Beyond Result Verification: Efficient Privacy-Preserving Spatial Keyword Query With Suppressed Leakage

Qiuyun Tong, Xinghua Li, *Member, IEEE*, Yinbin Miao, *Member, IEEE*, Yunwei Wang, Ximeng Liu, *Senior Member, IEEE*, and Robert H. Deng, *Fellow, IEEE*

***Abstract*— Boolean range query (BRQ) is a typical type of spatial keyword query that is widely used in geographic information systems, location-based services and other applications. It retrieves the objects inside the query range and containing all query keywords. Many privacy-preserving BRQ schemes have been proposed to support BRQ over encrypted data. However, most of them fail to achieve efficient retrieval and lightweight result verification while suppressing access and search pattern leakage. Thus, in this paper, we propose an efficient verifiable privacy-preserving Boolean range query with suppressed leakage. Firstly, we convert BRQ into multi-keyword query by using Gray code and Bloom filter. Then, we achieve efficient oblivious multi-keyword query by combining distributed point function and PRP-based Cuckoo hashing, which protects the access and search patterns. Moreover, we support lightweight and oblivious result verification based on oblivious query, aggregate MAC, keyed-hashing MAC and XOR-homomorphic pseudorandom function. It enables query users to verify the result integrity with a proof whose size is independent of the size of the outsourced dataset. Finally, formal security analysis and extensive experiments demonstrate that our proposed scheme is adaptively secure and efficient for practical applications, respectively.**

***Index Terms*— Privacy-preserving Boolean range query, access pattern, search pattern, result verification.**

I. Introduction

THE proliferation of GPS-equipped mobile devices (*e.g.,* smartphones, cars, UAVs) has facilitated the generation of massive spatio-textual data, which drives spatial keyword query services in diverse applications such as *Geographic Information System (GIS)* [1] and *Location-Based Service (LBS)* [2]. Boolean Range Query (BRQ) [3], as a typical spatial keyword query, retrieves the objects inside the query range and containing all query keywords. For example, a UAV issues a BRQ to find the landing sites within 10 km equipped with charging stations and repair shops. For flexibility and cost savings, the sptaio-textual data and query services tend to be outsourced to Cloud Service Providers (CSPs, *e.g., Amazon AWS, Microsoft Azure, etc.*). However, this raises privacy and security concerns, as CSPs are not fully trusted parties that may try to infer valuable information from outsourced data or queries, or return false results[1] due to various factors (*e.g.,* software/hardware failures, economic incentives, internal dishonesty, etc.) [4], [5], [6], [7]. To tackle these concerns, verifiable Privacy-preserving BRQ (PBRQ) is necessary to protect data and query privacy while achieving result verifiability.

It is also important to protect access pattern and search pattern in (verifiable) PBRQ. Access pattern reveals the identifiers of the objects that match the queries, and search pattern reveals whether two queries are identical. As shown in [8], [9], [10], and [11], given these leakages and some prior knowledge, a passive adversary (*e.g.,* an honest-but-curious CSP) can launch inference attacks or leakage-abuse attacks to reconstruct the dataset or queries. Although these reconstruction attacks are primarily designed for keyword queries and range queries, it is possible to apply them to PBRQ as well.[2] However, most existing PBRQ schemes [12], [13], [14], [15], [16], [17] fail to protect the access and search patterns. A possible solution to suppress these leakages is to employ more secure primitives such as ORAM [18], [19] or Private Information Retrieval (PIR) [20]. Unfortunately, both of them are not suitable for achieving efficient PBRQ. ORAM requires shuffling and encrypting the data blocks for each

Manuscript received 12 September 2023; revised 4 December 2023; accepted 11 January 2024. Date of publication 15 January 2024; date of current version 24 January 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62125205, Grant U23A20303, and Grant 62072361; in part by the Key Research and Development Program of Shaanxi under Grant 2023KXJ-190; and in part by the Fundamental Research Funds for the Central Universities under Grant YJSJ23007. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Meng Li. *(Corresponding author: Xinghua Li.)*

Qiuyun Tong, Yinbin Miao, and Yunwei Wang are with the State Key Laboratory of Integrated Services Networks and the School of Cyber Engineering, Xidian University, Xi'an 710071, China (e-mail: qytong0820@163.com; ybmiao@xidian.edu.cn; wywxidian@foxmail.com).

Xinghua Li is with the State Key Laboratory of Integrated Service Networks, School of Cyber Engineering, Xidian University, Xi'an 710071, China, and also with the Engineering Research Center of Big Data Security, Ministry of Education, Xi'an 710071, China (e-mail: xhli1@mail.xidian.edu.cn).

Ximeng Liu is with the Key Laboratory of Information Security of Network Systems, School of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China (e-mail: snbnix@gmail.com).

Robert H. Deng is with the School of Information Systems, Singapore Management University, Singapore 188065 (e-mail: robertdeng@smu.edu.sg).

Digital Object Identifier 10.1109/TIFS.2024.3354414

[1]False results may occur when CSPs tamper with or forge the outsourced dataset, or execute incorrect search operations.

[2]Similar to the scheme [10], we assume a basic form of leakage where the adversary has knowledge of the entire access pattern and search pattern (*i.e.,* learning which objects are returned for each query). Given the access and search patterns of the queries where the query keyword set $W^* = \varnothing$ or the query range R covers all the points, the adversary in [8], [9], [10], and [11] can launch reconstruction attacks to reveal data privacy or query privacy in PBRQ. Furthermore, for some PBRQ schemes such as [12] that do not build a unified index structure, the adversary has the same knowledge as when performing separate keyword queries and range queries. Thus, the attacks [8], [9], [10], [11] can also be applied to these PBRQ schemes.

query, which is cost-prohibitive for our setting, even for most practical ones. Existing PIR schemes [21], [22], [23], [24] only support private single-keyword queries natively, where an encrypted multi-map is built as the index structure. To perform multi-keyword queries, one has to execute multiple private single-keyword queries, which wastes a tremendous amount of computation resources, especially for homomorphic encryption based PIR. Moreover, these PIR schemes do not support spatial queries, which are essential for PBRQ. Therefore, how to achieve efficient oblivious PBRQ that suppresses the access and search pattern leakage remains a challenge.

Furthermore, the protection of access pattern and search pattern poses a challenge to result verification. The reason is that traditional result verification mechanisms, such as those based on Merkle Hash Tree (MHT) [25], [26], [27], accumulators [28], [29] and signatures [30], [31], are designed for the ciphertext retrieval schemes that leak the access and search patterns. For example, in the MHT-based verifiable retrieval schemes, the proof information is generated by CSP with the knowledge of query results, which consists of all components except the digest of each node on the search path and the digests of its sibling nodes. If we want to use these techniques to achieve result verification while protecting the access and search patterns, a naive method is to return the entire authenticated index structure as the proof information. This method, however, suffers from high communication overhead, which limits the scalability on large-scale datasets. Therefore, it is essential to design a lightweight and oblivious result verification mechanism.

To address the above issues, we propose an efficient and verifiable privacy-preserving Boolean range query with suppressed leakage (called $\mathsf{VPBRQ}_{\mathsf{SupL}}$), which supports efficient PBRQ and lightweight result verification while protecting the access and search patterns. We first convert BRQ into the inner product based multi-keyword query, such that the inner product results indicate whether the objects match the queries. Then, we construct Distributed Multi-Point Function (DMPF) by combining distributed point function and PRP-based Cuckoo hashing to calculate the inner product among multiple CSPs, where at least one CSP is not compromised. Each CSP only obtains one share of each inner product result, thereby concealing the information about which objects match the queries from CSPs and achieving oblivious PBRQ. Moreover, we design a lightweight and oblivious result verification mechanism based on HMAC and XOR-homomorphic pseudorandom function (XHPRF), aggregate MAC and oblivious query. This mechanism authenticates each index entry, and aggregates the authentication tags together before outsourcing to reduce the storage cost of the authenticated index structure. The proof information is obliviously generated based on the oblivious query, and its size does not depend on the size of the object dataset. It can be used to verify the correctness of index entries at the designated queried positions, thereby ensuring that the results passing the verification are correctly retrieved by CSPs and not forged. $\mathsf{VPBRQ}_{\mathsf{SupL}}$ also naturally achieves the result completeness verification as it returns all results. The main contributions of our work are as follows:

- We design a PBRQ construction with access and search pattern hidden. Specifically, we first achieve PBRQ by using Gray code, Bloom filter and one-time pad (or replicated secret sharing), then support efficient oblivious PBRQ based on the distributed multi-point function.
- We design a result verification mechanism to obliviously verify the result correctness. Specifically, we authenticate the index structure using HMAC, XHPRF and aggregate MAC, and obliviously generate the proof information based on the oblivious query, such that the lightweight and oblivious correctness verification can be performed.
- We provide a formal security analysis to show that $\mathsf{VPBRQ}_{\mathsf{SupL}}$ is adaptively secure. We also conduct extensive experiments using a real-world dataset to demonstrate that $\mathsf{VPBRQ}_{\mathsf{SupL}}$ suits resource-limited devices such as UAVs due to microsecond-level query generation and result verification time.

II. Related Work

Unlike textual data, spatio-textual data not only contains location information, but also often contains textual information. In addition to single category queries such as range queries and k-Nearest Neighbor (kNN) query, existing schemes are also proposed to simultaneously retrieve the location and textual information, namely Spatial Keyword Query (SKQ). There are many types of SKQ, such as Boolean range query, Boolean kNN query, top-k kNN query and top-k range query. To perform query processing with data and query privacy, several schemes supporting retrieval on encrypted spatio-textual data have been proposed [12], [13], [14], [15], [16], [17], [32], [33], [34], [35], [36]. The schemes [32], [33] designed an Order-Hiding Encoding (OHE) method to support privacy-preserving geographic range queries and privacy-preserving multi-range queries, respectively. Su et al. [34] designed a top-k kNN query over encrypted data, where the location and textual information are encrypted using Asymmetric Scalar-product-Preserving Encryption (ASPE). Based on similar encryption method, Cui et al. [12] proposed a PBRQ scheme and Tong et al. [35] proposed a privacy-preserving top-k range query scheme. These schemes are only secure against the ciphertext-only attacks. To enhance data confidentiality, Wang et al. [13] exploited the Symmetric-key Hidden Vector Encryption (SHVE) to encrypt data and query vectors, which can resist the selective chosen-plaintext attacks. Miao et al. [16] employed the enhanced ASPE to encrypt an unified index structure built based on Geohash algorithm and bitmap, whose security analysis claims that it has indistinguishability under the chosen plaintext attacks. Unfortunately, these schemes [12], [13], [14], [15], [16], [32], [33], [34], [35] leak access and search patterns.

For retrieval with suppressed leakage, Homomorphic Encryption (HE) based Privacy-preserving SKQ (PSKQ) are proposed. Wang et al. [17] presented a dynamic PBRQ based on Hilbert curve, prefix encoding, bitmap and Additive Homomorphic symmetric Encryption (AHE), where CSP cannot obtain the final object identifiers from the homomorphically encrypted bitmaps, thereby protecting access pattern. However, it leaks the search pattern of each query as CSP only retrieves

TABLE I
COMPARISON BETWEEN PRIOR WORKS AND $\mathsf{VPBRQ}_{\mathsf{SupL}}$

Schemes	Spatial condition	Textual condition	Pattern hiding	Result verification	Encryption method
[12]	Range	Boolean	✗	✗	ASPE
[13]	Range	Boolean	✗	✗	SHVE
[16]	Range	Boolean	✗	✗	Enhanced ASPE
[17]	Range	Boolean	AP	✗	AHE
[32]	Range	—	✗	✗	OHE
[36]	Range	Similarity	AP, SP	✗	FHE
[41]	RkNN	—	AP	✗	SHE
[42]†	kNN	Similarity	—	Correctness	—
[44]†	Range	Boolean	—	Integrity	—
[45]	—	Single	✗	Correctness	PRF
$\mathsf{VPBRQ}_{\mathsf{SupL}}$	Range	Boolean	AP, SP	Integrity	DMPF, RSS one-time pad

Notes. " † " : Retrieval in the plaintext setting; AP: Access pattern; SP: Search pattern; Integrity: Correctness and completeness.

specific index locations. Zhang et al. [36] designed a privacy-preserving top-k range query, where Fully HE (FHE) is used to encrypt the index structure and queries. It provides the access and search pattern privacy since CSP cannot learn which objects are selected as search results from the FHE ciphertexts and the FHE ciphertexts of the same query are different. Unfortunately, it incurs significant computation overhead due to the use of FHE. In addition, the techniques such as ORAM, Distributed Point Function (DPF) and differential privacy have been used to suppress the leakage in keyword query or spatial query [18], [19], [37], [38], [39], [40], [41]. The schemes [18], [19] stored the index structure in ORAM to achieve keyword query and spatial query with completely hiding access pattern. However, they incur significant communication overhead due to frequent reading, re-encrypting and rewriting operations for each query. The schemes [37], [38], [39] applied DPF to the encrypted database setting to achieve private keyword query without leaking access and search patterns. The scheme [40] adopted the differential privacy to obfuscate access pattern in keyword query such that the adversary cannot distinguish queries within a specified distance. The scheme [41] designed a private random tree permutation algorithm based on Symmetric HE (SHE), which supports Reverse kNN (RkNN) query over the encrypted high-dimensional data with access pattern protection.

In addition to data and query privacy, it is essential to support result verifiability. Su et al. [42] supported top-k kNN query processing and authentication by combining IR-tree with MHT. After receiving query responses and the proof information, the query user reconstructed the root hash of IR-tree to check if the spatio-textual objects in the result set are tampered with. The schemes [43], [44] provided verifiable BRQ in blockchain databases, where the correctness and completeness verification of query responses is achieved based on MHT and cryptographic multiset accumulator. However, these schemes [42], [43], [44] just support verifiable retrieval in the plaintext setting. For verifiable retrieval in the ciphertext setting, Guo et al. [45] took the smart contract as a trusted platform to store the encrypted digests of outsourced indexes, but it just supports dynamic single-keyword query.

TABLE II
NOTATION DESCRIPTIONS

Notations	Descriptions
$[\ell]$	$1, 2, \ldots, \ell$
$\|\|$	Concatenation operator
U	Number of CSPs
n	Size of spatio-textual dataset DB
N	Number of query sub-ranges
M	Size of traditional Cuckoo hash table
t	Number of hash functions for BF
κ	Number of hash functions for Cuckoo hash table
m_1	Length of BF storing spatial information
m_2	Length of BF storing textual information
m_1', m_2'	Number of segments for R and W^*
$\mathsf{DB}(Q)$	Objects in the dataset DB that match the query Q

III. PRELIMINARIES

In this section, we briefly introduce the primitives used in our scheme $\mathsf{VPBRQ}_{\mathsf{SupL}}$, which includes Bloom filter, PRP-based Cuckoo hashing, distributed point function and prefix constrained PRF. TABLE II gives a summary of notations used in the following paper.

A. Bloom Filter

Bloom Filter (BF) [46] is a space-efficient data structure that uses t independent uniform hash functions $\mathbf{h} = \{h_i | h_i : \{0,1\}^* \rightarrow [m], i \in [t]\}$ to map a set of elements $X = \{x_1, \ldots, x_n\}$ into an m-bit array b, and supports fast approximate set membership. Initially, all bits in the array b are set to 0. Then, for each element $x \in X$, set $\mathsf{b}[h_i(x)] = 1 (\forall i \in [t])$ and insert it into b. In this way, the membership of a query $\hat{x}$ can be tested by checking if all the bits at $\mathsf{b}[h_1(\hat{x})], \ldots, \mathsf{b}[h_t(\hat{x})]$ are 1. If not, $\hat{x}$ is definitely not in X. Otherwise, $\hat{x}$ is in X with a false positive probability $p \approx (1 - e^{-nt/m})^t$.

Garbled Bloom Filter (GBF) [47] is a variation of BF, where each array position $\mathsf{b}[j]$ stores a ψ-bit string instead of a single bit. For each element $x \in X$, set $\mathsf{b}[j] = \bigoplus_{i \in [t] \setminus \{i'\}} \mathsf{b}[h_i(x)] \oplus \mathtt{fp}(x)$, where $h_{i'}(x) = j$ and $\mathtt{fp}$ is a fingerprint algorithm mapping any length string to an ψ-bit string. In this way, the membership of a query $\hat{x}$ can be tested by checking if $\mathtt{fp}(\hat{x}) = \bigoplus_{i=1}^{t} \mathsf{b}[h_i(\hat{x})]$. Its false positive probability is identical to that of standard BF.

B. PRP-Based Cuckoo Hashing

PRP-based Cuckoo hashing [48] is an extension of traditional Cuckoo hashing. It produces a matrix of size $M \times D$ for a set of elements $X = \{x_1, \ldots, x_m\}$ by using κ pair independent PRP hash functions $\{(\hat{h}_i, v_i)\}_{i=1}^{\kappa}$, which is defined as follows:

$$\begin{aligned} \hat{h}_i(x) &= \lfloor \mathsf{PRP}(\zeta, x + m \cdot (i-1))/D \rfloor, \\ v_i(x) &= \mathsf{PRP}(\zeta, x + m \cdot (i-1)) \mod D, \end{aligned} \tag{1}$$

where ζ is the seed of the pseudorandom permutation $\mathsf{PRP} : \{0,1\}^{\lambda} \times [m\kappa] \rightarrow [m\kappa]$ and $D = \lceil m\kappa / M \rceil$. The hash functions $\hat{h}_1, \ldots, \hat{h}_{\kappa}$ can be used to construct the traditional Cuckoo hash table as follows. First, we initialize a hash table T with M empty buckets. Then, for each element $x_i \in X$,

we randomly select a hash function $\hat{h}_\iota$ and compute its hash value $\hat{h}_\iota(x_i)$. We try to insert x_i into the bucket $T[\hat{h}_\iota(x_i)]$. If the bucket is empty, we succeed. If the bucket is occupied by another element, say x_j, we evict x_j and replace it with x_i. The same process is repeated for x_j until either we find an empty bucket or we exhaust all κ hash functions. If we fail to insert any element, the construction fails. According to the scheme [49], the insertion failure probability is at most 2^{-40} when $\kappa = 3, M = 1.27m$. If the element x is mapped in T using $\hat{h}_\iota$, then its position in the PRP-based Cuckoo hash table is $(\hat{h}_\iota(x), v_\iota(x))$.

C. Distributed Point Function

Distributed Point Function (DPF) [50] enables additive secret sharing of a point function $f_{\alpha,\beta}$ (*i.e.,* for $\alpha, \beta \in \{0,1\}^*$, $f_{\alpha,\beta}(x) = \beta$ if $x = \alpha$ and $0^{|\beta|}$ otherwise) across two or more parties. More concretely, for an U-party DPF ($U \geq 2$), $f_{\alpha,\beta}$ is represented by U shares $\mathrm{k}^{(0)}, \mathrm{k}^{(1)}, \ldots, \mathrm{k}^{(U-1)}$. Any strict set of these shares reveals nothing about $f_{\alpha,\beta}$, but there is an efficient evaluation algorithm Eval holding that $f_{\alpha,\beta}(x) = \bigoplus_{\ell=0}^{U-1} \mathsf{Eval}(\mathrm{k}^{(\ell)}, x)$ for arbitrary input x. Formally, an U-party DPF is defined as the two following algorithms:

- $\mathsf{DPF.Gen}(1^\lambda, f_{\alpha,\beta})$: Given a security parameter λ and a point function $f_{\alpha,\beta}$, output an U-tuple of shares $\mathrm{k}^{(0)}, \mathrm{k}^{(1)}, \ldots, \mathrm{k}^{(U-1)}$.
- $\mathsf{DPF.Eval}(\mathrm{k}^{(\ell)}, x)$: Given a DPF share $\mathrm{k}^{(\ell)}$ and an input x, output an additive share of the value $f_{\alpha,\beta}(x)$.

The 2-party DPF can be applied to 2-server PIR. In such protocol, two parties P_0, P_1 hold identical copy of a dataset $X = \{x_1, \ldots, x_m\}$, and a user wants to retrieve x_α while hiding α from P_0, P_1. To achieve it, the user generates a pair of DPF shares $\mathrm{k}^{(0)}, \mathrm{k}^{(1)}$ by running $\mathsf{DPF.Gen}(1^\lambda, f_{\alpha,1})$. Given the share $\mathrm{k}^{(\ell)}$, P_ℓ returns $res_\ell = \bigoplus_{i=1}^{m} x_i \cdot \mathsf{DPF.Eval}(\mathrm{k}^{(\ell)}, i)$. Then, the user computes $res_0 \oplus res_1$ to reconstruct x_α.

Theorem 1: (Security of DPF) For any $\alpha, \beta \in \{0,1\}^*$, there exists a polynomial-probabilistic-time (PPT) algorithm Sim such that the outputs of $\mathsf{Sim}(1^\lambda, |\alpha|, |\beta|)$ and $\mathsf{DPF.Gen}(1^\lambda, f_{\alpha,\beta})$ are computationally indistinguishable in the presence of up to $U-1$ compromised parities.

D. Prefix Constrained PRF

Prefix constrained PRF [51] is a type of PRF that allows its evaluation to be delegated to an untrusted proxy according to a given prefix predicate. Formally, a prefix constrained PRF F_C with respect to a family of prefix predicates $\mathcal{P} = \{p_{\mathbf{v}} : \mathbf{v} \in \{0,1\}^{s-\iota}\{*\}^\iota, \iota \in [s]\}$ is a mapping $F_C : \mathbf{K} \times \{0,1\}^s \to \mathbf{Y}$, together with two efficient algorithms (F_C.Cons, F_C.Eval), defined as follows:

- $F_C.\mathsf{Cons}(K, p_{\mathbf{v}})$: On input a primary key $K \in \mathbf{K}$ and a prefix predicate $p_{\mathbf{v}} \in \mathcal{P}$, it outputs a constrained key $K_{\mathbf{v}}$.
- $F_C.\mathsf{Eval}(K_{\mathbf{v}}, x)$: On input a constrained key $K_{\mathbf{v}}$ for the prefix $\mathbf{v} \in \{0,1\}^{s-\iota}$ and an $x \in \{0,1\}^\iota$, it outputs the value $F_C(K, \mathbf{v}||x) \in \mathbf{Y}$.

IV. Problem Formulation

In this section, we introduce the system model, threat model, problem definition, security model and design goals.

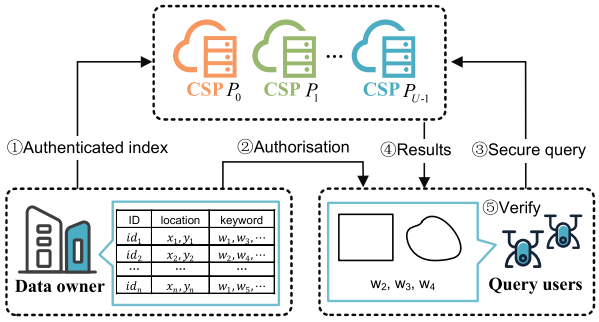


Fig. 1. System model of $\mathsf{VPBRQ_{SupL}}$.

A. System Model & Threat Model

In this paper, we consider a spatio-textual data outsourcing scenario in mobile computing where terminal users such as UAVs can issue search requests. We mainly focus on the index retrieval phase that retrieves the identifiers of the objects matching the queries. The system model of $\mathsf{VPBRQ_{SupL}}$ mainly consists of three entities: Cloud Service Providers (CSPs), Data Owner (DO) and Query Users (QUs), as shown in Fig. 1.

- **Cloud service providers**. With powerful storage and computation capacity, CSPs $P_0, P_1, \ldots, P_{U-1} (U \geq 2)$ provide storage and search services to DO and QUs, respectively.
- **Data owner**. DO generates an authenticated index for its spatio-textual dataset, and uploads it to each CSP.
- **Query users**. The authorized QU issues a search request by generating a secure query, and performs result verification when receiving query results.

DO may be an institution or organization that wants data outsourcing. The spatio-textual dataset contains n objects, each of which consists of spatial location, textual keywords and identifier information. For data confidentiality and searchability, DO builds an authenticated index for the spatio-textual dataset, and uploads it to each CSP (Step ①). DO also provides search permissions to authorized QUs and shares the secret key with them (Step ②). When an authorized QU (*e.g.,* UAV) wants to perform PBRQ, it generates a secure query and sends it to CSPs (Step ③). Each CSP then executes secure retrieval on the authenticated index and returns the results to QU (Step ④). Finally, QU verifies the integrity of received results before decrypting them (Step ⑤). Note that the multi-cloud architecture not only provides fault tolerance, but also facilitates the protection of access pattern and search pattern.

Threat Model: We consider that DO and QUs are fully trusted entities. As described in the schemes [52], [53], [54], they will follow the scheme to honestly construct the authenticated index and generate secure queries, respectively. Moreover, they will not leak the secret key to unauthorized parities or collude with CSPs. Such collusion would either inevitably violate data privacy or incur severe penalties. On the other hand, we assume that CSPs are untrusted entities, as they may be corrupted by malicious adversaries. The malicious adversaries can behave like semi-honest ones, but also take any actions (*e.g.,* tampering with or forging outsourced data,

executing incorrect search operations). We assume that there is at least one CSP not corrupted by such adversaries. That is, up to $U-1$ compromised CSPs may try to infer some valuable information from the outsourced data and retrieval process by launching passive attacks. Moreover, they may deviate from the secure retrieval protocol to cheat QUs by returning false results. In reality, multiple different CSPs (*e.g., Amazon AWS, Microsoft Azure, Linode, Alibaba Cloud, etc.*) can be employed to reduce the probability of all of them being compromised. There is a trade-off between security and performance in this multi-cloud architecture. The more CSPs are involved, the less likely it is that all of them are compromised, but the higher the communication and computation costs are.

B. Problem Definition

Let $\mathsf{DB} = \{O_1, \ldots, O_n\}$ be a spatio-textual dataset owned by DO, where each object O_i consists of a spatial point p_i, a keyword set W_i and a unique identifier id_i, namely $O_i = \{\mathsf{p}_i, W_i, id_i\}$. QU may issue a BRQ $Q = \{R, W^*\}$ in a privacy-preserving manner to retrieve the identifiers of the objects that fall within a geometric range R and contain a set of keywords W^*, *i.e.,* $\mathsf{DB}(Q) = \{id_i | \mathsf{p}_i \in R, W^* \subset W_i\}$. However, existing PBRQ schemes leak access and search patterns, and cannot achieve result verifiability. Techniques such as ORAM and PIR have been utilized to suppress the leakage, but they cannot achieve efficient PBRQ. Moreover, the suppression of the leakage poses a challenge to result verification, as traditional result verification mechanisms leak access and search patterns. A naive solution is to return the entire authenticated index structure as the proof information, but it incurs high communication overhead. Therefore, we study the problem of PBRQ supporting access and search pattern hidden and lightweight result verification, which is defined as follows:

Definition 1: ($\mathsf{VPBRQ_{SupL}}$) *Given a spatio-textual dataset* $\mathsf{DB} = \{O_1, \ldots, O_n\}$ *with* $O_i = \{\mathsf{p}_i, W_i, id_i\}$, $\mathsf{DB}(Q) = \{id_i | \mathsf{p}_i \in R, W^* \subset W_i\}$ *can be correctly retrieved for each query* $Q = \{R, W^*\}$ *without revealing access and search patterns to CSP and with the proof size is independent of n.*

C. Security Model

In this section, we first introduce the security model of $\mathsf{VPBRQ_{SupL}}$ consisting of three algorithms $\Pi = (\mathsf{Setup}, \mathsf{Search}, \mathsf{Verify})$. It ensures that no additional valuable information is revealed to the adversary $\mathcal{A}$ except what can be inferred from the leakage function $\mathcal{L} = \{\mathcal{L}_{\mathsf{Setup}}, \mathcal{L}_{\mathsf{Search}}\}$,[3] where $\mathcal{L}_{\mathsf{Setup}}$ and $\mathcal{L}_{\mathsf{Search}}$ correspond to the information leaked during Setup and Search, respectively. Then, we formally define the verifiability of $\mathsf{VPBRQ_{SupL}}$.

The security model of $\mathsf{VPBRQ_{SupL}}$ is based on the simulation-based game [17], [55]. In particular, we consider the adaptive security, where the PPT adversary $\mathcal{A}$ (corrupting up to $U-1$ CSPs) can adaptively choose each subsequent query based on the leakages of previous queries. Formally, the adaptive security of $\mathsf{VPBRQ_{SupL}}$ is defined as follows:

[3]The algorithm Verify is executed by QUs locally, which indicates that it will not leak any useful information to the adversary $\mathcal{A}$.

Definition 2: ($\mathcal{L}$-adaptive security of $\mathsf{VPBRQ_{SupL}}$) *For any PPT adversary* $\mathcal{A}$ *making a polynomial number of queries with a challenger* $\mathcal{C}$*, our scheme* $\Pi = (\mathsf{Setup}, \mathsf{Search}, \mathsf{Verify})$ *is said to be* $\mathcal{L}$*-adaptively secure if there exists a simulator* $\mathcal{S}$ *such that*

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

where the games $\textsc{Real}^{\Pi}_{\mathcal{A}}(\lambda)$, $\textsc{Ideal}^{\Pi}_{\mathcal{A},\mathcal{S},\mathcal{L}}(\lambda)$ are defined as:

- $\textsc{Real}^{\Pi}_{\mathcal{A}}(\lambda)$: $\mathcal{A}$ selects a dataset DB, then receives an authenticated index AUI from $\mathcal{C}$ who runs Setup. Next, $\mathcal{A}$ adaptively selects a polynomial number of queries. For each query, $\mathcal{C}$ executes Search and returns up to $U-1$ DMPF shares to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs a bit $b \in \{0, 1\}$.
- $\textsc{Ideal}^{\Pi}_{\mathcal{A},\mathcal{S},\mathcal{L}}(\lambda)$: $\mathcal{A}$ selects a dataset DB. Given the leakage $\mathcal{L}_{\mathsf{Setup}}$, $\mathcal{S}$ generates an authenticated index AUI, then sends them to $\mathcal{A}$. Next, $\mathcal{A}$ adaptively selects a polynomial number of queries. For each query, $\mathcal{S}$ generates up to $U-1$ DMPF shares according to the leakage $\mathcal{L}_{\mathsf{Search}}$. Finally, $\mathcal{A}$ outputs a bit $b \in \{0, 1\}$.

The verifiability of $\mathsf{VPBRQ_{SupL}}$ ensures that QUs can detect and reject false query responses. To define the verifiability of $\mathsf{VPBRQ_{SupL}}$, we consider a game between any PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, where $\mathcal{A}$ tries to deceive $\mathcal{C}$ with a false query response. The game consists of the following steps:

(1) $\mathcal{A}$ selects a dataset DB and sends it to $\mathcal{C}$.
(2) $\mathcal{C}$ runs Setup on DB and returns the output AUI to $\mathcal{A}$.
(3) $\mathcal{A}$ adaptively selects a query Q and sends it to $\mathcal{C}$.
(4) $\mathcal{C}$ executes Search with $\mathcal{A}$ and obtains $\mathsf{DB}'(Q)$.
(5) $\mathcal{C}$ executes Search on its own to obtain $\mathsf{DB}(Q)$.
(6) Taking $\mathsf{DB}'(Q)$ as the input of Verify, $\mathcal{C}$ checks whether its output is *true* when $\mathsf{DB}'(Q) \neq \mathsf{DB}(Q)$. If so, $\mathcal{A}$ wins the game. Otherwise, the game continues with the steps (3)-(6) until a polynomial number of queries are made by $\mathcal{A}$.

Definition 3: (Verifiability of $\mathsf{VPBRQ_{SupL}}$) *For any PPT adversary* $\mathcal{A}$ *making a polynomial number of queries with a challenger* $\mathcal{C}$*, our scheme* $\Pi = (\mathsf{Setup}, \mathsf{Search}, \mathsf{Verify})$ *is said to be verifiable if the probability that* $\mathcal{A}$ *wins the above game is negligible, i.e., Pr[*$\mathcal{A}$ *wins]* $\leq negl(\lambda)$.

D. Design Goals

In this paper, we aim to design an efficient and reliable privacy-preserving Boolean range query system, which enables at most $U-1$ compromised CSPs to provide retrieval services for resource-limited devices in a privacy-preserving and verifiable manner. The specific design goals are as follows:

- **Efficiency**. Our scheme should have low computation and communication overheads for QUs, and support fast query processing by CSPs.
- **Adaptive security**. Our scheme should protect the confidentiality of the outsourced data and queries from up to $U-1$ compromised CSPs, who may launch adaptive attack based on the observed information.
- **Lightweight result verification**. Our scheme should be able to verify the integrity of query results with minimal communication cost, such that the size of the proof information is independent of the size of the outsourced dataset.

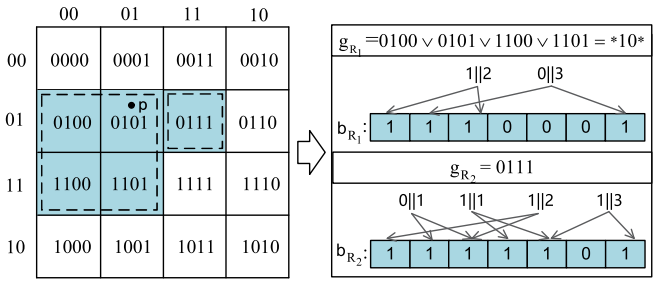


Fig. 2. Query range encoding example.

DMPF construction

- DMPF.Gen($1^\lambda, \{f_{\alpha_j,\beta_j}\}_{j\in[l]}$):
  (1) Insert $\alpha_1, \ldots, \alpha_l$ into a Cuckoo hash table with M buckets using κ hash functions $\hat{h}_1, \ldots, \hat{h}_\kappa$;
  (2) For each bucket i, run DPF.Gen to generate a tuple of DPF shares $(\mathtt{k}_i^{(0)}, \mathtt{k}_i^{(1)}, \ldots, \mathtt{k}_i^{(U-1)})$ for the point function $f_{v_{j'}(\alpha_j),\beta_j}$ if the bucket i holds α_j and $\hat{h}_{j'}(\alpha_j) = i$, or for the zero function $f_{0,0}$ if the bucket i is empty;
  (3) Output $\mathtt{k}^{(\ell)} = (\mathtt{k}_1^{(\ell)}, \ldots, \mathtt{k}_M^{(\ell)}), \ell \in \{0, 1, \ldots, U-1\}$.
- DMPF.Eval($\mathtt{k}^{(\ell)}, x$):
  (1) Compute κ candidate buckets $\hat{h}_1(x), \ldots, \hat{h}_\kappa(x)$;
  (2) Output $\bigoplus_{j=1}^{\kappa}$ DPF.Eval$(\mathtt{k}^{(\ell)}_{\hat{h}_j(x)}, v_j(x))$.

Fig. 3. Detailed procedure of DMPF construction.

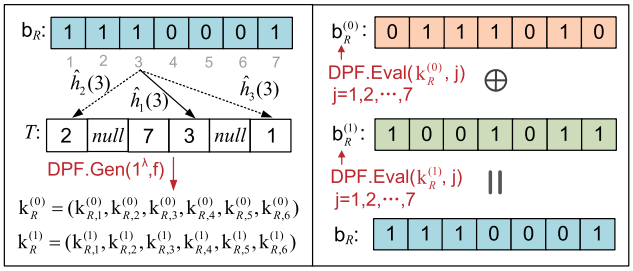


Fig. 4. DMPF example.

V. Our Proposed Scheme

We first introduce the technical overview of $\mathsf{VPBRQ}_{\mathsf{SupL}}$ and specify corresponding concrete construction, then extend it to enable retrieval without local decryption and propose another type of PSKQ based on it.

A. Technical Overview

Existing schemes fail to achieve efficient PBRQ and lightweight result verification while protecting access and search patterns. Techniques such as ORAM and PIR have been utilized to hide the access pattern or search pattern, but they are not suitable for PBRQ in mobile computing. The reason is that ORAM is cost-prohibitive, and PIR only supports private single-keyword queries natively. Furthermore, the protection of access and search patterns poses a challenge to result verification, as it requires CSP to generate the proof information without the knowledge of query results, which contradicts traditional result verification mechanisms. A naive solution is to return the whole authenticated index structure as the proof information, but it suffers from high communication overhead and limits the scalability on large-scale datasets. Therefore, how to achieve efficient PBRQ with suppressed access and search pattern leakage and lightweight result verification remains a challenge.

To solve the above challenge, we first transform BRQ to inner product based multi-keyword query in the building block *Spatio-textual data encoding*. Then, we design Distributed Multi-Point Function (DMPF) in the building block *DMPF-based oblivious search* that applies PRP-based Cuckoo hashing to DPF, such that the inner product can be efficiently calculated without revealing access and search patterns. To achieve lightweight and oblivious result verification, we authenticate the index structure based on HMAC, XHPRF, aggregate MAC and oblivious query in the building block *Lightweight result verification mechanism*. These three building blocks are presented as follows:

1) Spatio-Textual Data Encoding: We first grid the spatial space into $L \times L$ cells, and then encode each cell into a unique code using the Gray code [56]. After that, we represent each spatial point p as the code of the cell it located, and the query range R as the minimum code set of the cells inside R. For example, in Fig. 2 the spatial space is divided into 4×4 cells. The spatial point p is encoded as $\mathsf{g_p} = 0101$ and the query range (blue area) is encoded as $\mathsf{g}_{R_1} = 0100 \vee 0101 \vee 1100 \vee 1101 = *10*$, $\mathsf{g}_{R_2} = 0111$. We then transform the Gray codes of each spatial point p and the query range R into keywords, and store them into Bloom filters. For $\mathsf{g_p} = 0101$, we take $\mathcal{W}(\mathsf{g_p}) = \{0||1, 1||2, 0||3, 1||4\}$ as four keywords, and map them to a garbled Bloom filter $\mathsf{b_p}$ such that $\bigoplus_{i=1}^{t} \mathsf{b_p}[h_i(w)] = \mathtt{fp}(w)$ for each keyword $w \in \mathcal{W}(\mathsf{g_p})$. Similarly, for $\mathsf{g}_{R_1} = *10*, \mathsf{g}_{R_2} = 0111$, we obtain $\mathcal{W}(\mathsf{g}_{R_1}) = \{1||2, 0||3\}, \mathcal{W}(\mathsf{g}_{R_2}) = \{0||1, 1||2, 1||3, 1||4\}$, then map them to the Bloom filters $\mathsf{b}_{R_1}, \mathsf{b}_{R_2}$ respectively, such that $\mathsf{b}_{R_t}[h_i(w)] = 1 (\forall i \in [t])$ for each keyword $w \in \mathcal{W}(\mathsf{g}_{R_t})$. As a result, the computation result $\mathsf{b_p} \cdot \mathsf{b}_{R_1} = \bigoplus_{i=1}^{7} \mathsf{b_p}[i]\mathsf{b}_{R_1}[i]$ is equal to $\mathtt{fp}(1||2) \oplus \mathtt{fp}(0||3)$, which indicates that the spatial point p is inside the query range R. Furthermore, if the query keyword set W^* is the subset of the keyword set W, then the object $O = \{\mathsf{p}, W, id\}$ is a BRQ result of the query $Q = \{R, W^*\}$. Note that the keyword query can also be achieved based on BF, *i.e.,* checking $\mathsf{b}_W \cdot \mathsf{b}_{W^*} = \bigoplus_{w\in W^*} \mathtt{fp}(w)$.

DMPF-based Oblivious Search. Derived from [48], we combine DPF and PRP-based Cuckoo hashing to construct DMPF, as shown in Fig. 3. We then utilize it to achieve efficient oblivious range queries. Suppose that the query range R is represented by a Bloom filter b_R with length of m_1. Let S_R be the set of queried/non-zero columns in b_R. QU first generates

$$(\mathtt{k}_R^{(0)}, \mathtt{k}_R^{(1)}, \ldots, \mathtt{k}_R^{(U-1)}) \leftarrow \mathsf{DMPF.Gen}(1^\lambda, \{f_{i,\mathsf{b}_R[i]}\}_{i\in S_R}).$$

Then, QU sends $\mathtt{k}_R^{(\ell)}$ to CSP P_ℓ for each $\ell \in \{0, 1, \ldots, U-1\}$. To evaluate each column $j \in [m_1]$, CSP P_ℓ runs DMPF.Eval $(\mathtt{k}_R^{(\ell)}, j)$ to output $\langle \mathsf{b}_R[j] \rangle_\ell$. Taking an example in Fig. 4 with $U = 2$, assume that b_R is equal to $\mathsf{b}_{R_1} = (1, 1, 1, 0, 0, 0, 1)$, then we have $S_R = \{1, 2, 3, 7\}$ and QU inserts it into a Cuckoo hash table T using 3 hash functions. The size of T is $M = \lceil 1.27 \cdot 4 \rceil = 6$. To evaluate the column 3, each CSP gets 3 candidate buckets $\hat{h}_1(3) = 4, \hat{h}_2(3) = 1, \hat{h}_3(3) = 6$.

Then, P_0 outputs $\mathsf{DPF.Eval}(\mathrm{k}_{R,4}^{(0)}, 3) \oplus \mathsf{DPF.Eval}(\mathrm{k}_{R,1}^{(0)}, 3) \oplus \mathsf{DPF.Eval}(\mathrm{k}_{R,6}^{(0)}, 3) = 1$ and P_1 outputs $\mathsf{DPF.Eval}(\mathrm{k}_{R,4}^{(1)}, 3) \oplus \mathsf{DPF.Eval}(\mathrm{k}_{R,1}^{(1)}, 3) \oplus \mathsf{DPF.Eval}(\mathrm{k}_{R,6}^{(1)}, 3) = 0$. In this way, P_0, P_1 obtain $\mathsf{b}_R^{(0)} = (0, 1, 1, 1, 0, 1, 0)$, $\mathsf{b}_R^{(1)} = (1, 0, 0, 1, 0, 1, 1)$, respectively. Then, P_ℓ computes the inner product $\mathsf{b_p} \cdot \mathsf{b}_R^{(\ell)}$, and sends the computation result to QU. QU computes $\mathsf{b_p} \cdot \mathsf{b}_R^{(0)} \oplus \mathsf{b_p} \cdot \mathsf{b}_R^{(1)}$ and checks if the result is equal to $\mathsf{b_p} \cdot \mathsf{b}_R = \mathtt{fp}(1||2) \oplus \mathtt{fp}(0||3)$. Similarly, we can achieve efficient oblivious keyword queries via DMPF.

During the retrieval, each CSP only knows a portion of the computation result for each object, so it cannot infer which objects matching the queries, thereby protecting the access pattern. Moreover, $\mathsf{DPF.Gen}$ is a non-deterministic algorithm, which means that any subset of the compromised CSPs (up to $U-1$) cannot infer whether two queries are the same based on the DMPF shares owned by them. Therefore, DMPF-based BRQ also protects the search pattern.[4] In addition, one of the advantages of using DMPF over DPF for oblivious search is that DMPF conceals the number of non-zero columns in the query Bloom filters, while DPF reveals this information to CSPs. This enhances query privacy.

2) Lightweight Result Verification Mechanism: After encoding n spatio-textual objects, DO obtains n Bloom filters with length of m_1 for storing spatial points and n Bloom filters with length of m_2 for storing textual information, denoting as $n \times m_1$ spatial matrix $\mathsf{B_{spa}}$ and $n \times m_2$ textual matrix $\mathsf{B_{tex}}$, respectively. To ensure the correctness of computation results, we use MAC with a secret key $\mathcal{K}$ to authenticate each entry of the matrix $\mathsf{B}[i][j]$ as well as its position (i, j) and identifier id_i, *e.g.,* $\mathsf{MAC}(\mathcal{K}, \mathsf{B}[i][j], \mathrm{N}_{i,j,id_i})$, where N_{i,j,id_i} contains the position and identity information of the entry $\mathsf{B}[i][j]$ and $\mathsf{B} \in \{\mathsf{B_{spa}}, \mathsf{B_{tex}}\}$. However, the tag matrix introduces additional storage overhead for CSPs. It also increases the communication overhead, as n proofs generated based on the tag matrix need to be returned. To reduce it, we use aggregate MAC [58] to generate an aggregate tag for each column of the matrix, *e.g.,* $\sigma_j = \bigoplus_{i=1}^{n} \mathsf{MAC}(\mathcal{K}, \mathsf{B}[i][j], \mathrm{N}_{i,j,id_i})$. Now, QU runs *DMPF-based oblivious search* on both the matrix B and the tag vector $\sigma = (\sigma_1, \ldots \sigma_m)$ to obtain n computation results $\mathbf{res_b}$ and a single proof Γ_b, where $\mathbf{res_b}[i] = \bigoplus_{j=1}^{m}(\mathsf{B}[i][j] \cdot \mathsf{b}[j])$ for $i \in [n]$ and $\Gamma_\mathsf{b} = \bigoplus_{j=1}^{m}(\sigma_j \cdot \mathsf{b}[j])$. Note that $\mathsf{b} = \mathsf{b}_R$, $m = m_1$ if $\mathsf{B} = \mathsf{B_{spa}}$ and $\mathsf{b} = \mathsf{b}_{W^*}$, $m = m_2$ if $\mathsf{B} = \mathsf{B_{tex}}$. Then, we can obtain the BRQ result $\mathbf{res_{BRQ}} = \mathbf{res}_R \oplus \mathbf{res}_{W^*}$ and corresponding proof $\Gamma_{\mathsf{BRQ}} = \Gamma_R \oplus \Gamma_{W^*}$. QU can verify the correctness of $\mathbf{res_{BRQ}}$ by checking

$$\Gamma_{\mathsf{BRQ}} = \bigoplus\nolimits_{i\in[n]} \mathsf{MAC}(\mathcal{K}, \mathbf{res_{BRQ}}[i], \bigoplus\nolimits_{j\in S_R \cup S_{W^*}} \mathrm{N}_{i,j,id_i}).$$

To support such verification, MAC must have the property of XOR homomorphism, which means that $\mathsf{MAC}(\mathcal{K}, x_1 \oplus x_2, \mathrm{N}_1 \oplus \mathrm{N}_2) = \mathsf{MAC}(\mathcal{K}, x_1, \mathrm{N}_1) \oplus \mathsf{MAC}(\mathcal{K}, x_2, \mathrm{N}_2)$. Therefore, we define

$$\mathsf{MAC}(\mathcal{K}, \mathsf{B}[i][j], \mathrm{N}_{i,j,id_i}) = F_{\mathrm{X}}(K_i, \mathsf{B}[i][j]) \oplus \mathrm{N}_{i,j,id_i}, \quad (2)$$

[4]As described in [57], the search pattern leakage occurs when the query always generates the same trapdoor or the same access pattern. Our solution avoids these two situations, so it protects the search pattern.

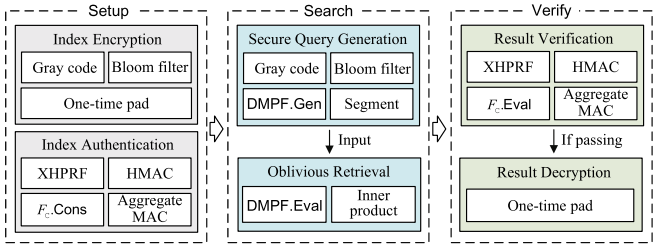


Fig. 5. Framework of $\mathsf{VPBRQ_{SupL}}$.

where F_{X} is an XOR-homomorphic PRF and $\mathcal{K} = (K_1, \ldots, K_n, K_h)$. We have

$$\sigma_j = \bigoplus\nolimits_{i\in[n]} F_{\mathrm{X}}(K_i, \mathsf{B}[i][j]) \oplus \mathrm{N}_{j,ID}. \quad (3)$$

Here, $\mathrm{N}_{j,ID} = \bigoplus_{i=1}^{n} \mathrm{N}_{i,j,id_i}$ is defined as $\mathsf{HMAC}(K_h, j||id_1||\ldots||id_n)$, which ensures that the BRQ result is obtained by retrieved at designated positions derived from the query. To reduce QU's storage cost, K_i can be generated by using the prefix constrained PRF $F_{\mathrm{C}} : \{0,1\}^\lambda \times \{0,1\}^s \to \{0,1\}^\lambda$ with respect to a family of prefix predicates $\mathcal{P} = \{p_{\mathbf{v}} : \mathbf{v} \in \{0,1\}^{s-\lceil \log n \rceil}\{*\}^{\lceil \log n \rceil}\}$, such that the secret key sent to QU is $\mathcal{K} = (K_{\mathbf{v}}, K_h)$ rather than $\mathcal{K} = (K_1, \ldots, K_n, K_h)$. Specifically, DO runs $F_{\mathrm{C}}.\mathsf{Cons}(K, p_{\mathbf{v}})$ to generate a constrained key $K_{\mathbf{v}}$ and sends it to QU. Then, QU runs $F_{\mathrm{C}}.\mathsf{Eval}(K_{\mathbf{v}}, i)$ to obtain $K_i = F_{\mathrm{C}}(K, \mathbf{v}||i)$ for $i \in [n]$.

B. Concrete Construction of $\mathsf{VPBRQ_{SupL}}$

The construction of $\mathsf{VPBRQ_{SupL}}$ can be divided into three algorithms: Setup, Search and Verify, as shown in Fig. 5. In Setup, DO generates an authenticated index structure through two steps: index encryption and index authentication. In Search, QU first generates a secure query and sends it to CSPs, who then perform oblivious retrieval and return the results to QU. In Verify, QU first performs result verification, then decrypts the verified results to extract matched identifiers. The specific process is shown as follows.

$\mathsf{Setup}(1^\lambda, \mathsf{DB})$: We consider a spatio-textual dataset $\mathsf{DB} = \{O_1, \ldots, O_n\}$. To protect data confidentiality and achieve result integrity verification, DO constructs, encrypts and authenticates an index structure shown in **Algorithm 1**, which consists of the following steps.

- For each object $O_i = \{\mathsf{p}_i, W_i, id_i\}$, DO sets up two Bloom filters: $\mathsf{b}_{\mathsf{p},i}$ with length m_1 to represent the spatial point p_i and $\mathsf{b}_{W,i}$ with length m_2 to represent the keyword set W_i. The details of the Bloom filter construction are described in *Spatio-textual data encoding*.
- DO randomly picks a secret key $K_\mathrm{e} \in \{0,1\}^\lambda$ for the PRF $F : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^{(m_1+m_2)\psi}$, and computes a one-time pad pad_i for each object O_i. Then, DO uses pad_i to encrypt $\mathsf{b}_{\mathsf{p},i}$ and $\mathsf{b}_{W,i}$ as $\mathsf{Eb}_{\mathsf{p},i}$ and $\mathsf{Eb}_{W,i}$ respectively (Lines 5-8). Correspondingly, an $n \times m_1$ encrypted spatial matrix I_{spa} and an $n \times m_2$ encrypted textual matrix I_{tex} is constructed, where each row corresponds to an object.
- DO first randomly selects a primary key $K \in \{0,1\}^\lambda$ and a prefix $\mathbf{v} \in \{0,1\}^{s-\lceil \log n \rceil}$, then uses them to generate $K_i (i \in [n])$ for $F_{\mathrm{X}} : \{0,1\}^\lambda \times \{0,1\}^\psi \to \{0,1\}^\psi$.

Algorithm 1 Setup Process

Input: Security parameter λ, the spatio-textual dataset DB
Output: Authentication index AUI, secret key $\mathcal{K}$

1 **foreach** $O_i \in \mathsf{DB}$ **do**
2 $\quad \mathsf{b}_{\mathsf{p},i}, \mathsf{b}_{W,i} \xleftarrow{encode} O_i$;
3 $\quad K_{\mathrm{e}} \xleftarrow{\$} \{0,1\}^{\lambda}$;
4 $\quad$ Compute $\mathsf{pad}_i = F(K_{\mathrm{e}}, i||id_i)$;
5 $\quad$ **for** $j = 1; j \le m_1; j++$ **do**
6 $\quad\quad \mathsf{Eb}_{\mathsf{p},i}[j] = \mathsf{b}_{\mathsf{p},i}[j] \oplus \mathsf{pad}_i[(j-1)\psi : j\psi]$;
7 $\quad$ **for** $j = 1; j \le m_2; j++$ **do**
8 $\quad\quad \mathsf{Eb}_{W,i}[j] = \mathsf{b}_{W,i}[j] \oplus \mathsf{pad}_i[(j+m_1-1)\psi : (j+m_1)\psi]$;
9 Set $I_{\mathsf{spa}} = (\mathsf{Eb}_{\mathsf{p},1}, \ldots, \mathsf{Eb}_{\mathsf{p},n})$, $I_{\mathsf{tex}} = (\mathsf{Eb}_{W,1}, \ldots, \mathsf{Eb}_{W,n})$;
10 $K, K_h \xleftarrow{\$} \{0,1\}^{\lambda}$, $\mathbf{v} \xleftarrow{\$} \{0,1\}^{s-\lceil \log n \rceil}$;
11 Compute $K_i = F_{\mathrm{C}}(K, \mathbf{v}||i)$ for $i \in [n]$;
12 **for** $j = 1; j \le m_1; j++$ **do**
13 $\quad \sigma_{\mathsf{spa},j} = \bigoplus_{i\in[n]} F_{\mathrm{X}}(K_i, I_{\mathsf{spa}}[i][j]) \oplus \mathsf{HMAC}(K_h, j||id_1||\ldots||id_n)$;
14 **for** $j = 1; j \le m_2; j++$ **do**
15 $\quad \sigma_{\mathsf{tex},j} = \bigoplus_{i\in[n]} F_{\mathrm{X}}(K_i, I_{\mathsf{tex}}[i][j]) \oplus \mathsf{HMAC}(K_h, (j+m_1)||id_1||\ldots||id_n)$;
16 Set $\sigma_{\mathsf{spa}} = (\sigma_{\mathsf{spa},1}, \ldots, \sigma_{\mathsf{spa},m_1})$, $\sigma_{\mathsf{tex}} = (\sigma_{\mathsf{tex},1}, \ldots, \sigma_{\mathsf{tex},m_2})$;
17 Set $\tilde{I}_{\mathsf{spa}} = (I_{\mathsf{spa}}, \sigma_{\mathsf{spa}})$, $\tilde{I}_{\mathsf{tex}} = (I_{\mathsf{tex}}, \sigma_{\mathsf{tex}})$, $AUI = \{\tilde{I}_{\mathsf{spa}}, \tilde{I}_{\mathsf{tex}}\}$;
18 Compute $K_{\mathbf{v}} = F_{\mathrm{C}}.\mathsf{Cons}(K, p_{\mathbf{v}})$;
19 Set $\mathcal{K} = (K_{\mathrm{e}}, K_{\mathbf{v}}, K_h)$;
20 **return** $AUI, \mathcal{K}$

Moreover, DO creates the secret key $K_h \in \{0,1\}^{\lambda}$ for $\mathsf{HMAC} : \{0,1\}^{\lambda} \times \{0,1\}^{*} \to \{0,1\}^{\psi}$. Next, DO authenticates I_{spa} (*resp.* I_{tex}) by computing an aggregate tag $\sigma_{\mathsf{spa},j}$ (*resp.* $\sigma_{\mathsf{tex},j}$) for its each column j (Lines 12-15). In particular, F_{X} can be constructed as $F_{\mathrm{X}}(K_i, u) = \bigoplus_{\iota=1}^{\psi} \mathsf{PRF}(K_i, u[\iota]||\iota)$, where $u[\iota]$ is the ι-th bit of u and $\mathsf{PRF} : \{0,1\}^{\lambda} \times \{0,1\}^{*} \to \{0,1\}^{\psi}$. Obviously, F_{X} preserves the XOR homomorphism for odd number of XOR operations. To preserve this property for even number, XOR $F_{\mathrm{X}}(K_i, 0^{\psi})$ additionally.

- DO sets $\tilde{I}_{\mathsf{spa}} = (I_{\mathsf{spa}}, \sigma_{\mathsf{spa}})$, $\tilde{I}_{\mathsf{tex}} = (I_{\mathsf{tex}}, \sigma_{\mathsf{tex}})$, and uploads the authenticated index $AUI = \{\tilde{I}_{\mathsf{spa}}, \tilde{I}_{\mathsf{tex}}\}$ to each CSP P_ℓ. DO also generates a constrained key $K_{\mathbf{v}}$, and shares the secret key $\mathcal{K} = (K_{\mathrm{e}}, K_{\mathbf{v}}, K_h)$ with authorized QUs. Note that $\tilde{I}_{\mathsf{spa}}$ and $\tilde{I}_{\mathsf{tex}}$ are $(n+1) \times m_1$ and $(n+1) \times m_2$ matrices, respectively.

$\mathsf{Search}(AUI, Q)$: As shown in **Algorithm 2**, given the query $Q = \{R, W^*\}$, QU constructs a set of Bloom filters $B_R = \{\mathsf{b}_{R_1}, \ldots, \mathsf{b}_{R_N}\}$, each with length m_1, to represent the query range R. QU also constructs a Bloom filter b_{W^*} with length m_2 to represent the query keyword set W^*. The details of the Bloom filter construction are given in *Spatio-textual data encoding*. Then, QU and CSPs execute the *DMPF-based oblivious search* protocol to retrieve the matching object identifiers without revealing the access and search patterns. The specific process is shown as follows. Here, we optimize DMPF using the segmentation method [39] to further improve retrieval efficiency.

Algorithm 2 Search Process

Input: Query Q, authenticated index AUI
Output: Result shares $\mathbf{R}^{(0)}, \mathbf{R}^{(1)}, \ldots, \mathbf{R}^{(U-1)}$

1 QU:
2 $B_Q = \{\mathsf{b}_{R_1}, \ldots, \mathsf{b}_{R_N}, \mathsf{b}_{W^*}\} \xleftarrow{encode} Q$;
3 **foreach** $\mathsf{b} \in B_Q$ **do**
4 $\quad$ Split b into m' segments $\mathbf{seg} = (seg_1, \ldots, seg_{m'})$;
5 $\quad$ Set $S' = \{i | seg_i \neq 0^{L_{seg}}\}$;
6 $\quad$ $(\mathrm{k}_{\mathsf{b}}^{(0)}, \mathrm{k}_{\mathsf{b}}^{(1)}, \ldots, \mathrm{k}_{\mathsf{b}}^{(U-1)}) \leftarrow \mathsf{DMPF.Gen}(1^{\lambda}, \{f_{i,seg_i}\}_{i\in S'})$;
7 Send $\mathrm{K}^{(\ell)} = \{\mathrm{k}_{\mathsf{b}}^{(\ell)}\}_{\mathsf{b}\in B_Q}$ to P_ℓ;
8 P_ℓ **for** $\ell \in \{0, 1, \ldots, U-1\}$:
9 **for** $j = 1; j \le m_2'; j++$ **do**
10 $\quad seg_j^{(\ell)} \leftarrow \mathsf{DMPF.Eval}(\mathrm{k}_{W^*}^{(\ell)}, j)$;
11 Set $\bar{\mathsf{b}}_{W^*}^{(\ell)} = (seg_1^{(\ell)}, \ldots, seg_{m_2'}^{(\ell)})$;
12 Compute $\mathbf{res}_{W^*}^{(\ell)} = \bigoplus_{i=1}^{m_2} (\tilde{I}_{\mathsf{tex}}[:, i] \cdot \bar{\mathsf{b}}_{W^*}^{(\ell)}[i])$;
13 **foreach** $\mathrm{k}_{R_\iota}^{(\ell)} \in \{\mathrm{k}_{R_1}^{(\ell)}, \ldots, \mathrm{k}_{R_N}^{(\ell)}\}$ **do**
14 $\quad$ **for** $j = 1; j \le m_1'; j++$ **do**
15 $\quad\quad seg_j^{(\ell)} \leftarrow \mathsf{DMPF.Eval}(\mathrm{k}_{R_\iota}^{(\ell)}, j)$;
16 $\quad$ Set $\bar{\mathsf{b}}_{R_\iota}^{(\ell)} = (seg_1^{(\ell)}, \ldots, seg_{m_1'}^{(\ell)})$;
17 $\quad$ Compute $\mathbf{res}_{R_\iota}^{(\ell)} = \bigoplus_{i=1}^{m_1} (\tilde{I}_{\mathsf{spa}}[:, i] \cdot \bar{\mathsf{b}}_{R_\iota}^{(\ell)}[i])$;
18 $\quad$ Compute $\mathbf{res}_{\iota}^{(\ell)} = \mathbf{res}_{R_\iota}^{(\ell)} \oplus \mathbf{res}_{W^*}^{(\ell)}$;
19 Set $\mathbf{R}^{(\ell)} = \{\mathbf{res}_1^{(\ell)}, \ldots, \mathbf{res}_N^{(\ell)}\}$;
20 **return** $\mathbf{R}^{(0)}, \mathbf{R}^{(1)}, \ldots, \mathbf{R}^{(U-1)}$

For each Bloom filter $\mathsf{b} \in B_R \cup \{\mathsf{b}_{W^*}\}$ with corresponding length $m \in \{m_1, m_2\}$, QU splits it into $m' = \lceil m/L_{seg} \rceil$ segments $\mathbf{seg} = (seg_1, \ldots, seg_{m'})$, where $L_{seg} \in [m]$ is the length of each segment.[5] Then, QU extracts the positions of the non-zero segments to form the set $S' = \{i | seg_i \neq 0^{L_{seg}}\}$. Taking the point functions $\{f_{i,seg_i}\}_{i\in S'}$ as inputs, QU runs $\mathsf{DMPF.Gen}$ to generate U DMPF shares $(\mathrm{k}_{\mathsf{b}}^{(0)}, \mathrm{k}_{\mathsf{b}}^{(1)}, \ldots, \mathrm{k}_{\mathsf{b}}^{(U-1)})$. Fianlly, QU sends $\mathrm{k}_{\mathsf{b}}^{(\ell)}$ to CSP P_ℓ for each $\ell \in \{0, 1, \ldots, U-1\}$. We find that the segmentation method reduces QU's computation and bandwidth overheads, as QU only needs to generate and upload $1.27|S'|$ pair DPF shares instead of $1.27|S|(> 1.27|S'|)$ pairs. In addition, the use of PRP-based Cuckoo hashing shrinks the domain size of DPF from m to $D(< m)$, which also reduces QU's computation and bandwidth overheads. The reason is that the performance of $\mathsf{DPF.Gen}$ and the size of the DPF share linearly increase with the domain size of DPF [59]. Fig. 6 gives an example of the optimized DMPF with the parameters $U = 2, \kappa = 3, m = 7$,

[5]The length of the last segment $seg_{m'}$ may be less than L_{seg}. We can add zeros on the left to make it L_{seg} in length.

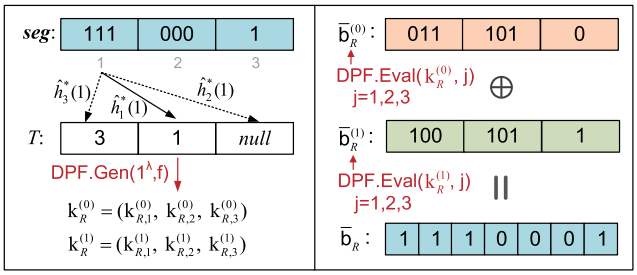


Fig. 6. Optimized DMPF example.

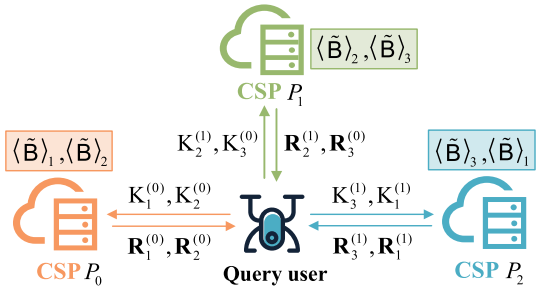


Fig. 7. Query evaluation with RSS.

$L_{seg} = 3, S'_R = \{1, 3\}, M = \lceil 1.27 \cdot 2 \rceil = 3, m' = \lceil 7/3 \rceil = 3, D = \lceil 2 \cdot 3/3 \rceil = 2$. QU only performs DPF.Gen three times instead of six times (2× speed up) to split the Bloom filter b_R into $(\mathtt{k}_R^{(0)}, \mathtt{k}_R^{(1)})$, and the domain size of DPF is shrunk from 7 to 2.

Upon receiving the search request, CSP P_ℓ first runs DMPF.Eval to evaluate the DMPF share $\mathtt{k}_{W^*}^{(\ell)}$ at each column $i \in [m'_2]$ and obtains a vector $\bar{\mathsf{b}}_{W^*}^{(\ell)}$ of length m_2. Then, CSP P_ℓ computes the inner product of $\bar{\mathsf{b}}_{W^*}^{(\ell)}$ and $\tilde{I}_{\mathsf{tex}}$ to obtain $\mathbf{res}_{W^*}^{(\ell)}$ (Lines 9-12). Similarly, CSP P_ℓ evaluates $\mathtt{k}_{R_\iota}^{(\ell)} (\iota \in [N])$ to obtain a vector $\bar{\mathsf{b}}_{R_\iota}^{(\ell)}$ of length m_1, which is used to compute the inner product with $\tilde{I}_{\mathsf{spa}}$ to obtain $\mathbf{res}_{R_\iota}^{(\ell)}$ (Lines 13-17). Finally, CSP P_ℓ computes the BRQ result share $\{\mathbf{res}_\iota^{(\ell)}\}_{\iota \in [N]}$ by combining the textual and spatial ones. The proof information is $\{\Gamma_\iota^{(\ell)}\}_{\iota \in [N]}$, where $\Gamma_\iota^{(\ell)} = \mathbf{res}_\iota^{(\ell)}[n+1]$. Thanks to the optimized DMPF, each CSP only needs to execute $\kappa(Nm'_1 + m'_2)$ DPF evaluations under the domain size D, instead of $\kappa(Nm_1 + m_2)$ as in the original DMPF. Fig. 6 shows that P_ℓ performs $3 \cdot 3$ DPF.Eval instead of $3 \cdot 7$ (2.3× speed up) to obtain $\bar{\mathsf{b}}_R^{(\ell)}$.

Verify$(\mathcal{K}, Q, \{\mathbf{R}^{(\ell)}\}_{\ell=0}^{U-1}, ID)$: QU first reconstructs

$$\mathbf{R} = \mathbf{R}^{(0)} \oplus \mathbf{R}^{(1)} \oplus \cdots \oplus \mathbf{R}^{(U-1)}. \tag{4}$$

Then, QU verifies the correctness of each result $\mathbf{res}_\iota \in \mathbf{R} (\iota \in [N])$. The proof information is $\Gamma_\iota = \mathbf{res}_\iota[n+1]$. QU first computes the HMACs of corresponding queried column set $S_\iota = S_{R_\iota} \cup \{m_1 + s\}_{s \in S_{W^*}}$, and aggregates them together as $\mathtt{N}_{S_\iota, ID} = \bigoplus_{j \in S_\iota} \mathsf{HMAC}(K_h, j || id_1 || \ldots || id_n)$. QU also computes the key $K_i = F_{\mathrm{C}}.\mathsf{Eval}(K_{\mathbf{v}}, i)$ for $i \in [n]$. After that, QU checks whether the Eq. 5 holds or not, where $Aux = \bigoplus_{i \in [n]} F_{\mathrm{X}}(K_i, 0^\psi)$ if $|S_\iota|$ is a even number and 0 otherwise.

$$\Gamma_\iota = \bigoplus\nolimits_{i \in [n]} F_{\mathrm{X}}(K_i, \mathbf{res}_\iota[i]) \oplus \mathtt{N}_{S_\iota, ID} \oplus Aux. \tag{5}$$

If so, the result $\mathbf{res}_\iota$ are correct and QU decrypts it to extract the matched identifiers. Specifically, QU first computes the mask $\Delta_{S_\iota, i}$ for $\mathbf{res}_\iota[i] (i \in [n])$ in Eq. 6.

$$\Delta_{S_\iota, i} = \bigoplus\nolimits_{j \in S_\iota} F(K_{\mathrm{e}}, i || id_i)[(j-1) \cdot \psi : j \cdot \psi]. \tag{6}$$

Then, QU checks Eq. 7 holds or not. If it holds, then add id_i to the matched identifier set.

$$\mathbf{res}_\iota[i] = \bigoplus\nolimits_{w \in \mathcal{W}(\mathsf{g}_{R_\iota}) \cup W^*} \mathtt{fp}(w) \oplus \Delta_{S_\iota, i}. \tag{7}$$

Remark. When obtaining the matched identifier set, QU can fetch corresponding object ciphertexts from CSPs. To prevent CSPs from learning the access and search patterns during this process, some techniques such as PathORAM [37] and keyword PIR [22], [23] can be directly used to achieve oblivious fetch. In addition, considering that the compromised CSPs may temper with or forge the returned objects, we can combine keyword PIR with authenticated PIR [60], which ensures both obliviousness and verifiability of the object ciphertext retrieval.

However, a remaining challenge for $\mathsf{VPBRQ}_{\mathsf{SupL}}$ is to handle large-scale datasets efficiently. The main bottleneck is the result decryption time, which grows linearly with the size of the outsoucred dataset n and dominates the Verify time. According to our experiments, it takes about 36 s for a Raspberry Pi to perform result decryption when $n = 100000$, indicating that the result decryption consumes a lot of computation resources of resource-limited devices. In addition, considering the needs of users for different query types, it is desirable to support other types of PSKQ. For these concerns, we extent $\mathsf{VPBRQ}_{\mathsf{SupL}}$ to support retrieval without local decryption and propose an efficient and oblivious Privacy-preserving Boolean kNN Query (PBkQ).

C. $\mathsf{VPBRQ}_{\mathsf{SupL}}$ Without Local Decryption

To reduce QUs' computation overheads, we adopt Replicated Secret Sharing (RSS) [61] instead of one-time pad to protect index confidentiality, which enables QUs to extract the matched identifiers without local decryption. $\mathsf{VPBRQ}_{\mathsf{SupL}}$ with RSS allows QUs to perform efficient oblivious PBRQ without communication between CSPs, as shown in Fig. 7.

In Setup, when encoding each object $O_i = \{\mathsf{p}_i, W_i, id_i\}$ into two Bloom filters $\mathsf{b}_{\mathsf{p},i}, \mathsf{b}_{W,i}$, DO authenticates $\mathsf{B}_{\mathsf{spa}} = (\mathsf{b}_{\mathsf{p},1}, \ldots, \mathsf{b}_{\mathsf{p},n})$ and $\mathsf{B}_{\mathsf{tex}} = (\mathsf{b}_{W,1}, \ldots, \mathsf{b}_{W,n})$ as does in the building block *Lightweight result verification mechanism* to obtain two aggregate tag vectors $\sigma_{\mathsf{spa}}, \sigma_{\mathsf{tex}}$. Then, DO sets $\tilde{\mathsf{B}}_{\mathsf{spa}} = (\mathsf{B}_{\mathsf{spa}}, \sigma_{\mathsf{spa}}), \tilde{\mathsf{B}}_{\mathsf{tex}} = (\mathsf{B}_{\mathsf{tex}}, \sigma_{\mathsf{tex}})$ to form the authenticed index $AUI = \{\tilde{\mathsf{B}}_{\mathsf{spa}}, \tilde{\mathsf{B}}_{\mathsf{tex}}\}$. Next, DO uses RSS to secretly share AUI among three non-colluding CSPs P_0, P_1, P_2. For each $\tilde{\mathsf{B}} \in AUI$, DO randomly splits it into three shares $\langle\tilde{\mathsf{B}}\rangle_1, \langle\tilde{\mathsf{B}}\rangle_2, \langle\tilde{\mathsf{B}}\rangle_3 \in \mathbb{Z}_{2^\psi}^{(n+1)\times m}$ such that $\langle\tilde{\mathsf{B}}\rangle_1 \oplus \langle\tilde{\mathsf{B}}\rangle_2 \oplus \langle\tilde{\mathsf{B}}\rangle_3 = \tilde{\mathsf{B}}$, where $m = m_1$ if $\tilde{\mathsf{B}} = \tilde{\mathsf{B}}_{\mathsf{spa}}$ and $m = m_2$ otherwise. Each CSP receives a pair of shares: P_0 gets $(\langle\tilde{\mathsf{B}}\rangle_1, \langle\tilde{\mathsf{B}}\rangle_2)$, P_1 gets $(\langle\tilde{\mathsf{B}}\rangle_2, \langle\tilde{\mathsf{B}}\rangle_3)$, and P_2 gets $(\langle\tilde{\mathsf{B}}\rangle_3, \langle\tilde{\mathsf{B}}\rangle_1)$.

In Search, QU generates three pairs of DMPF shares $(\mathtt{K}_1^{(0)}, \mathtt{K}_1^{(1)}), (\mathtt{K}_2^{(0)}, \mathtt{K}_2^{(1)}), (\mathtt{K}_3^{(0)}, \mathtt{K}_3^{(1)})$ by executing DMPF.Gen three times. Then, QU sends $(\mathtt{K}_1^{(0)}, \mathtt{K}_2^{(0)}), (\mathtt{K}_2^{(1)}, \mathtt{K}_3^{(0)}), (\mathtt{K}_3^{(1)}, \mathtt{K}_1^{(1)})$ to

CSPs P_0, P_1, P_2, respectively. CSP P_0 executes **Algorithm 2** from line 9 to 17 to evaluate $(\langle\tilde{\mathsf{B}}\rangle_1, \langle\tilde{\mathsf{B}}\rangle_2)$, and returns the evaluation result $(\mathbf{R}_1^{(0)}, \mathbf{R}_2^{(0)})$ to QU. CSPs P_1, P_2 perform similar operations to return $(\mathbf{R}_2^{(1)}, \mathbf{R}_3^{(0)})$, $(\mathbf{R}_3^{(1)}, \mathbf{R}_1^{(1)})$, respectively.

In Verify, QU first computes

$$\mathbf{R}_\tau = \mathbf{R}_\tau^{(0)} \oplus \mathbf{R}_\tau^{(1)}, \tau \in \{1, 2, 3\}$$
$$\mathbf{R} = \mathbf{R}_1 \oplus \mathbf{R}_2 \oplus \mathbf{R}_3. \tag{8}$$

Then, QU performs the result verification and the matched identifier extraction as does in Eq. 5 and Eq. 7, respectively.

D. Efficient and Oblivious PBkQ

PBkQ aims to retrieve the objects that contain all query keywords and are kNN to the query point. It faces similar challenges as existing PBRQ schemes, *i.e.,* how to achieve efficient retrieval and lightweight result verification while suppressing access and search pattern leakage. Thus, we aim to design a verifiable and oblivious efficient PBkQ. The main idea is to transform kNN query into the inner product based multi-keyword query, and then use *DMPF-based oblivious search* and *Lightweight result verification mechanism* that we introduced in Section V-A. The transformation process is described as follows.

We adopt the composite projection functions [62] to encode each spatial point into a series of regions with increasing interval lengths $d_1, d_2, \ldots, d_l$, then store them to Bloom filters. A projection function $g : \mathbb{R}^2 \to \mathbb{Z}$ is defined as

$$g_{\mathsf{a},b,d}(\mathsf{p}) = \left\lfloor \frac{\mathsf{a} \cdot \mathsf{p} + b}{d} \right\rfloor, \tag{9}$$

where $\mathsf{a} = (\theta, r)$ is a vector with angle $\theta \in [0, 2\pi)$ and magnitude $r = 1$ in the polar coordinate system, and b is a random number chosen from $[0, d)$. By first u AND-composition and then v OR-composition of multiple projection functions, a spatial point $\mathsf{p} = (x, y)$ is encoded into a code $C_{d_i}(\mathsf{p}) = \{\mathtt{H}_{i,1}(\mathsf{p}), \mathtt{H}_{i,2}(\mathsf{p}), \ldots, \mathtt{H}_{i,v}(\mathsf{p})\}$ for each interval length d_i, where $\mathtt{H}_{i,j}(\mathsf{p}) = i||j||g_{\mathsf{a}_1,b_i,d_i}(\mathsf{p}) \,||\cdots||\, g_{\mathsf{a}_u,b_i,d_i}(\mathsf{p})$. The code $C_{d_i}(\mathsf{p})$ represents a feasible region $R_{d_i}(\mathsf{p})$ that contains all the possible points p' having at least one common element with p (*i.e.,* $\exists j \in [v], s.t.\ \mathtt{H}_{i,j}(\mathsf{p}) = \mathtt{H}_{i,j}(\mathsf{p}')$). Similarly, we encode the query point q into l codes $\mathcal{C}_\mathsf{q} = \{C_{d_1}(\mathsf{q}), \ldots, C_{d_l}(\mathsf{q})\}$, each corresponding to a feasible region $R_{d_i}(\mathsf{q})$. These regions have the successive inclusion property, *i.e.,* $R_{d_1}(\mathsf{q}) \subset \cdots \subset R_{d_l}(\mathsf{q})$. We can test whether a spatial point p is in $R_{d_i}(\mathsf{q})$ by checking if $C_{d_i}(\mathsf{q}) \cap C_{d_i}(\mathsf{p}) \neq \varnothing$. If the spatial point p_1 is in $R_{d_1}(\mathsf{q})$ and the spatial point p_2 is in $R_{d_2}(\mathsf{q})$ but not in $R_{d_1}(\mathsf{q})$, then p_1 is closer to q than p_2. Based on this observation, we can retrieve the kNN of q by scanning its feasible regions from the smallest to the largest until k spatial points are found. Next, we take the elements in $\mathcal{C}_\mathsf{p} = \{C_{d_1}(\mathsf{p}), \ldots, C_{d_l}(\mathsf{p})\}$ as $l \cdot v$ keywords and map them to a garbled Bloom filter b_p such that $\bigoplus_{i=1}^{t} \mathsf{b}_\mathsf{p}[h_i(w)] = \mathtt{fp}(w)$ for each keyword w in $\mathcal{C}_\mathsf{p}$. Moreover, we create $l \cdot v$ Bloom filters $\{\mathsf{b}_{\mathsf{q}_{i,j}}\}_{i\in[l], j\in[v]}$, one for each element in $\mathcal{C}_\mathsf{q}$. $\mathsf{b}_\mathsf{p} \odot \mathsf{b}_{\mathsf{q}_{1,1}} = \mathtt{fp}(\mathtt{H}_{1,1}(\mathsf{p}))$ indicates that the spatial point p is in the smallest feasible region of q. If the number of such objects is less than k, we repeat the process with $\mathsf{b}_{\mathsf{q}_{1,2}}$ and so on, until we find k objects or exhaust all the Bloom filters.

One limitation of the above construction is that it may return approximate results instead of accurate ones. Due to the similarity of the design process to $\mathsf{VPBRQ}_{\mathsf{SupL}}$, we will specify it in the future work.

VI. Security Analysis

In this section, we first prove the $\mathcal{L}$-adaptive security of $\mathsf{VPBRQ}_{\mathsf{SupL}}$ defined in Definition 2, and then prove the verifiability of $\mathsf{VPBRQ}_{\mathsf{SupL}}$ defined in Definition 3.

Before proving the $\mathcal{L}$-adaptive security in **Theorem 2**, we will explain the specific information leaked by our scheme $\mathsf{VPBRQ}_{\mathsf{SupL}}$, namely the specific components of $\mathcal{L}_{\mathsf{Setup}}$ and $\mathcal{L}_{\mathsf{Search}}$. In $\mathsf{VPBRQ}_{\mathsf{SupL}}$, CSPs receive an authenticated index AUI at the phase Setup, which reveals the size of spatio-textual dataset n, the length of BF storing spatial information m_1, the length of BF storing textual information m_2 and the length of each item in GBF ψ. At the phase Search, each CSP P_ℓ receives the DMPF share set $\mathtt{K}^{(\ell)}$, which reveals the number of BFs used to represent the query range N, the length of each segment L_{seg}, the number of segments and the size of PRP-based Cuckoo hash table for the query range m_1', M_1 and for the query keyword set m_2', M_2. In other words, we have $\mathcal{L}_{\mathsf{Setup}} = (n, m_1, m_2, \psi)$ and $\mathcal{L}_{\mathsf{Search}} = (N, L_{seg}, m_1', m_2', M_1, M_2)$.

Theorem 2: $\mathsf{VPBRQ}_{\mathsf{SupL}}$ is $\mathcal{L}$-adaptively secure if F, F_{X} are two secure pseudorandom functions, HMAC is a secure message authentication code and DPF is a simulation-secure distributed point function.

Proof: The proof proceeds with a sequence of games below, and we demonstrate the computational indistinguishability between each other.

Game G_0: G_0 is identical to the real-world game $\mathrm{REAL}_{\mathcal{A}}^{\Pi}(\lambda)$:

$$\Pr[\mathrm{REAL}_{\mathcal{A}}^{\Pi}(\lambda) = 1] = \Pr[G_0 = 1]. \tag{10}$$

Game G_1: G_1 is identical to G_0 except that the simulator $\mathcal{S}$ generates a random string using a truly random function $g_\lambda : \{0,1\}^* \to \{0,1\}^{(m_1+m_2)\psi}$, and uses it to replace $\mathsf{Eb}_\mathsf{p}, \mathsf{Eb}_W$ of each object. Since $\mathcal{A}$ cannot computationally distinguish between g_λ and PRF F with a non-negligible probability, the advantage of $\mathcal{A}$ in distinguishing G_0 and G_1 is bound by the advantage of breaking the security of PRF $Adv_{\mathcal{A},F}^{\mathsf{PRF}}(\lambda)$, *i.e.,*

$$\Pr[G_0 = 1] - \Pr[G_1 = 1] \leq Adv_{\mathcal{A},F}^{\mathsf{PRF}}(\lambda). \tag{11}$$

Game G_2: G_2 is identical to G_1 except that $\mathcal{S}$ uniformly chooses $m_1 + m_2$ random strings from the space $\{0,1\}^\psi$, and uses them to replace the aggregate tags in σ_{spa} and σ_{tex}. Since HMAC can be modeled as a Random Oracles (RO) and the secret keys $K_1, \ldots, K_n, K_h$ are randomly generated and keep secret from $\mathcal{A}$, HMAC and F_{X} are computationally indistinguishable from a truly random function with output space $\{0,1\}^\psi$. Thus, the advantage of $\mathcal{A}$ in distinguishing G_2 and G_1 is bound by the advantage of breaking the security of PRF $Adv_{\mathcal{A},F_{\mathrm{X}}}^{\mathsf{PRF}}(\lambda)$ and the security of HMAC $Adv_{\mathcal{A},\mathsf{HMAC}}^{\mathsf{RO}}(\lambda)$, *i.e.,*

$$\Pr[G_1 = 1] - \Pr[G_2 = 1] \leq Adv_{\mathcal{A},F_{\mathrm{X}}}^{\mathsf{PRF}}(\lambda) \cdot Adv_{\mathcal{A},\mathsf{HMAC}}^{\mathsf{RO}}(\lambda).$$

Game G_3: G_3 is identical to G_2 except that $\mathcal{S}$ replaces $\mathsf{DPF.Gen}(1^\lambda, f_{\alpha,\beta})$ with the PPT algorithm $\mathsf{Sim}(1^\lambda, |\alpha|, |\beta|)$.

TABLE III
THEORETICAL COMPUTATION AND COMMUNICATION COSTS IN DIFFERENT SCHEMES

Types	Schemes	Setup	Search	Verify
		DO	QU and CSPs	QU
Comp. costs	PPSKS	$n(m_x+m_y+m_2)\mathsf{T}_{\mathsf{SHE.Enc}}$	$(n_x m_x+n_y m_y+m_2\lvert W^*\rvert)\mathsf{T}_{\mathsf{SHE.Enc}}+$ $n(n_x+n_y+\lvert W^*\rvert)(\mathsf{T}_{\mathsf{Dot}}+\mathsf{T}_{\mathsf{Poly}})+n\mathsf{T}_{\mathsf{Scom}}$	—
	$\mathsf{VPBRQ}_{\mathsf{SupL}}$	$(m_1+m_2)(n\mathsf{T}_{F_{\mathsf{X}}}+\mathsf{T}_{\mathsf{HMAC}})$ $+n\mathsf{T}_F+n\mathsf{T}_{F_{\mathsf{C}}}$	$(NM_1+M_2)\mathsf{T}_{\mathsf{DPF.Gen}}+U(Nm_1'+m_2')\kappa\mathsf{T}_{\mathsf{DPF.Eval}}$ $+U(n+1)(N+1)\mathsf{T}_{\mathsf{I}}$	$n(\mathsf{T}_{F_{\mathsf{C}}.\mathsf{Eval}}+\mathsf{T}_{F_{\mathsf{X}}})+nN\mathsf{T}_F+$ $\sum_{\iota\in[N]}\lvert S_\iota\rvert\mathsf{T}_{\mathsf{HMAC}}$
Comm. costs	PPSKS	$2k_0n(m_x+m_y+m_2)$	$(k_1+2k_0)(n_x m_x+n_y m_y+\lvert W^*\rvert m_2)+2k_0n$	—
	$\mathsf{VPBRQ}_{\mathsf{SupL}}$	$U(m_1+m_2)(n+1)\psi$	$(NM_1+M_2)b_{key}+U(n+1)N\psi$	—

Notes. m_x, m_y: Length of BF storing the data point (x, y) respectively; n_x, n_y: Number of BFs representing the ranges R_x, R_y respectively; $\mathsf{T}_{\mathsf{Dot}}$: Time complexity of inner product; $\mathsf{T}_{\mathsf{Poly}}$: Time complexity of polynomial computation; $\mathsf{T}_{\mathsf{Scom}}$: Time complexity of secure comparison; $\mathsf{T}_{\mathsf{SHE}}$: Time complexity of symmetric homomorphic encryption; k_0, k_1: Security parameters of SHE.

According to **Theorem 1**, the advantage of $\mathcal{A}$ in distinguishing G_3 and G_2 is bound by the advantage of breaking the security of DPF $Adv_{\mathcal{A}}^{\mathsf{DPF}}(\lambda)$, *i.e.,*

$$\Pr[G_2=1]-\Pr[G_3=1]\leq Adv_{\mathcal{A}}^{\mathsf{DPF}}(\lambda) \tag{12}$$

Game G_4: G_4 is identical to G_3 except that $\mathcal{S}$ gets rid of redundant steps that do not provide any advantage to $\mathcal{A}$ in distinguishing between G_4 and G_3. Thus, we have

$$\Pr[G_4=1]=\Pr[G_3=1] \tag{13}$$

The simulator $\mathcal{S}$ of G_4 is given in **Algorithm 3**. Essentially, $\mathcal{S}$ simulates the ideal-world game $\textsc{Ideal}_{\mathcal{A},\mathcal{S},\mathcal{L}}^{\Pi}(\lambda)$. Obviously, the inputs are the pre-defined leakages $\mathcal{L}_{\mathsf{Setup}}, \mathcal{L}_{\mathsf{Search}}$, and the outputs are the same to $\textsc{Ideal}_{\mathcal{A},\mathcal{S},\mathcal{L}}^{\Pi}(\lambda)$. Thus, we have

$$\Pr[G_4=1]=\Pr[\textsc{Ideal}_{\mathcal{A},\mathcal{S},\mathcal{L}}^{\Pi}(\lambda)=1] \tag{14}$$

Combining all the games above, the advantage of any PPT adversary $\mathcal{A}$ in distinguishing the real-world game $\textsc{Real}_{\mathcal{A}}^{\Pi}(\lambda)$ from the ideal-world game $\textsc{Ideal}_{\mathcal{A},\mathcal{S},\mathcal{L}}^{\Pi}(\lambda)$ is

$$\begin{aligned}&\lvert\Pr[\textsc{Real}_{\mathcal{A}}^{\Pi}(\lambda)=1]-\Pr[\textsc{Ideal}_{\mathcal{A},\mathcal{S},\mathcal{L}}^{\Pi}(\lambda)=1]\rvert\\&\leq Adv_{\mathcal{A},F}^{\mathsf{PRF}}(\lambda)+Adv_{\mathcal{A},F_{\mathsf{X}}}^{\mathsf{PRF}}(\lambda)\cdot Adv_{\mathcal{A},\mathsf{HMAC}}^{\mathsf{RO}}(\lambda)+Adv_{\mathcal{A}}^{\mathsf{DPF}}(\lambda).\end{aligned}$$

■

Similarly, $\mathsf{VPBRQ}_{\mathsf{SupL}}$ proposed in Section V-C also can be proven to be $\mathcal{L}$-adaptively secure if F_{X} is a secure pseudo-random functions, HMAC is a secure message authentication code, DPF is a simulation-secure distributed point function, and RSS is a computationally secure protocol in the presence of one semi-honest corrupted party, so does the efficient and oblivious PBkQ proposed in Section V-D.

Theorem 3: $\mathsf{VPBRQ}_{\mathsf{SupL}}$ is verifiable for the result correctness and completeness if F_{X} is a secure pseudorandom function and HMAC is a secure message authentication code.

Proof: Suppose that the adversary $\mathcal{A}$ outputs a valid response $\mathbf{R}$ for a query $Q_i(\exists i\in[poly(\lambda)])$, which implies that $\mathcal{A}$ is able to generate a n-dimensional vector $\mathbf{res}$ and a value σ such that

$$\begin{aligned}\sigma&=\bigoplus\nolimits_{i\in[n]}F_{\mathsf{X}}(K_i,\mathbf{res}[i])\oplus\mathtt{N}_{S,ID}\oplus Aux,\\ \mathtt{N}_{S,ID}&=\bigoplus\nolimits_{j\in S}\mathsf{HMAC}(K_h,j||id_1||\ldots||id_n).\end{aligned} \tag{15}$$

Since the secret keys $K_1,\ldots,K_n$ keep secret from $\mathcal{A}$, PRF F_{X} is computationally indistinguishable from a truly random function. The randomness of F_{X} hides $\mathtt{N}_{S,ID}$ such that $\mathcal{A}$ cannot infer any information about $\mathtt{N}_{S,ID}$. Moreover, HMAC can be modeled as a RO. If the secret key K_h keeps secret for $\mathcal{A}$, HMAC is also computationally indistinguishable from a truly random function. Therefore, the probability that $\mathcal{A}$ can deceives QU with a false query response successfully is

$$\Pr[\mathcal{A}\text{ wins}]\leq\sum_{i=1}^{poly(\lambda)}\lvert B_{Q_i}\rvert\cdot Adv_{\mathcal{A},P}^{\mathsf{PRF}}(\lambda)\cdot Adv_{\mathcal{A},\mathsf{HMAC}}^{\mathsf{RO}}(\lambda).$$

■

VII. PERFORMANCE ANALYSIS

We analyze the performance of $\mathsf{VPBRQ}_{\mathsf{SupL}}$ theoretically and experimentally by comparing it with PPSKS [36].

A. Theoretical Analysis

We present the computation and communication costs of our scheme and the comparison scheme in TABLE III.

As for computation cost of our scheme, we mainly consider several time-consuming operations: PRFs F_{X}, F, HMAC, DPF and inner product. Let $\mathsf{T}_{F_{\mathsf{X}}}, \mathsf{T}_F, \mathsf{T}_{\mathsf{HMAC}}, \mathsf{T}_{\mathsf{DPF}}, \mathsf{T}_{\mathsf{I}}$ be corresponding time complexities. In Setup, DO computes a one-time pad for each object, generates m_1+m_2 aggregate tags for the encrypted index and generates n secret keys $K_1,\ldots,K_n$, which costs $n\mathsf{T}_F+(m_1+m_2)(n\mathsf{T}_{F_{\mathsf{X}}}+\mathsf{T}_{\mathsf{HMAC}})+n\mathsf{T}_{F_{\mathsf{C}}}$. In Search, QU first runs the algorithm $\mathsf{DMPF.Gen}$ to generate DMPF shares for N query sub-ranges and a query keyword set, which costs $(NM_1+M_2)\mathsf{T}_{\mathsf{DPF.Gen}}$. Then, each CSP runs the algorithm $\mathsf{DPF.Eval}$ to generate the vector shares for N query sub-ranges and the query keyword set, and uses them to evaluate the authenticated index, which costs $\kappa(Nm_1'+m_2')\mathsf{T}_{\mathsf{DPF.Eval}}+(n+1)(N+1)\mathsf{T}_{\mathsf{I}}$. In Verify, QU verifies the correctness of Nn computation results and computes n one-time pads for decryption if the verification passes, which costs $n\mathsf{T}_{F_{\mathsf{C}}.\mathsf{Eval}}+nN\mathsf{T}_{F_{\mathsf{X}}}+\sum_{\iota\in[N]}\lvert S_\iota\rvert\mathsf{T}_{\mathsf{HMAC}}+n\mathsf{T}_F$ under the assumption that $\lvert S_\iota\rvert$ is an odd number for $\iota\in[N]$.

As for communication cost of our scheme, we represent the size of U DPF shares as b_{key}, which is equal to $\lceil\log D\rceil(\lambda+2)+\log L_{seg}$ in [59] when $U=2$. In Setup, the communication cost originates from uploading the authenticated index to U CSPs, which costs $U(m_1+m_2)(n+1)\psi$. In Search, the communication cost is derived from issuing the search request and returning U query

Algorithm 3 Simulator of G_4

SimSetup(1^λ, $\mathcal{L}_{\mathsf{Setup}}$):
for $i = 1; i \le n; i++$ **do**
  Select $\mathsf{str} \xleftarrow{\$} \{0,1\}^{(m_1+m_2)\psi}$;
  Set $I_{\mathsf{spa}}[i] = (\mathsf{str}[0:\psi], \ldots, \mathsf{str}[(m_1-1)\psi : m_1\psi])$;
  Set $I_{\mathsf{tex}}[i] = (\mathsf{str}[m_1\psi : (m_1+1)\psi], \ldots, \mathsf{str}[(m_1+m_2-1)\psi : (m_1+m_2)\psi])$;
for $j = 1; j \le m_1; j++$ **do**
  $\sigma_{\mathsf{spa},j} \xleftarrow{\$} \{0,1\}^{\psi}$;
for $j = 1; j \le m_2; j++$ **do**
  $\sigma_{\mathsf{tex},j} \xleftarrow{\$} \{0,1\}^{\psi}$;
Set $\tilde{I}_{\mathsf{spa}} = (I_{\mathsf{spa}}, \sigma_{\mathsf{spa}})$, $\tilde{I}_{\mathsf{tex}} = (I_{\mathsf{tex}}, \sigma_{\mathsf{tex}})$, $AUI = \{\tilde{I}_{\mathsf{spa}}, \tilde{I}_{\mathsf{tex}}\}$;
return AUI
SimSearch(1^λ, $\mathcal{L}_{\mathsf{Search}}$):
for $\iota = 1; \iota \le N; \iota++$ **do**
  for $i = 1; i \le M_1; i++$ **do**
    $(\mathrm{k}_i^{(0)}, \ldots, \mathrm{k}_i^{(U-1)}) \leftarrow \mathsf{Sim}(1^\lambda, \lceil \log D_1 \rceil, L_{seg})$;
  Set $\mathrm{k}_{R_\iota}^{(\ell)} = (\mathrm{k}_1^{(\ell)}, \ldots, \mathrm{k}_{M_1}^{(\ell)})$ for $\ell \in \{0,1,\ldots,U-1\}$;
for $i = 1; i \le M_2; i++$ **do**
  $(\mathrm{k}_{W^*,i}^{(0)}, \ldots, \mathrm{k}_{W^*,i}^{(U-1)}) \leftarrow \mathsf{Sim}(1^\lambda, \lceil \log D_2 \rceil, L_{seg})$;
Set $\mathrm{k}_{W^*}^{(\ell)} = (\mathrm{k}_{W^*,1}^{(\ell)}, \ldots, \mathrm{k}_{W^*,M_2}^{(\ell)})$ for $\ell \in \{0,1,\ldots,U-1\}$;
Set $\mathrm{K}^{(\ell)} = \{\mathrm{k}_{R_1}^{(\ell)}, \ldots, \mathrm{k}_{R_N}^{(\ell)}, \mathrm{k}_{W^*}^{(\ell)}\}$ for $\ell \in \{0,1,\ldots,U-1\}$;
return $\mathrm{K}^{(0)}, \mathrm{K}^{(1)}, \ldots, \mathrm{K}^{(U-1)}$

responses, which costs $(NM_1 + M_2)b_{key} + U(n+1)N\psi$. In Verify, the result verification and decryption are completed locally by QU, which does not incur any communication cost.

In TABLE III, we compare our scheme $\mathsf{VPBRQ}_{\mathsf{SupL}}$ with previous scheme PPSKS [36]. Note that the setup and data outsourcing phases in PPSKS is equivalent to the Setup phase in our scheme, and the token generation and search phases in PPSKS is equivalent to the Search phase in our scheme. Since FHE and the secure comparison operation are much expensive, the computation cost of $\mathsf{VPBRQ}_{\mathsf{SupL}}$ is significantly lower than that of PPSKS.

B. Experimental Tests

We implement $\mathsf{VPBRQ}_{\mathsf{SupL}}$ and PPSKS with Python and C programming language. All experiments are conducted on the personal computer[6] (PC) and a Raspberry Pi[7] (Pi). In our experiments, we randomly select 10000 items from a real-world Yelp business dataset[8] as the test dataset, where

[6]PC with 3.20GHz 3.19GHz Intel(R) Core(TM) i7-8700K CPU.
[7]Raspberry Pi 4B with 1.5Ghz ARM Cortex A72 CPU and 2GB memory.
[8]https://www.yelp.com/dataset

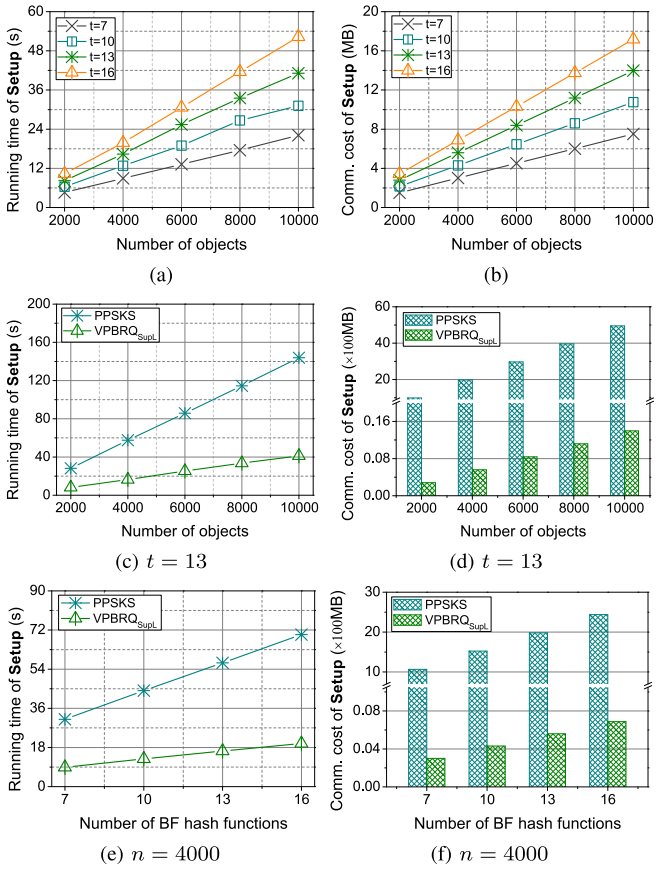


Fig. 8. Performance of Setup.

each item includes spatial location and category description. The spatial locations are within the longitude range of $[-115.450, -72.947]$ and latitude range of $[33.218, 51.299]$. The category descriptions are processed to extract up to 19 keywords for each item. In $\mathsf{VPBRQ}_{\mathsf{SupL}}$, we grid the space into 1000×1000 cells, and encode each cell as a Gray code of length $2\lceil \log 1000 \rceil = 20$. Other parameters in $\mathsf{VPBRQ}_{\mathsf{SupL}}$ are set as follows: $U = 2$, $\psi = 16$, $\kappa = 3$, $M = \lceil 1.27|S'| \rceil$, $L_{seg} = 32$, $|\mathcal{W}(\mathsf{g}_R)| = 12$. In addition, the PRF F, F_{X} and HMAC are instantiated with SHA-256. In PPSKS, we encode the longitude and latitude of each object as 18 and 17 keywords respectively (*i.e.*, $m_x = 17, m_y = 18$) and the query range as $n_x = 5, n_y = 4$. In addition, we take SHE as FHE and set its security parameters to $k_0 = 2048$, $k_1 = 20$, $k_2 = 160$. In both schemes, we set the length of Bloom filter to $m = t \cdot N_{BF} / \ln 2$, the number of query keywords to $|W^*| = 5$, and the security parameter to $\lambda = 128$, where N_{BF} is the number of elements stored in BF. In this way, the false positive probability is $p \approx (1/2)^t$.

Performance of Setup. The setup time and communication cost of $\mathsf{VPBRQ}_{\mathsf{SupL}}$ depend on the number of objects n and the number of BF hash functions t. We vary these parameters to evaluate the performance of $\mathsf{VPBRQ}_{\mathsf{SupL}}$ and compare it with PPSKS, which does not support result verification. As shown in Figs. 8a, 8b, both the setup time and communication cost of $\mathsf{VPBRQ}_{\mathsf{SupL}}$ grow linearly with n and t. We set $t = 7$, 10, 13, 16 to achieve different false positive probabilities of BF, which are 0.78×10^{-2}, 0.98×10^{-3}, 1.22×10^{-4}, 1.53×10^{-5} respectively. TABLE IV shows that more than half

TABLE IV

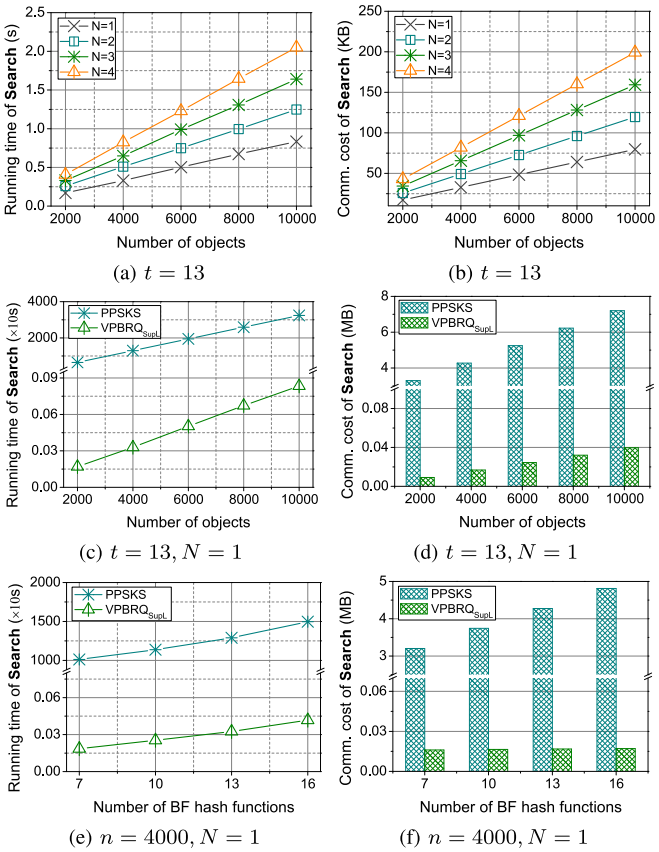
RUNNING TIME OF ENCRYPTING AND AUTHENTICATING THE INDEX STRUCTURE WHEN $t = 13$ (SECONDS)

n	2000	4000	6000	8000	10000
Encryption	3.159	6.344	9.341	12.433	15.773
Authentication	5.064	10.025	16.058	21.104	25.368

TABLE V

RUNNING TIME OF QU AND EACH CSP IN SEARCH WHEN $n = 4000$, $N = 1$ (MILLISECONDS)

t	7	10	13	16
QU (Pi)	1.55	2.07	2.65	3.14
QU (PC)	0.27	0.34	0.40	0.46
CSP P_ℓ	185.48	253.32	329.52	415.90



(a) $t = 13$ (b) $t = 13$ (c) $t = 13, N = 1$ (d) $t = 13, N = 1$ (e) $n = 4000, N = 1$ (f) $n = 4000, N = 1$

Fig. 9. Performance of Search.

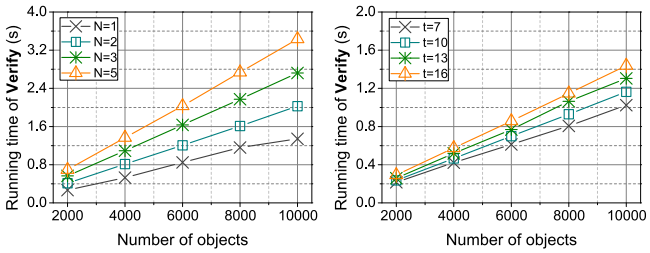
of the setup time is spent on generating aggregate tags. Figs. 8c, 8d, 8e, 8f demonstrate that $\mathsf{VPBRQ}_{\mathsf{SupL}}$ outperforms PPSKS. When $n = 10000, t = 13$, the running time and communication cost of $\mathsf{VPBRQ}_{\mathsf{SupL}}$ is about 40 s and 13 MB, which are at least 3× and 350× lower than those of PPSKS respectively, although PPSKS does not consider result verification.

Performance of Search. We evaluate the search time and communication cost of $\mathsf{VPBRQ}_{\mathsf{SupL}}$ under different values of n, N and t. Figs. 9a, 9b show that they increase linearly with both n and N. Figs. 9c, 9d, 9e, 9f indicate that $\mathsf{VPBRQ}_{\mathsf{SupL}}$ achieves significant savings over PPSKS in both computation

TABLE VI

RUNNING TIME OF RESULT VERIFICATION AND DECRYPTION WHEN $t = 13$, $N = 1$ (SECONDS)

n	2000	4000	6000	8000	10000
Verification (PC)	0.023	0.044	0.064	0.085	0.106
Decryption (PC)	0.237	0.457	0.685	0.914	1.148
Verification (Pi)	0.070	0.135	0.203	0.270	0.340
Decryption (Pi)	0.655	1.397	2.159	2.909	3.621



(a) $t = 13$ (b) $N = 1$

Fig. 10. Performance of verify on PC.

and communication costs. When $n = 10000, t = 13, N = 1$, the search time and communication cost of $\mathsf{VPBRQ}_{\mathsf{SupL}}$ is about 0.8 s and 40 KB, which are at least 30000× and 180× lower than those of PPSKS, respectively. The communication cost of $\mathsf{VPBRQ}_{\mathsf{SupL}}$ includes two parts: the cost for sending the search request from QU to CSPs, and the cost for returning the results from CSPs to QU. The former is 1.54 KB, and the latter is 19.53 KB per CSP, of which the cost of the proof information accounts for 2 B. TABLE V shows that the time of generating a secure query is microsecond-level whether on PC or Pi.

Performance of Verify. Unlike PPSKS, $\mathsf{VPBRQ}_{\mathsf{SupL}}$ needs extra operations to check the integrity of query results. Fig. 10 illustrates that the Verify time increases linearly with n, t and N, where the Verify time consists of result verification time and result decryption time. As shown in TABLE VI, the result decryption time dominates the Verify time, accounting for more than 90%. Moreover, it grows linearly with the number of objects n. When $n = 10000, t = 13, N = 1$, the result decryption time is about 3.6 s on Pi, which is twice as long as that on PC. When processing large-scale datasets, the heavy decryption will consume a large amount of computation resources of resource-limited devices.

VIII. CONCLUSION

In this work, we propose an efficient verifiable privacy-preserving Boolean range query with suppressed leakage, namely $\mathsf{VPBRQ}_{\mathsf{SupL}}$. It achieves privacy-preserving Boolean range query by using Gray code, Bloom filter, one-time pad. Then, it supports efficient retrieval with access and search pattern hidden based on distributed multi-point point function and segmentation method. Moreover, it achieves lightweight result verification without revealing access and search patterns by using HMAC, XOR-homomorphic PRF, aggregate MAC and oblivious query. In addition, we extend it to enable retrieval without local decryption by using replicated secret sharing, and propose an efficient and oblivious privacy-preserving Boolean kNN query. Finally, the formal security proves its

adaptive security, and the performance analysis demonstrates its efficiency and feasibility for practical application.

REFERENCES

[1] R. Hariharan, B. Hore, C. Li, and S. Mehrotra, "Processing spatial-keyword (SK) queries in geographic information retrieval (GIR) systems," in *Proc. 19th Int. Conf. Sci. Stat. Database Manage. (SSDBM)*, Jul. 2007, p. 16.

[2] D. Negi, S. Ray, and R. Lu, "Pystin: Enabling secure LBS in smart cities with privacy-preserving top-*k* spatial–textual query," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7788–7799, Oct. 2019.

[3] Z. Chen, L. Chen, G. Cong, and C. S. Jensen, "Location- and keyword-based querying of geo-textual data: A survey," *VLDB J.*, vol. 30, no. 4, pp. 603–640, Jul. 2021.

[4] B. Sheng and Q. Li, "Verifiable privacy-preserving range query in two-tiered sensor networks," in *Proc. IEEE 27th Conf. Comput. Commun. (INFOCOM)*, Apr. 2008, pp. 46–50.

[5] C. Xu, J. Xu, H. Hu, and M. H. Au, "When query authentication meets fine-grained access control: A zero-knowledge approach," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, May 2018, pp. 147–162.

[6] J. Wang, S.-F. Sun, T. Li, S. Qi, and X. Chen, "Practical volume-hiding encrypted multi-maps with optimal overhead and beyond," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Nov. 2022, pp. 2825–2839.

[7] Z. Wang, L. Zhang, X. Ding, K. R. Choo, and H. Jin, "A dynamic-efficient structure for secure and verifiable location-based skyline queries," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 920–935, 2023.

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

[9] L. Blackstone, S. Kamara, and T. Moataz, "Revisiting leakage abuse attacks," *Cryptol. ePrint Arch.*, pp. 1–43, Oct. 2019.

[10] F. Falzon et al., "Full database reconstruction in two dimensions," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Oct. 2020, pp. 443–460.

[11] E. A. Markatou, F. Falzon, R. Tamassia, and W. Schor, "Reconstructing with less: Leakage abuse attacks in two dimensions," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Nov. 2021, pp. 2243–2261.

[12] N. Cui, J. Li, X. Yang, B. Wang, M. Reynolds, and Y. Xiang, "When geo-text meets security: Privacy-preserving Boolean spatial keyword queries," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 1046–1057.

[13] X. Wang et al., "Search me in the dark: Privacy-preserving Boolean range query over encrypted spatial data," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jul. 2020, pp. 2253–2262.

[14] X. Wang, J. Ma, F. Li, X. Liu, Y. Miao, and R. H. Deng, "Enabling efficient spatial keyword queries on encrypted data with strong security guarantees," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4909–4923, 2021.

[15] Q. Tong, X. Li, Y. Miao, X. Liu, J. Weng, and R. H. Deng, "Privacy-preserving Boolean range query with temporal access control in mobile computing," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 5, pp. 5159–5172, May 2023.

[16] Y. Miao, Y. Yang, X. Li, L. Wei, Z. Liu, and R. H. Deng, "Efficient privacy-preserving spatial data query in cloud computing," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 1, pp. 122–136, Jan. 2024.

[17] X. Wang, J. Ma, X. Liu, Y. Miao, Y. Liu, and R. H. Deng, "Forward/backward and content private DSSE for spatial keyword queries," *IEEE Trans. Depend. Sec. Comput.*, vol. 20, no. 4, pp. 3358–3370, Jul./Aug. 2023.

[18] L. Assouline and B. Minaud, "Weighted oblivious RAM, with applications to searchable symmetric encryption," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn. (EUROCRYPT)*. Cham, Switzerland: Springer, 2023, pp. 426–455.

[19] Z. Chang, D. Xie, F. Li, J. M. Phillips, and R. Balasubramonian, "Efficient oblivious query processing for range and kNN queries," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 12, pp. 5741–5754, Dec. 2022.

[20] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *J. ACM*, vol. 45, no. 6, pp. 965–981, 1998.

[21] S. M. Hafiz and R. Henry, "Querying for queries: Indexes of queries for efficient and expressive IT-PIR," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Oct. 2017, pp. 1361–1373.

[22] R. A. Mahdavi and F. Kerschbaum, "Constant-weight PIR: Single-round keyword PIR via constant-weight equality operators," in *Proc. USENIX Secur. Symp. (USENIX Security)*, 2022, pp. 1723–1740.

[23] I. Ahmad, D. Agrawal, A. E. Abbadi, and T. Gupta, "Pantheon: Private retrieval from public key-value store," *Proc. VLDB Endowment*, vol. 16, no. 4, pp. 643–656, 2022.

[24] S. Patel, J. Y. Seo, and K. Yeo, "Don't be dense: Efficient keyword PIR for sparse databases," *Cryptol. ePrint Arch.*, pp. 1–18, Jun. 2023.

[25] S. Wu, Q. Li, G. Li, D. Yuan, X. Yuan, and C. Wang, "ServeDB: Secure, verifiable, and efficient range queries on outsourced database," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 626–637.

[26] Q. Tong, Y. Miao, X. Liu, K. R. Choo, R. H. Deng, and H. Li, "VPSL: Verifiable privacy-preserving data search for cloud-assisted Internet of Things," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2964–2976, Oct. 2022.

[27] Q. Meng et al., "Verifiable spatial range query over encrypted cloud data in VANET," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 12342–12357, Dec. 2021.

[28] 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. 2015, pp. 2110–2118.

[29] Q. Liu, X. Nie, X. Liu, T. Peng, and J. Wu, "Verifiable ranked search over dynamic encrypted data in cloud computing," in *Proc. IEEE/ACM 25th Int. Symp. Quality Service (IWQoS)*, Jun. 2017, pp. 1–6.

[30] Q. Zheng, S. Xu, and G. Ateniese, "VABKS: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2014, pp. 522–530.

[31] Y. Zhang, T. Zhu, R. Guo, S. Xu, H. Cui, and J. Cao, "Multi-keyword searchable and verifiable attribute-based encryption over cloud data," *IEEE Trans. Cloud Comput.*, vol. 11, no. 1, pp. 971–983, Jan. 2023.

[32] Y. Guo, H. Xie, C. Wang, and X. Jia, "Enabling privacy-preserving geographic range query in fog-enhanced IoT services," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 5, pp. 3401–3416, Sep. 2022.

[33] Y. Guo, H. Xie, M. Wang, and X. Jia, "Privacy-preserving multi-range queries for secure data outsourcing services," *IEEE Trans. Cloud Comput.*, vol. 11, no. 3, pp. 2431–2444, Jul./Sep. 2023.

[34] S. Su, Y. Teng, X. Cheng, K. Xiao, G. Li, and J. Chen, "Privacy-preserving top-*k* spatial keyword queries in untrusted cloud environments," *IEEE Trans. Services Comput.*, vol. 11, no. 5, pp. 796–809, Sep. 2018.

[35] Q. Tong, Y. Miao, H. Li, X. Liu, and R. H. Deng, "Privacy-preserving ranked spatial keyword query in mobile cloud-assisted fog computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 6, pp. 3604–3618, Jun. 2023.

[36] S. Zhang, S. Ray, R. Lu, Y. Guan, Y. Zheng, and J. Shao, "Efficient and privacy-preserving spatial keyword similarity query over encrypted data," *IEEE Trans. Depend. Sec. Comput.*, vol. 20, no. 5, pp. 3770–3786, Sep./Oct. 2023.

[37] E. Dauterman, E. Feng, E. Luo, R. A. Popa, and I. Stoica, "DORY: An encrypted search system with distributed trust," in *Proc. USENIX Conf. Operating Syst. Design Implement. (USENIX)*, 2020, pp. 1101–1119.

[38] E. Dauterman, M. Rathee, R. A. Popa, and I. Stoica, "Waldo: A private time-series database from function secret sharing," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2022, pp. 2450–2468.

[39] C. Huang, D. Liu, A. Yang, R. Lu, and X. Shen, "Multi-client secure and efficient DPF-based keyword search for cloud storage," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 1, pp. 353–371, Jan./Feb. 2024.

[40] G. Chen, T.-H. Lai, M. K. Reiter, and Y. Zhang, "Differentially private access patterns for searchable symmetric encryption," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 810–818.

[41] Y. Zheng et al., "PHRkNN: Efficient and privacy-preserving reverse kNN query over high-dimensional data in cloud," *IEEE Trans. Dependable Secure Comput.*, pp. 1–15, Jul. 2023, doi: 10.1109/TDSC.2023.3291715.

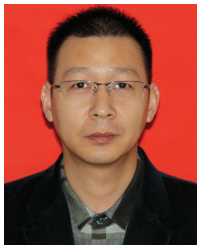
[42] S. Su, H. Yan, X. Cheng, P. Tang, P. Xu, and J. Xu, "Authentication of top-*k* spatial keyword queries in outsourced databases," in *Proc. Database Syst. Adv. Appl. (DASFAA)*. Cham, Switzerland: Springer, 2015, pp. 567–588.

[43] C. Xu, C. Zhang, and J. Xu, "VChain: Enabling verifiable Boolean range queries over blockchain databases," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, Jun. 2019, pp. 141–158.

[44] H. Wang, C. Xu, C. Zhang, J. Xu, Z. Peng, and J. Pei, "VChain+: Optimizing verifiable blockchain Boolean range queries," in *Proc. IEEE 38th Int. Conf. Data Eng. (ICDE)*, May 2022, pp. 1927–1940.

[45] Y. Guo, C. Zhang, C. Wang, and X. Jia, "Towards public verifiable and forward-privacy encrypted search by using blockchain," *IEEE Trans. Depend. Sec. Comput.*, vol. 20, no. 3, pp. 2111–2126, May/Jun. 2023.
[46] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.
[47] C. Dong, L. Chen, and Z. Wen, "When private set intersection meets big data: An efficient and scalable protocol," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 789–800.
[48] L. de Castro and A. Polychroniadou, "Lightweight, maliciously secure verifiable function secret sharing," in *Proc. Theory Appl. Cryptograph. Techn. (EUROCRYPT)*. Cham, Switzerland: Springer, 2022, pp. 150–179.
[49] B. Pinkas, T. Schneider, and M. Zohner, "Scalable private set intersection based on OT extension," *ACM Trans. Privacy Secur.*, vol. 21, no. 2, pp. 1–35, May 2018.
[50] N. Gilboa and Y. Ishai, "Distributed point functions and their applications," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn. (EUROCRYPT)*. Cham, Switzerland: Springer, 2014, pp. 640–658.
[51] D. Boneh and B. Waters, "Constrained pseudorandom functions and their applications," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur. (ASIACRYPT)*. Cham, Switzerland: Springer, 2013, pp. 280–300.
[52] Z. Wan and R. H. Deng, "VPSearch: Achieving verifiability for privacy-preserving multi-keyword search over encrypted cloud data," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 6, pp. 1083–1095, Nov. 2018.
[53] J. Shu, X. Jia, K. Yang, and H. Wang, "Privacy-preserving task recommendation services for crowdsourcing," *IEEE Trans. Services Comput.*, vol. 14, no. 1, pp. 235–247, Jan. 2021.
[54] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Towards practical and privacy-preserving multi-dimensional range query over cloud," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 5, pp. 3478–3493, Sep. 2022.
[55] S. Kamara and T. Moataz, "Computationally volume-hiding structured encryption," in *Proc. Theory Appl. Cryptograph. Techn. (EUROCRYPT)*. Cham, Switzerland: Springer, 2019, pp. 183–213.
[56] J. R. Bitner, G. Ehrlich, and E. M. Reingold, "Efficient generation of the binary reflected gray code and its applications," *Commun. ACM*, vol. 19, no. 9, pp. 517–521, Sep. 1976.
[57] S. Oya and F. Kerschbaum, "Hiding the access pattern is not enough: Exploiting search pattern leakage in searchable encryption," in *Proc. USENIX Secur. Symp. (USENIX Security)*, 2021, pp. 127–142.
[58] B. Cogliati and Y. Seurin, "EWCDM: An efficient, beyond-birthday secure, nonce-misuse resistant MAC," in *Proc. Int. Cryptol. Conf. (CRYPTO)*. Cham, Switzerland: Springer, 2016, pp. 121–149.
[59] E. Boyle, N. Gilboa, and Y. Ishai, "Function secret sharing: Improvements and extensions," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2016, pp. 1292–1303.
[60] S. Colombo, K. Nikitin, C. Tech, H. Corrigan-Gibbs, D. J. Wu, and B. Ford, "Authenticated private information retrieval," in *Proc. USENIX Conf. Operating Syst. Design Implement. (USENIX)*, 2023, pp. 3835–3851.
[61] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, "High-throughput semi-honest secure three-party computation with an honest majority," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Oct. 2016, pp. 805–817.
[62] X. Lei, A. X. Liu, R. Li, and G.-H. Tu, "SecEQP: A secure and efficient scheme for SkNN query problem over encrypted geodata on cloud," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 662–673.

Qiuyun Tong received the B.S. degree from the Department of Information and Computing Science, Shaanxi Normal University, Xi'an, China, in 2019. She is currently pursuing the Ph.D. degree with the Department of Cyber Engineering, Xidian University, Xi'an. Her research interests include information security and applied cryptography.



Xinghua Li (Member, IEEE) received the M.E. and Ph.D. degrees in computer science from Xidian University, Xi'an, China, in 2004 and 2007, respectively. He is currently a Professor with the School of Cyber Engineering, Xidian University. His research interests include wireless networks security, privacy protection, cloud computing, and security protocol formal methodology.

Yinbin Miao (Member, IEEE) received the B.E. degree from the Department of Telecommunication Engineering, Jilin University, Changchun, China, in 2011, and the Ph.D. degree from the Department of Telecommunication Engineering, Xidian University, Xi'an, China, in 2016. He is currently a Professor with the Department of Cyber Engineering, Xidian University. His research interests include information security and applied cryptography.

Yunwei Wang received the Ph.D. degree in security of cyberspace from Xidian University, China, in 2022. He is currently a Lecturer in security of cyberspace with Xidian University. His research interests include the IoT, data security, and privacy protection.

Ximeng Liu (Senior Member, IEEE) received the B.E. degree in electronic engineering and the Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2010 and 2015, respectively. He is currently a full Professor with the College of Computer and Data Science, Fuzhou University. He was a Research Fellow with the School of Information System, Singapore Management University, Singapore. His research interests include cloud security, applied cryptography, and big data security.

Robert H. Deng (Fellow, IEEE) has been a AXA Chair Professor of cybersecurity and a Professor of information systems with the School of Information Systems, Singapore Management University, since 2004. His research interests include data security and privacy, multimedia security, and network and system security. He has received the Distinguished Paper Award from NDSS 2012 and the Best Paper Award from CMS 2012. He has served on the editorial boards for many international journals, including IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY and IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING.