we build a volume-hiding range SSE scheme (named VH-RSSE) with the optimal trade-off between search efficiency and storage overhead. Specifically, our proposed scheme VH-RSSE achieves the optimal size of query tokens (i.e., $O(1)$) for any range queries, and $O(m)$ storage overhead on the server's side, where m is the number of keywords in the datasets. The comparison of VH-RSSE with prior arts is given in Table I. Our contributions are listed as follows.

- We design a novel index, referred to as the order-accumulated inverted index, by combining the order-weighted inverted index and bitmap index. Our proposed order-accumulated index achieves the optimal size of query tokens (i.e., $O(1)$) and can also hide the volume pattern, thereby avoiding volume-leakage attacks.
- To further support large-scale datasets, we split the whole range into disjoint partitions, and design a local search tree to store the partition information. By leveraging the advantage of the local search tree, the client only needs to send at most two search tokens for any range query and retrieve document identifiers locally, which significantly decreases the storage overhead on the server's side.
- Based on the leakage function and the simulation paradigm, we provide formal security proof of our proposed scheme. Besides, we apply real-world datasets to our experiments and conduct comprehensive analyses in terms of computation and storage overhead. By comparing to the state-of-the-art works, we claim that our proposed VH-RSSE is more secure and efficient, especially on large-scale datasets.

The rest of this article is organized as follows. Section II introduces the related work. The problem statement is given in Section III. In Section IV, we describe the SSE scheme and related index structures. Section V illustrates our proposed schemes in detail. Then, we give a formal security analysis and performance evaluation in Sections VI and VII, respectively. Finally, we conclude this article in Section VIII.

II. Related Work

Secure range query has attracted extensive attention in the database research domain, and it can be achieved straightforwardly with the aid of Order Preserving Encryption (OPE) [24], e.g., CryptDB [25]. Due to the property of preserving the order for encrypted values, the schemes based on OPE have high compatibility with existing database management systems. However the security of OPE is still debatable, an attacker can easily launch an inference attack through the leakage of order information [26], [27]. To provide a stronger privacy assurance for range queries, some studies [28], [29] adopt a two-server architecture and choose the Paillier cryptosystem instead of OPE. Although these schemes can achieve higher search efficiency and stronger privacy assurance compared with the scheme based on OPE, the assumption that two servers are required to be non-colluding seems too strong. Intuitively, no one can assure that two servers will never collude in practice [30].

Another line of secure range query is to utilize SSE for a better balance between privacy protection and search efficiency. The concept of SSE was first introduced by Song, Wagner, and Perrig [1] in 2000. To formalize the security definitions and fulfill the practical requirement, Curtmola et al. [2] presented the semantic security definitions of SSE and adopted an inverted-index structure to improve keyword search efficiency. After that,

TABLE I
COMPARISON WITH PRIOR ARTS

Schemes	Size of Query Tokens	Storage Overhead (Server's Side)	Volume-Pattern Preservation	Large-Scale Datasets Support
SchemeA [11]	$O(\log \tilde{q})$	$O(d \cdot \lceil \log m \rceil)$	✗	✗
SchemeB [11]	$O(\log \tilde{q})$	$O(d \cdot m)$	✓	✗
Range SSE-I [12]	$O(q)$	$O(d + m)$	✗	✓
Range SSE-II [12]	$O(1)$	$O(d + m^2)$	✗	✗
FBDSSE-RQ [13]	$O(\log q)$	$O(d \cdot m)$	✓	✗
VH-RSSE	$O(1)$	$O(m)$	✓	✓

Note that d and m represent the number of documents and the number of keywords in the dataset, respectively. $\tilde{q}$ is the size of a range query, q is the number of distinct values inside a range query. Loosely speaking, $\tilde{q} \geq q$, cf., [12].

a series of studies have begun to concentrate on providing richer search functionality, such as Boolean queries [3], [4], [5], fuzzy queries [6], [7], [8], and range queries [9], [10], [11], [12], [13], [14]. As one of the most well-known SSE schemes with support for Boolean queries, OXT protocol [3] first achieves multiple keyword conjunctive queries in sublinear time. Later, Faber et al. [9] further extended OXT protocol to support range queries. The core idea is to convert a range query into disjunctions of exact keywords, and then invoke OXT protocol to execute multiple keywords. A similar method is also conducted in research by Demertzis et al. [10], but Demertzis et al. argued that OXT protocol is not suitable for handling keyword disjunctive queries since the search time is linear with the number of documents. In addition, they formalize range queries in the context of SSE [10] and also propose numerous range SSE schemes with various trade-offs between efficiency and security. Different from the above two schemes, the research [12] achieves efficient range queries by constructing an order-weighted inverted index instead of the original inverted index.

Nevertheless, to balance efficiency and privacy, all the above SSE schemes need to reveal some information about the query and the corresponding response, which are usually called *leakage profiles* [15], [16], [17]. The leakage profiles can be categorized into three types: *search-pattern leakage*, *access-pattern leakage*, and *volume-pattern leakage*. Recent studies [17], [18], [19], [20], [21], [22] emphasize that current range SSE schemes are vulnerable to attacks based on volume-pattern leakage. Although Zuo et al. [13] proposed a dynamic range SSE scheme that can inherently avoid leaking the volume pattern due to the natural property of bitmap structure. However, Wang and Chow [14] pointed out that the bitmap structure adopted in [13] restricts the size of files, which would lead to impracticality for large-scale datasets.

A naïve method to resist volume-leakage attacks is adopting padding countermeasures in general SSE schemes [31], [32], [33]. However, the padding strategies cannot be directly integrated in the range SSE schemes, especially when the dataset's size is very large. The reason is that, in the existing range SSE schemes [9], [10], [11], [12], [13], [14], there always exists at least one keyword which is mapped to the total documents in the dataset. Therefore, directly applying padding strategies (i.e., each keyword contains the same size of corresponding documents) will result in too much storage overhead. One recent line of works [34], [35], [36] has focused on the design of volume-hiding encrypted multi-maps that can provide better storage overhead than the naïve padding method. Unfortunately, Ando and George [36] argued that the length of each encrypted response produced by a minimally-leaking encrypted multi-map scheme must be at least the maximum response length. That is to say, when applying volume-hiding encrypted multi-maps into existing range SSE schemes, the response length must be at least as long as the number of the whole documents.

In a relatively recent work [37], Ren et al. designed a hybrid index framework to eliminate volume-pattern leakage for range queries by utilizing the Trusted Execution Environment (TEE) such as an SGX-enabled storage server. Like many other TEE-assisted applications (e.g., [38]), these schemes are vulnerable to side-channel attacks [39]. So far, there still lacks a volume-hiding range SSE scheme that supports large-scale datasets without relying on TEE.

III. PRELIMINARIES

A. SSE Scheme Syntax

In a typical SSE scheme, a client can outsource its encrypted documents to an untrustworthy cloud server preserving with keyword searchability. Following the definitions in [3], [10], each document has a unique identifier (denoted by ind) and contains a set of keywords (denoted by W). Assuming that the number of total documents is d, we use a database $\mathsf{DB} = (\mathsf{ind}_i, \mathsf{W}_i)_{i=1}^{d}$ to represent the list of identifier-keywords pairs. We denote the document identifiers that contain the keyword w by $\mathsf{DB}(w)$. The client first encrypts DB and then uploads the encrypted database EDB to the cloud server. The search protocol is executed between the client and cloud server, where the client first obtains the search token according to the search keyword, then the cloud server uses the search token to retrieve encrypted document identifiers from EDB. In general, an SSE scheme contains the following five algorithms:

- $K \leftarrow \mathsf{Setup}(1^\lambda)$: is a probabilistic algorithm run by the client. Given a security parameter λ, this algorithm outputs a secret key K.
- $\mathsf{EDB} \leftarrow \mathsf{BuildIndex}(K, \mathsf{DB})$: is a probabilistic algorithm run by the client. Upon inputting a database DB and the

Inverted Index		**Order-Weighted Inverted Index**	
Keyword	Document Identifiers	Keyword	Document Identifiers
6	$\{\mathsf{ind}_2\}$	6	$\{\mathsf{ind}_2\|\|1\}$
18	$\{\mathsf{ind}_2, \mathsf{ind}_6, \mathsf{ind}_7\}$	18	$\{\mathsf{ind}_2\|\|2, \mathsf{ind}_6\|\|1, \mathsf{ind}_7\|\|1\}$
21	$\{\mathsf{ind}_4\}$	21	$\{\mathsf{ind}_2\|\|2, \mathsf{ind}_4\|\|1, \mathsf{ind}_6\|\|1, \mathsf{ind}_7\|\|1\}$
28	$\{\mathsf{ind}_6, \mathsf{ind}_7\}$	28	$\{\mathsf{ind}_2\|\|2, \mathsf{ind}_4\|\|1, \mathsf{ind}_6\|\|2, \mathsf{ind}_7\|\|2\}$
33	$\{\mathsf{ind}_2\}$	33	$\{\mathsf{ind}_2\|\|3, \mathsf{ind}_4\|\|1, \mathsf{ind}_6\|\|2, \mathsf{ind}_7\|\|2\}$

Fig. 2. Example of order-weighted inverted index.

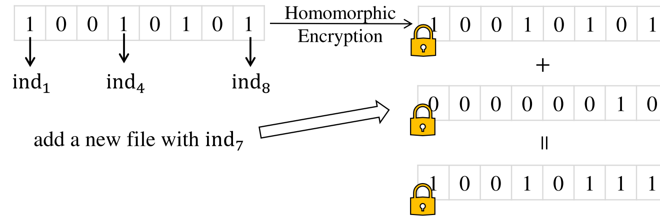


Fig. 3. Example of the bitmap index.

secret key K, this algorithm outputs an encrypted database EDB.

- $st \leftarrow \mathsf{TokenGen}(K, w)$: is a deterministic algorithm run by the client. Upon inputting a search keyword w and the secret key K, this algorithm outputs a search token st.
- $RS \leftarrow \mathsf{Search}(st, \mathsf{EDB})$: is a deterministic algorithm run by the server. For a search token st and the encrypted database EDB, this algorithm outputs a set of search results RS.
- $I \leftarrow \mathsf{Dec}(K, RS)$: is a deterministic algorithm run by the client. For the secret key K and the search result RS, this algorithm outputs a set of document index I.

We say that an SSE scheme is correct if for any K output by $\mathsf{Setup}(1^\lambda)$, given any DB, after running $\mathsf{EDB} \leftarrow \mathsf{IndexBuild}(K, \mathsf{DB})$, $st \leftarrow \mathsf{TokenGen}(K, w)$, $RS \leftarrow \mathsf{Search}(st, \mathsf{EDB})$, and $I \leftarrow \mathsf{Dec}(K, RS)$, the set I contains the same document identifiers as that in $\mathsf{DB}(w)$. To improve the search efficiency, a common approach is to use the inverted index as the EDB's structure, where each (encrypted) keyword is regarded as an index and mapped to a set of (encrypted) document identifiers.

In the context of range SSE, the syntax definition is slightly different. For the sake of simplicity, many studies (e.g., [10], [12], [13]) only consider single-dimensional range queries, because the multi-dimensional range queries can be reduced to several independent single-dimensional range queries. Following this assumption, each document is characterized by an attribute (e.g., age), whose value v is regarded as the document's keyword. That is, $\mathsf{DB} = (\mathsf{ind}_i, w_i)_{i=1}^d$ and $\mathsf{W} = \bigcup_{i=1}^d \{w_i\} = \{v_1, v_2, \ldots, v_m\}$, where $v_1 < v_2 < \ldots < v_m$, m is the number of distinct values in DB. Since each document only corresponds to one keyword, it is clear to infer that $m \leq d$. The range query is denoted by $Q = [v_\ell, v_r]$, where $\ell, r \in \{1, 2, \ldots, m\}$ and $\ell \leq r$. Therefore, the corresponding inverted index $\mathbf{I}$ can be parsed as $(v_i, I_i)_{i=1}^m$, where $I_i = \mathsf{DB}(v_i)$ is the set of identifiers containing v_i.

B. Order-Weighted Inverted Index

As a special construction of the inverted index, the order-weighted inverted index is originally designed for range queries in [12]. An example of order-weighted inverted index is shown in Fig. 2.

At a high level, given an inverted index $\mathbf{I} = (v_i, I_i)_{i=1}^m$, the order-weighted inverted index can be represented as $(v_i, I_i^*)_{i=1}^m$, where $I_i^* = \bigcup_{j=1}^i \{I_j\}$ means the collection of identifiers containing values no greater than v_i. Each document identifier in order-weighted inverted index is attached with a weight information (e.g., $\mathsf{ind}_i || \omega_i$, where ω_i is ind_i's weight). The weight information represents the number of occurrences of the corresponding document identifier. This structure has the benefit in reducing the size of search tokens (token size is $O(1)$). That is to say, for any range query $Q = [v_\ell, v_r]$, the client only needs to generate two search tokens: one is for $v_{\ell-1}$, and the other is for v_r. The search result can be calculated by $I_r^* \backslash I_{\ell-1}^*$. For more details of the order-weighted inverted index, we refer the readers to [12].

C. Bitmap Index

Bitmap index has been widely used to represent document identifiers in several SSE schemes [11], [13], [40], [41], [42]. As illustrated in Fig. 3, assume that there are 8 documents, then we can set a bit string bs with length 8. Consider an inverted index $(v_i, I_i)_{i=1}^m$, we can use a bit string bs to represent I_i. That is, if there exists document identifier ind_j in I_i, we set the jth bit of bs to 1. Otherwise, it is set to 0.

This index is usually encrypted by an additively homomorphic encryption scheme so that the server can update the bit string in the ciphertext domain. Since for each keyword v_i, the length of bs is identical, which implies that the bitmap index can inherently prevent the volume-pattern leakage.

IV. System Model and Problem Statement

A. System Definition

In our model, there are two participants, i.e., a client and a cloud server (as shown in Fig. 1). The client first outsources its encrypted documents to the cloud server. Then the client can generate search tokens to retrieve the documents from the server. From the view of the client, the cloud server is considered to be *honest-but-curious*, which means the cloud server will execute the protocol honestly, but it will also try to infer what range the client is searching for or what identifiers have been retrieved by leveraging the leakage profile. Assuming that d is the number of documents, we use $\mathsf{DB} = (\mathsf{ind}_i, \mathsf{w}_i)_{i=1}^d$ to represent the list of identifier-keywords pairs. Following the range SSE syntax, each document contains a unique identifier ind and a distinct value $v \in \{v_1, v_2, \ldots, v_m\}$, where m is the number of distinct values in DB. The collection of all keywords is denoted by

TABLE II
NOTATIONS

Symbol	Description
λ	The security parameter
BT	The local binary tree
DB	A database held by the client, where $\mathsf{DB} = (\mathsf{ind}_i, w_i)_{i=1}^{d}$
$\mathbf{I}$	The inverted index of DB, where $\mathbf{I} = (v_i, I_i)_{i=1}^{m}$
EDB	The encrypted database of $\mathbf{I}$
bs	The bit string used to represent the document identifiers
L	The length of bs
e	The encrypted bit string
F	A secure Pseudorandom Function (PRF)
H_1, H_2	The keyed hash functions
K	The secret key held by the client
C	A collection used to record each subrange's information
N	A node with a left child $N.left$ and a right child $N.right$
Q	A range query with lower bound v_ℓ and upper bound v_r
ST	The search tokens list issued by the client
RS	The search result returned by the server
$a\|\|b$	String concatenation of a and b

$\mathsf{W} = \bigcup_{i=1}^{d} \mathsf{w}_i = \{v_1, v_2, \ldots, v_m\}$. Notations used in this article are given in Table II.

B. Design Goal

Our goal is to design a volume-hiding range SSE scheme which can support efficient operations on large-scale datasets. Despite the bitmap index can prevent volume-pattern leakage due to its inherent property, it also has limitations when the number of documents is large, especially in the case of range SSE. Our design aims to resolve the issue of the trade-off between privacy preservation and large-scale dataset support. In particular, the following objectives should be achieved:

- *Privacy Preservation*: Our design aims to protect *query privacy* (i.e., what range the client is searching for) and *data privacy* (i.e., what data is being retrieved for each query) from the cloud server. Considering that the cloud server would launch volume-leakage attacks to infer the query privacy and data privacy, we need to hide the volume pattern to avoid these leakage-abuse attacks.
- *Large-Scale Dataset Support*: Our design should be highly scalable in terms of supporting large documents. In addition, no matter how large the dataset is, or what range is queried, the protocol should maintain high efficiency on both the client's side and the server's side.

C. Security of Range SSE

Similar to the previous studies [10], [12], [13], we define a leakage function $\mathcal{L}$ to depict the information which would be learned by the cloud server. The rigorous definition of leakage function $\mathcal{L}$ is given in Section VI. Based on the leakage function, the security of our proposed scheme can be formalized by the simulation paradigm [2] (also referred to as the ideal/real model). That is to say, for a range scheme Π, the behavior in the real world is essentially the same as the protocol in Π. While in the ideal world, the behavior is simulated by a simulator $\mathcal{S}$ which exploits the leakage information (i.e., the outputs of $\mathcal{L}$) as the input. Intuitively, to prove that Π is secure, we only need to prove that for every probabilistic polynomial time (PPT) adversary $\mathcal{A}$, the probability to distinguish these two worlds is negligible.

We use $\mathbf{Real}_{\mathcal{A}}^{\Pi}$ and $\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{\Pi}$ to denote two security games in real world and ideal world, respectively. The definitions of $\mathbf{Real}_{\mathcal{A}}^{\Pi}$ and $\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{\Pi}$ are described as follows.

- $\mathbf{Real}_{\mathcal{A}}^{\Pi}$: Upon inputting a security parameter λ, the challenger invokes $\mathsf{Setup}(\lambda)$ to generate a secret key K. For a database DB provided by the adversary $\mathcal{A}$, the challenger returns $\mathsf{EDB} \leftarrow \mathsf{IndexBuild}(K, \mathsf{DB})$ to $\mathcal{A}$. Then adversary $\mathcal{A}$ sends a polynomial number of range queries to the challenger. For each range query Q, the challenger returns a corresponding search tokens list ST. Eventually, $\mathcal{A}$ outputs a bit $b \in \{0, 1\}$ as the result of the game.
- $\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{\Pi}$: For a database DB provided by the adversary $\mathcal{A}$, the simulator $\mathcal{S}$ returns EDB by exploiting the leakage function $\mathcal{L}(\mathsf{DB})$. Then adversary $\mathcal{A}$ sends a polynomial number of range queries to $\mathcal{S}$. For each range query Q, the simulator $\mathcal{S}$ returns a corresponding search tokens list ST by exploiting the leakage function $\mathcal{L}(Q, \mathsf{DB})$. Eventually, $\mathcal{A}$ outputs a bit $b \in \{0, 1\}$ as the result of the game.

Definition 1 (Security of Range SSE): For a range SSE scheme Π and the security games described above, the scheme is $\mathcal{L}$-adaptively-secure if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, there is a PPT simulator $\mathcal{S}$ such that

$$|\Pr[\mathbf{Real}_{\mathcal{A}}^{\Pi}(\lambda) = 1] - \Pr[\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{\Pi}(\lambda) = 1]| \leq \mathsf{negl}(\lambda), \quad (1)$$

where $\mathsf{negl}(\lambda)$ is a negligible function.

V. The Proposed Scheme

A. Overview

In our system, the client first partition the range $[v_1, v_m]$ into disjoint subranges. For each subrange, we utilize a bitmap index to represent the document identifiers, and ensure that the number of documents in each subrange could not exceed the bit string's length. We also use a collection C to record the subranges' information. For example, assuming $C = \{C^{(1)}, C^{(2)}, \ldots\}$, $C^{(i)}$ records the ith subrange's information. The information includes the subrange's boundary values (denoted by $[C_{\min}^{(i)}, C_{\max}^{(i)}]$), the volume information within the subrange (denoted by $C_{info}^{(i)}$), and the distinct values within the subrange (denoted by $C_v^{(i)}$).

According to collection C, the client can build a binary tree BT for local search. As shown in Algorithm 1, for each node N, we use $[N_{\min}, N_{\max}]$ to represent the boundary of the range associated with N, where $N_{\min} = N.left_{\min}$ and $N_{\max} = N.right_{\max}$. After building the local search tree BT, the client begins to construct an order-accumulated inverted index for each subrange. A concrete example of the order-accumulated inverted index is given in Fig. 4. The idea behind our proposed order-accumulated inverted index is from the fact that each document is characterized by only a keyword value in the range query scenario. Compared to the original order-weight

Algorithm 1: Local Binary Tree.

BinaryTree.BuildTree(C):
Input: A Collection C
Output: A Binary Tree BT
Client:
  if $|C| = 0$ **then**
    Return $\perp$
  else if $|C| = 1$ **then**
    Assign $C^{(1)}$ to a root node and set it as BT;
    Return BT
  else
    Generate 2^{κ} leaf nodes ; // $2^{\kappa-1} < |C| \leq 2^{\kappa}$
    Assign each element in C to each leaf node;
    for $i = \kappa - 1$ *to 0* **do**
      Generate 2^i nodes;
      for *each node* N **do**
        Set its left and right child to two consecutive unassigned nodes of previous level;
        $N_{min} := N.left_{min}$;
        $N_{max} := N.right_{max}$;
      end
    end
    Set the root node as BT;
  end
  Return BT
end

BinaryTree.LocalSearch(Q, BT):
Input: Range Query $Q = [v_\ell, v_r]$ and binary tree BT
Output: C_s
Client:
  Temp $\leftarrow$ Empty Set;
  for $v \in \{v_\ell, v_r\}$ **do**
    Let N be the root node;
    while N *is not a leaf node* **do**
      if $v \leq N.left_{max}$ **then**
        $N := N.left$;
      else
        $N := N.right$;
      end
    end
    Temp := Temp $\cup N$;
  end
  Parse Temp as $\{C^{(left)}, C^{(right)}\}$; // $left \leq right$
  $C_s := \{C^{(left)}, C^{(left+1)}, \ldots, C^{(right)}\}$;
  Return C_s
end

inverted index, we remove the redundant weight information and use the bitmaps to represent the document identifiers, so that the volume pattern of each keyword can be hidden. After that, the client encrypts the order-accumulated inverted indexes as EDB and sends it to the server.

To search the documents in a range $Q = [v_\ell, v_r]$, the client first performs the local search based on BT (cf., Algorithm 1). If a node of BT precisely matches the queried range, then the client can directly obtain the bitmap index locally without interacting with the server. Otherwise, the client needs to generate a search token(s) list ST and send ST to the server.

Upon receiving ST, the server starts to execute the search algorithm with each token in ST and returns the final results to the client. Finally, the client decrypts the final results and obtains the bitmap index from BT. The whole procedure is depicted in

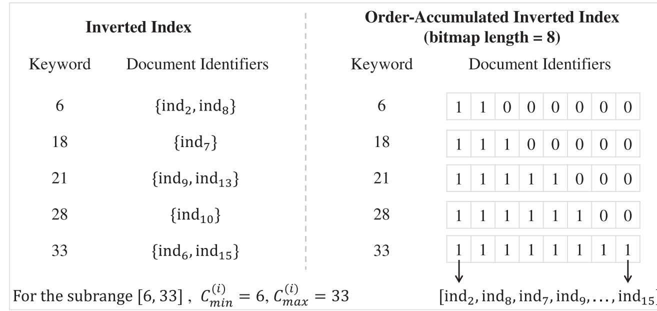


Fig. 4. Example of our order-accumulated inverted index structure.

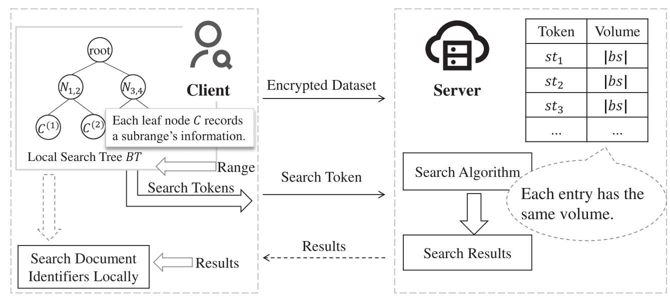


Fig. 5. Overview of VH-RSSE.

Algorithm 2: Setup.

Input: Security Parameter λ, $\mathcal{V} = \max\{|\mathsf{DB}(w)|, \forall w \in \mathsf{W}\}$
Output: System Parameters
Client:
  Initialize L $(L \geq \max\{\lambda, \mathcal{V}\})$, $K \leftarrow \{0,1\}^{\lambda}$;
  Initialize two hash functions H_1, H_2 and a PRF F;
  Initialize an empty map EDB, a binary tree BT;
  Return $(L, K, \mathsf{EDB}, \mathsf{BT}, H_2, H_2, F)$
end

Fig. 5. More specifically, we divide the whole procedure into three phases: *Setup*, *Building*, and *Search*.

B. Setup Phase

In this phase, the client first initializes the secret key K and the bitmap's length L according to the security parameter λ (e.g., 80 bits) and the max volume $\mathcal{V}$ of database DB. In our system, we require that L is not less than λ or $\mathcal{V}$, i.e., $L \geq \max\{\lambda, \mathcal{V}\}$. Afterward, the client generates two hash functions H_1, H_2, a secure Pseudorandom Function (PRF) F, an empty map EDB and a binary tree BT. Note that EDB is an inverted index where we use a bitmap to represent the document identifiers, and the binary tree BT stores the range information. The procedure is described in Algorithm 2.

C. Building Phase

For $\mathsf{DB} = (\mathsf{ind}_i, \mathsf{w}_i)_{i=1}^{d}$, the client first parses it as an inverted index $\mathbf{I} = (v_i, I_i)_{i=1}^{m}$. As we explained in Section IV, in the case of a range query, each document identifier corresponds to only a single keyword, and each keyword w has a distinct value v.

Algorithm 3: BuildIndex.

Input: Database $\mathsf{DB} = (\mathsf{ind}_i, \mathsf{w}_i)_{i=1}^{d}$, System Parameters
Output: Encrypted Database EDB, Local Search Tree BT
Client:
Parse DB into an inverted index $\mathbf{I} = (v_i, I_i)_{i=1}^{m}$;
Initialize an empty set C, set $ct \leftarrow 1$;
Initialize an offset value $\varphi \leftarrow 0$;
Initialize an empty inverted index $\mathbf{I}^*$ and set $\mathbf{I}^*[v_0] \leftarrow \emptyset$;
Initialize an empty list vlist;
for $i \leftarrow 1$ *to* m **do**
  if $|\mathbf{I}^*[v_i]| + |I_i| \leq L$ **then**
    $\mathbf{I}^*[v_i] \leftarrow I_i \cup \mathbf{I}^*[v_{i-1}]$;
    Append v_i into vlist;
  else
    Let $C_{info}^{(ct)} \leftarrow (|\mathbf{I}^*[v_i]|, \varphi)$, $C_v^{(ct)} \leftarrow$ vlist;
    Let $C_{max}^{(ct)} \leftarrow \max(\text{vlist})$, $C_{min}^{(ct)} \leftarrow \min(\text{vlist})$;
    Append $C^{(ct)}$ into C;
    $\mathbf{I}^*[v_i] \leftarrow I_i$, $\varphi \leftarrow \varphi + |\mathbf{I}^*[v_i]|$;
    vlist $\leftarrow \emptyset$, $ct \leftarrow ct + 1$;
  end
end
BT $\leftarrow$ BinaryTree.BuildTree(C);
for $ct \leftarrow 1$ *to* $|C|$ **do**
  for v_i *in range* $[C_{min}^{(ct)}, C_{max}^{(ct)}]$ **do**
    $K_1||K_2 \leftarrow F(K, v_i)$;
    Generate bs based on $C_{ind}^{(ct)}$ and $\mathbf{I}^*[v_i]$;
    $e \leftarrow bs \oplus H_1(K_1, v_i)$, $st \leftarrow H_2(K_2, v_i)$;
    $\mathsf{EDB}[st] := e$;
  end
end
Return EDB, BT
end

Afterward, the client begins by partitioning the range $[v_1, v_m]$ into disjoint subranges and stores them in a collection C. Then, based on C, the client can build BT and EDB. Finally, the client retains BT locally and sends EDB to the cloud server. To alleviate the storage burden, we assert that the client only needs to store an offset value and the size of document identifiers for each collection C. This is due to the fact that the bitmap's non-zero entries are consecutive in our proposed order-accumulated inverted index. Therefore, the client can reconstruct the bitmap using the offset value and the size of document identifiers. The details of the procedure are shown in Algorithm 3.

D. Search Phase

For a range query $Q = [v_\ell, v_r]$, the client first obtains a subrange set C_s via local search tree BT. If v_ℓ and v_r are exactly the low and high bounds of C_s, then the client can generate the bitmaps within the range Q locally. Otherwise, the client needs to generate the search tokens and send them to the cloud server. Upon receiving the search tokens list ST, the server retrieves all encrypted bitmap index e corresponding to ST and sends search results to the client. Finally, the client decrypts the search result and restores all plain bitmap index via C_s. The details of the procedure is shown in Algorithm 4.

Algorithm 4: Search.

Input: Search Range $Q = [v_\ell, v_r]$
Output: Bitmap Index Collection
Client:
Initialize two empty set ST, I_{RS};
$C_s \leftarrow$ BinaryTree.LocalSearch(Q, BT);
// Parse C_s as $\{C^{(s_1)}, C^{(s_2)}, \ldots, C^{(s_t)}\}$;
if $v_\ell = C_{min}^{(s_1)}$ *and* $v_r = C_{max}^{(s_t)}$ **then**
  for $i \leftarrow 1$ *to* t **do**
    $(\nu, \varphi) \leftarrow C_{info}^{(s_i)}$, $bs_i \leftarrow 1^\nu||0^{L-\nu}$;
    Append (bs_i, φ) into I_{RS};
  end
  Return I_{RS}
else
  Initialize an empty set V;
  Let $C_{s'} \leftarrow C_s$;
  if $v_\ell \neq C_{min}^{(s_1)}$ **then**
    Find $v \in C_v^{(s_1)}$ where v is closest to v_ℓ and $v \geq v_\ell$, let $v_\ell \leftarrow v$;
    $K_1||K_2 \leftarrow F(K, v_\ell)$, $st_\ell \leftarrow H_2(K_2, v_{\ell-1})$;
    Append st_ℓ into ST;
    Append $v_{\ell-1}$ into V;
    $C_{s'} \leftarrow C_{s'} \backslash C^{(s_1)}$;
  end
  if $v_r \neq C_{max}^{(s_t)}$ **then**
    Find $v \in C_v^{(s_t)}$ where v is closest to v_r and $v \leq v_r$, let $v_r \leftarrow v$;
    $K_1||K_2 \leftarrow F(K, v_r)$, $st_r \leftarrow H_2(K_2, v_r)$;
    Append st_r into ST;
    Append v_r into V;
    $C_{s'} \leftarrow C_{s'} \backslash C^{(s_t)}$;
  end
  Send search tokens ST to Server;
end
end
Server:
Initialize an empty set RS;
for st *in* ST **do**
  $e \leftarrow \mathsf{EDB}[st]$;
  Append e into RS;
end
Send search result RS to Client.
end
Client:
for e *in* RS, *and corresponding* v *in* V **do**
  $K_1||K_2 \leftarrow F(K, v)$, $bs \leftarrow e \oplus H_1(K_1, v)$;
  if v *equals to* $v_{\ell-1}$ **then**
    $(\nu, \varphi) \leftarrow C_{info}^{(s_1)}$;
    Append $(bs \oplus (1^\nu||0^{L-\nu}), \varphi)$ into I_{RS};
  else
    $(\nu, \varphi) \leftarrow C_{info}^{(s_t)}$;
    Append (bs, φ) into I_{RS};
  end
end
Return $I_{\mathsf{RS}} \cup \{(1^\nu||0^{L-\nu}, \varphi)\}_{\forall C^{(s_i)} \in C_{s'}, (\nu,\varphi) \leftarrow C^{s_i}_{info}}$
end

E. Discussions

Like many other SSE schemes [3], [4], [5] that focus on the search document index, our design only retrieves the bitmap index corresponding to the documents that match the queried range. This model is also deemed as structure-only SSE [43], which means the process of document retrieval is left out. This is because the client can retrieve the corresponding documents by PIR or ORAM techniques, after obtaining the document index. In our proposed VH-RSSE, in order to retrieve the documents, we need to establish a connection between the bitmap and the positions of documents stored on the server. For a bitmap bs with an offset value φ, assume that the ith bit of bs corresponds to ind_j, we only need to set the position of document ind_j as $\mathcal{P}(K_p, i+\varphi)$ in the setup phase, where $\mathcal{P}$ is a pseudorandom permutation. In this way, the client can calculate the positions of documents according to the bitmap index, and further retrieve the documents from the server.

Then we need to discuss the strength of VH-RSSE when applied to large-scale datasets. As we mentioned earlier, no matter the BRC technique applied in [10], [13] or the order-weighted inverted index utilized in [12], there always exists a keyword containing the whole document identifiers. To address this issue, in our proposed scheme, we design a hybrid index structure called order-accumulated inverted index by combining the bitmap index and order-weighted inverted index. Through integrating the partitioning strategy, each keyword in EDB corresponds to a bit string with a constant size, where the size depends on the partitioning strategy and is far smaller than the size in Zuo et al.'s scheme [13]. Concretely, our proposed scheme requires that each bit string's length $L \geq \max\{|\mathsf{DB}(w)|, \forall w \in \mathsf{W}\}$, while Zuo et al.'s scheme requires that each bit string's length $L' \geq \sum |\mathsf{DB}(w)|, \forall w \in \mathsf{W}$. Although Zuo et al. also mentioned that they can divide a large bit string into several shorter ones by taking the method in [40], the size of total bit strings for each keyword has no changes.

From the aspect of index storage overhead, it is easy to find that the size of EDB only depends on the length of the bitmap and the number of keywords, which implies that the number of document identifiers will have little impact on it. The token generation time is $O(\log|C|)$, i.e., the height of the local search tree BT. For any range query, the number of search tokens is at most 2. Since the search complexity is directly correlated with the number of search tokens, we claim that VH-RSSE achieves high search efficiency.

Our proposed scheme can also be extended to support dynamic datasets by employing the idea from existing dynamic SSE schemes (such as [44], [45], [46]). Taking the idea of [46] as an example, we only need to make slight changes to our proposed scheme for supporting dynamic updates, where the procedure for updating and searching is shown in Fig. 6. In the setup phase, the client needs to additionally initialize an empty map Σ, a hash function H_3 and a pseudorandom permutation P. The empty map Σ is used to store the states for each keyword, and P is used to establish a connection between every two states. For each keyword v, $\Sigma[v]$ records a tuple (st_c, c), where c is a counter to record the number of updates and st_c is the state. In each state st_c, the client needs to generate a random k_{c+1} to calculate the next state, namely, $st_{c+1} \leftarrow P(k_{c+1}, st_c)$. The bit string bs corresponding to st_{c+1} is encrypted as $e_c \leftarrow bs \oplus H_1(K_1, v||c+1)$. Then the client just set $\mathsf{EDB}[H_2(K_2, st||st_{c+1})] := (k_{c+1}||e_{c+1}) \oplus H_3(st||st_{c+1})$. As for the search phase, the server only needs to take $st||st_{c+1}$ as the input, and obtain the previous state by computing $st_c \leftarrow P^{-1}(k_{c+1}, st_{c+1})$. After retrieving $[e_1, e_2, \ldots, e_{c+1}]$, the server just needs to return the final result $RS = e_1 \oplus e_2 \oplus \ldots \oplus e_{c+1}$. One of the main challenges in dynamic SSE is preserving forward privacy, which means the newly added documents cannot be matched by the previous search tokens. In this extended version, the newly added document in state st_{c+1} cannot be matched by the previous search tokens because the server cannot calculate $P_{k_{c+1}}(st_c)$ without the knowledge of k_{c+1} in probabilistic polynomial time. Since our purpose is to design a volume-hiding range SSE for large-scale datasets, here we omit a full analysis of dynamic updates for brevity.

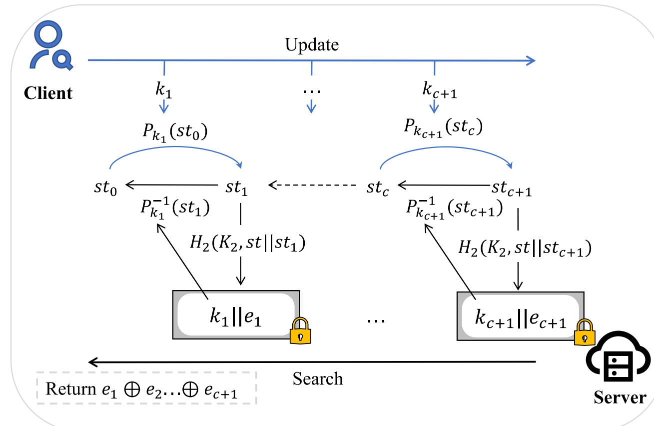


Fig. 6. Extension of VH-RSSE to support dynamic updates.

VI. Security Analysis

A. Leakage Function

As we mentioned in Section IV, the security of our proposed scheme is defined on the basis of the leakage function. Therefore, before analyzing the security, we need to discuss the leakage function first. In particular, the leakage function of VH-RSSE is defined by $\mathcal{L} = \{\mathcal{L}^{\mathsf{Index}}, \mathcal{L}^{\mathsf{Query}}\}$, where $\mathcal{L}^{\mathsf{Index}}$ represents the leakage information from the procedure of building index and $\mathcal{L}^{\mathsf{Query}}$ represents the leakage information from the procedure of querying. According to the output of building index, the server only knows the number of keywords, i.e., m. For a range query Q, the server only knows the size of ST rather than the size of the matched identifiers. Thus, we use $\mu(Q) = |\mathsf{ST}|$ to denote the token size pattern of Q. The definitions of $\mathcal{L}^{\mathsf{Index}}$ and $\mathcal{L}^{\mathsf{Query}}$ are described as follows:

- $\mathcal{L}^{\mathsf{Index}}(\mathsf{DB}) = (m)$,
- $\mathcal{L}^{\mathsf{Query}}(\mathsf{EDB}, Q) = (\mu(Q))$.

Note that the token size pattern would reveal the information that whether a range query contains the minimal or maximal value of a subrange when the token size is 1. We argue that this leakage is acceptable and can be eliminated by randomly generating a "fake" token for the minimal (or maximal) value in each subrange.

B. Security Proof

Based on the leakage function, we now analyze the security of our VH-RSSE (denoted by RSSE hereafter for brevity).

Theorem 1: Let $\mathcal{L} = \{\mathcal{L}^{\mathsf{Index}}, \mathcal{L}^{\mathsf{Query}}\}$ be the leakage function defined in Section VI-A, our proposed scheme $\mathsf{RSSE} = (\mathsf{Setup}, \mathsf{BuildIndex}, \mathsf{Search})$ is $\mathcal{L}$-adaptively-secure, assuming that H_1 and H_2 are two random oracles [47], and F is a secure PRF.

Proof: Similar to the proof in the previous works [12], [13], we define several games G_0, G_1, G_2, G_3, G_4 from $\mathbf{Real}^{\mathsf{RSSE}}_{\mathcal{A}}$ to $\mathbf{Ideal}^{\mathsf{RSSE}}_{\mathcal{A},\mathcal{S}}$. According to the definition in Section IV, we need to prove $\mathbf{Real}^{\mathsf{RSSE}}_{\mathcal{A}}$ and $\mathbf{Ideal}^{\mathsf{RSSE}}_{\mathcal{A},\mathcal{S}}$ are indistinguishable. A common approach is to prove that every two consecutive games from $\mathbf{Real}^{\mathsf{RSSE}}_{\mathcal{A}}$ to $\mathbf{Ideal}^{\mathsf{RSSE}}_{\mathcal{A},\mathcal{S}}$ are indistinguishable.

Game G_0: G_0 straightforwardly performs our proposed scheme. Namely, G_0 is the same as the experiment in $\mathbf{Real}^{\mathsf{RSSE}}_{\mathcal{A}}$. Thus, we have

$$\Pr[G_0 = 1] = \Pr[\mathbf{Real}^{\mathsf{RSSE}}_{\mathcal{A}}(\lambda) = 1]. \tag{2}$$

Game G_1: In G_1, we replace $K_1||K_2 \leftarrow F(K, v)$ by choosing K_1 and K_2 at random and using a map $\mathbb{K}$ to store the pair $(v, K_1||K_2)$. More specifically, given an input v, if there exists an entry v in map $\mathbb{K}$, the corresponding $K_1||K_2$ is returned; otherwise, a random $K_1||K_2$ is returned and stored with the value v in $\mathbb{K}$. We denote this process by $K_1||K_2 \leftarrow \mathbb{K}(v)$ for simplicity.

Assuming that there is a PPT adversary $\mathcal{A}_1$ that can distinguish between G_0 and G_1, then we can construct another PPT adversary $\mathcal{B}_1$ to distinguish $K_1||K_2 \leftarrow F(K, v)$ from the random selection $K_1||K_2 \leftarrow \mathbb{K}(v)$. Since F is a secure PRF, we have

$$\Pr[G_1 = 1] - \Pr[G_0 = 1] \leq \mathsf{Adv}^{\mathsf{prf}}_{F,\mathcal{B}_1}(\lambda), \tag{3}$$

where $\mathsf{Adv}^{\mathsf{prf}}_{F,\mathcal{B}_1}(\lambda)$ is a negligible function that represents the advantage for the adversary $\mathcal{B}_1$ to distinguish F from a truly random function.

Game G_2: We replace the hash function H_1 by a random oracle $\mathbb{H}_1$ such that $bs' \leftarrow \mathbb{H}_1(K_1, v_i)$. Since the length of bs' is L, the probability of a correct guess for bs' by the adversary is $1/2^L$. According to the assumption of random oracle, if the adversary $\mathcal{A}$ issues polynomial number $p(\lambda)$ of queries, then we have

$$\Pr[G_2 = 1] - \Pr[G_1 = 1] \leq p(\lambda)/2^L. \tag{4}$$

Game G_3: Similarly, we replace the hash function H_2 by a random oracle $\mathbb{H}_2$ such that $st \leftarrow \mathbb{H}_2(K_2, v_i)$. As the length of search token st is λ, the probability of a correct guess for search token st by the adversary is also $1/2^\lambda$. According to the assumption of random oracle, if the adversary $\mathcal{A}$ issues polynomial number $p(\lambda)$ of queries, then we have

$$\Pr[G_3 = 1] - \Pr[G_2 = 1] \leq p(\lambda)/2^\lambda. \tag{5}$$

Game G_4: We replace $e \leftarrow bs \oplus \mathbb{H}_1(K_1, v_i)$ by just setting $e \leftarrow \mathbb{H}_1(K_1, v_i)$ in the building phase. According to the assumption of random oracle and the definition of perfectly secret [47], if the adversary $\mathcal{A}$ issues polynomial number $p(\lambda)$ of queries,

Algorithm 5: Simulator $\mathcal{S}$.

$\mathcal{S}.\mathsf{Setup}(1^\lambda)$:
Input: Security Parameter λ.
Output: System Parameters.
Client:
  Initialize L, two maps $\mathbb{K}$ and $\mathbb{V}$; // $L \geq \lambda$
  Initialize two random oracles $\mathbb{H}_1$ and $\mathbb{H}_2$;
  Initialize an empty map EDB, a binary tree BT;
  Return $(L, \mathsf{EDB}, \mathsf{BT})$.
end
$\mathcal{S}.\mathsf{BuildIndex}(\mathcal{L}^{\mathsf{Index}})$:
Input: $\mathcal{L}^{\mathsf{Index}}(\mathsf{DB}) = (m)$.
Output: Encrypted Database EDB.
Client:
  for $i \leftarrow 1$ *to* m **do**
    Generate a random value v_i;
    $K_1||K_2 \leftarrow \mathbb{K}(v_i)$; // Definied in G_1
    $e \leftarrow \mathbb{H}_1(K_1, v_i)$; // Definied in G_4
    $st \leftarrow \mathbb{H}_2(K_2, v_i)$; // Definied in G_3
    $\mathsf{EDB}[st] := e$.
  end
  Return EDB to the adversary $\mathcal{A}$.
end
$\mathcal{S}.\mathsf{Search}(\mathcal{L}^{\mathsf{Query}})$:
Input: $\mathcal{L}^{\mathsf{Query}}(Q) = (\mu(Q))$.
Output: Document Identifiers.
Client:
  if $\mu(Q) = 0$ **then**
    Return $\perp$.
  else
    Initialize an empty search token set ST;
    for $i \leftarrow 1$ *to* $\mu(Q)$ **do**
      Choose the ith value v in $\mathbb{V}(Q)$;
      $K_1||K_2 \leftarrow \mathbb{K}(v)$; // Definied in G_1
      $st \leftarrow \mathbb{H}_2(K_2, v)$; // Definied in G_3
      Append st to ST.
    end
    Return ST to the adversary $\mathcal{A}$.
  end
end

then we have

$$\Pr[G_4 = 1] - \Pr[G_3 = 1] \leq p(\lambda)/2^L. \tag{6}$$

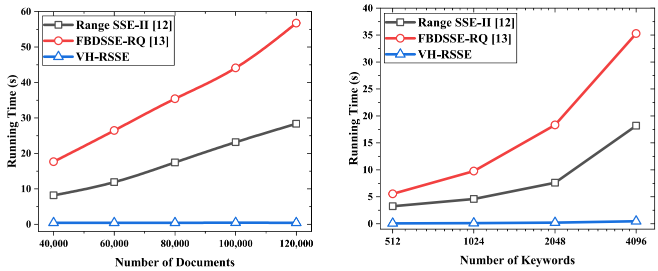
Simulator $\mathcal{S}$: Now we construct the simulator $\mathcal{S} = \{\mathcal{S}.\mathsf{Setup}, \mathcal{S}.\mathsf{BuildIndex}, \mathcal{S}.\mathsf{Search}\}$. Here we use a map $\mathbb{V}$ to store the pair $(Q, \{v_i\}_{i=1}^{\mu(Q)})$, which is similar to the map $\mathbb{K}$. The details of the algorithm is described in Algorithm 5.

Note that the simulator $\mathcal{S}$ takes λ and the leakage function $\mathcal{L} = \{\mathcal{L}^{\mathsf{Index}}, \mathcal{L}^{\mathsf{Query}}\}$ as input, and returns a simulated EDB and a simulated token list ST to the adversary $\mathcal{A}$. Since the procedure of constructing EDB is the same as that in game G_4, then we have

$$\Pr[\mathbf{Ideal}^{\mathsf{RSSE}}_{\mathcal{A},\mathcal{S}}(\lambda) = 1] = \Pr[G_4 = 1]. \tag{7}$$

By using the standard hybrid argument technique [47], it is clear to see that

$$\begin{aligned}\Pr[\mathbf{Ideal}^{\mathsf{RSSE}}_{\mathcal{A},\mathcal{S}}(\lambda) = 1] - \Pr[\mathbf{Real}^{\mathsf{RSSE}}_{\mathcal{S}}(\lambda) = 1]\\ \leq \mathsf{Adv}^{\mathsf{prf}}_{F,\mathcal{B}_1}(\lambda) + p(\lambda)/2^\lambda + 2p(\lambda)/2^L\end{aligned}$$



(a) Running Time of Generating Index when $m = 4,096$ (b) Running Time of Generating Index when $d = 80,000$

Fig. 7. Comparison of running time of generating index.

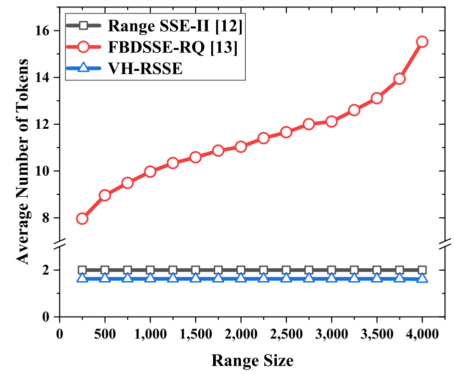


Fig. 8. Number of tokens for different range sizes.

$$\leq \mathsf{Adv}^{\mathsf{prf}}_{F,\mathcal{B}_1}(\lambda) + 3p(\lambda)/2^{\lambda}, \qquad (8)$$

which indicates that our proposed VH-RSSE is secure. ■

VII. Performance Evaluation

In this section, we evaluate the performance of VH-RSSE from various aspects by comparing it with the recently proposed schemes [12], [13]. All experiments are conducted on a computer with an Intel Core i5-10400 CPU (4.30 GHz) and 16 GB RAM running the Linux OS. We adopt the Python Crypto module to implement the cryptographic operations, such as the hash function and the PRFs. For the sake of simplicity, we simulate the instances of the client and server on the same machine.

We choose Gowalla location checkin dataset[1] as the test dataset. Specifically, the Gowalla location checkin dataset contains 6,442,892 entries, and each entry is recorded in the form of *(user, check-in time, latitude, longitude, location id)*. For our experiments, we treat each entry as a document and the value of latitude in each entry as the keyword,[2] which is similar to the experiment setting in [12]. In this way, we assign each entry a unique document identifier and parse the whole dataset as $\mathsf{DB} = (\mathsf{ind}_i, \mathsf{w}_i)_{i=1}^{|\mathsf{DB}|}$. The total number of documents is 6,442,892 and the number of keywords is 335,362. Unless otherwise stated, the default bitmap's lenght L is set to 6,264.

As shown in Table I, FBDSSE-RQ [13] outperforms SchemeA and SchemeB [11], and Range SSE-II [12] is a modified version of SSE-I [12]. Consequently, for our comparative experiments, we only need to choose Range SSE-II [12] and FBDSSE-RQ [13] to evaluate their performance in comparison to our proposed VH-RSSE. To compare with these two schemes, we generate several datasets with various sizes of documents and keywords from the whole dataset. Note that the number of keywords is denoted by m and the number of document identifiers is denoted by d. We first compare the time of generating encrypted index and local binary tree. The result is shown in Fig. 7. It is clear to see that VH-RSSE and Range SSE-II [12] are more efficient than FBDSSE-RQ [13] in generating index. This is because, in FBDSSE-RQ [13], the identifiers for each keyword are disorganized, and the bit string's length is extremely large, which results in time-consuming computation in generating the bitmap index for each keyword. In contrast, both Range SSE-II [12] and our VH-RSSE use a specially constructed inverted index, which is more regular than the normal inverted index, thus leading to efficiency. When the number of keywords is fixed to 4,096, the running time of generating index grows linearly with the number of documents except for VH-RSSE (see Fig. 7(a)). This is due to the fact that the inverted index structure in our scheme only depends on the number of keywords and the bitmap's length. We also observe that when the number of documents is fixed to 80,000, the running time of VH-RSSE grows much slower than that of others (see Fig. 7(b)). The reason behind that is the height of the local tree in VH-RSSE only depends on the number of documents and the bitmap's length.

As illustrated in Fig. 8, we compare the average number of search tokens with respect to different range sizes. Here we set the number of documents to $d = 80,000$, and the number of keywords to $m = 4,096$. The range size represents the number of distinct values within the range. Assume that the range size is q, a straightforward solution is to transform each value into a search token, where the number of search tokens is also q. FBDSSE-RQ [13] adopts the BRC technique to reduce the number of search tokens from q to $O(\log q)$. Benefiting from the order-weighted inverted index, Range SSE-II [12] and VH-RSSE have a smaller number of search tokens. To be specific, the number of search tokens in Range SSE-II [12] is always 2, and in our proposed scheme, the number of search tokens is even smaller, which is at most 2.

As for the storage overhead, Table III lists the size of the encrypted index (stored on the server's side) and local binary tree (stored on the client's side) for each scheme. We can see that when the number of keywords is fixed, the encrypted index size increases linearly with the number of documents except for VH-RSSE. In other words, we eliminate the impact of large document identifiers by splitting the whole range into disjoint subranges. In this way, each keyword in our inverted index only needs to carry a small set of document identifiers within the corresponding subrange. Therefore, the encrypted index size in VH-RSSE is only related to the number of keywords. In addition, we find that the encrypted index size in Range SSE-II [12] is extremely huge, which will cause inefficiency when uploading it to the server. In the comparison of the local binary tree, as

[1] http://snap.stanford.edu/data/loc-gowalla.html.
[2] All keyword values are rounded to 4 decimal places.

TABLE III
COMPARISON OF STORAGE OVERHEAD

Dataset Settings		Range SSE-II [12]		FBDSSE-RQ [13]		VH-RSSE	
		Client	Server	Client	Server	Client	Server
$m = 4,096$	$d = 60,000$	193 KB	527 MB	193 KB	61 MB	37 KB	3,297 KB
	$d = 100,000$	193 KB	928 MB	193 KB	100 MB	38 KB	3,297 KB
	$d = 120,000$	193 KB	1,164 MB	193 KB	120 MB	38 KB	3,297 KB
$d = 80,000$	$m = 2,048$	93 KB	309 MB	93 KB	41 MB	19 KB	1,649 KB
	$m = 4,096$	193 KB	732 MB	193 KB	80 MB	37 KB	3,297 KB
	$m = 8,192$	401 KB	1,538 MB	401 KB	159 MB	73 KB	6,593 KB

TABLE IV
PERFORMANCE OF VH-RSSE ON LARGE-SCALE DATASET

Dataset Setting	Running Time for BuildIndex	Storage Overhead (Client's Side)	Storage Overhead (Server's side)
$m = 2^{16}, d = 10^6$	7.09 s	588 KB	52 MB

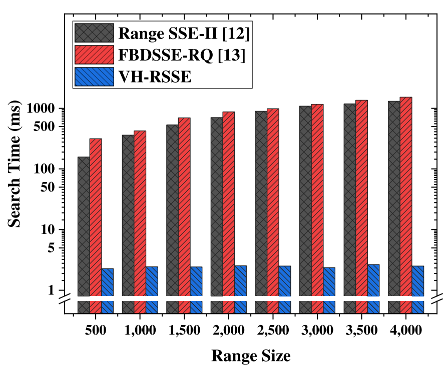


Fig. 9. Comparison of search time for different range sizes.

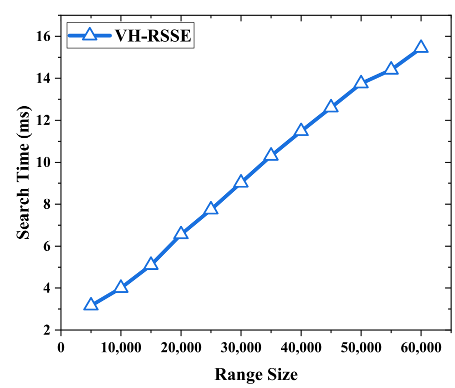


Fig. 10. Search time of VH-RSSE on large-scale dataset.

for VH-RSSE, since we only need to assign each subrange collection to each leaf node, the the storage of local binary tree in VH-RSSE is much lower than that in the other schemes.

We also evaluate the search time for the three schemes. Considering that except for VH-RSSE, the other two schemes are not suitable for large-scale datasets. Thus, we choose a smaller dataset in this experiment, i.e., the number of keywords is 4,096, and the number of documents is 80,000. According to the results in Fig. 9, we find that all three range SSE schemes have an efficient search time. Nevertheless, our proposed scheme achieves the best performance among them.

To examine the performance of VH-RSSE on a large-scale dataset, we also conduct additional experiments under a large-scale dataset where the number of documents is set to 10^6 and the number of keywords is set to 2^{16}. The results are shown in Table IV and Fig. 10. We find that even in a dataset with a million records, the computation time of generating index is still efficient (around 7.09 s), and the storage overhead on the server's side (i.e., the size of EDB) is around 52 MB, which is still small compared to the results of prior arts in Table III. As for the search time, the results in Fig. 10 also demonstrate that our proposed scheme remains highly efficient in search even the range size is extremely large.

VIII. CONCLUSION

In this article, we proposed a novel range SSE scheme that avoids volume-pattern leakage and supports efficient operations on large-scale datasets. By leveraging the order-weighted inverted index and bitmap structure, we designed an order-accumulated inverted index that provides optimal token size and simultaneously conceals volume patterns. Through our partitioning strategy, the proposed scheme achieves high efficiency for range queries on large-scale datasets. Based on the leakage function, we demonstrated the security of our scheme under the ideal/real model simulation paradigm. Finally, we conducted a series of experiments that highlight the practicality of our proposed scheme compared to prior methods, especially for large-scale datasets.

REFERENCES

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

[2] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 79–88.

[3] 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. 33rd Annu. Cryptol. Conf.*, Springer, 2013, pp. 353–373.

[4] S. Lai et al., "Result pattern hiding searchable encryption for conjunctive queries," in *Proc. 25th ACM Conf. Comput. Commun. Secur.*, 2018, pp. 745–762.

[5] S. Kamara and T. Moataz, "Boolean searchable symmetric encryption with worst-case sub-linear complexity," in *Proc. 36th Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, Springer, 2017, pp. 94–124.

[6] 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.

[7] Z. Fu, L. Xia, X. Sun, A. X. Liu, and G. Xie, "Semantic-aware searching over encrypted data for cloud computing," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 9, pp. 2359–2371, Sep. 2018.

[8] L. Chen, Y. Xue, Y. Mu, L. Zeng, F. Rezaeibagha, and R. Deng, "CASE-SSE: Context-aware semantically extensible searchable symmetric encryption for encrypted cloud data," *IEEE Trans. Serv. Comput.*, vol. 16, no. 2, pp. 1011–1022, Mar./Apr. 2023.

[9] S. Faber, S. Jarecki, H. Krawczyk, Q. Nguyen, M. Rosu, and M. Steiner, "Rich queries on encrypted data: Beyond exact matches," in *Proc. 20th Eur. Symp. Res. Comput. Secur.*, Springer, 2015, pp. 123–145.

[10] I. Demertzis, S. Papadopoulos, O. Papapetrou, A. Deligiannakis, and M. Garofalakis, "Practical private range search revisited," in *Proc. Int. Conf. Manage. Data*, 2016, pp. 185–198.

[11] C. Zuo, S.-F. Sun, J. K. Liu, J. Shao, and J. Pieprzyk, "Dynamic searchable symmetric encryption schemes supporting range queries with forward (and backward) security," in *Proc. 23rd Eur. Symp. Res. Comput. Secur.*, Springer, 2018, pp. 228–246.

[12] B. Wang and X. Fan, "Search ranges efficiently and compatibly as keywords over encrypted data," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 6, pp. 1027–1040, Nov./Dec. 2018.

[13] C. Zuo, S. Sun, J. K. Liu, J. Shao, J. Pieprzyk, and L. Xu, "Forward and backward private DSSE for range queries," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 1, pp. 328–338, Jan./Feb. 2022.

[14] J. Wang and S. S. M. Chow, "Forward and backward-secure range-searchable symmetric encryption," in *Proc. Privacy Enhancing Technol.*, vol. 2022, no. 1, pp. 28–48, 2022.

[15] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proc. 22nd ACM Conf. Comput. Commun. Secur.*, 2015, pp. 668–679.

[16] S. Kamara, A. Kati, T. Moataz, T. Schneider, A. Treiber, and M. Yonli, "SoK: Cryptanalysis of encrypted search with LEAKER–A framework for LEakage AttacK Evaluation on Real-world data," in *Proc. IEEE 7th Eur. Symp. Secur. Privacy*, 2022, pp. 90–108.

[17] E. M. Kornaropoulos, C. Papamanthou, and R. Tamassia, "Response-hiding encrypted ranges: Revisiting security via parametrized leakage-abuse attacks," in *Proc. IEEE Symp. Secur. Privacy*, 2021, pp. 1502–1519.

[18] P. Grubbs, M.-S. Lacharite, B. Minaud, and K. G. Paterson, "Pump up the volume: Practical database reconstruction from volume leakage on range queries," in *Proc. 25th ACM Conf. Comput. Commun. Secur.*, 2018, pp. 315–331.

[19] M.-S. Lacharité, B. Minaud, and K. G. Paterson, "Improved reconstruction attacks on encrypted data using range query leakage," in *Proc. IEEE Symp. Secur. Privacy*, 2018, pp. 297–314.

[20] Z. Gui, O. Johnson, and B. Warinschi, "Encrypted databases: New volume attacks against range queries," in *Proc. 26th ACM Conf. Comput. Commun. Secur.*, 2019, pp. 361–378.

[21] I. Demertzis, D. Papadopoulos, C. Papamanthou, and S. Shintre, "SEAL: Attack mitigation for encrypted databases via adjustable leakage," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 2433–2450.

[22] J. Ning et al., "LEAP: Leakage-abuse attack on efficiently deployable, efficiently searchable encryption with partially known dataset," in *Proc. 28th ACM Conf. Comput. Commun. Secur.*, 2021, pp. 2307–2320.

[23] M. Xu, A. Namavari, D. Cash, and T. Ristenpart, "Searching encrypted data with size-locked indexes," in *Proc. 30th USENIX Secur. Symp.*, 2021, pp. 4025–4042.

[24] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order-preserving symmetric encryption," in *Proc. 28th Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, Springer, 2009, pp. 224–241.

[25] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting confidentiality with encrypted query processing," in *Proc. 23rd ACM Symp. Operating Syst. Princ.*, 2011, pp. 85–100.

[26] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *Proc. 22nd ACM Conf. Comput. Commun. Secur.*, 2015, pp. 644–655.

[27] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart, "Leakage-abuse attacks against order-revealing encryption," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 655–672.

[28] K. Xue, S. Li, J. Hong, Y. Xue, N. Yu, and P. Hong, "Two-cloud secure database for numeric-related SQL range queries with privacy preserving," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 7, pp. 1596–1608, Jul. 2017.

[29] K. Cheng et al., "Strongly secure and efficient range queries in cloud databases under multiple keys," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2019, pp. 2494–2502.

[30] G. Xu, H. Li, S. Liu, M. Wen, and R. Lu, "Efficient and privacy-preserving truth discovery in mobile crowd sensing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3854–3865, Apr. 2019.

[31] L. Xu, X. Yuan, C. Wang, Q. Wang, and C. Xu, "Hardening database padding for searchable encryption," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2019, pp. 2503–2511.

[32] L. Blackstone, S. Kamara, and T. Moataz, "Revisiting leakage abuse attacks," in *Proc. 27th Annu. Netw. Distrib. Syst. Secur. Symp.*, 2020.

[33] V. Vo, X. Yuan, S. Sun, J. K. Liu, S. Nepal, and C. Wang, "ShieldDB: An encrypted document database with padding countermeasures," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 4236–4252, Apr. 2023.

[34] S. Kamara and T. Moataz, "Computationally volume-hiding structured encryption," in *Proc. 38th Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, Springer, 2019, pp. 183–213.

[35] S. Patel, G. Persiano, K. Yeo, and M. Yung, "Mitigating leakage in secure cloud-hosted data structures: Volume-hiding for multi-maps via hashing," in *Proc. 26th ACM Conf. Comput. Commun. Secur.*, 2019, pp. 79–93.

[36] M. Ando and M. George, "On the cost of suppressing volume for encrypted multi-maps," in *Proc. Privacy Enhancing Technol.*, vol. 2022, no. 4, pp. 44–65, 2022.

[37] K. Ren et al., "HybrIDX: New hybrid index for volume-hiding range queries in data outsourcing services," in *Proc. 40th Int. Conf. Distrib. Comput. Syst.*, 2020, pp. 23–33.

[38] Y. Chen, Q. Zheng, Z. Yan, and D. Liu, "QShield: Protecting outsourced cloud data queries with multi-user access control based on SGX," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 485–499, Feb. 2021.

[39] J. Gharehchamani, Y. Wang, D. Papadopoulos, M. Zhang, and R. Jalili, "Multi-user dynamic searchable symmetric encryption with corrupted participants," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 114–130, Jan./Feb. 2023.

[40] C. Zuo, S.-F. Sun, J. K. Liu, J. Shao, and J. Pieprzyk, "Dynamic searchable symmetric encryption with forward and stronger backward privacy," in *Proc. 24th Eur. Symp. Res. Comput. Secur.*, Springer, 2019, pp. 283–303.

[41] J. Wang, R. Zhang, J. Li, Y. Xiao, and H. Ma, "SeUpdate: Secure encrypted data update for multi-user environments," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 6, pp. 3592–3606, Nov./Dec. 2022.

[42] F. Li, J. Ma, Y. Miao, Q. Jiang, X. Liu, and K.-K. R. Choo, "Verifiable and dynamic multi-keyword search over encrypted cloud data using bitmap," *IEEE Trans. Cloud Comput.*, vol. 11, no. 1, pp. 336–348, First Quarter 2023.

[43] Z. Gui, K. G. Paterson, and S. Patranabis, "Rethinking searchable symmetric encryption," in *Proc. IEEE Symp. Seucr. Privacy*, 2023, pp. 485–502.

[44] R. Bost, ": Forward secure searchable encryption," in *Proc. 23rd ACM Conf. Comput. Commun. Secur.*, 2016, pp. 1143–1154.

[45] M. Du, Q. Wang, M. He, and J. Weng, "Privacy-preserving indexing and query processing for secure dynamic cloud storage," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 9, pp. 2320–2332, Sep. 2018.

[46] X. Song, C. Dong, D. Yuan, Q. Xu, and M. Zhao, "Forward private searchable symmetric encryption with optimized I/O efficiency," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 5, pp. 912–927, Sep./Oct. 2020.

[47] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in *Proc. 27th Annu. Cryptol. Conf.*, Springer, 2007, pp. 535–552.

Feng Liu received the master's degree from the Department of Computer Science and Engineering, Southern University of Science and Technology (SUSTech), jointly with Harbin Institute of Technology (HIT), in 2019. He is currently working toward the PhD degree with the School of Cyber Science and Technology, University of Science and Technology of China (USTC). From 2019 to 2020, he was a research assistant with the Department of Computer Science and Engineering, SUSTech. His research interests include network security and applied cryptography.

Kaiping Xue (Senior Member, IEEE) received the bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003, and the doctor's degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. From 2012 to 2013, he was a postdoctoral researcher with the Department of Electrical and Computer Engineering, University of Florida. Currently, he is a professor with the School of Cyber Science and Technology, USTC. He is also the director of Network and Information Center, USTC. His research interests include next-generation Internet architecture design, transmission optimization, and network security. He serves on the editorial board of several journals, including *IEEE Transactions on Dependable and Secure Computing (TDSC)*, *IEEE Transactions on Wireless Communications (TWC)*, and *IEEE Transactions on Network and Service Management (TNSM)*. He has also served as a (lead) guest editor for many reputed journals/magazines, including *IEEE Journal on Selected Areas in Communications (JSAC)*, *IEEE Communications Magazine*, and *IEEE Network*. He is an IET fellow.

Jinjiang Yang received the BS degree in information security from the School of Cyber Science and Technology, University of Science and Technology of China (USTC), in 2021. He is currently working toward the graduate degree in information security with the School of Cyber Science and Technology, USTC. His research interests include network security and applied cryptography.

Jing Zhang received the BS degree in electronic information science and technology from the Hefei University of Technology, China, in 2007, and the MS degree in control theory and control engineering from the Anhui University of Science and Technology, Huainan, in 2010. He is currently working toward the PhD degree with Science Island Branch, Graduate School of USTC, Hefei. From 2017 to 2021, he was a communication algorithm engineer with the 38th Research Institute of China Electronics Technology Group Corporation, Hefei. His research interests include space information networks, satellite communication system design, and Ad-hoc networks.

Zixuan Huang received the bachelor's degree in electronic information science and technology from the Hefei University of Technology, Hefei, China, in 2016, and the MS degree in electrical and information engineering from the Harbin Institute of Technology, Shenzhen, in 2019. From 2019 to 2021, she was a communication algorithm engineer with Huawei, ShenZhen and Shanghai. She is currently a communication algorithm engineer with the Institute of Space Integrated Ground Network, Hefei. Her research interests include space information networks, ERR control codes, and Ad-hoc networks.

Jian Li (Senior Member, IEEE) received the bachelor's degree from the Department of Electronics and Information Engineering, Anhui University, in 2015, and the doctor's degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 2020. From 2019 to 2020, he was a visiting scholar with the Department of Electronic and Computer Engineering, University of Florida. From 2020 to 2022, he was a post-doctoral researcher with the School of Cyber Science and Technology, USTC. He is currently an associate researcher with the School of Cyber Science and Technology, USTC. He also serves as an editor of *China Communications*. His research interests include wireless networks, next-generation Internet, and quantum networks.

David S.L. Wei (Senior Member, IEEE) received the PhD degree in computer and information science from the University of Pennsylvania, in 1991. He is currently a professor with the Computer and Information Science Department, Fordham University. He was a lead guest editor or a guest editor for several special issues in *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Cloud Computing*, and *IEEE Transactions on Big Data*. He also served as an associate editor of *IEEE Transactions on Cloud Computing*, 2014–2018, an editor of *IEEE Journal on Selected Areas in Communications* for the Series on Network Softwarization & Enablers, 2018 – 2020, and an associate editor of *Journal of Circuits, Systems and Computers*, 2013–2018. He is the recipient of IEEE Region 1 Technological Innovation Award (Academic), 2020, for contributions to information security in wireless and satellite communications and cyber-physical systems. He is a member of ACM and AAAS, IEEE Computer Society, and IEEE Communications Society. Currently, he focuses his research efforts on cloud and edge computing, cybersecurity, and quantum computing and communications.