

Enabling Efficient Spatial Keyword Queries on Encrypted Data With Strong Security Guarantees

Xiangyu Wang¹, Jianfeng Ma¹, *Member, IEEE*, Feng Li¹, Ximeng Liu¹, *Member, IEEE*,
Yinbin Miao², *Member, IEEE*, and Robert H. Deng², *Fellow, IEEE*

Abstract—Structured Encryption (STE), which allows a server to provide secure search services on encrypted data structures, has been widely investigated in recent years. To meet expressive search requirements in practical applications, a large number of STE constructions have been proposed either on textual keywords or spatial data. However, STE on *spatio-textual* data, which are widely used in location-based services, has not been fully investigated. In this paper, we formally define the notion of Spatial Keyword Structured Encryption (SKSE) and propose several concrete SKSE constructions with various efficiency-security trade-offs. Firstly, we propose a basic construction with linear search complexity, which only leaks the private files matching both spatial range query and all query keywords. Then, to improve the search efficiency on large-scale datasets, we present a novel tree-based construction with sub-linear search complexity. Finally, we introduce a post-validation approach to remove false positives and further improve storage and search performance. Our constructions are general in the sense that they can be constructed from any hidden vector encryption schemes, including public-key setting and symmetric-key setting, which can meet different sharing requirements. Our rigorous security analysis and comprehensive performance evaluation demonstrate that the proposed constructions are secure and outperform the start-of-the-art solutions.

Index Terms—Structured encryption, spatio-textual data, hidden vector encryption.

Manuscript received January 23, 2021; revised May 20, 2021 and August 10, 2021; accepted September 16, 2021. Date of publication October 8, 2021; date of current version October 25, 2021. This work was supported in part by the Shaanxi Science & Technology Coordination & Innovation Project under Grant 2016TZC-G-6-3; in part by the National Natural Science Foundation of China under Grant 62072361, Grant 62072109, and Grant U1804263; in part by the Key Research and Development Program of Shaanxi Province under Grant 2019ZDLGY12-04 and Grant 2020ZDLGY09-06; in part by the Fundamental Research Funds for the Central Universities under Grant JB191506; in part by the Natural Science Foundation of Shaanxi Province under Grant 2019JQ-005; in part by the Guangxi Key Laboratory of Trusted Software under Grant KX202028; in part by the CCF-Tencent Open Research Fund; in part by the CCF-NSFOCUS Kunpeng Research Fund; and in part by the Hong Kong Scholar Program. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Lejla Batina. (*Corresponding author: Xiangyu Wang.*)

Xiangyu Wang, Jianfeng Ma, and Feng Li are with the School of Cyber Engineering, and the Shaanxi Key Laboratory of Network and System Security, Xidian University, Xi'an 710071, China (e-mail: xywang_xidian@163.com; jfma@mail.xidian.edu.cn; fl1996@stu.xidian.edu.cn).

Ximeng Liu is with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China (e-mail: snbnix@gmail.com).

Yinbin Miao is with the School of Cyber Engineering, Xidian University, Xi'an 710071, China, also with the Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China, and also with the Department of Computer Science, City University of Hong Kong, Hong Kong, SAR, China (e-mail: ybmiao@xidian.edu.cn).

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.2021.3118880

I. INTRODUCTION

A. Background and Motivation

RECENTLY, spatial keyword queries have received increasingly consideration with the popularity of location-based services and social networks. As a result, several systems have been proposed to store, manage, and query *spatio-textual* data, such as PostGIS¹ and Oracle Spatial.² A typical type of spatial keyword query is receiving particular attention, namely Boolean Range Query (BRQ) [1]: Given a set of *spatio-textual* objects, on input a query range R associated with a set of query keywords, a search user can obtain all objects inside R such that every object contains all query keywords. For example, a *Twitter* user can find all friends with the same interests in an area; a *Google Map* user can find all shops with the specific services in an area. Fig. 1 shows several examples of BRQ, where the logo represents the locations of users and the keywords represent the list of services. There are five objects (i.e., users) $\{O_1, \dots, O_5\}$ in the *spatio-textual* dataset, and a *Google Map* user wants to find all the shops providing *coffee*, *pizza*, and *spaghetti* inside a certain area. He/She can submit the query ranges (i.e., the “red area”) and the three keywords, and obtain $\{O_2, O_4\}$ from the server.

Due to the economic and flexible advantages of cloud computing, more and more data owners are motivated to outsource their *spatio-textual* data services to a cloud server. However, increasing concerns about data privacy disclosure in the cloud server, have curbed many data owners’ enthusiasm about outsourcing their data services to the cloud. To address this issue, Cui *et al.* [2] proposed ELCBFR+, a privacy-preserving solution to support BRQ on encrypted data. They presented a Bloom Filter (BF) [3] based *spatio-textual* data encoding method and evaluated BFs using Asymmetric Scalar Product-preserving Encryption (ASPE) [4]. Unfortunately, the ASPE has been proven to be too weak to resist even Ciphertext-Only Attack (COA) [5], which means that their solution cannot guarantee data security and query privacy. In addition, Wang *et al.* [6] proposed a strongly secure BRQ solution namely PBRQ based on Gray code and bitmap encoding, but the performance is limited. Specifically, their PBRQ-Q construction costs 3.14 seconds to find the BRQ results from an encrypted database with 1×10^5 objects, which

¹<http://www.postgis.org>

²<https://www.oracle.com/database/>

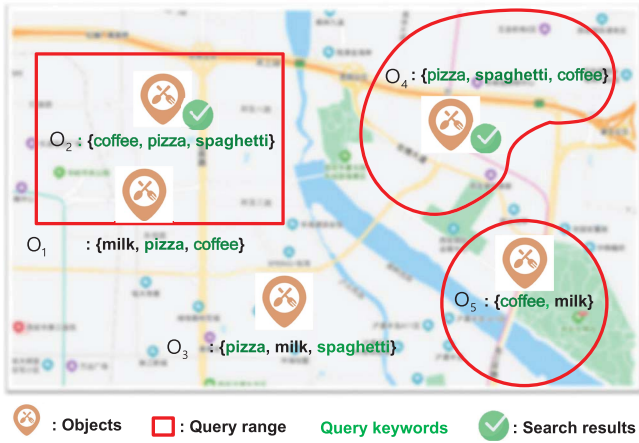


Fig. 1. BRQ examples.

is not suitable for large-scale datasets. Structured Encryption [7] (STE) is a promising cryptographic primitive used to protect the privacy of cloud-hosted data structures. Using STE, a data owner can perform query operations on outsourced encrypted data structures without revealing any information to the server beyond some well-defined reasonable leakage functions. However, existing studies primarily focus on the design of STE for either textual keyword (i.e., searchable encryption [8]–[13]) or spatial data [14]–[17], which cannot be directly applied to solve the BRQ problem.

B. Challenges

1) *STE Supporting BRQ*: As an important example of STE, Searchable Encryption (SE) [8]–[13] has been well-studied in the literature. SE enables a data owner to encrypt a corpus of documents and store them to an untrusted server while maintaining the searchability for documents containing specified keywords. Specifically, a server needs to perform equality testing and returns all records matching the query keywords. However, such a method cannot work on a *spatio-textual* database. To perform BRQ on a *spatio-textual* database, a server needs to perform not only the keyword query for textual information but also the geometric range query for spatial data. Different from the keyword query, a geometric range query on spatial data requires *compute-then-compare* operations.³ Combining existing SE schemes with secure geometric range query schemes [16]–[18] seems a simple way to address this issue. Unfortunately, it will produce a lot of useless intermediate results, resulting in considerable computational overhead.⁴ In addition, such a method will leak both the query information of the textual keywords and spatial data separately, so that an attacker can infer the textual keywords or spatial data using the existing attack approaches (e.g., file-injection attacks [19] for textual keywords, reconstruct attacks [20] and

³For example, to determine whether a point is inside a circle, a server needs to calculate the distance from this point to the center of the circle, and then compare this distance with the radius of the circle.

⁴In this case, the cloud server needs to obtain corresponding results according to the textual keyword query and the spatial range query separately, and take their intersection as the final BRQ result. Thus, the entire database is searched at least twice.

volume-based attacks [21] for spatial data). How to construct STE constructions supporting BRQ is still challenging.

2) *Practicality vs Information Leakage*: To achieve high performance, existing STE schemes do efficiency-security trade-offs. Several encrypted database systems [22], [23] and secure spatial search schemes [4], [14], [18] use property-preserving encryption schemes such as deterministic encryption [24], Order-Preserving Encryption (OPE) [25], and Order-Revealing Encryption (ORE) [26]. Recent attacks [27] show that the property-preserving encryption based schemes have security vulnerabilities. In addition, most of the STE schemes leak some “partial” information (e.g., storage leakage, query pattern leakage, and result pattern leakage.) to the server for the sake of efficiency. Such partial information, however, can sometimes be exploited in practical applications, especially when attackers can access some auxiliary side information [19]–[21]. An adapted version of the recent proposed file-injection attacks [12], [19] can exploit Keyword-Pair Result Pattern (KPRP) leakage [10]⁵ to reveal all keywords of a conjunctive queries. Moreover, the reconstruct and volume-based attacks [20], [21] are able to reconstruct spatial databases with high accuracy from query pattern or volume leakage of range queries. Oblivious RAM (ORAM) [28] is able to hide the query pattern and result pattern, but it also induces multiple interactives and larger computational overhead. How to construct practical STE constructions that can resist the above attacks with a single interactive is a challenging problem.

C. Our Contributions

In this paper, we discuss the application of Hidden Vector Encryption (HVE) [12], [29] for practical and secure BRQ on encrypted data. Our constructions allow BRQ on encrypted data with only a single interactive. The main contributions of the paper can be summarized as follows:

- 1) We investigate the problem of secure BRQ on encrypted *spatio-textual* data, propose a formal notion of Spatial Keyword Structured Encryption (SKSE) and formalize its security (see Section III).
- 2) We propose several general constructions for SKSE with different efficiency-privacy trade-offs, which can be instantiated using any HVE schemes. Firstly, we present a linear complexity construction SKSE-I with minimal result pattern leakage (i.e., the Whole Result Pattern (WRP), which only leaks the identifiers of files matching both geometric range and all query keywords.). In our second construction, SKSE-II, we construct a novel tree-structure index to achieve sub-linear search complexity. Besides the WRP, SKSE-II also leaks query pattern (i.e., whether two queries traverse the same tree node). Finally, we present a post-validation approach to guarantee query accuracy and further improve search and storage performance (see Section IV).
- 3) We provide rigorous security analysis and demonstrate that our constructions are secure under Selective

⁵For a n keywords conjunctive query $w_1 \wedge w_2 \wedge \dots \wedge w_n$ on a database $\mathbf{DB}$, KPRP leaks the partial result sets $\mathbf{DB}(w_1) \cap \mathbf{DB}(w_i)$ for each pair (w_1, w_i) .

TABLE I
NOTATIONS AND TERMINOLOGIES

Notation	Meaning
λ	a security parameter
ind_i	the identifier of the i -th private file
$\mathbf{p}_i = \{x_i, y_i\}$	a spatial point
$\mathcal{W}_i$	a set of textual keywords
N	# of objects in the database
$\text{DB} = \{\mathbf{p}_i, \mathcal{W}_i, \text{ind}_i\}_{i=1}^N$	a <i>spatio-textual</i> database
$Q = \{R_Q, \mathcal{W}_Q\}$	a search query
$R_Q, \mathcal{W}_Q$	query range and textual keywords
t	# of hash functions in a Bloom filter
m	the length of the Bloom filter
f_p	false positive probability
$\mathcal{H}(\cdot)$	Hilbert code of a point or range
$S(\cdot)$	prefix elements of a range
$\mathcal{F}(\cdot)$	prefix elements of a spatial data
β	size of prefix elements
M	max number of children in index tree
δ	depth of the index tree
$[n]$	the set of integers $\{1, 2, \dots, n\}$
$\mathcal{R}$	result set
$ \cdot $	number of elements in a set
$\text{negl}(\lambda)$	a negligible function in λ

Chosen-Plaintext Attacks (SCPA) with reasonable leakages. Then we discuss the state-of-the-art attacks for STE and show that the leakages in our constructions are acceptable that can resist these attacks (see Section V).

- 4) We implement the proposed SKSE constructions and evaluate them using different real-world datasets and HVE schemes. The experimental results show that our constructions are practical on large-scale datasets. The comparative analysis explains that our constructions outperform the state-of-the-art solutions (see Section VI).

II. PRELIMINARIES

In this section, we introduce some primitives that are used in our work, a summary of notations and terminologies used in this paper is given in Table I.

A. Bloom Filter

Bloom Filter (BF) [3] is a memory-saving probabilistic data structure to check whether a query is contained in a set. Assume that there is a set $\mathcal{V} = \{s_1, s_2, \dots, s_n\}$ of n elements. The BF is initialized as a m -bits binary vector B . To set up BF for $\mathcal{V}$, choose t independent hash functions $\{H_i\}_{1 \leq i \leq t}$ each with range $[1, m]$. For each element $s \in \mathcal{V}$, the bits at positions $\{H_i(s)\}_{1 \leq i \leq t}$ are set to 1. To test the existence of q , check if B has 1's in all positions $\{H_i(q)\}_{1 \leq i \leq t}$, and if so, $q \in \mathcal{V}$ with high probability; otherwise, $q \notin \mathcal{V}$. A *false positive* event is happened when the existence test returns 1, but $q \notin \mathcal{V}$. Given a BF of $\mathcal{V}$ and a query $q \notin \mathcal{V}$, the *false positive* probability is

$$f_p \leq (1 - e^{-t \cdot n/m})^t. \quad (1)$$

Given n, f_p , the optimal choice of t is $t \approx \log_2(1/f_p)$, while the required $m \approx 1.44 \cdot \log_2(1/f_p) \cdot n$.

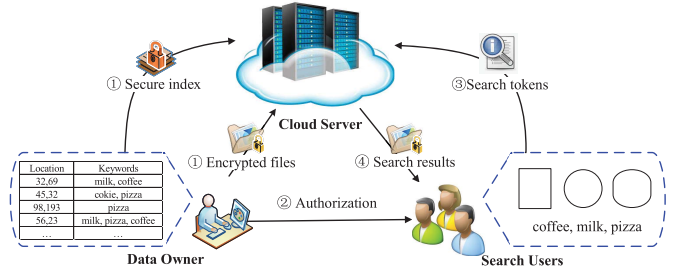


Fig. 2. System model of SKSE.

B. Hidden Vector Encryption

Hidden Vector Encryption (HVE) [12], [29] is able to support conjunctive, equality, comparison, and subset queries on encrypted data. Let $\Gamma \in \{0, 1\}$ and $*$ be a wildcard symbol (“don’t care” value) not in Γ , $\Gamma_* = \Gamma \cup \{*\}$. Let $\mathcal{M} = \{\text{true}, \text{false}\}$ be the message space, the HVE contains following four algorithms:

- **HVE.Setup**(1^λ) $\rightarrow$ msk : On input a security parameter λ , it returns a master secret key msk .
- **HVE.KeyGen**($msk, \mathbf{v} \in \Gamma_*^n$) $\rightarrow$ $\mathbf{s}$: On input a vector $\mathbf{v}$ and a secret key msk , it returns a decryption key $\mathbf{s}$.
- **HVE.Enc**($msk, \mu \in \mathcal{M}, \mathbf{x} \in \Gamma^n$) $\rightarrow$ $\mathbf{c}$: On input the secret key msk , a message μ , and an index vector $\mathbf{x}$, it returns the ciphertext $\mathbf{c}$ associated with $(\mathbf{x}, \mu)$.
- **HVE.Query**($\mathbf{c}, \mathbf{s}$) $\rightarrow$ $\mu(\perp)$: On input the ciphertext $\mathbf{c}$ and the decryption key $\mathbf{s}$, it returns μ if $P_{\mathbf{v}}^{\text{HVE}}(\mathbf{x}) == 1$; otherwise outputs $\perp$.

For each $\mathbf{v} = \{v_1, \dots, v_n\} \in \Gamma_*^n$, $\mathbf{x} = \{x_1, \dots, x_n\} \in \Gamma^n$, we have

$$P_{\mathbf{v}}^{\text{HVE}}(\mathbf{x}) = \begin{cases} 1, & \forall 1 \leq i \leq n (v_i == x_i \text{ or } v_i == *) \\ 0, & \text{otherwise} \end{cases}$$

III. PROBLEM FORMULATION

In this section, we formally describe the definition and security of Spatial Keyword Structured Encryption (SKSE).

A. System Model

As shown in Fig. 2, SKSE has three entities: a data owner, a cloud server, and search users. Data owner stores an encrypted *spatio-textual* database on a cloud server, and shares secret keys with search users to delegate the query service. Specifically, *spatio-textual* data objects consist of spatial location, textual keywords, and private files (e.g., private images, documents, etc.). To guarantee security, searchability and search efficiency, data owner generates a secure index according to spatial data and textual keywords and encrypts the private files using secret keys (Step ①)⁶ and shares secret keys with legal search users (Step ②). It is worth noting that each secure index item for *spatio-textual* objects contains a file identifier to record the association between the index and encrypted files. After receiving a search query from an

⁶Note that search queries are processed on the secure index, the private files can be encrypted by any standard encryption algorithms (e.g., AES, 3DES.).

authorized search user (Step ③), the cloud server performs a secure search on the index, obtaining the file identifiers of search results, and returns the corresponding encrypted private files to the search user (Step ④). Finally, the search user can decrypt the result files using the shared key from data owner. In the rest of this paper, we focus on how to generate indexes and complete the query process securely and no longer describe the encryption and decryption process of private files.

B. SKSE Definition

Based on the above model, we now describe the definition of Secure Boolean Range Query (SBRQ) and SKSE. A *spatio-textual* database DB is defined as: $DB = \{p_i, \mathcal{W}_i, \text{ind}_i\}_{i=1}^N$, where N is the database size, $p_i = (x_i, y_i)$ is a location point, $\mathcal{W}_i$ is a set of keywords, and ind_i is a file identifier.

Definition 1 (SBRQ): Given an encrypted *spatio-textual* database *EDB* and an encrypted search token T_Q , a SBRQ is to retrieve a subset $\{\text{ind}_i^*\}_{i=1}^c$ from *EDB*, such that $\forall 1 \leq i \leq c$, $\mathcal{W}_Q \in \mathcal{W}_i^*$, $p_i^* \in R_Q$.

Definition 2 (SKSE): A SKSE construction is a tuple of four algorithms $\Sigma = (\text{Setup}, \text{IndexBuild}, \text{TokGen}, \text{Search})$ such that:

- **Setup**(1^λ) $\rightarrow K_\Sigma$: On input a security parameter λ , this algorithm outputs a secret key set K_Σ .
- **IndexBuild**(K_Σ, DB) $\rightarrow (K_\Sigma, EDB)$: On input a secret set K_Σ and a database *DB*, this algorithm outputs an encrypted database *EDB*.
- **TokGen**($K_\Sigma, Q = \{\mathcal{W}_Q, R_Q\}$) $\rightarrow T_Q$: On input a secret set K_Σ and a BRQ query Q , this algorithm outputs an encrypted search token T_Q .
- **Search**(T_Q, EDB) $\rightarrow \mathcal{R}$: On input a search token T_Q and an encrypted database *EDB*, this algorithm outputs a result set $\mathcal{R}$.

Security: In a SKSE construction, the cloud server is *honest-but-curious*, which honestly performs the protocols and provides services but curiously captures the information of encrypted queries and data. We assume the data owner and search users are trusted. They do not collude with the cloud server and the transmission channel is secure. Since our constructions are built from HVE [12], [29], the formalized notion of security is presented in the simulation-setting, which widely and consistently subsumes the traditional security definitions for HVE in the indistinguishability setting. An adversary $\mathcal{A}$ should not learn any information about the content of the data and the queries beyond some explicit leakages. A leakage function $\mathcal{L}$ is used to capture the information learned by adversary and its components express the information leaked by **Setup**, **IndexBuild**, **TokGen**, and **Search**, respectively. The experiments **REAL** and **SIM** are described in Figure 3. Denote $\text{View}_{\mathcal{A}, \text{REAL}}$, $\text{View}_{\mathcal{A}, \text{SIM}}$ to be the views of the adversary $\mathcal{A}$ in the real and simulation experiments, respectively, the security of SKSE is defined as:

Definition 3 (Security of SKSE): Let $\mathcal{L}$ be a leakage function, a SKSE construction $\Sigma = (\text{Setup}, \text{IndexBuild}, \text{TokGen}, \text{Search})$ is said to be *selectively simulation-secure*, if for all Probabilistic Polynomial Time (PPT) adversary $\mathcal{A}$ that make a polynomial $q_2(\lambda)$ of queries and a PPT

$\text{SIM}_{\mathcal{A}}^\Sigma$: The simulation experiment for SKSE involves a simulator $\mathcal{S}$ and a stateful Probabilistic Polynomial Time (PPT) adversary $\mathcal{A}$, who interact as follows:

- **Setup**: The adversary $\mathcal{A}$ chooses a *spatio-textual* database *DB*.
- **Phase 1**: The adversary $\mathcal{A}$ adaptively chooses a number of queries Q_j for $j \in [q_1]$. For each such query, the simulator $\mathcal{S}$ only receives as input the leakage function $\mathcal{L}(DB, Q_j)$. It responds with the corresponding search token set T_{Q_j} .
- **Challenge**: The simulator $\mathcal{S}$ flips a coin $b \in \{0, 1\}$, sends the challenge encrypted database *EDB* corresponding to *DB* to $\mathcal{A}$.
- **Phase 2**: The adversary $\mathcal{A}$ runs a protocol identical to **Phase 1** and receives T_{Q_j} for $q_1 + 1 \leq j \leq q_2$.
- **Guess**: The adversary takes a guess of b .

$\text{REAL}_{\mathcal{A}}^\Sigma$: The real experiment for SKSE involves a challenger $\mathcal{C}$ and a stateful PPT adversary $\mathcal{A}$, who interact as follows:

- **Setup**: The adversary $\mathcal{A}$ submits a *spatio-textual* database *DB* to challenger $\mathcal{C}$. In response, $\mathcal{C}$ runs $K_\Sigma \leftarrow \text{Setup}(1^\lambda)$.
- **Phase 1**: The adversary $\mathcal{A}$ adaptively chooses a number of queries Q_j for $j \in [q_1]$. The challenger $\mathcal{C}$ then runs $\text{TokGen}(K_\Sigma, Q_j)$, and responds with the corresponding search token set T_{Q_j} .
- **Challenge**: The challenger $\mathcal{C}$ flips a coin $b \in \{0, 1\}$, runs $\text{IndexBuild}(K_\Sigma, DB)$ and obtains the encrypted database *EDB*, which is given to $\mathcal{A}$.
- **Phase 2**: The adversary $\mathcal{A}$ runs a protocol identical to **Phase 1** and receives T_{Q_j} for $q_1 + 1 \leq j \leq q_2$.
- **Guess**: The adversary takes a guess of b .

Fig. 3. The real and simulation experiments.

distinguisher $\mathcal{D}$, there exists an efficient simulator $\mathcal{S}$ such that:

$$|\Pr[\mathcal{D}(\text{View}_{\mathcal{A}, \text{REAL}}) = 1] - \Pr[\mathcal{D}(\text{View}_{\mathcal{A}, \text{SIM}}) = 1]| \leq \text{negl}(\lambda).$$

IV. OUR SKSE CONSTRUCTIONS

In this section, we describe our SKSE constructions for *spatio-textual* data. Note that our constructions are general that can construct SKSE schemes from any HVE schemes, including public-key setting and symmetric-key setting, which can meet different sharing requirements. Here, we first propose our SKSE constructions in symmetric-key setting [12] and then describe how to construct SKSE schemes in public-key setting [29].

A. SKSE-I: Basic SKSE Construction

Here, we present our basic SKSE construction with linear search complexity and reveal only the whole result pattern. Instead of using the heavy (e.g., homomorphic encryption [30]) or weak (e.g., OPE [25] or ORE [26]) cryptographic primitives, we set spatial and textual elements into BFs, and then encrypt and securely evaluate BFs using HVE. With BFs, we can easily determine whether an object contains all the query textual keywords. To achieve BRQ using BFs, the key problem is to cover the testing of whether a spatial point falls into a geometric range to the testing of whether two sets have common elements. To address this issue, we encode the spatial data and geometric range queries into prefix elements using the Hilbert curve [31] and prefix membership verification scheme [32].

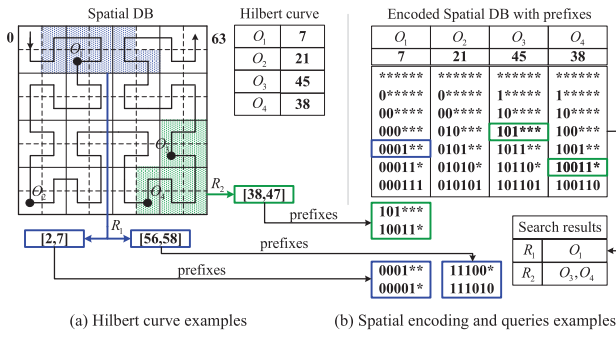


Fig. 4. Hilbert curve and spatial encoding examples.

1) *Spatial Data and Queries Encoding*: Hilbert curve [31] is a classic space-filling curve that has superior clustering properties, that is, two close data points in d -dimensional space are also close to each other after mapping into one-dimensional. For a d -dimensional spatial space in which each dimension is partitioned into 2^h regions averagely, the whole space will be partitioned into 2^{dh} regions averagely. Each region is regarded as a dh -bits value, the h is the order of Hilbert curve. Fig. 4(a) shows the order 3 Hilbert curve for two-dimensional space. If a user wants to search all objects inside the R_1 or R_2 , he/she can generate a query range as $R_1 = \{[2, 7], [56, 58]\}$ or $R_2 = \{[38, 47]\}$. In this way, any geometric range queries over the d -dimensional space can be easily transformed into multiple range queries on one-dimensional space. Denote the Hilbert curve of a point p or a range R be $\mathcal{H}(p)$, $\mathcal{H}(R)$, respectively. In this case, we can set all encoded elements contained in a certain range into a BF and test whether an encoded spatial point is contained in it. However, the size of BF will linearly increase with the increasing number of elements. Note that the number of elements in a range query is much larger than that in a data object. To minimize the size of BF, we need to reduce the number of elements in range queries. Thus, we adopt the prefix membership verification scheme [32]. Given a ω -bits data item $Y = b_1 b_2 \dots b_\omega$, its prefix family is a set of $\omega + 1$ elements $\mathcal{F}(Y) = \{b_1 b_2 \dots b_\omega, b_1 b_2 \dots b_{\omega-1}*, \dots, b_1* \dots *, ** \dots *\}$, where the i -th prefix elements is $b_1 b_2 \dots b_{\omega-i+1} * \dots *$. Given a range $[y_{min}, y_{max}]$, the query $S([y_{min}, y_{max}])$ is the minimum set of prefix elements that cover the range $[y_{min}, y_{max}]$. For any item Y and query range $[y_{min}, y_{max}]$, $Y \in [y_{min}, y_{max}]$ if and only if $\mathcal{F}(Y) \cap S([y_{min}, y_{max}]) \neq \emptyset$. For example, the prefix family of a 4-bits data item 5 is $\mathcal{F}(5) = \{0101, 010*, 01**, 0***, ****\}$. To query all data item in range $[4, 7]$, the search query is generated as $S([4, 7]) = \{01**\}$. We have $\mathcal{F}(5) \cap S([4, 7]) = 01** \neq \emptyset$, thus 5 falls in range $[4, 7]$. It is worth noting that the number of prefix elements in $S([y_{min}, y_{max}])$ is at most $2\omega - 2$, where y_{min} and y_{max} are two ω -bits data items. Fig. 4(b) shows examples of geometric range queries over a spatial database through our spatial data and query encoding method. Suppose there are four spatial objects in the spatial database, each object is encoded by Hilbert curve and represented as prefix family. To query all objects in R_1 , a search user first encodes query range into $\{[2, 7], [56, 58]\}$. Then, the search user generates the corresponding query prefixes as $\{0001**, 00001*\}$ and

Algorithm 1 SKSE-I Construction

Setup(1^λ) $\rightarrow K_\Sigma$:

$msk \leftarrow \text{HVE.Setup}(1^\lambda)$;
Randomly selects $HK = \{H_i(\cdot)\}_{i=1}^t$;
return $K_\Sigma = \{msk, HK\}$.

IndexBuild(K_Σ, DB) $\rightarrow \text{EDB}$:

foreach data object $O_i = \{p_i, \mathcal{W}_i, \text{ind}_i\}$ **do**
 $e_{O_i} \leftarrow \mathcal{F}(\mathcal{H}(p_i))$;
 $B_{O_i} \leftarrow \text{BF}(HK, e_{O_i} \cup \mathcal{W}_i)$;
 $\text{cHVE}(B_{O_i}) \leftarrow \text{HVE.Enc}(msk, \text{"true"}, B_{O_i})$;
return $\text{EDB} = \{\text{cHVE}(B_{O_i}), \text{ind}_i\}_{i=1}^N$.

TokGen($K_\Sigma, Q = \{\mathcal{W}_Q, R_Q\}$) $\rightarrow T_Q$:

$S_Q \leftarrow S(\mathcal{H}(R_Q))$;
foreach $s_k \in S_Q (1 \leq k \leq \beta)$ **do**
 $B_{Q_k} \leftarrow \text{BF}(HK, s_k \cup \mathcal{W}_Q)$;
 $\text{sHVE}(B_{Q_k}) \leftarrow \text{HVE.KeyGen}(msk, B_{Q_k})$;
return $T_Q = \{\text{sHVE}(B_{Q_k})\}_{k=1}^\beta$.

Search(T_Q, EDB) $\rightarrow \mathcal{R}$:

foreach encrypted index $\{\text{cHVE}(B_{O_i}), \text{ind}_i\} \in \text{EDB}$ **do**
 foreach search token $\text{sHVE}(B_{Q_k}) \in T_Q$ **do**
 Flag $\leftarrow \text{HVE.Query}(\text{cHVE}(B_{O_i}), \text{sHVE}(B_{Q_k}))$;
 if Flag == true **then** $\mathcal{R}.\text{Append}(\text{ind}_i)$;
return $\mathcal{R}$.

$\{11100*, 111010\}$, separately. Since $0001** \in \mathcal{F}(\mathcal{H}(O_1))$, we have $O_1 \in R_1$. To query all objects in R_2 , a search user first encodes query range into $\{[38, 47]\}$. Next, the search user generates the corresponding query prefixes as $\{101**, 10011*\}$. We have $O_3, O_4 \in R_2$ via $101*** \in \mathcal{F}(\mathcal{H}(O_3))$, and $10011* \in \mathcal{F}(\mathcal{H}(O_4))$.

2) *Formal Description of SKSE-I*: We now formally present our basic SKSE construction for *spatio-textual* data as shown in Algorithm 1. In particular, our construction uses hash functions $H(\cdot)$ and an HVE scheme HVE as follows:

Setup(1^λ) $\rightarrow K_\Sigma$: Given a security parameter λ , it randomly selects t hash functions $HK = \{H_i(\cdot)\}_{i=1}^t$, and generates a secret key $msk \leftarrow \text{HVE.Setup}(1^\lambda)$. $K_\Sigma = \{msk, HK\}$ is the secret key set shared by data owner and search users.

IndexBuild(K_Σ, DB) $\rightarrow \text{EDB}$: Given a database DB and a secret key set K_Σ , this algorithm generates the EDB. For each object $O_i = \{p_i, \mathcal{W}_i, \text{ind}_i\}$ in DB, encoding its spatial data p_i to $\mathcal{F}(\mathcal{H}(p_i))$, and then set up a BF containing all textual keyword in $\mathcal{W}_i$ and encoded spatial data. Finally, the BF of every object is encrypted by HVE.Enc according to msk .

TokGen(K_Σ, Q) $\rightarrow T_Q$: Given a search query $Q = \{\mathcal{W}_Q, R_Q\}$, this algorithm generates an encrypted search token T_Q . First, it encodes R_Q into $S(\mathcal{H}(R_Q))$. Then, for each prefix element $s_k \in S(\mathcal{H}(R_Q)) (1 \leq k \leq \beta)$, set up a BF containing $\mathcal{W}_Q \cup s_k$ and encrypt it using HVE.KeyGen, where the '0' positions are set to '*', and the '1' positions are still '1'. The search token T_Q is the set of all encrypted BFs.

Search(T_Q, EDB) $\rightarrow \mathcal{R}$: Given a search token T_Q and an encrypted database EDB, it evaluates encrypted BFs one

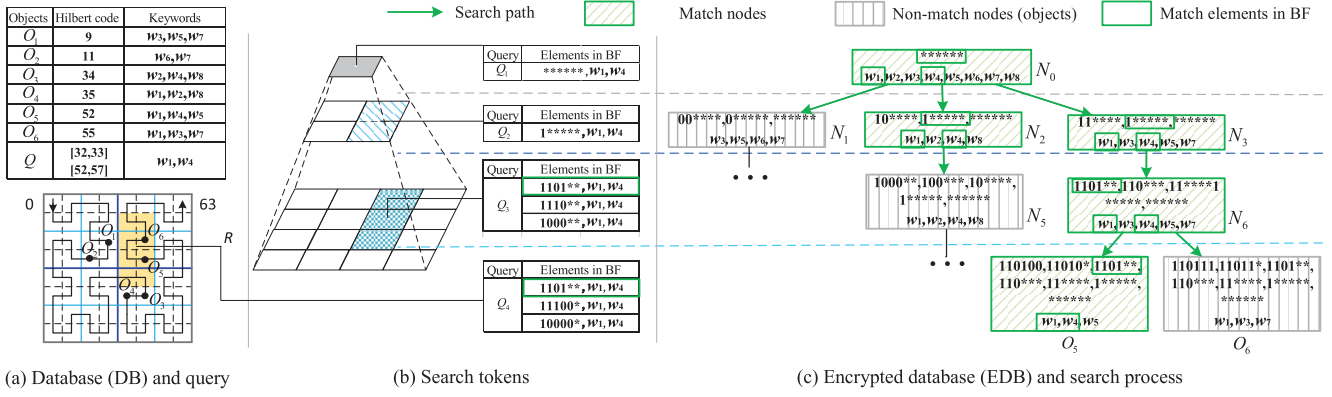


Fig. 5. BRQ over BH-tree.

by one using HVE.Query according to search token. If the evaluation result is “true”, add the corresponding ind_i into the result set $\mathcal{R}$.

3) *Complexity*: Suppose there are β query BF, since the ciphertext of every bit in encrypted BF is λ -bits, the token size of SKSE-I is $m\lambda\beta$ -bits. As described above, the number of prefix elements in a range is at most $2\omega - 2$. Obviously, the min number of encoded range and prefix elements are both 1, thus the min number of query BF $\beta_{\min} = 1$. Suppose that the search query is converted into a ranges, we have the max number of query BF $\beta_{\max} = a \cdot (2\omega - 2)$. As for storage cost, the EDB contains N objects, which cost $m\lambda N$ -bits. In the search process, every encrypted index should be searched by all encrypted BF in search token, which cost $O(\beta N)$ HVE query operations.

B. SKSE-II: Efficiency Improved SKSE Construction

In the basic SKSE construction described above, the search complexity is $O(\beta N)$. However, such a linear search complexity construction is difficult to cope with the challenges of large-scale datasets. Here, we present a sub-linear search complexity SKSE construction.

1) *Overview of SKSE-II*: To achieve sub-linear search complexity, we organize the encoded data objects as a tree structure, called Bloom filter Hierarchical tree (BH-tree). In a BH-tree, every *spatio-textual* object is a leaf node, and every non-leaf node covers the spatial range and textual keywords of its children. All nodes of BH-tree are represented using BF. In the search process, the BH-tree is traversed from the root node, and only the subtree that satisfies the query conditions will be continue queried. Hence, the search space is pruned dynamically, which greatly reduces retrieval complexity. If we set the BF in all levels with the same length (i.e., set the BF of higher levels to the superimpositions of the lower levels), that will lead to more false positives in the higher levels.⁷ Higher false positive will reduce the pruning ability, thus increasing the search space. A simple idea is to increase the length of all BF to satisfy the requirements of the highest level. However, it will lead to meaningless storage and computation cost at lower levels. To address this issue,

⁷There will be more ‘1’ positions in higher levels’ BF.

we can use varying BF lengths for different levels to reduce the false positive, particularly in non-leaf nodes. Note that the number of spatial prefixes and textual keywords in higher levels is larger than that of those in lower levels. To minimize the BF size, we need to reduce the number of spatial prefixes at higher levels since the number of textual keywords cannot be reduced. Instead of setting all spatial prefixes of the lower levels, we only set the minimal prefixes that cover its range into the BF. As an example, the prefixes of a node covers [4, 7] is generated as {01 **, 0 ***, * * **}, instead of {0100, 0101, 0110, 0111, 011*, 010*, 01 **, 1 ***, 0 ***, * * **}. Let M be the max number of children in each non-leaf node, we set $M = 4^{\vartheta}$ to minimize the prefixes. Specifically, the BF length for each level is obtained from all elements cover the subtree of each node, instead of all elements of the children nodes as before. In this case, to securely evaluate the BF at different levels, we generate different query BF for different levels according to the minimal cover range (i.e., all nodes in this level that intersect with the query range.) and the query keywords.

Example: Figure 5 shows an example of BH-tree with order-3 Hilbert curve, where $M = 4$. As shown in Figure 5(a), the database DB consists of six objects with eight textual keywords in total, and the search query is $Q = \{R = \{[32, 33], [52, 57]\}, \mathcal{W}_Q = \{w_1, w_4\}\}$. To perform BRQ on the encrypted database EDB, we generate query BF for different levels of the index tree, where the textual elements are $\mathcal{W}_Q$ and the spatial elements is the minimal area at this level that covers the query range R . For example, the minimal area that covers R at the 2-th level and 3-th are {1 * * * *} and {1101 **, 1110 **, 1000 **}, respectively (See Figure 5(b)). Figure 5(c) shows the search path on EDB. Starting from the 1-th level (i.e., the root node), since $Q_1 \in N_0$, continue to search its children. At the 2-th level, the N_1 is pruned since $Q_2 \notin N_1$ and the N_2 ’s and N_3 ’s children are searched. Next, the N_5 is pruned since $Q_3 \notin N_5$ at the 3-th level. Finally, O_5 is returned as the result.

2) *Formal Description of SKSE-II*: The formal description of SKSE-II is shown in Algorithm 2.

Setup(1^λ) $\rightarrow K_\Sigma$: Given a security parameter λ , it generates a secret $\text{msk} \leftarrow \text{HVE.Setup}(1^\lambda)$. Then, for each level of BH-tree, it randomly selects t independent hash functions

Algorithm 2 SKSE-II Construction

```

Setup( $1^\lambda$ )  $\rightarrow K_\Sigma$ :
   $msk \leftarrow \text{HVE.Setup}(1^\lambda)$ ;
   $HK^j = \{H_i^j(\cdot)\}_{i=1}^t$ ;
  return  $K_\Sigma = \{msk, HK = \{HK^j\}_{j=1}^\delta\}$ .

IndexBuild( $K_\Sigma, \text{DB}$ )  $\rightarrow \text{EDB}$ :
   $\mathcal{T} \leftarrow \text{TreeBuild}(\text{DB})$ ;
  foreach leaf node  $O_i = \{p_i, \mathcal{W}_i, \text{ind}_i\}$  do
     $e_{O_i} \leftarrow \mathcal{F}(\mathcal{H}(p_i))$ ;
     $B_{O_i} \leftarrow \text{BF}(HK^\delta, e_{O_i} \cup \mathcal{W}_i)$ ;
     $c_{\text{HVE}}^\delta(B_{O_i}) \leftarrow \text{HVE.Enc}(msk, \text{"true"}, B_{O_i})$ ;
  foreach  $j$ -th level non-leaf node  $\mathcal{N} = \{R_\mathcal{N}, \mathcal{W}_\mathcal{N}\}$  do
     $e_\mathcal{N} \leftarrow \mathcal{F}(\mathcal{H}(R_\mathcal{N}))$ ;
     $B_\mathcal{N} \leftarrow \text{BF}(HK^j, e_\mathcal{N} \cup \mathcal{W}_\mathcal{N})$ ;
     $c_{\text{HVE}}^j(B_\mathcal{N}) \leftarrow \text{HVE.Enc}(msk, \text{"true"}, B_\mathcal{N})$ ;
  return  $\text{EDB}$ .

TokGen( $K_\Sigma, Q = \{\mathcal{W}_Q, R_Q\}$ )  $\rightarrow T_Q$ :
  foreach  $j$ -th level of the BH-tree do
     $S_Q^j \leftarrow S_j(\mathcal{H}(R_Q))$ ;
    foreach  $s_k^j \in S_Q^j$  ( $1 \leq k \leq \beta$ ) do
       $B_{Q_k}^j \leftarrow \text{BF}(HK^j, s_k^j \cup \mathcal{W}_Q)$ ;
       $s_{\text{HVE}}(B_{Q_k}^j) \leftarrow \text{HVE.KeyGen}(msk, B_{Q_k}^j)$ ;
     $T_Q^j = \{s_{\text{HVE}}(B_{Q_k}^j)\}_{k=1}^\beta$ ;
  return  $T_Q = \{T_Q^j\}_{j=1}^\delta$ .

Search( $T_Q, \text{EDB}$ )  $\rightarrow \mathcal{R}$ :
  Start from the root node of  $\text{EDB}$ ;
  foreach  $j$ -th level non-leaf node  $\mathcal{N}$  in search path do
    for  $1 \leq k \leq \beta$  do
       $\text{Flag} \leftarrow \text{HVE.Query}(c_{\text{HVE}}^j(B_\mathcal{N}), s_{\text{HVE}}(B_{Q_k}^j))$ ;
      if  $\text{Flag} == \text{true}$  then
        Continue to search its children;
      while  $k == \beta$  do Stop to search this subtree;
  foreach leaf node  $O_i$  in search path do
    foreach search token  $s_{\text{HVE}}(B_{Q_j}) \in T_Q^\delta$  do
       $\text{Flag} \leftarrow \text{HVE.Query}(c_{\text{HVE}}^\delta(B_{O_i}), s_{\text{HVE}}(B_{Q_j}^\delta))$ ;
      if  $\text{Flag} == \text{true}$  then  $\mathcal{R}.\text{Append}(\text{ind}_i)$ ;
  return  $\mathcal{R}$ .

```

$HK^j = \{H_i^j(\cdot)\}_{i=1}^t$. The secret key set is $K_\Sigma = \{msk, HK = \{HK^j\}_{j=1}^\delta\}$, where $\delta = \log_M N$ is the depth of the BH-tree.

IndexBuild(K_Σ, DB) $\rightarrow \text{EDB}$: Firstly, this algorithm generates an hierarchical tree according to the DB, where every non-leaf node contains M children. Next, for each leaf node $O_i = \{p_i, \mathcal{W}_i, \text{ind}_i\}$, it encodes the spatial data p_i as $\mathcal{F}(\mathcal{H}(p_i))$, and sets up a BF containing all textual keywords in $\mathcal{W}_i$ and encoded spatial data according to the δ -th level BF HK^δ ; for each j -th level non-leaf node $\mathcal{N}$, this algorithm encodes the spatial range as $\mathcal{F}(\mathcal{H}(R_\mathcal{N}))$, and sets up a BF containing all textual keywords in $\mathcal{W}_\mathcal{N}$ and encoded spatial

data according to HK^j . Finally, the BF of every node is encrypted by HVE.Enc according to msk .

TokGen(K_Σ, Q) $\rightarrow T_Q$: Given a search query $Q = \{\mathcal{W}_Q, R_Q\}$, this algorithm generates the encrypted search token T_Q . For j -th level, it generates query prefixes according to query range R and sets up BFs containing all textual keywords in $\mathcal{W}_Q$ and the query prefixes by using HK^j .

Search(T_Q, EDB) $\rightarrow \mathcal{R}$: For the j -th level, it checks the encrypted BF of each node using HVE.Query , if anyone of the query BFs matches with a node, continue to search its children; otherwise, stop search this subtree. When arrival a leaf node, check the encrypted BF using HVE.Query , if the result is "true", add the corresponding ind_i into $\mathcal{R}$.

3) *Complexity*: For the convenience of description, we assume that the average size of BFs is m , and the number of non-leaf nodes in BH-tree is $\mathcal{X}$. The token size of SKSE-II is $\delta\beta m\lambda$ -bits since the search token is a set of $\delta\beta$ BFs. As for storage cost, the EDB contains $\mathcal{X}$ non-leaf nodes and N leaf nodes (data objects), which cost $m\lambda(N + \mathcal{X})$ -bits. In the search process, the search space will be pruned dynamically. The worst case is that all objects in the dataset match the search query, which means the cloud server traverses the entire tree, which cost $O(\beta(N + \mathcal{X}))$ operations⁸; the best case happens when only a single object matches the search query, which means the cloud server traverses a single node at every level of the index tree, which costs $O(\beta M \log N)$ operations. Let $\bar{A}$ be the average number of the nodes at each level that traversed by cloud server, the average search complexity is $O(\bar{A}\beta M \log N)$.

C. SKSE-II-PV: SKSE Construction Without False Positive

Although we have carefully designed the index structure to reduce the number of elements in the BFs, we still need to maintain the length of the BFs to reduce the false positive. In HVE, every bit of a BF is encrypted to λ -bits ciphertext and costs a single evaluate operations. The long length of BFs incurs huge storage and computational cost. In addition, the false positive also leads to dirty results (i.e., non-matched files in the results set). The dirty results cannot be found unless the user decrypts all result files. However, the size of private files is usually much larger than that of the index. *Is it possible to present an SKSE construction with lower overhead, and without false positives?* Here, we propose a post-verification approach that allows search users to remove dirty results without decrypting private files. Specifically, a bitmap is stored for each private file to verify the results set. Besides, we introduce a compression algorithm to reduce the storage overhead of bitmaps. Using this approach, the length of BFs can also be significantly reduced, thereby reducing storage and computational cost.

1) *Proposed Post-Verification Approach*: To verify the results set, we store a bitmap for each file associated with the identifier ind . The bitmap is a simple and efficient index structure, which stores information using 0,1 bits [33]. The simplest and most common method of bitmap indexing associates a bitmap with the position of elements, i.e., if an

⁸In real-world applications, the range of query area and textual keywords is much smaller than that of the entire database, that the worst case is impossible.

element exists in a set, then set the corresponding position to 1; otherwise, set the corresponding position to 0. For example, given a textual keyword set $\{w_1, w_2, w_3, w_4, w_5\}$, if w_3 exists, then set the 3-th position to 1, the bitmap is 00100; as for a digital number with ω -bits width, generate a 2^ω -bits bitmap and set the corresponding position to 1. We can achieve complex queries with logical operations over bitmaps, which are computationally efficient in a machine. Although the bitmap is computationally efficient, it causes huge space cost in both storage and computation. For instance, for a set containing 1000 keywords, it will cost 1000 bits to store a single textual keyword using a bitmap, however, only several Bytes using ASCII code. To address this issue, we introduce the Enhanced Word-Aligned Hybrid (EWAH) compression algorithm [33]. EWAH is made of two different types of words: the first type is a 32-bit dirty word⁹; the second type of word is a marker word, where the first bit indicates which clean word¹⁰ will follow, half the bits (32 bits) are used to store the number of clean words, and the rest of the bits (31 bits) are used to store the number of dirty words following the clean words.¹¹ EWAH bitmaps begin with a marker word. For instance, if the input bitmap is $0^{194}1^80^{115}0^3$ (320 bits), then EWAH partitions it into three groups: $G_1(0^{192})$, $G_2(0^21^80^{54})$, and $G_3(0^{61}0^3)$. Furthermore, it generates a marker word as $00^{30}110^{29}10$, followed by two dirty word $0^21^80^{54}$, $0^{61}0^3$. It is worth noting that the bitmap of the Hilbert curves can be compressed into three groups since there is only a single dirty word. As for the textual keyword, the size of bitmaps can be significant compression compressed since only several keywords are contained in each object. Our approach consists of three main steps: Given the dataset **DB**, in the first step, we generate a bitmap for each object and compress it. The second step encrypts the compressed bitmaps and stores them associated with the **ind** as verify tokens. When a search query is performed, the verify tokens are returned to the search user together with the results set. After receiving the tokens, the search user decrypts them and verifies the results set according to the compressed bitmaps.¹² In SKSE-II, to minimize the dirty results, BFs should be set as long as possible. For instance, to keep the false positive $f_p \leq 10^{-4}$, we need to set $t = 14$, $m/n = 20$. Using the post-verification approach, it is unnecessary to set false positive to such a small value. We only need to keep the basic filterability of BFs and obtain accurate results according to the post-verification approach. For instance, we only need to keep the false positive $f_p \leq 3\%$, which set $t = 5$ and $m/n = 6$. Thus the storage and computational overhead can be significantly reduced.

2) *Formal Description of SKSE-II-PV*: Here, we formally describe the SKSE-II with the post-verification approach (namely SKSE-II-PV). Denote **sym.Enc**, **sym.Dec** be the

⁹Dirty word consists of both “0” and “1” bits.

¹⁰Clean word consists of either “0” bit or “1” bit, i.e., 00000000000000000000000000000000 (0³² in short) and 1³².

¹¹For simplicity, we limit our exposition to 32-bits words, which satisfies the 64-bits operation system.

¹²For more details about the faster operations over compressed bitmaps, see [33].

encryption and decryption algorithms of a symmetric encryption scheme (e.g., AES), separately.

Setup(1^λ) $\rightarrow K_\Sigma$: Besides the keys generated in the SKSE-II, a symmetric key sk is generated to encrypt verify tokens.

IndexBuild(K_Σ, DB) $\rightarrow \text{EDB}$: Besides the **EDB** generated in the SKSE-II, this algorithm generates a verify token for each *spatio-textual* object. Firstly, it generates a bitmap as $\text{bm}_i \leftarrow \text{Bitmap}(\mathcal{H}(\text{p}_i) \cup \mathcal{W}_i)$. Then, this algorithm compresses the bitmap as $\mathcal{C}_i \leftarrow \text{Compress}(\text{bm}_i)$ and encrypts the compressed bitmap using **sym.Enc**. Finally, the encrypted bitmap (verify token) tk_i is stored associated with the ind_i .

TokGen(K_Σ, Q) $\rightarrow T_Q$: The same as SKSE-II.

Search(T_Q, EDB) $\rightarrow \mathcal{R}$: In the search process, cloud server adds $\{\text{ind}_i, \text{tk}_i\}$ into the results set $\mathcal{R}'$. After receiving the $\mathcal{R}'$, the search user decrypts the verify tokens using **sym.Dec** and verifies the tokens according to the bitmaps of the query, if the bitmaps are matched, keeps the corresponding ind_i in the result set; otherwise, remove the corresponding ind_i .

3) *Complexity*: The token size of SKSE-II-PV is still $\delta m \lambda$ -bits. Since a verify token needed to be stored for each data object, which costs $\mu \lambda$ -bits, the storage cost is $(\mu + mN)\lambda$ -bits, where μ is the number of compressed groups. The search process cost additional $O(|\mathcal{R}'|)$ operations for verify token decryption and verification at the user side.

D. SKSE in Public-Key Setting

The above-mentioned constructions are constructed in the symmetric-key setting, which are suitable for a single-user sharing scenario (i.e., only a single user or multiple users who fully trust each other). However, in a practical scenario, the *spatio-textual* data services may have multiple users who do not trust each other, that is, they do not want to disclose their search queries to other users. The search token in symmetric-key setting can be decrypted by other users, which is unsuitable. To ensure the user's query privacy in multi-user setting, we can expend our constructions by using public-key HVE. Specifically, in the public-key SKSE, a data owner only shares the public key of HVE with search users and keeps the private key secret. In this way, search users can only generate search tokens according to the public key and cannot decrypt them, thereby preventing the query privacy of a search user from leaking to other search users.

V. SECURITY ANALYSIS

Our SKSE constructions follow the security properties from HVE that only leak the fact that whether an encrypted vector matches the other encrypted vector or not. In our constructions, we set the indexes (including leaf nodes and non-leaf nodes) and search queries into Bloom filters, and then secure evaluate them using HVE. If the HVE used in our constructions is secure, our constructions are secure.

A. Leakage Functions

Before formal security analysis, we define a few simple leakage functions to evaluate the privacy leaks in SKSE.

Size Pattern (SP): $SP(DB) \rightarrow N$, where N is the number of objects in a database DB .

Whole Result Pattern (WRP): In SKSE, the search query contains not only textual keyword, but also spatial range. Given a BRQ query Q , the whole result pattern is all identifiers that matching Q :

$$WRP(Q, DB) \rightarrow \{ind_j\}_{j \in [N], p_j \in \mathcal{R}_Q \wedge \mathcal{W}_Q \in \mathcal{W}_j}.$$

For a set of queries, the adversary can obtain the intersection of any different queries' results.

Query Pattern (QP): The adversary can obtain which nodes are traversed during the search process:

$$QP(Q, DB) \leftarrow \{\mathcal{N}_j\}_{j \in [N], \mathcal{W}_Q \in \mathcal{W}_{\mathcal{N}_j} \wedge \mathcal{R}_Q \cap \mathcal{R}_{\mathcal{N}_j} \neq \emptyset}.$$

For a set of queries, an adversary can distinguish whether two queries traverse a common node.

B. Security of Proposed SKSE Constructions

Here, we prove the security of our SKSE constructions. Generally speaking, our SKSE constructions are indistinguishability under Selective Chosen-Plaintext Attacks (IND-SCPA) with different leakage functions. The details are described as follows.

Theorem 1 (Security of SKSE-I): Define $\mathcal{L}_1$ as:

$$\mathcal{L}_1(Q, DB) = \{SP(DB), WRP(Q, DB)\},$$

SKSE-I is IND-SCPA secure if HVE is IND-SCPA secure.

Proof: The idea of this proof is to simulate the experiments of our SKSE-I construction with an adversary $\mathcal{A}$ from the experiments of HVE, and demonstrate that compromise the security of SKSE-I is equivalent to compromise the security of HVE. Specifically, we have

- **Setup:** The adversary $\mathcal{A}$ chooses a database $DB = \{p_i, \mathcal{W}_i, ind_i\}_{i=1}^N$ and sends it to challenger. The challenger $\mathcal{C}$ randomly selects t hash functions $HK = \{H_i(\cdot)\}_{i=1}^t$, and generates a secret key $msk \leftarrow HVE.Setup(1^\lambda)$. Finally, $\mathcal{C}$ keeps $\{msk, HK\}$ private.
- **Phase 1:** The adversary $\mathcal{A}$ adaptively chooses a number of queries $Q_j = \{\mathcal{R}_{Q_j}, \mathcal{W}_{Q_j}\}$ for $j \in [q_1]$. For each such query, the challenger $\mathcal{C}$ encodes the query range $\mathcal{R}_{Q_j}$ into $S(\mathcal{H}(\mathcal{R}_{Q_j}))$. Then, for each prefix element $s_k \in S(\mathcal{H}(\mathcal{R}_{Q_j}))$ ($1 \leq k \leq \beta$), $\mathcal{C}$ sets up a BF containing $\mathcal{W}_{Q_j} \cup s_k$ and encrypts it using $HVE.KeyGen$, where the '0' positions are set to '*', and the '1' positions are still '1'. Finally, $\mathcal{C}$ responses the search token T_{Q_j} to $\mathcal{A}$.
- **Challenge:** The challenger $\mathcal{C}$ flips a coin $b \in \{0, 1\}$. For each data object O_i , $\mathcal{C}$ generates the corresponding BF as $BF_{O_i} \leftarrow BF(HK, \mathcal{F}(\mathcal{H}(p_i) \cup \mathcal{W}_i))$. Then, $\mathcal{C}$ provides $\mathcal{A}$ with the challenge encrypted database EDB according to $HVE.Enc(msk, BF_{O_i})$ for $i \in [N]$.
- **Phase 2:** The adversary $\mathcal{A}$ runs a protocol identical to **Phase 1** and receives T_{Q_j} for $q_1 + 1 \leq j \leq q_2$.
- **Guess:** The adversary takes a guess of b .

The simulator can be generated by adding the leakage functions of SKSE-I and replacing the HVE algorithm using the corresponding HVE simulator. Since we successfully simulate the experiments of our SKSE-I with an adversary $\mathcal{A}$ from the

experiments of HVE, the indistinguishability of the encrypted database EDB and the search query Q_j for $j \in [q_2]$ from the real and simulate experiment follows directly from the facts that the indistinguishability of the HVE. The experiment of our SKSE-I is essentially simulated by q' instances of HVE, where q' is the total number of HVE instances. As a result, the adversary $\mathcal{A}'$ could distinguish the encrypted database EDB and the search query Q_j as long as it could distinguish any pair of two vectors of HVE, which can be interpreted as:

$$\begin{aligned} \text{Adv}_{\text{SKSE-I}, \mathcal{A}}^{\text{IND-SCPA}}(1^\lambda) &\leq \text{Adv}_{\text{HVE}, \mathcal{A}'}^{\text{IND-SCPA}}(1^\lambda) \\ &\leq q' \cdot \text{negl}(\lambda) \end{aligned}$$

which proves the IND-SCPA secure of our SKSE-I. $\square$

Theorem 2 (Security of SKSE-II): Define $\mathcal{L}_2$ as:

$$\mathcal{L}_2(Q, DB) = \{SP(DB), WRP(Q, DB), QP(Q, DB)\},$$

SKSE-II is IND-SCPA secure if HVE is IND-SCPA secure.

Proof: We can still use a similar way to our SKSE-II as we did in the security proofs of SKSE-I to demonstrate the security of SKSE-II. Specifically, we can still simulate the experiments of SKSE-II with an adversary $\mathcal{A}$ in the experiments of HVE, and prove that if adversary $\mathcal{A}$ is able to break the security experiments, it is equivalent to break the security experiments of HVE, which contradicts to the assumption that HVE is secure. Due to space limitations, we skip further details. $\square$

C. Discussion of Attacks

Most existing STE schemes leak the set of results matching the query (i.e., result pattern) and query pattern (unless an ORAM is used, but it will incur multiple interactions) to the cloud server. Our SKSE only leaks the result set and query pattern matching both spatial range and textual keywords. As described in [12], the file-injection attacks [19] focus on conjunctive query SE schemes are unusable on the STE constructions with the whole result pattern. The volume-based or reconstruct attacks [20], [21], [34] can reconstruct the whole spatial database according to a large number of records of different range queries' volume information (i.e., the number of all objects matching a range query) or query pattern. In our constructions, an attacker can only obtain the volume information or query pattern of all objects (nodes) matching both geometric query range and all the query keywords. This makes it difficult for an attacker to complete the attack unless every data object contains all query keywords, which is almost impossible in practical applications.

Here, we give a simple example to clarify the leakages in our constructions, and intuitive display why our constructions leak less information than combine a secure spatial search construction and a secure textual search construction. The example dataset is shown in Table II. Given a search query $Q_1 = \{R_1 = [10, 50], \mathcal{W}_1 = \{w_4\}\}$, our constructions only leaks the result $O_1, O_3 \leftarrow EDB(Q_1)$. If we combine a secure spatial search construction and a secure textual search construction, the spatial database will leak $O_1, O_2, O_3 \leftarrow EDB(R_1)$ and the textual database will leak $O_1, O_3, O_5 \leftarrow EDB(\mathcal{W}_1)$. In this case, our constructions protect the spatial information of O_2

TABLE II
EXAMPLE DATASET

Object	Hilbert Code	Textual Keywords
O_1	13	w_1, w_3, w_4
O_2	28	w_2, w_5
O_3	46	w_1, w_2, w_4
O_4	54	w_2, w_3, w_5
O_5	77	w_2, w_3, w_4

and the textual information of O_4, O_5 . Similarly, for a search query $Q_2 = \{R_2 = [30, 80], \mathcal{W}_2 = \{w_4\}\}$, our constructions only leaks the result $O_3, O_5 \leftarrow \text{EDB}(Q_2)$. In a combined scheme, it leaks $O_3, O_4, O_5 \leftarrow \text{EDB}(R_2)$ and $O_1, O_3, O_5 \leftarrow \text{EDB}(\mathcal{W}_2)$. In this case, our construction protect the spatial information of O_4 and the textual information of O_1 . After obtaining the above queries Q_1 and Q_2 , the adversary can only learn the intersect of the query results $\text{EDB}(Q_1) \cap \text{EDB}(Q_2) = O_3$.

Next, we explain why such leakages can resist the above-mentioned leakage attacks used in spatial and textual databases. Let D be the spatial range in the spatial database. As described in [34], for a *desnse* database (i.e., all values appear in at least one record), an attacker needs to see at least $O(D \log D)$ distinct queries to reconstruct the whole database; as for a *sparse* database, an attacker needs to see at least $O(D^2 \log D)$ distinct queries to reconstruct the whole database. It is worth noting that the attack succeeds after the attacker has seen each of the possible query results at least once, independent of their distribution. In our constructions, however, the attacker cannot obtain enough information as much as in a range query scheme. As described in the above example, our constructions can protect the spatial information of some objects that do not match query keywords (i.e., O_2 for Q_1 and O_4 for Q_2). Thus, the attacker has to monitor more search queries that contain all possible range queries and textual keywords to obtain all possible results. Suppose each BRQ query only contains a single keyword, to obtain all possible search results, the attack complexity grows to $O(|W|D \log |W|D)$ and $O(|W|D^2 \log |W|D)$, where $|W|$ is the size of query keyword dictionary. Suppose each BRQ query contains m keyword, the attack complexity is $O(\frac{|W|!D}{m!(|W|-m)!} \log \frac{|W|!D}{m!(|W|-m)!})$ and $O((\frac{|W|!D}{m!(|W|-m)!})^2 \log \frac{|W|!D}{m!(|W|-m)!})$. In our construction, we set the Hilbert order to $h = 24$. Thus, all of the databases used in evaluation are *sparse* databases, and the attacker has to monitor at least $O((3000 \times 2^{24})^2 \log(3000 \times 2^{24})) \approx 2^{75}$ queries to reconstruct the database. Sometimes, the adversary might be able to observe many queries within a particular range. For a particular range covers 1‰ space of the whole database the attacker still needs to collect at least $O(3000 \times 2^{14})^2 \log(3000 \times 2^{14}) \approx 2^{34}$ queries to reconstruct the database. Therefore, our constructions are strong enough to resist the existing state-of-the-art attacks and guarantee data and query security.

VI. PERFORMANCE EVALUATION

In this section, we evaluate the performance of our proposed constructions and compare them with state-of-the-art solutions with various settings.

TABLE III
COMPARE OF BF LENGTH (BITS)

	$ R = 2^{14}$	$ R = 2^{16}$	$ R = 2^{18}$	$ R = 2^{20}$
B_T	327,680	1,310,720	5,242,880	20,971,520
B_{TL}	10,920	26,480	84,510	184,640
Our BF	520	600	680	760

TABLE IV
DETAILS OF THE DATASETS USED IN THE EVALUATION

Description	NY	FLA	UF
# of objects N	264,346	1,070,376	1×10^6
Avg. keywords per object	12	12	10
total # of keyword	3,173,451	12,692,685	1×10^7
keyword dictionary size	3,000	3,000	3,000

A. BF Parameters and Query Accuracy

In all of our SKSE constructions, we use BFs to support BRQ. Let t be the number of hash functions in BFs, m be the length of BFs, n be the number of query elements. In SKSE-I and SKSE-II, the false positive is $f_p \leq (1 - e^{-t \cdot n/m})^t$ as described in Section II-A. We set $t = 14, m/n = 20$ so that the false positive $f_p < 10^{-4}$. For SKSE-II-PV, search user can clean up dirty results using the post-verification approach to ensure the accuracy of the final results without relying on the accuracy of BFs. We only need to keep the basic filterability of BFs and obtain accurate results according to the post-verification approach. Thus, we set $t = 5, m/n = 6$ to reduce the storage and computational cost while maintaining the basic pruning ability of BH-tree. We also compare the BF in our constructions with traditional BF and the linear transformed BF in ELCBFR+ [2] (denoted as B_T and B_{LT} respectively) with varying query range $|R|$, where $t = 14, m/n = 20$. We can observe from Table III that the BF length in our constructions is extremely smaller than that of B_T and B_{LT} . In our constructions, the number of prefix elements in a range is at most $2\omega - 2$, where ω is the bit-length of the range, which significantly reduces the length of the BF.

B. Experimental Settings

1) *Implementation and Setup*: The implement of our constructions is written in Java. We use the opensource code of HVEs and BF from [12]. Although we have experimented with both symmetric-key HVE [12] and public-key HVE [29], due to the limited space, we only show some public-key setting results in the evaluation of Search, and most results are in symmetric-key setting. The security parameter of symmetric-key HVE is set to $\lambda = 128$, and the security parameter of public-key HVE is $\lambda = 160$. The experiments are run on a machine with Intel Xeon CPU E5-2640 v3 2.6GHz and 128 GB RAM. Although our constructions can support and geometric range queries, we set query range as a square for easy to control range size. Each experimental result is an average of 10 trials.

2) *Datasets*: As shown in Table IV, we use three datasets to evaluate the performance of our constructions, where NY and FLA are real-world datasets, and UF is a synthetic dataset.

TABLE V
EXPERIMENTAL PARAMETER SETTINGS

Description	Setting
# of query keywords $ \mathcal{W}_Q $	1 2 3 4 5
max number of children M	4 16 64
query range r (% of the map)	1% 2% 3% 4% 5%
dataset	NY FLA UF

As commonly used in spatial keyword researches [1], [2], [6], NY and FLA are a combination of two real spatial datasets¹³ and a real document dataset 20 Newsgroups.¹⁴ The spatial data of NY and FLA are from USA road network nodes of New York City and Florida, separately, and textual keywords of each spatial location are randomly selected from the top-3000 frequently keywords in 20 Newsgroups document dataset. In addition, we generate a synthetic dataset UF that has 1 million objects from the uniform distribution, where the keywords allocated to each object are completely random. Other parameter settings are shown in Table V, where the default value is bolded. NY and FLA are used to evaluate the performance of our constructions varying r , $\mathcal{W}_Q$, and M . UF is a synthetic dataset used to evaluate the performance of our constructions varying N .

C. Evaluation of Index Generation

There are two main factors affecting **IndexBuild**: (1) dataset size N ; (2) max number of children M in BH-tree. For each both aspects, we evaluate index generation time and EDB size.

Varying N : As shown in Fig. 6(a) and Fig. 6(b), when N increases, the index generation time and EDB size grow linearly. To improve search efficiency, SKSE-II generates a large amount of non-leaf nodes, which incurs additional storage and computational overhead compared with SKSE-I. In SKSE-II-PV, we adopt the post-verification approach that eliminates false positives and hence greatly reduces the length of BFs. In addition, the compressed bitmap introduced by the post-validation approach is storage-saving, therefore the EDB size and index generation time of SKSE-II-PV are optimal. We can see that compare with SKSE-II, our SKSE-II-PV has at least 60% storage and computation cost saving.

Varying M : From Fig. 6(c-f), we can observe that SKSE-I is not affected by M because it does not need to generate an index tree. Since the number of non-leaf nodes is decreased when M increases, both the index generation time and EDB size decrease in SKSE-II and SKSE-II-PV.

D. Evaluation of Token Generation

Here, we evaluate the impact of dataset size N , and the max number of children M in **TokGen**. For each both aspects, we evaluate token generation time and token size.

Varying N : Fig. 7(a) and Fig. 7(b) show that the token size and token generate time is constant in SKSE-I. As for SKSE-II and SKSE-II-PV, the number of their query BFs increases as the tree height increases, resulting in an increase in token

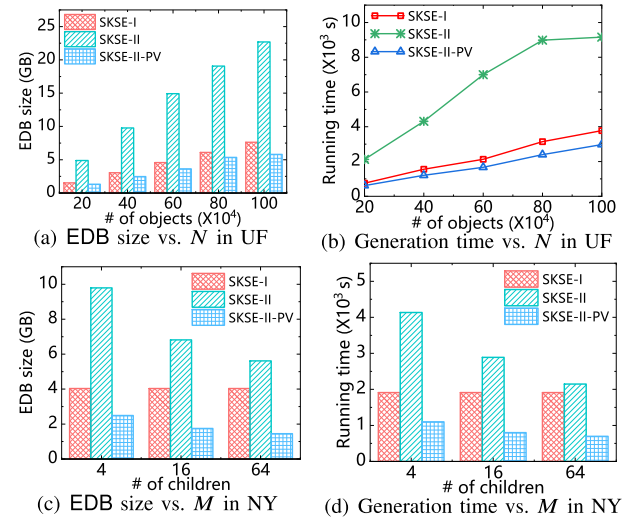


Fig. 6. Performance of IndexBuild.

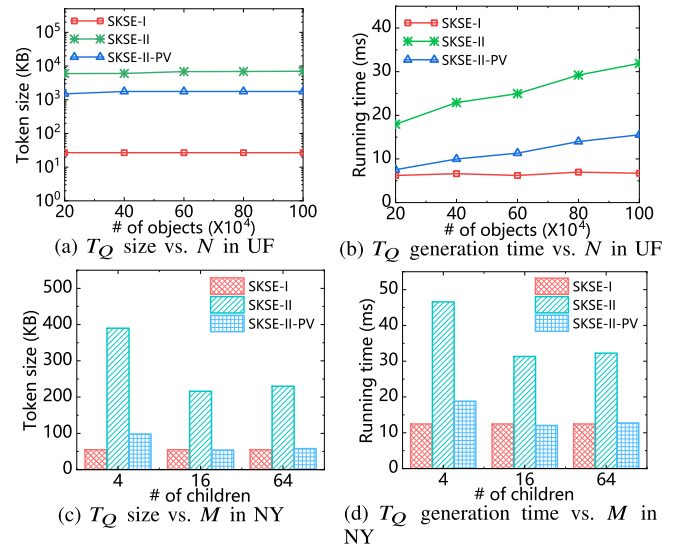


Fig. 7. Performance of TokGen.

size and token generate time. Since the length of BFs in SKSE-II-PV is much smaller than that of SKSE-II, SKSE-II-PV is more efficient than SKSE-II.

Varying M : As shown in Fig. 7(c-d), both the token generation time and token size decrease varying increasing M in both SKSE-II and SKSE-II-PV since the number of non-leaf nodes decreases varying increasing M . In SKSE-I, the average number of query tokens is 3.42. As for SKSE-II and SKSE-II-PV, the average number of query tokens is 24.46, 13.5, 14.38, separately, when $M = 4, 16, 64$, separately.

E. Evaluation of Search

There are five main factors affecting **Search**: (1) the size of dataset N ; (2) the max number of children M in BH-tree; (3) the query range r ; (4) the number of query keyword $|\mathcal{W}_Q|$.

Varying N : Fig. 8(a) shows that the running time of **Search** in all of our constructions grows with varying N . The running time in SKSE-I grows linearly with varying N , and the running time in SKSE-II and SKSE-II-PV has sub-linear

¹³<http://users.diag.uniroma1.it/challenge9/>

¹⁴<http://people.csail.mit.edu/jrennie/20Newsgroups>

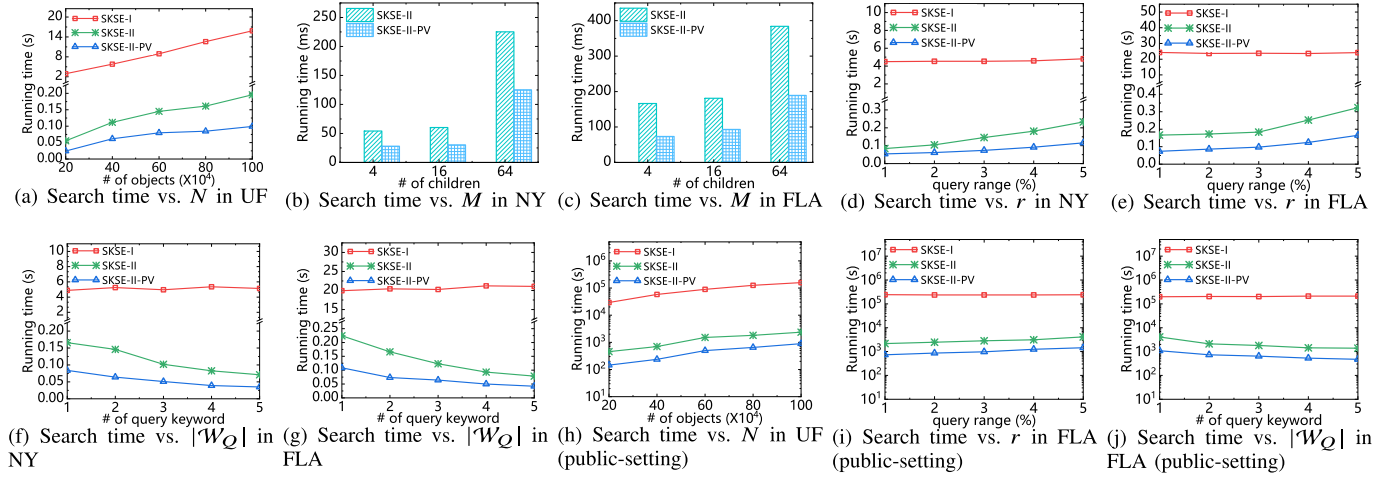


Fig. 8. Performance of Search.

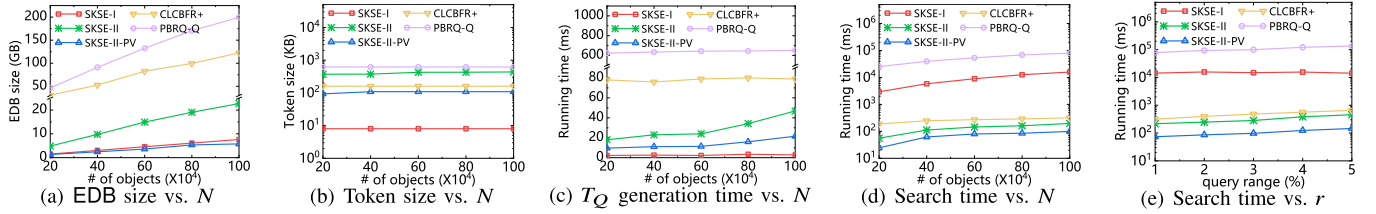


Fig. 9. Comparing with the existing solutions in UF.

growth with varying N , which satisfies the complexity analysis. Specifically, the running time of three constructions on one million data are 15839 ms, 196 ms, and 99 ms, respectively.

Varying M : From Fig. 8(b) and Fig. 8(c), we can observe that the running time of Search in SKSE-II and SKSE-II-PV grows with varying M . This is because a larger M results in a smaller depth of index tree, but larger search space per node, which leads to higher computational cost.

Varying r : As shown in Fig. 8(d) and Fig. 8(e), the search time in SKSE-II and SKSE-II-PV has linear-like growth since the search space has linear growth with varying r . As for SKSE-I, the running time of Search is constant since it has to traverse all objects. We can see that the maximums of the running time are 24181 ms, 323 ms, and 165 ms for our three constructions, respectively.

Varying $|\mathcal{W}_Q|$: For SKSE-II and SKSE-II-PV, more query keywords lead to stronger pruning capability, which reduces the search space and thus reduces search time. As for SKSE-I, it has to traverse all objects, which cost constant search time. The experimental results are shown in Fig. 8(f) and Fig. 8(g), we can observe that the maximums of the running time are 21218 ms, 224 ms, and 108 ms for our three constructions, respectively.

Performance in Public-Key Setting: The above evaluations are performed in a symmetric-key setting. Our constructions are generic that can be constructed from any HVE schemes, including symmetric-key and public-key settings. To fully evaluate the performance of our constructions, we simulate the search performance in public-key setting. The experimental result is shown in Fig. 8(h-j). We can see the search time of

each construction in public-key setting is at least $10^4 \times$ slower than that in symmetric-key setting.

F. Comparative Analysis

Here, we compare our constructions with the state-of-the-art solutions ELCBFR+ [2] and PBRQ-Q [6]. As described in Section I-A, the ELCBFR+ is based on the broken cryptography tool ASPE [4], which means it cannot guarantee data security. The security and leakages of PBRQ-Q are similar to those of our constructions. As shown in Fig. 9(a), the EDB size of our constructions are smaller than that in both ELCBFR+ and PBRQ-Q. Specifically, compared to PBRQ-Q and ELCBFR+, our SKSE-II-PV have 97% and 95% storage cost saving on one million data, respectively. As for TokGen, we can observe from Fig. 9(b) and Fig. 9(c) that the token size in our SKSE-II and SKSE-II-PV is larger than that in ELCBFR+, but the running time in both constructions is faster than that in ELCBFR+. SKSE-I is optimal in both token size and running time of TokGen, and PBRQ-Q is the worst in both respects. Fig. 9(d) and Fig. 9(e) show that the running time of Search in our SKSE-II and SKSE-II-PV outperforms the ELCBFR+ and PBRQ-Q. Specifically, our SKSE-II-PV is at least $3 \times$ faster than ELCBFR+ and $400 \times$ faster than PBRQ-Q. In summary, the performance of our SKSE-II and SKSE-II-PV outperform the ELCBFR+, except for the token generation time and token size (which is acceptable). The performance of PBRQ-Q is outperformed by all of our constructions.

In TABLE VI, we compare our constructions with existing constructions. The “Sub-linear Complexity” means the search

TABLE VI
COMPARISON WITH EXISTING CONSTRUCTIONS

Construction	CRSE [35]	GRSE [16]	FastGeo [17]	ELCBFR+ [2]	PBRQ-Q [6]	SKSE-I	SKSE-II
Sub-linear Complexity	✗	✓	✓	✓	✓	✗	✓
Performance	Very Low	Low	Low	Very High	Medium	High	Very High
Search Method	Circle	Geometric	Geometric	Geometric	BRQ	BRQ	BRQ
Dynamic Update	✗	✗	✓	✗	✗	✗	✗
Forward/backward Privacy	N/A	N/A	✗	N/A	N/A	N/A	N/A
Security Model	IND-SCPA	IND-SCPA	IND-SCPA	Broken	IND-SCPA	IND-SCPA	IND-SCPA

complexity is faster than linear, which means the construction is able to support search over large-scale datasets. The “Performance” is a qualitative comparison of search time on real-world datasets. The “Forward/backward Privacy” is a security property for dynamically updatable STE constructions. Note that only FastGeo supports dynamic updates but it cannot achieve forward/backward privacy, how to achieve dynamically updatable STE constructions for BRQ is still a challenging problem, we will discuss this problem in the next section. As shown in TABLE VI, to the best of our knowledge, our constructions achieve the best performance with strong security for BRQ.

VII. LIMITATION AND DISCUSSION

In this section, we clarify the limitations of our SKSE constructions, discuss the challenges to fix these limitations, and give some ideas to solve these limitations. We believe that the discussion in this section will be helpful for researchers who are interested in the problem of secure spatial keyword queries in future work.

A. Dynamic Update

In the above method SKSE constructions, we do not consider the dynamic update of outsourced *spatio-textual* database. The static setting is widely used in both plaintext and encrypted *spatio-textual* indexes [1], [2], [6]. Dynamic update, however, may be an important property for some frequently updated applications. There is a naive way to support updates over our encrypted outsourced constructions, that is, rebuilding the encrypted Bloom filters from the updated object to the root node. The rebuilt nodes including all parent nodes of the updated object but not all nodes of the entire database. However, such an approach incurs unignorable computation and communication costs. In addition, dynamic data updates also incur more information leakages than search. Specifically, data insertion reveals relations between newly inserted data and the previously searched data; data deletion reveals relations between newly deleted data and the previously inserted or searched data. Practical attacks in [19], [36], [37] showed that the above information learned in data update could be exploited to compromise the data privacy. Therefore, there are two main challenges to design a secure BRQ construction that supports dynamic update:

Challenge 1: How to minimize computation and communication overhead?

Challenge 2: How to minimize information leakages in update operations?

Here, we show that our spatial data and queries encoding method may be of independent interest to construct a secure BRQ construction that supports dynamic updates. Using our spatial data and queries encoding method as described in Section IV-A, any spatial range queries can be translated into multiple prefixes. Thereby, we can construct a secure dynamic BRQ by using two secure Dynamic Searchable Encryption (DSE) constructions. Specifically, we use one DSE construction (namely KDB) to store textual keyword indexes and another DSE construction (namely SDB) to store spatial prefix indexes. To update a *spatio-textual* object, we can update KDB and SDB according to the textual keywords and spatial prefixes, separately. As for search, a search user submits keyword queries and spatial queries to KDB and SDB, separately. Next, search user obtains keyword query results $\mathcal{R}_k$ from KDB and spatial query results $\mathcal{R}_s$ from SDB. Finally, the BRQ result is the intersection of $\mathcal{R}_k$ and $\mathcal{R}_s$. Such construction is efficient and secure if the chosen DSE constructions are efficient and secure to resist the above-mentioned leakage-based attacks. Certainly, such construction does not combine the space and text index tightly. A well-designed dynamic index can provide a better performance, but this is beyond the scope of this paper, and we will try to solve it in future work.

B. Hiding Query Pattern and Result Pattern

In our constructions, to minimize communication round-trip and computation cost, we leak the query pattern and result pattern. *Is there any way to hide the query and result patterns?* ORAM is an ideal cryptography primitive to address this problem. Here, we give a simple idea to support BRQ using ORAM-based searchable encryption schemes [38]–[40]. Our idea is similar to the dynamic update idea mentioned above. We can use two ORAM-based searchable encryption constructions to store prefix indexes and keyword indexes, separately. Then, BRQ results can be obtained by intersecting the search results of these two indexes. Similarly, such an approach does not combine the space and text index tightly, which does not achieve the best performance. We welcome researchers who are interested in this issue to explore and solve it.

VIII. RELATED WORK

Searchable Encryption (SE) is a typical STE construction for textual keyword first proposed by Song *et al.* [8]. Next,

Curtmola *et al.* [9] developed the security model of SE and designed the first inverted-index-based SE construction. Then, many SE schemes have been proposed to enrich functionalities [10], [41], improve performance [42], and enhance security [12], [43]. However, almost all of the above schemes only consider the keyword data rather than the spatial data.

Secure range queries on encrypted data has been well-studied in recent years. Li *et al.* [15] proposed a novel PBtree which can achieve index indistinguishability against chosen keyword attack. However, PBtree only supports one-dimensional range queries. Wang *et al.* [14] presented a novel R-tree index for encrypted half-space range queries by using the broken ASPE [4]. Afterwards, Demertzis *et al.* [44] translated range search into multi-keyword search, so that they can support range queries using any SE schemes. Unfortunately, their approaches introduced false positives. To achieve geometric range queries, several schemes [16], [17], [45] were proposed based on homomorphic encryption, pairing-based encryption, or ASPE. But either the performance or security of these schemes is limited.

Encrypted databases are another important application of STE. CryptDB [22] is an earlier attempt at encrypted databases use Property-Preserving Encryption (PPE) schemes such as deterministic encryption [24] and order-preserving encryption [25]. In [27], Grubbs *et al.* have shown that the PPE-based schemes have security vulnerabilities. To address this issue, Popa *et al.* [46] respectively proposed semantically secure SQL encrypted databases without using PPE. However, these schemes cannot hide metadata or access patterns during execution, if an attacker can observe such information from many queries, the attacker can reconstruct data [20], [21]. Moreover, all of the above encrypted databases are dedicated to solving the SQL query problem on encrypted data, and cannot deal with BRQ.

IX. CONCLUSION

In this paper, we presented several SKSE constructions to achieve various security-efficiency tradeoffs. Specially, we first presented a linear complexity SKSE construction with minimizing results pattern leakage. Next, we constructed a novel tree-structure index to achieve sub-linear search complexity. Finally, we proposed a post-validation approach to ensure query accuracy and further improve search and storage performance. We provided rigorous security proof and comprehensive performance evaluating to prove the security and efficiency of our constructions. In our future work, we will explore more secure and functionality spatial keyword STE constructions.

REFERENCES

- [1] L. Chen, G. Cong, C. S. Jensen, and D. Wu, "Spatial keyword query processing: An experimental evaluation," *Proc. VLDB Endowment*, vol. 6, no. 3, pp. 217–228, 2013.
- [2] 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.
- [3] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.
- [4] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis, "Secure KNN computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2009, pp. 139–152.
- [5] R. Li, A. X. Liu, Y. Liu, H. Xu, and H. Yuan, "Insecurity and hardness of nearest neighbor queries over encrypted data," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 1614–1617.
- [6] 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.
- [7] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur. (ASIACRYPT)*, 2010, pp. 577–594.
- [8] D. Xiaoding Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy. (SP)*, May 2000, pp. 44–55.
- [9] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *J. Comput. Secur.*, vol. 19, no. 5, pp. 895–934, 2011.
- [10] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for Boolean queries," in *Proc. Annu. Cryptol. Conf.*, 2013, pp. 353–373.
- [11] W. Xiangyu, J. Ma, M. Yinbin, X. Liu, and Y. Ruikang, "Privacy-preserving diverse keyword search and online pre-diagnosis in cloud computing," *IEEE Trans. Services Comput.*, early access, Dec. 16, 2019, doi: 10.1109/TSC.2019.2959775.
- [12] S. Lai *et al.*, "Result pattern hiding searchable encryption for conjunctive queries," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 745–762.
- [13] X. Wang, J. Ma, X. Liu, and Y. Miao, "Search in my way: Practical outsourced image retrieval framework supporting unshared key," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 2485–2493.
- [14] P. Wang and C. V. Ravishanker, "Secure and efficient range queries on outsourced databases using Rp-trees," in *Proc. IEEE 29th Int. Conf. Data Eng. (ICDE)*, Apr. 2013, pp. 314–325.
- [15] R. Li, A. X. Liu, A. L. Wang, and B. Bruhadeshwar, "Fast and scalable range query processing with strong privacy protection for cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2305–2318, Aug. 2016.
- [16] B. Wang, M. Li, and H. Wang, "Geometric range search on encrypted spatial data," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 704–719, Apr. 2016.
- [17] B. Wang, M. Li, and L. Xiong, "FastGeo: Efficient geometric range queries on encrypted spatial data," *IEEE Trans. Depend. Sec. Comput.*, vol. 16, no. 2, pp. 245–258, Mar. 2019.
- [18] X. Wang, J. Ma, X. Liu, Y. Miao, and D. Zhu, "Spatial dynamic searchable encryption with forward security," in *Proc. Int. Conf. Database Syst. Adv. Appl. (DASFAA)*. Springer, 2020, pp. 746–762.
- [19] Y. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," in *Proc. USENIX Secur. Symp. (USENIX Security)*, 2016, pp. 707–720.
- [20] P. Grubbs, M.-S. Lacharite, B. Minaud, and K. G. Paterson, "Learning to reconstruct: Statistical learning theory and encrypted database attacks," in *Proc. IEEE Symp. Secur. Privacy. (SP)*, May 2019, pp. 1067–1083.
- [21] Z. Gui, O. Johnson, and B. Warinschi, "Encrypted databases: New volume attacks against range queries," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 361–378.
- [22] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting confidentiality with encrypted query processing," in *Proc. 23rd ACM Symp. Operating Syst. Princ.*, 2011, pp. 85–100.
- [23] X. Yuan, Y. Guo, X. Wang, C. Wang, B. Li, and X. Jia, "EncKV: An encrypted key-value store with rich queries," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Apr. 2017, pp. 423–435.
- [24] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in *Proc. Annu. Int. Cryptol. Conf.*, 2007, pp. 535–552.
- [25] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order-preserving symmetric encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2009, pp. 224–241.
- [26] K. Lewi and D. J. Wu, "Order-revealing encryption: New constructions, applications, and lower bounds," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 1167–1178.
- [27] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 644–655.

- [28] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious RAMs," *J. ACM*, vol. 43, no. 3, pp. 431–473, 1996.
- [29] V. Iovino and G. Persiano, "Hidden-vector encryption with groups of prime order," in *Proc. Int. Conf. Pairing-Based Cryptogr.*, 2008, pp. 75–88.
- [30] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn. (EUROCRYPT)*, 1999, pp. 223–238.
- [31] H. Sagan, *Space-Filling Curves*. Springer, 2012.
- [32] A. X. Liu and F. Chen, "Privacy preserving collaborative enforcement of firewall policies in virtual private networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 887–895, May 2011.
- [33] D. Lemire, O. Kaser, and K. Aouiche, "Sorting improves word-aligned bitmap indexes," *Data Knowl. Eng.*, vol. 69, no. 1, pp. 3–28, Jan. 2010.
- [34] E. A. Markatou and R. Tamassia, "Full database reconstruction with access and search pattern leakage," in *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 395, 2019.
- [35] B. Wang, M. Li, H. Wang, and H. Li, "Circular range search on encrypted spatial data," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Sep. 2015, pp. 182–190.
- [36] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 668–679.
- [37] J. Ning, G. S. Poh, X. Huang, R. Deng, S. Cao, and E.-C. Chang, "Update recovery attacks on encrypted database within two updates using range queries leakage," *IEEE Trans. Depend. Sec. Comput.*, early access, Aug. 12, 2020, doi: [10.1109/TDSC.2020.3015997](https://doi.org/10.1109/TDSC.2020.3015997).
- [38] T. Hoang, A. A. Yavuz, F. B. Durak, and J. Guajardo, "Oblivious dynamic searchable encryption on distributed cloud systems," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy (DBSec)*, 2018, pp. 113–130.
- [39] S. Garg, P. Mohassel, and C. Papamanthou, "TWRAM: Efficient oblivious RAM in two rounds with applications to searchable encryption," in *Proc. CRYPTO*, vol. 9816, M. Robshaw and J. Katz, Eds. Springer, 2016, pp. 563–592.
- [40] T. Hoang, A. A. Yavuz, and J. Guajardo, "Practical and secure dynamic searchable encryption via oblivious access on distributed data structure," in *Proc. 32nd Annu. Conf. Comput. Secur. Appl. (ACSAC)*, Dec. 2016, pp. 302–313.
- [41] S. Faber, S. Jarecki, H. Krawczyk, Q. Nguyen, M. Rosu, and M. Steiner, "Rich queries on encrypted data: Beyond exact matches," in *Proc. Eur. Symp. Res. Comput. Secur. (ESORICS)*, 2015, pp. 123–145.
- [42] I. Demertzis, R. Talapatra, and C. Papamanthou, "Efficient searchable encryption through compression," *Proc. VLDB Endowment*, vol. 11, no. 11, pp. 1729–1741, Jul. 2018.
- [43] R. Bost, B. Minaud, and O. Ohrimenko, "Forward and backward private searchable encryption from constrained cryptographic primitives," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, 2017, pp. 1465–1482.
- [44] I. Demertzis, S. Papadopoulos, O. Papapetrou, A. Deligiannakis, and M. Garofalakis, "Practical private range search revisited," in *Proc. ACM Int. Conf. Manage. Data (SIGMOD)*, Jun. 2016, pp. 185–198.
- [45] H. Zhu, R. Lu, C. Huang, L. Chen, and H. Li, "An efficient privacy-preserving location-based services query scheme in outsourced cloud," *IEEE Trans. Veh. Technol.*, vol. 65, no. 9, pp. 7729–7739, Sep. 2016.
- [46] R. Poddar, T. Boelter, and R. A. Popa, "Arx: An encrypted database using semantically secure encryption," *Proc. VLDB Endowment*, vol. 12, no. 11, pp. 1664–1678, Jul. 2019.



Xiangyu Wang received the B.E. degree from the School of Cyber Engineering, Xidian University, Xi'an, Shannxi, China, in 2017. He is currently pursuing the Ph.D. degree with Xidian University. His research interests include data security and secure computation outsourcing.



Jianfeng Ma (Member, IEEE) received the B.S. degree in mathematics from Shaanxi Normal University, Xi'an, China, in 1985, and the M.S. and Ph.D. degrees in computer software and telecommunication engineering from Xidian University, Xi'an, in 1988 and 1995, respectively. He is currently a Professor with the School of Cyber Engineering, Xidian University. He is also the Director of the Shaanxi Key Laboratory of Network and System Security. His current research interests include information and network security and mobile computing systems.



Feng Li received the B.E. degree from the School of Cyberspace Security, Hangzhou Dianzi University, Hangzhou, China, in 2018. He is currently pursuing the Ph.D. degree with the School of Cyber Engineering, Xidian University. His current research interest includes cloud computing security.



Ximeng Liu (Member, IEEE) received the B.Sc. degree in electronic engineering and the Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2010 and 2015, respectively. He was a Research Fellow with the School of Information System, Singapore Management University, from 2016 to 2019. He is currently a Full Professor with the College of Mathematics and Computer Science, Fuzhou University, China. His research interests include cloud security, applied cryptography, and big data security.



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 Lecturer with the Department of Cyber Engineering, Xidian University. His research interests include information security and applied cryptography.



Robert H. Deng (Fellow, IEEE) is currently the AXA Chair Professor of cybersecurity and the Director of the Secure Mobile Centre, School of Information Systems, Singapore Management University (SMU). His research interests include data security and privacy, cloud security, and the Internet of Things security. He received the Outstanding University Researcher Award from the National University of Singapore, the Lee Kuan Yew Fellowship for Research Excellence from SMU, and the Asia-Pacific Information Security Leadership Achievements Community Service Star from the International Information Systems Security Certification Consortium. His professional contributions include an extensive list of positions in several industry and public services advisory boards, editorial boards, and conference committees. These include the editorial boards of *IEEE Security & Privacy Magazine*, *IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING*, *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY*, and *Journal of Computer Science and Technology*, and the Steering Committee Chair of the ACM Asia Conference on Computer and Communications Security.