Range Search over Encrypted Multi-Attribute Data

Francesca Falzon*
francesca_falzon@brown.edu
Brown University
University of Chicago

Evangelia Anna Markatou*
markatou@brown.edu
Brown University

Zachary Espiritu
zesp@brown.edu
Brown University

Roberto Tamassia
roberto@tamassia.net
Brown University

ABSTRACT

This work addresses expressive queries over encrypted data by presenting the first systematic study of multi-attribute range search on a symmetrically encrypted database outsourced to an honest-but-curious server. Prior work includes a thorough analysis of single-attribute range search schemes (e.g. Demertzis et al. 2016) and a proposed high-level approach for multi-attribute schemes (De Capitani di Vimercati et al. 2021). We first introduce a flexible framework for building secure range search schemes with an arbitrary number of attributes (dimensions) by adapting a broad class of geometric search data structures to operate on encrypted data. Our framework encompasses widely used data structures such as multi-dimensional range trees and quadtrees, and has strong security properties that we formally prove. We then develop six concrete highly parallelizable range search schemes within our framework that offer a sliding scale of efficiency and security tradeoffs to suit the needs of the application. We evaluate our schemes with a formal complexity and security analysis, a prototype implementation, and an experimental evaluation on real-world datasets.

KEYWORDS

Encrypted Database; Database Reconstruction; Attack

1 INTRODUCTION

With the rise of cloud services, there is a growing need for schemes that support complex privacy-preserving queries. In this paper, we study the security of schemes that support range-queries over multi-attribute data. We consider a two party setting in which a client outsources their data to a cloud provider. The server is untrusted and assumed to be a ***persistent, honest-but-curious*** adversary, and the client wishes to query their data *privately*. One solution is to use strong cryptographic primitives like fully-homomorphic encryption [22] or oblivious RAM [24]. While they offer the best security guarantees, these solutions are not yet practical. As an alternative, solutions for private range queries have been proposed using ***searchable symmetric encryption*** (SSE) (see, e.g., [7–11, 13, 23, 34–36, 51, 53, 59]). SSE schemes offer the following tradeoff: in exchange for efficiency they reveal some well-defined information, or *leakage*, about the queries and underlying data.

Existing efficient schemes support range queries on only single-attribute (1D) data or lack formal leakage analysis. In this paper, we present the security of a broad class of schemes that support range queries over multi-attribute data. We first give a general framework for building such schemes and give a security proof for all schemes that fit this framework. The schemes from prior work most related to those developed within our framework are by Demertzis et al. [14, 15], who present 1D range schemes with storage and security trade-offs, and by Faber et al. [18], who build on the SSE scheme in [10] to support 1D range, substring, wild-card, and phrase queries. We extend these schemes to support multi-attribute range queries, formalize the leakage of our schemes, and evaluate our schemes using real-world datasets. Our work is the first to systematically study schemes for encrypted range search on more than two attributes.

At a high level, our framework operates by decomposing the multi-dimensional domain $\mathcal{D}$ into subranges. Given a database over domain $\mathcal{D}$, each subrange can be mapped to the corresponding set of records with values in that subrange. The resulting map can then be encrypted using a standard SSE (e.g. [10]). The schemes within our framework are highly parallelizable and updates can be supported by batching updates as done in [14].

Our work systematizes the construction of schemes that support private range queries. For our concrete schemes, we opted for data-independent data structures – such as balanced range trees – since this prevents the adversary from inferring information about the data distribution from the underlying data structure. This is a consequence of the fact that leakage mitigation and scheme efficiency are fundamentally at odds with each other; that said, our framework encompasses a wide range of classical data structures including data-dependent data structures such as *kd*-trees.

In our work, we address a number of challenges from phenomena that arise only in multiple dimensions, also known as the *curse of dimensionality.* Classic range-supporting data structures in multiple dimensions are significantly more complex than their 1D variants. Extending such data structures and their range covering algorithms to better suit the encrypted setting requires new techniques.

Despite a large body of work on SSE-based range schemes spanning 7 years and dating back to 2015 [18], there are only a handful of papers on 2D schemes, and none in higher dimensions (see Section 1.1). We present the first concrete SSE range schemes in arbitrary dimensions, providing a number of efficiency and security tradeoffs. We give a thorough performance and leakage analysis of our schemes, thus aiding practitioners to make informed choices in the deployment of encrypted range search based on the application context and risk assessment. Our schemes are rooted in classic data structures that are well studied, optimized, efficient, extendable to other query types, and easily implementable.

1.1 Prior and Related Work

Schemes. *Order preserving encryption* (OPE) supports range queries by enabling order comparisons of the underlying plaintexts with out decrypting. However, OPE schemes have been shown vulnerable to several leakage abuse attacks [4, 17, 28, 50]. SSE leaks strictly less than *order preserving encryption*, and a number of range schemes using SSE have also been suggested (e.g., [14, 18, 31, 64]). SSE achieves

*Both authors contributed equally to this research. Preliminary eprint in [20].

its efficiency by using light-weight cryptographic primitives like *pseudorandom functions* (PRFs) and *hash functions*.

Several SSE schemes have been developed for single-attribute (1D) range queries on encrypted databases (see, e.g., [31, 33, 57, 64] and for a comparative evaluation of secure range schemes see [6]. The schemes most related to those developed within our framework are by Demertzis et al. [14, 15], who present a 1D range schemes with storage and security trade-offs, and by Faber et al. [18], who extend the SSE scheme in [10] to support range, substring, wildcard, and phrase queries. Both works build a binary tree on the domain, build an index based on this tree, and then use a black-box SSE scheme to encrypt the index. Bogatov et al. [5] present a differentially private solution that combines oblivious RAM and differentially private sanitizers to mitigate leakage. Wang and Chow [63] describe a framework for range-supporting schemes, however, they do not prove different properties of different range covering schemes as we do here. They then describe two forward-secure, dynamic schemes for 1D range queries. Falzon et al. [19] sketch out what we refer to as the quadratic scheme, however, this scheme is inefficient and does not scale well to higher dimensions.

De Capitani di Vimercati et al. [16] propose to index multi-dimensional symmetrically encrypted data by recursively partitioning records into boxes, thus taking steps toward a general scheme. However, their approach doesn't fall into the standard SSE definition [13]. Also, they do not provide a formal leakage analysis. Thus, we do not include [16] in our comparison with prior work. Additional prior work on multi-attribute range query schemes does not use symmetric encryption: Shi et al. [58] and Wang et al. [62] use public-key cryptography, whereas Kermanshahi et al. [39] use homomorphic encryption to support multi-attribute range queries however they leak significantly more than schemes built on SSE.

Leakage abuse attacks. Common types of SSE scheme leakage are access pattern (the adversary can identify the encrypted records in each response), volume pattern (the adversary can observe the number of encrypted records in each response), and search pattern (the adversary can determine if two issued queries are equal. Database reconstruction attacks have been presented against schemes supporting 1D range queries. The first such attack by Kellaris et al. [38] was followed by more efficient attacks for one-dimensional range queries using access (e.g. [27, 41, 44, 48]) and volume pattern (e.g. [26, 29, 42]). Two attack works on generic two-dimensional (2D) or two-attribute database reconstruction have also been presented [19, 47] using access and search pattern leakage.

1.2 Contributions

New challenges arise in higher dimensions due to increased structural complexity and addressing them requires fundamentally new techniques, including formalizing the leakage and developing data structures that can efficiently support range queries in the encrypted setting. One such problem we address is extending the standard quadtree to a novel tree data structure with a provable reduction in the number of false positives.

We introduce a ***general framework*** for building private range search schemes by adapting a broad class of geometric search data structures (e.g. range trees and quadtrees) to operate on encrypted data. Our schemes reduce a range query to a set of queries on an

Table 1: Comparison of selected non-interactive range search schemes from [14] and [18] with our schemes (bold). We assume that all schemes are instantiated using an EMM or SSE scheme that hides access pattern and leaks search and volume pattern. We show the asymptotic complexity of the schemes. Notation: range size R, result size r, database size n, domain size m, number of dimensions (query attributes) d. We evaluate the security of the schemes based on token co-occurrences (reduce security) and false positives (increase security). Notation for token co-occurrences: ● high; ◐ moderate; ○ none. Note that the query size for Quad-BRC is worst-case.

Scheme	Complexity				Security	
	Query Size	Resp. Size	Storage	Dim.	Token Co-occur.	False pos.
Range tree [18]	$\log R$	r	$m + n \log m$	1	◐	-
Quadratic [14]	1	r	$m + n^2$	1	○	-
Constant [14]	$\log R$	r	$m + n$	1	●	-
Log-URC [14]	$\log R$	r	$m + n \log m$	1	◐	-
Log-BRC [14]	$\log R$	r	$m + n \log m$	1	◐	-
Log-SRC [14]	1	$r + R$	$m + n \log m$	1	○	✓
Quadratic	1	r	$m + n^2$	d	○	-
Linear	R	r	$m + n$	d	●	-
Range-URC	$\log^d R$	r	$m + n \log^d m$	d	◐	-
Range-BRC	$\log^d R$	r	$m + n \log^d m$	d	◐	-
Quad-BRC	$m^{\frac{d-1}{d}}$	r	$m + n \log m$	d	◐	-
Range-SRC	1	$r + R$	$m + n \log^d m$	d	○	✓
Quad-SRC	1	$r + R^2$	$m + n \log m$	d	○	✓

underlying encrypted multimap. We provide a generic security proof for *any* scheme derived from our framework (Section 3).

We then present six concrete range search schemes that fit within our framework and support queries on an arbitrary number of dimensions (attributes). The schemes offer a sliding scale of efficiency security tradeoffs to suit the needs of the application (Table 1 and Section 4). One scheme reduces range queries to point queries. Three schemes are based on the range tree and build upon the 1D schemes by Demertzis et al. [14, 15] and by Faber et al. [18]. One is based based on the 2D quadtree [21], another fundamental spatial search data structure. The last scheme is a novel data structure that is also based on the quadtree, with modifications designed to reduce the number of search tokens needed at query time.

Our schemes employ ***range covering methods*** that reduce the given range query to a set of precomputed queries whose answers are stored in the multimap, extending to arbitrary dimensions the 1D single range cover (SRC), best range cover (BRC), and uniform range cover (URC) [40] methods presented in [14, 15].

We identify a form of leakage that we call *structure pattern*, i.e., the pattern of co-occurrence of subqueries. Structure pattern leakage is inherent in schemes derived from standard multidimensional search data structures (e.g., the 1D logarithmic scheme in [14], which is derived from the range tree). We evaluate our schemes with a theoretical complexity analysis, prototype implementation, and experimental evaluation on real-world datasets (Section 6).

We summarize our contributions as follows:

- We are the first to build a *formal general framework* for building SSE schemes that support range search on *multiple* attributes.
- We build a number of schemes that fit our framework, are based on *data-independent* data structures, and offer different *trade-offs* between bandwidth, storage and security. Our schemes are highly parallelizable and efficient in practice.
- We formally define the leakage of any scheme that fits our framework and analyze the relative security of our schemes with examples of what information an adversary can recover given scheme-specific leakage.
- We re-examine the relative security of the basic and uniform range covers, in the context of token co-occurrence.
- We support our results with a *prototype implementation*, and *experimental evaluation* on real-world databases.

Table 1 compares our concrete schemes for multi-attribute range queries on an encrypted database with selected prior work.

2 PRELIMINARIES

Given integers a, b with $a \leq b$, let $[a] = \{1, 2, \ldots, a\}$ and let $[a, b] = \{a, a+1, \ldots, b\}$. Let $m_1, \ldots, m_d$ be positive integers and $d \geq 2$. A ***d*-attribute database**, or a ***d*-dimensional database**, D is an injective mapping from a domain $\mathcal{D} = [m_1] \times \cdots \times [m_d]$ to a set of records of $O(1)$ size. We denote the set of records with domain value $x = (x_1, \ldots, x_d) \in \mathcal{D}$ as $D[x]$. A ***d*-dimensional range query** is a hyper-rectangle $[a_1, b_1] \times \cdots \times [a_d, b_d]$ where $[a_i, b_i] \subseteq [1, m_i]$ denotes the range in the i-th dimension.

PRFs. A ***pseudorandom function*** (PRF) family is a polynomial-time computable algorithm F that takes a key $K \in \{0,1\}^\lambda$ and an input $x \in \{0,1\}^*$ and returns $y \in \{0,1\}^k$ for some integer k. A PRF should be indistinguishable from random functions to any polynomial-time adversary with all but negligible probability.

2.1 Classic Range-Supporting Data Structures

In this work, we use and build upon two classic range-supporting data structures: the range tree and the quad tree.

Range Tree [3] A *range tree* G_{RT} on a d-dimensional domain is a recursively defined tree. Start with a binary search tree on $[m_1]$. Each node v in this tree is associated with a binary tree on $[m_2]$. More generally, there is an edge from each vertex of the binary trees on $[m_i]$ to the root of a binary tree on $[m_{i+1}]$. A ***dyadic range*** is an interval with a power-of-two length $\ell = 2^k$ and its start index is $1 \mod \ell$. A binary search tree on $[m]$ can thus be viewed as a tree whose nodes are each associated with a dyadic range in $[m]$. The source s of G_{RT} is such that $s.range = \mathcal{D}$. For a node v of a binary tree on $[m_i]$, let $v.dyadic$ denote the dyadic range in $[m_i]$ that v is associated with. Let $w.range = w_1 \times \cdots \times w_d$ be the canonical range of the root w of v's binary subtree on $[m_i]$. We have

$$\boxed{v.range = w_1 \times \ldots \times w_{i-1} \times v.dyadic \times [m_{i+1}] \times \ldots \times [m_d]} \quad (1)$$

See Figure 2 for an example of 2D range tree.

Region Quad Tree [21] The quadtree recursively subdivides the domain into 2^d quadrants or orthants. Unlike the range tree, each node of the quadtree has four children, each corresponding to one of the four quadrants covering the node's range. See Figure 3 for an example of a 2D quadtree.

2.2 EMM Definition and Security Model

EMM scheme syntax. Our range search schemes on encrypted data are built using an *encrypted multimap* (EMM) scheme in a generic manner. A *multimap* is a map that takes labels from a label space $\mathbb{L}$ to sets of values from a value space $\mathbb{V}$ i.e. $\mathsf{MM} : \mathbb{L} \mapsto 2^{\mathbb{V}} \cup \{\perp\}$ where $\perp$ indicates an uninitialized value. Given a multimap MM, we denote the set of values associated to label ℓ as $\mathsf{MM}[\ell]$.

Definition 1 ([11]). *An* ***encrypted multimap scheme*** *(EMM) is a tuple of algorithms* $\Sigma = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Eval}, \mathsf{Result})$, *where*

- $\Sigma.\mathsf{Setup}$*: (probabilistic) takes a security parameter and a multimap, and returns a secret key and an encrypted multimap.*
- $\Sigma.\mathsf{Query}$*: takes a key and label, and returns a search token.*
- $\Sigma.\mathsf{Eval}$*: takes an encrypted multimap and a search token, and returns a ciphertext.*
- $\Sigma.\mathsf{Result}$*: takes a key and a ciphertext, and returns a set of values.*

Our label space $\mathbb{L}$ is the set of possible ranges over the desired domain, and the value space $\mathbb{V}$ is the set of possible record values, i.e. $\{0,1\}^*$. We use the terms SSE and EMM interchangeably.

EMM security model. The security of EMMs is traditionally proven using the real-ideal paradigm [11]. The definition of adaptive security for an EMM scheme Σ is parameterized by a leakage function $\mathcal{L}^\Sigma = (\mathcal{L}_\mathsf{S}^\Sigma, \mathcal{L}_\mathsf{Q}^\Sigma)$ which describes the exact information that a passive adversary may learn about the underlying database. In particular, $\mathcal{L}_\mathsf{S}^\Sigma$ captures the leakage at setup and $\mathcal{L}_\mathsf{Q}^\Sigma$ captures the leakage when a sequence of queries is issued. Using this security framework, we refer to an adaptively $(\mathcal{L}_\mathsf{S}^\Sigma, \mathcal{L}_\mathsf{Q}^\Sigma)$-secure EMM scheme. The goal is to prove that the EMM scheme is indistinguishable from an ideal setting in which an algorithm simulates the response of the setup and query algorithms using only the leakage. Adaptive security of an EMM scheme is formally defined below.

Definition 2. *Let* $\Sigma = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Eval}, \mathsf{Result})$ *be an EMM scheme and let* $\mathcal{L}^\Sigma = (\mathcal{L}_\mathsf{S}^\Sigma, \mathcal{L}_\mathsf{Q}^\Sigma)$ *be a tuple of stateful algorithms. For distinct algorithms* $\mathcal{A}$, $\mathcal{S}$, *and* C *we describe two experiments below.*

$\mathbf{Real}_\mathcal{A}^\Sigma(1^\lambda)$

1. *The adversary* $\mathcal{A}$ *selects a multimap* MM *and gives it to the challenger* C.
2. *The challenger* C *runs the setup algorithm with* 1^λ *and* MM *as input,* $(K, \mathsf{EMM}) \leftarrow \Sigma.\mathsf{Setup}(\mathsf{MM})$. *The challenger* C *sends the encrypted multimap* EMM *to the adversary* $\mathcal{A}$.
3. $\mathcal{A}$ *adaptively chooses labels* $\ell_1, \ldots, \ell_{\mathsf{poly}(\lambda)}$*; for each label* ℓ_i *the adversary sees the token* $t_i \leftarrow \Sigma.\mathsf{Query}(K, \ell_i)$.
4. $\mathcal{A}$ *eventually outputs a bit* $b \in \{0,1\}$.

$\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^\Sigma(1^\lambda)$

1. *The adversary* $\mathcal{A}$ *selects a multimap* MM *and sends* MM *to challenger* C*;* C *sends* $\mathcal{L}_\mathsf{S}^\Sigma(\mathsf{MM})$ *to the simulator* $\mathcal{S}$.
2. *The simulator* $\mathcal{S}$ *generates an encrypted multimap* EMM *and gives it to the adversary* $\mathcal{A}$.
3. $\mathcal{A}$ *adaptively chooses labels* $\ell_1, \ldots, \ell_{\mathsf{poly}(\lambda)}$*; for each label* ℓ_i, C *gives* $\mathcal{L}_\mathsf{Q}^\Sigma(\mathsf{MM}, \ell_i)$ *to* $\mathcal{S}$, *and* $\mathcal{S}$ *outputs a token* t_i *to* $\mathcal{A}$.
4. $\mathcal{A}$ *eventually outputs a bit* $b \in \{0,1\}$.

Scheme Σ *is* ***adaptively*** $\mathcal{L}^\Sigma$***-secure*** *if for all polynomial-time adversaries* $\mathcal{A}$, *there exists a poly-time simulator* $\mathcal{S}$ *such that:*

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

FORMALIZING LEAKAGE EMMs are parameterized by different leakage functions, which output information about the underlying data structure and its contents. Below, we define two common leakage functions of EMM schemes relevant to this work. Let MM be a multimap with label space $\mathbb{L}$ and volume space $\mathbb{V}$.

- The ***search pattern*** reveals when two queries are equal. It takes as input a multimap MM and a label $\ell \in \mathbb{L}$, and outputs an ID. Without loss of generality we assume a 1-to-1 correspondence between range queries and identifiers: $\mathsf{SP}(\mathsf{MM}, \ell) \mapsto i \in [|\mathbb{L}|]$.
- The ***volume pattern*** of a label ℓ reveals the number of records in $\mathsf{MM}[\ell]$: $\mathsf{Vol}(\mathsf{MM}, \ell) = |\mathsf{MM}[\ell]|$.

In this paper we assume that the underlying EMM scheme is response-hiding. Attacks on multi-dimensional range queries have leveraged access and search pattern [19, 47], and we thus chose an underlying EMM scheme that does not leak access pattern.

2.3 REMM Definition and Security Model

DEFINITION 3. *A **range encrypted multimap scheme** is a tuple of four algorithms* $\mathsf{REMM} = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Eval}, \mathsf{Result})$. *The syntax of the algorithms is defined as those in Definition 1 with the following two changes:*

- REMM.Setup *takes as input a security parameter 1^λ and a multi-attribute database D and outputs a key K and an encrypted database* EMM.
- REMM.Query *takes as a input a key K and a range query q on the domain of D and outputs a token t.*

In order to support range queries, we take the label space of the underlying multimap to be the set of all possible range queries and the value space to be the set of all possible records.

For correctness we require that for all d-dimensional databases D with domain $\mathcal{D}$, all d-dimensional range queries q over $\mathcal{D}$, and all security parameters 1^λ, we have $\{D[x] : x \in q\} \subseteq V$, where $(K, \mathsf{EMM}) \leftarrow \mathsf{REMM.Setup}(1^\lambda, D)$, $t \leftarrow \mathsf{REMM.Query}(K, q)$, $C \leftarrow \mathsf{REMM.Eval}(\mathsf{EMM}, t)$, and $V \leftarrow \mathsf{REMM.Eval}(K, C)$.

REMM SECURITY MODEL. We extend the security model in [14] to encrypted multidimensional range schemes with games $\mathbf{Real}^{\mathsf{REMM}}_{\mathcal{A}}$ and $\mathbf{Ideal}^{\mathsf{REMM}}_{\mathcal{A},\mathcal{S}}$. They are identical to the game in Definition 2 except that in step (1) the adversary selects a multi-attribute database D on domain $\mathcal{D}$, in step (2) the adversary selects a polynomial number of range queries on the domain $\mathcal{D}$, and Σ is replaced by an encrypted multi-dimensional range scheme REMM. The adaptive security of REMM schemes is defined analogously to Definition 2.

3 GENERIC FRAMEWORK

We now present a framework for building range encrypted multimap schemes from data structures for multidimensional range search based on a search DAG. This framework generalizes many 1D range schemes and captures commonly used data-structures for range search such as range-trees, kd-trees, and quad-trees. These schemes provide a variety of trade-offs with respect to query size, response size, storage, and security. We introduce the family of range-supporting data structures and explain how to build an encrypted index from a data structure in this family. This framework allows us to characterize entire classes of schemes. Furthermore, we are able prove the properties of DAGs and range covers that are necessary to guarantee e.g. that the response has no false positives.

DEFINITION 4. *A **range-supporting data structure** for a database D with domain $\mathcal{D}$ is a pair* (G, RC), *where:*

(i) G is a connected directed acyclic graph (DAG).
*(ii) Each vertex v of G corresponds to a range on domain $\mathcal{D}$, which we denote as $v.range$ and refer to as a **canonical range**.*
(iii) G has a single source vertex s whose range is the entire domain, i.e., $s.range = \mathcal{D}$. For each non-sink (non-leaf) vertex v of G, we have $v.range = \bigcup_{(v,w)\in G} w.range$.
(iv) RC*, called **range covering algorithm**, is a polynomial-time algorithm that takes as input DAG G and a range query q on domain $\mathcal{D}$, and returns a subset W of vertices of G, called a **cover** of range q, such that the union of the canonical ranges of W includes range q, i.e., $q \subseteq \bigcup_{w\in W} w.range$.*

A range-supporting data structure can be used to perform range queries by precomputing and storing the responses to all the canonical ranges of the scheme. To perform a range query q, we use the range covering function to find a cover W of q, retrieve the responses to the canonical queries for the nodes of W, and return their union as the response to q.

The response to a range query q may have ***false positives***, i.e., points of the database outside of range q, which will have to be filtered out to obtain the exact response. To avoid false positives, we use a data structure where the cover W returned by the range covering function is such that the union of the canonical ranges of W is equal to range q, i.e., $q = \bigcup_{v\in W} v.range$.

The classic 1D range tree with the basic range cover (Algorithm 1) is an example of a range-supporting data structure without false positives. The theorem below gives a necessary condition for a range-supporting data structure to be without false positives.

THEOREM 1. *Let* (G, RC) *be a range-supporting data structure for a domain $\mathcal{D}$ such that the answer to any range query has no false positives. Then, for every domain point $x \in \mathcal{D}$, there is a node v of G with canonical range $v.range = x$.*

Note that, for any DAG, there is a trivial range cover that returns the source node of the DAG for every query, i.e., the trivial cover consisting of the entire domain. In general, this algorithm will cause false positives. To avoid false positives one needs to use a cover with multiple nodes. When the DAG is a tree, T, whose leaves are associated with the domain points (Theorem 1) and for each internal node v, the canonical ranges of the children of v are a partition of $v.range$, Algorithm 1, the ***best range cover*** (BRC), produces a cover of the query range with the minimum number of nodes. See Figure 2 for an example.

THEOREM 2. *Let* (T, BRC) *be a range-supporting data structure whose DAG is a tree T such that*

(1) the canonical range of the leaves (sinks) of T are in 1-1 correspondence with the domain points $x \in \mathcal{D}$;
(2) for each internal node v of T, the canonical ranges of the children of v are a non-trivial partition of the canonical range of v, i.e., $v.range = \bigcup_{(v,w)\in T} w.range$ and $\bigcap_{(v,w)\in T} w.range = \emptyset$.

Then for any range query q, the cover returned by BRC *has no false positives and is unique and minimal (i.e. smallest number of nodes).*

Algorithm 1: $\mathsf{BRC}(T, q, v)$

// Invoked with $\mathsf{BRC}(T, q, s)$, where s is the root (source) of T
Label v as explored
$W \leftarrow \emptyset$
if $v.range \subseteq q$ **then**
  $W \leftarrow \{v\}$
else
  if $v.range \cap q \neq \emptyset$ **then**
    for $(v, w) \in T$ and w is not labeled as explored **do**
      $W \leftarrow W \cup \mathsf{BRC}(T, q, w)$
return W

Algorithm 2: $\mathsf{SRC}(G, q, v)$

// Invoked with $\mathsf{SRC}(G, q, s)$, where s is the source of G
Label v as explored
$cand \leftarrow null$
if $q \subseteq v.range$ **then**
  $cand \leftarrow v$
  for $(v, w) \in G$ and w is not labeled as explored **do**
    $\{t\} \leftarrow \mathsf{SRC}(G, q, w)$
    if $|t.range| < |cand.range|$ **then**
      $cand \leftarrow t$
return $\{cand\}$

Corollary 1. *Let* (T, BRC) *be a range-supporting data structure whose DAG is a tree* T *such that only (2) holds. For any range query* q *that is the union of canonical ranges of leaves, the cover returned by* BRC *has no false positives and is the unique minimal cover.*

Observe that for queries of the same size, BRC can produce covers of different sizes (see Figure 2). Cover size may reveal some information about the location of the queried range. Kiayias et al. [40] introduce the notion of a ***uniform range cover*** (URC) to resolve this problem by making the size of the tokenset depend only on the size of the range, and not on its location in the domain. Let URC denote the uniform range cover algorithm for 1D ranges from [40]. It takes as input a 1D range tree G_{RT}, a range query q, and the source node of s, and returns the uniform range cover of q in G_{RT}. One way to compute URC on q is to compute the best range cover of q and then decompose the cover's nodes into their children until the desired number of nodes is reached.

When false positives are acceptable in the query answer, it may be desirable to have a cover consisting of a single node, which is accomplished by Algorithm 2, called ***single range cover*** (SRC).

Theorem 3. *Let* (G, SRC) *be a range-supporting data structure. For any range query* q*, the cover* v *returned by* SRC *minimizes the number of domain points of the cover outside of* q*, i.e. the number of potential false positives.*

The theorem can be proven by showing that at the end of every iteration, *cand* (Algorithm 2) is a cover of q that minimizes the number of false positives. The proof can be found in the full version.

Since many different domain-dependent data structures can be encoded as a DAG with the properties of Definition 4, we develop a generic scheme that supports range queries given any range-supporting data structure and which makes black box use of an SSE scheme. We give implementation details of the SSE scheme in Section 3.3 and describe our generic scheme below.

$\mathsf{GenericRS}(1^\lambda, D, \mathcal{D}, \Sigma, (G, \mathsf{RC}))$

$\mathsf{Setup}(1^\lambda, D)$:
  Initialize empty multimap MM.
  for node $w \in G$ **do**
    $\mathsf{MM}[w.range] \leftarrow \{D[x] : x \in w.range\}$
  $(K, \mathsf{EMM}) \leftarrow \Sigma.\mathsf{Setup}(1^\lambda, \mathsf{MM})$; **return** (K, EMM)
$\mathsf{Query}(K, q)$:
  $W \leftarrow \emptyset$; $\mathbf{t} \leftarrow \emptyset$
  $W \leftarrow \mathsf{RC}(G, q)$
  for $w \in W$ **do** $\mathbf{t} \leftarrow \mathbf{t} \cup \Sigma.\mathsf{Query}(K, w.range)$
  permute and **return t**
$\mathsf{Eval}(\mathbf{t}, \mathsf{EMM})$:
  $\mathbf{c} \leftarrow \emptyset$; **for** $t \in \mathbf{t}$ **do** $\mathbf{c} \leftarrow \mathbf{c} \cup \Sigma.\mathsf{Eval}(t, \mathsf{EMM})$
  return c
$\mathsf{Result}(K, \mathbf{c})$:
  $\mathbf{v} \leftarrow \emptyset$; **for** $c \in \mathbf{c}$ **do** $\mathbf{v} \leftarrow \mathbf{v} \cup \Sigma.\mathsf{Result}(K, c)$
  return v

Figure 1: Algorithm of the generic range encrypted multimap scheme (Definition 3) for a database D with domain $\mathcal{D}$ built from an encrypted multimap scheme Σ (Definition 1) and a range-supporting data structure (G, RC) (Definition 4).

Given a range-supporting data structure (G, RC) for a database D over domain $\mathcal{D}$, a range encrypted multimap scheme can be derived as follows. The client initializes a multimap MM and for every node v of G sets $\mathsf{MM}[v.range] \leftarrow \{D[x] : x \in v.range\}$. The resulting multimap is encrypted using the underlying EMM and outsourced to the server. To perform a range query q, the client computes $\mathsf{RC}(G, q, s)$ to obtain a cover W of q. Using the underlying EMM scheme, the client computes search token t_w for the canonical range $w.range$ of each node $w \in W$. This set of tokens $\mathbf{t}$ is permuted to remove any ordering information stemming from the range covering algorithm. The set $\mathbf{t}$ is then sent to the server, who then retrieves the corresponding encrypted sets of records (i.e., the encrypted responses for the canonical ranges) and returns them to the client. ***Generic range encrypted multimap scheme scheme*** $\mathsf{GenericRS}(1^\lambda, D, \mathcal{D}, \Sigma, (G, \mathsf{RC}))$ is formally described in Figure 1.

3.1 Security

Our generic range encrypted multimap scheme built from a range-supporting data structure leaks the size of the domain and the total size of the entries stored in the EMM. For each query, it leaks the tokenset and the sizes of the partial responses of each token. Thus, this scheme gives rise to an additional leakage resulting from the chosen DAG and range covering scheme. As one of the contributions of this work, we extend the notion of "partitioning" of IDs from Demertzis et al. [14], which refers specifically to the 1D range tree scheme with an underlying EMM scheme that leaks search and access pattern, and introduce ***structure pattern leakage***, a general and broadly applicable characterization of leakage from queries in arbitrary dimensions on a scheme instantiated with any range-supporting data structure (G, RC) and any EMM scheme.

Definition 5. *Let (G, RC) be a range-supporting data structure for a d-dimensional database D with domain $\mathcal{D}$, let* MM *be the multimap resulting from* GenericRS*, and let Σ be the encrypted multimap scheme. Let s be the source node of G. The* ***structure pattern*** *of a range query q is*

$$\boxed{\mathsf{Str}(D, q) \mapsto (\mathcal{L}_{\mathsf{Q}}^{\Sigma}(\mathsf{MM}, v.range))_{v \in \mathsf{RC}(G,q)}.}$$

The leakage of GenericRS is formally characterized in the following theorem.

Theorem 4. *Given an adaptively secure EMM scheme Σ that leaks search pattern and volume pattern, and a range-supporting data structure (G, RC) for a database D with domain $\mathcal{D}$ of size m, the generic range encrypted multimap scheme* GenericRS *built from Σ and (G, RC) is adaptively $(\mathcal{L}_{\mathsf{S}}, \mathcal{L}_{\mathsf{Q}})$-secure, where:*

$$\boxed{\begin{gathered}\mathcal{L}_{\mathsf{S}}(D, \mathcal{D}) = \mathcal{L}_{\mathsf{S}}^{\Sigma}(\mathsf{MM}) \\ \mathcal{L}_{\mathsf{Q}}(D, q^{(1)}, \ldots, q^{(t)}) = (\mathsf{Str}(D, q^{(i)}))_{i \in [t]}.\end{gathered}}$$

We prove this theorem using a standard hybrid argument. The goal is to find a sequence of games that starts in the real world and ends in the ideal world; there must be a polynomial number of such games in the sequence and they must all be indistinguishable from each other with all but negligible probability.

Proof. We construct a stateful simulator $\mathcal{S}$ for Setup and Query.

$\mathsf{EMM} \leftarrow \mathcal{S}.\mathsf{SimSetup}(1^{\lambda}, \mathcal{L}_{\mathsf{S}}^{\Sigma}(\mathsf{MM}))$

(1) Invoke the simulator of the underlying EMM scheme on $\mathcal{L}_{\mathsf{S}}^{\Sigma}(\mathsf{MM})$ to initialize an encrypted multimap EMM.
(2) Return EMM.

$\mathsf{Resp} \leftarrow \mathcal{S}.\mathsf{SimQuery}(1^{\lambda}, (\mathsf{Str}(Dcq^{(i)}))_{i \in [t]})$

(1) Initialize an empty set Resp.
(2) For each $\mathcal{L}_{\mathsf{Q}}^{\Sigma}(\mathsf{MM}, v.range)$ such that $v \in \mathsf{RC}(G, s, q^{(t)})$:
    (a) Simulator $\mathcal{S}$ uses the RC algorithm and invokes the simulator of the EMM scheme Σ on $\mathcal{L}_{\mathsf{Q}}^{\Sigma}(\mathsf{MM}, v.range)$, obtains the response for $v.range$ and adds it to Resp.

Note that the simulator uses RC and G to correctly simulate the structure pattern.

It remains to show that for all probabilistic poly-time adversaries $\mathcal{A}$, the probability $|\Pr[\mathbf{Real}_{\mathcal{A}}^{\mathsf{GenericRS}} = 1] - \Pr[\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{\mathsf{GenericRS}} = 1]|$ is negligibly small. We define the following two games and conclude with a hybrid argument.

Hyb0: This is identical to $\mathbf{Real}_{\mathcal{A}}^{\mathsf{GenericRS}}$.
Hyb1: This is identical to **Hyb0**, except that instead of invoking Σ.Setup and Σ.Query we invoke the simulator of the underlying EMM scheme.

$|\Pr[\mathbf{Hyb0}] - \Pr[\mathbf{Hyb1}]|$ is negligibly small, otherwise the security of the underlying EMM scheme would be broken with non-negligible probability. Since the distribution of **Hyb1** is identical to $\mathbf{Ideal}_{\mathcal{A}}^{\mathsf{GenericRS}}$ this concludes our proof. □

Note that the leakage of the generic scheme is heavily dependent on the DAG and range covering algorithm used.

3.2 Choosing the Data Structure

One important choice we made is using ***data-independent data structures***. This stands in contrast to the long line of work that has developed more efficient data structures for *plaintext databases*, such as R-trees (e.g. [1, 2, 30, 37, 56]) and learned indexes (e.g. [43, 46, 55]). That said, the leakage profile of the generic scheme (GenericRS) is highly dependent on the underlying DAG and range covering algorithm. Using data-dependent data structures would inherently leak more information. For example, consider a 1D database D that can be more efficiently encrypted using a range-supporting data structure whose DAG is an unbalanced binary tree that decomposes each canonical range $[i, j]$ into subranges $[i, (j - i + 1)/2]$ and $[(j - i + 1)/2 + 1, j]$. If the adversary were able to infer the tree structure, then it would additionally learn something about the data distribution of the underlying records. Many private 1D range schemes opt for balanced range trees for this reason [14, 15, 18]. We leave open the question of reconciling the development of data-dependent data structures and leakage minimization.

3.3 Implementing an EMM

We instantiate our schemes with an EMM scheme. In our implementation, we use Π_{bas} from [9]. To query a label with Π_{bas}, the client computes two per-label keys $K_1 \| K_2$ which are sent to the server as the search token. For each $i \in [n]$ in increasing order, the server computes the PRF value $F(K_1, i)$ and uses it to retrieve the corresponding encrypted value. The server then uses $F(K_2, i)$ to decrypt the value which it then returns to the client. It increments i and repeats the look up until $\perp$ is returned. We can turn this scheme intro a response-hiding one with the following, simple modification: we instead use a separate key to encrypt the values. This key is kept private and the values are decrypted client-side.

Now, we can use the specific range-supporting data structure to create a plaintext multimap, which maps each canonical range to the records inside that range. This plaintext multimap is encrypted using an EMM scheme. Whenever the client issues a query, they use an appropriate range covering algorithm to determine which plaintext canonical ranges they need to query. Then, they can use the EMM scheme to compute search tokens and decrypt the result.

4 SCHEMES

We now give concrete examples of schemes that fit the general framework; these schemes include generalizations of schemes that were presented by Demertzis et al. [14, 15] and Faber et al. [18]. We present six schemes for range search on encrypted databases. All the schemes are instances of range encrypted multimap schemes based on range-supporting data structures presented in Section 3. The schemes support multidimensional range searches on an arbitrary fixed number of dimensions (attributes), d. The first scheme we present is the linear scheme, which provides optimal server storage at the expense of client bandwidth. In order to reduce the required bandwidth, we turn to classic data structures for geometric searching, specifically the range tree and the quadtree. Both data structures are an improvement upon the linear scheme's bandwidth, and offer trade-offs for bandwidth and storage. In order to query our data structures, we have developed the following multi-dimensional range covering techniques, extending 1D covers in novel ways: (i)

Best Range Cover (BRC), which optimizes both required query bandwidth and false positives; (ii) Uniform Range Cover (URC), which in the 1D case, Demertzis et al. [14, 15] suggest is safer than BRC at a small bandwidth expense; and (iii) Single Range Cover (SRC), which is the most secure. Finally, we develop variations of our data structures to minimize the false positive rate when a SRC is used.

For completeness, we also include the multi-dimensional ***quadratic scheme*** in Table 1. This scheme comprises of a DAG whose nodes are in 1-1 correspondence with the set of all possible range queries, together with either BRC or SRC. While this scheme offers the best possible security, it also requires quadratic storage and hence we omit it from our formal analysis.

Following the notation used throughout the paper, we denote the database with D and the domain with $\mathcal{D}$. We denote their sizes as $n = |D|$ and $m = |\mathcal{D}|$. The number of domain points in a query range is referred to as ***range size*** and denoted with R. The number database of records within a query range is referred to as ***result size*** and denoted with r. In our schemes, a query is issued by the client to the server as a tokenset. We refer to the number of search tokens in the tokenset as the ***query size***. A response is returned by the server to the client as a collection of encrypted sets of records (one set per search token), whose total number of records is referred to as ***response size***. Note that the response size is equal to the result size plus the number of false positives returned.

4.1 Linear Scheme

We first present a simple scheme, called ***linear scheme***, that offers the smallest storage at the expense of the least security.

The linear scheme indexes each record by its domain value. Intuitively, every time the client wishes to query a range r, they send a search token for every domain point of r to the server. The linear scheme's DAG, G_L, is a star comprising a source s connected to m sinks. We have that $s.range = \mathcal{D}$ and each sink v is associated with a distinct domain point $x \in \mathcal{D}$ such that $v.range = x$. For this scheme, the generic BRC algorithm takes $O(m)$ time. To increase efficiency, we use the ***linear range covering algorithm*** (LRC), where in a preprocessing step, the sinks of G_L are stored in a d-dimensional array, $V[\mathcal{D}]$, indexed by its domain point. We summarize the scheme and its complexity with the following theorem.

Theorem 5. *Let G_L be the star DAG for a database D of size n on a d-dimensional domain $\mathcal{D}$ of size m, and let* LRC *be the linear covering algorithm defined above. Then (G_L, LRC) is a range-supporting data structure (Definition 4) and the range encrypted multimap scheme derived from it (Definition 5) uses space $O(n + m)$. Also, a query with range size R and result size r has query size R and response size r.*

Proof. It is straightforward to verify properties (i) to (iv). It remains to show that algorithm LRC runs in polynomial-time when invoked on the source of G_L. LRC is implemented such that the sinks of G_L are stored in an array, $V[\mathcal{D}]$, indexed by domain point. Let q be any range query on the domain. Initializing an empty set W can be done in constant time. Looping through $x \in q$ and adding $V[x]$ to W takes time $O(R)$, where R is the size of the range q.

The linear scheme generates a multimap with m labels, one for each sink in G_L. Each record is stored once with its corresponding point value. The index has size $n+m$. When the client issues a range query of size R, the client computes R search tokens and sends them to the server. Each search corresponds to the domain points of the query and thus the response has size $O(r)$. □

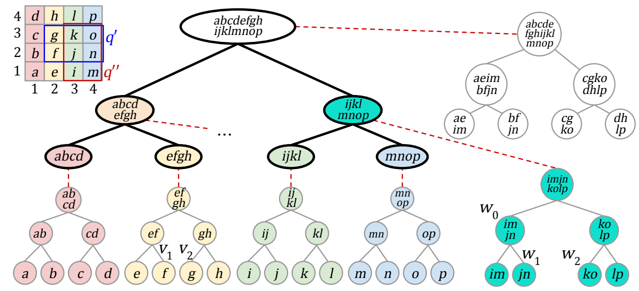


Figure 2: Example of a range tree for a database over a $[4] \times [4]$ domain. The binary tree over the first dimension is shown with black thick lines and the binary trees over the second dimension are shown with gray thin lines. Using BRC_{RT}, query range $q' = [2,4] \times [2,3]$ (in blue) corresponds to cover $\{v_1, v_2, w_1, w_2\}$ and query range $q'' = [3,4] \times [2,3]$ (in red) corresponds to cover $\{w_0, w_2\}$. The queries have the same size, but correspond to best range covers of different sizes.

The proofs for the analogous theorems of the other schemes use similar techniques and we thus defer their proofs to the full version.

4.2 Range-BRC and Range-URC Schemes

In an effort to decrease the bandwidth of the linear scheme, we present the ***Range-BRC*** and ***Range-URC schemes*** based on the range tree [3].

The multi-dimensional range tree for $d > 1$ does not satisfy the properties of the tree in Theorem 2. The multi-dimensional range tree can be viewed as being composed of subtrees that subdivide the domain along different dimensions; these subtrees do satisfy the properties of Theorem 2. For example, note that the three children of the root in Figure 2 correspond to the ranges $\{abcdefgh\}$, $\{ijklmnop\}$, $\{abcdefghijklmnop\}$ and thus do not correspond to a partition of the root's canonical range. However, the children outlined in bold correspond to the ranges $\{abcdefgh\}$ and $\{ijklmnop\}$, and belong to the same subtree (which divides the domain along the first dimension). Hence they form a partition of the root's canonical range.

We thus design a ***best range cover*** for multi-dimensional range trees, Algorithm 3 (BRC_{RT}), that calls BRC as a subroutine on these subtrees. In particular, as we traverse the subtrees starting from the root, we compute BRC of the range along the corresponding dimension; this cover corresponds to a collection of subtrees rooted at each vertex in the cover. We then recurse and apply BRC to each of these subtrees, until we reach the final set of subtrees that partition the range along the n-th dimension. For example, in Figure 2, we have query $q'' = [3,4] \times [2,3]$. First, we find the best range cover in the bold tree, which corresponds to the range along the first dimension [3,4], which is node $\{ijklmnop\}$. Then, we find the best range cover in the subtree branching off of $\{ijklmnop\}$, which corresponds to the range along the second dimension [2,4], which is nodes w_0 and w_2.

Theorem 6. *Let G_{RT} be a multi-dimensional range tree. For any range query q, the cover returned by BRC_{RT} has no false positives and is the unique minimal cover.*

Algorithm 3: BRC_{RT} URC_{RT} (T, q, v)

1: // Invoked with input (T, q, s), where s is the source of T
2: $W \leftarrow \{v\}$
3: $q_1 \times \cdots \times q_d \leftarrow q$
4: **for** $i \in [d]$ **do**
5: $\quad W' \leftarrow \emptyset$
6: $\quad$ **for** $w \in W$ **do**
7: $\quad\quad w_1 \times \cdots \times w_d \leftarrow w.range$
8: $\quad\quad \hat{q} \leftarrow w_1 \times \cdots \times w_{i-1} \times q_i \times [m_{i+1}] \times \cdots \times [m_d]$
9: $\quad\quad$ Let $\widehat{T} \subseteq T$ be the subtree on $[m_i]$ rooted at w.
10: $\quad\quad W' \leftarrow W' \cup \mathsf{BRC}(\widehat{T}, \hat{q}, w)$ $\quad W' \leftarrow W' \cup \mathsf{URC}(\widehat{T}, \hat{q}, w)$
11: $\quad W \leftarrow W'$
12: **return** W

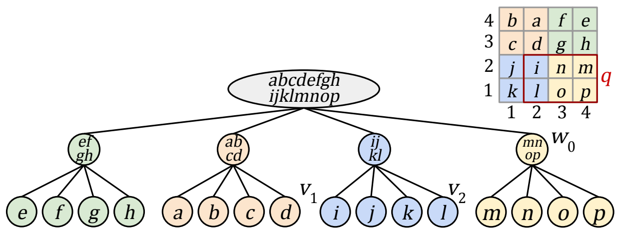


Figure 3: A quadtree for a database on domain $[1,4] \times [1,4]$**. With** BRC**, query range** $q = [2,4] \times [1,2]$ **(in red) corresponds to the cover** $\{v_1, v_2, w_0\}$**.**

We can prove the statement inductively by showing that at the end of the i-th iteration, we have the best range cover of the first i dimensions of the queried range. The proof follows from standard methods in geometric search and can be found in the full version.

We extend the 1D URC algorithm to higher dimensions. Our uniform range cover algorithm for multi-dimensional range trees, denoted URC_{RT}, is identical to BRC_{RT} except that on line 10, it calls URC as a subroutine instead of BRC (See Algorithm 3 for psuedocode).

Theorem 7. *Let G_{RT} be a range tree on $[m_1] \times \cdots \times [m_d]$ and σ be any permutation on $[d]$. If q is a range query of size $R = R_1 \times \cdots \times R_d$ and q' is a range query of size $R = R_{\sigma(1)} \times \cdots \times R_{\sigma(d)}$, then their respective URC_{RT} covers W and W' are such that $|W| = |W'|$.*

We recall that, given a 1D range tree G_{RT} on domain $[m]$, and range queries q and q' of the same size, then URC returns covers of the same size for q and q' [40]. We prove Theorem 7 by extending the 1D version to arbitrary dimensions. The proof can be found in the full version. We now state and prove a theorem summarizing the complexity of the range tree scheme.

Theorem 8. *Let G_{RT} be the range tree for a database D of size n on a d-dimensional domain $\mathcal{D}$ of size m, and let BRC_{RT} and URC_{RT} be the range covering algorithms defined in Section 4.2. We have that $(G_{RT}, \mathsf{BRC}_{RT})$ and $(G_{RT}, \mathsf{URC}_{RT})$ are range-supporting data structures (Definition 4) and the range encrypted multimap schemes derived from them (Definition 5) use space $O(n + m \log^d m)$. Also, a query with range size R and result size r has query size $O(\log^d R)$ and response size r.*

4.3 Quad-BRC Scheme

We base our ***Quad-BRC scheme*** on the ***region quadtree***. The client computes a region quadtree $G_{QT} = (V, E)$ on domain $\mathcal{D}$. G_{QT} is a 2^d-ary tree with one source vertex $s \in V$ such that $s.range = \mathcal{D}$. Each node $v \in V$ is associated with a d-dimensional hypercube $w.range$ and the canonical range of each child of v corresponds to one of 2^d equal-sized partitions of $v.range$. G_{QT} will have m leaves, which are in 1-1 correspondence with the domain values (see Figure 3).

To carry out a range query q on the database the client computes the best range cover $\mathsf{BRC}(G_{QT}, q, s)$. We now state and prove a theorem summarizing the complexity of the quadtree scheme.

Theorem 9. *Let G_{QT} be the quadtree built over d-dimensional domain $\mathcal{D}$ of size m and let BRC be the range covering algorithm defined in Algorithm 1. We have that (G_{QT}, BRC) is a range-supporting data structure (Definition 4). Also, to store a database of size n on domain $\mathcal{D}$, the range encrypted multimap scheme built from (G_{QT}, BRC) according to Theorem 4 uses space $O(m + n \log m)$ and has query token size $m^{\frac{d-1}{d}}$ and ciphertext response size r, where r is the size of the plaintext query response.*

4.4 Range-SRC Scheme

One downside of the previous schemes is that the client must often generate multiple search tokens. To overcome this, we consider *single range covers* that cover the range with only one node at the expense of false positives. Naively covering the range can result in a worst case false positive of $O(m)$ in both the range tree and quadtree constructions. Demertzis et al. propose injecting a small number of additional nodes into the range tree to create a ***tree-like DAG*** (TDAG) which reduces the false positive rate to $O(R)$.

To build a TDAG in one dimension, build a range tree over the domain $[m]$ and inject one extra node between every two nodes at every level of the tree. Then add edges from this node to the two nodes below it in the next level and add an edge from the node directly above in the previous level. Given a TDAG over domain $[m]$ and a range query $q \subseteq [m]$ of size R, there is a set of nodes that cover q whose canonical ranges sum to size $O(R)$ [14].

We can extend this to multiple dimensions in the following manner. Let $\mathcal{D} = [m_1] \times \cdots \times [m_d]$ be the domain. Build a range tree over $\mathcal{D}$, then for each subtree on $[m_i]$ for $i \in [d]$, inject the nodes as described before (See Figure 4a). To issue a range query, the client uses Algorithm 2 (SRC).

Corollary 2. *Given d TDAGs constructed over the domain $\mathcal{D} = [m_1] \times \cdots \times [m_d]$ and any range query $q \subseteq \mathcal{D}$ of size R, there is a set of nodes that cover q and whose canonical ranges sum to size $O(R)$.*

A TDAG is only a constant factor larger than the original range trees and hence the storage requires $O(m+n \log^d m)$ space. To query this scheme, the client generates a single search token, yielding a query complexity of 1. By Corollary 2, the total number of false positives for any given query is $O(R)$ where R is the size of the query issued. We state and prove the following theorem summarizing the complexity of the ***Range-SRC scheme***.

Theorem 10. *Let G_{RS} be the TDAG built over d-dimensional domain $\mathcal{D}$ of size m and let SRC be the single range covering algorithm defined in Algorithm 2. We have that (G_{RS}, SRC) is a range-supporting data structure (Definition 4). Also, to store a database of size n on domain $\mathcal{D}$, the range encrypted multimap scheme built from (G_{RS}, SRC) according to Theorem 4 uses space $O(m + n \log^d m)$ and has query token size 1 and ciphertext response size $r + R$, where r is the size of the plaintext query response and R is the size of the queried range.*

4.5 Quad-SRC Scheme

We present the ***Quad-SRC scheme***, which is derived from the quadtree, supports single range covers at the expense of $O(R^d)$ false positives, and also extends the one-dimensional TDAG scheme [14] to higher dimensions. The response size overhead is an inherent limitation of schemes that index the domain using only hypercubes.

Given a multi-dimensional database D with domain $\mathcal{D}$ we build the data structure the bottom up, starting with m leaves corresponding to points of domain $\mathcal{D}$. At level j we add nodes for all hypercubes of size 2^{n-j} tiling the domain, as well as each of these hypercubes shifted by 2^{n-j-1} along each dimension. For each node at level j, we add directed edges to all nodes in level $j-1$ which it covers. We recursively build the structure until we reach the (source) root node that corresponds to the entire domain. Each node v in this DAG is associated with a d-dimensional hypercube. To execute a range query, the client computes its SRC cover.

The Quad-SRC scheme is illustrated in Figure 4b. Below we prove that the false positive rate is $O(R^d)$ and the QDAG size is $O(m)$.

Lemma 1. *Given QDAG $G_{QS} = (V, E)$ over domain $\mathcal{D} = [m_1] \times \cdots \times [m_d]$ and any range q in $\mathcal{D}$ of size $R = R_1 \times \cdots \times R_d$, there exists a vertex $v \in V$ such that $q \subseteq v.range$ and $v.range$ has size $O(R^d)$.*

Proof. Recall that the number of nodes in a quadtree scheme is $O(m)$. In the quadtree, at the j-th level, each of the d dimensions is partitioned into 2^{n-j} axis-aligned segments. Thus, at the j-th level, we have partitioned the domain into $2^{(n-j)d}$ hypercubes.

In the QDAG, we shift each hypercube by the legnth of half of the edge of the hypercube along each dimension. Thus at the j-th level of the QDAG we have at most $2^d 2^{(n-j)d} = O(2^{(n-j)d})$ hypercubes. Each hypercube corresponds to a node in the QDAG, so size of the QDAG is upper bounded by a constant factor of 2^d times the size of the region quadtree. □

Lemma 2. *Let D be a database with n records and a domain $\mathcal{D}$ of size m. Then the size of the QDAG on $\mathcal{D}$ is $O(m)$.*

Proof. Let q be any range query of size R. We will show that this range can be covered by a vertex $v \in V$ such that $v.range$ has size $O(R^d)$. First note, there exists some minimal integer j such that for all i, $R_i \leq 2^j \leq 2R_i$.

Case 1: range query q is covered by a vertex $v \in V$ such that $v.range$ has size 2^{jd}. Thus, q is covered by a range of size $O(R^d)$.

Case 2: range query q is *not* covered by a vertex $v \in V$ such that $v.range$ has size 2^{jd}. Along each dimension, the range q must intersect with at most 2 distinct ranges $v.range, v'.range$ each of size 2^{jd} where $v, v' \in V$. Since ranges of the same size are shifted by lengths of 2^{j-1} there must exist a hypercube in $\mathcal{D}$ with edges of length $2^j + 2^{j-1}$ that completely contains q. Note this hypercube does not correspond to a vertex of G_{QS}.

Let $q' = [a_1, b_1] \times \cdots \times [a_d, b_d]$ define this hypercube. Consider an edge $[a_i, b_i]$ of this hypercube. For each $i \in [d]$ there is a set of vertices $V_i \subset V$ such that for all $v \in V_i$, $v.range$ is of size $2^{(j+1)d}$ and it covers $[a_i, b_i]$. In the i-th dimension $v.range$ must either start at a_i or at $a_i - 2^j$. Since the hypercubes of size $2^{(j+1)d}$ are tiling the entire domain with shifts of 2^j along each dimension we can thus find a vertex $v^* \in T$ that contains q' and thus also contains q. The range $v^*.range$ has size $2^{(j+1)d} = O(2^{jd}) = O(R^d)$. □

We now state and prove a theorem summarizing the complexity of the Quad-SRC scheme. The proof is similar to that of Theorem 10 and has thus been omitted.

Theorem 11. *Let G_{QS} be the QDAG for a database D of size n on a d-dimensional domain $\mathcal{D}$ of size m, and let SRC be the linear covering algorithm defined in Algorithm 2. We have that (G_{QS}, SRC) is a range-supporting data structure (Definition 4) and the range encrypted multimap scheme derived from it (Definition 5) uses space $O(m + n \log m)$. Also, a query with range size R and result size r has query size 1 and response size $O(r + R^2)$.*

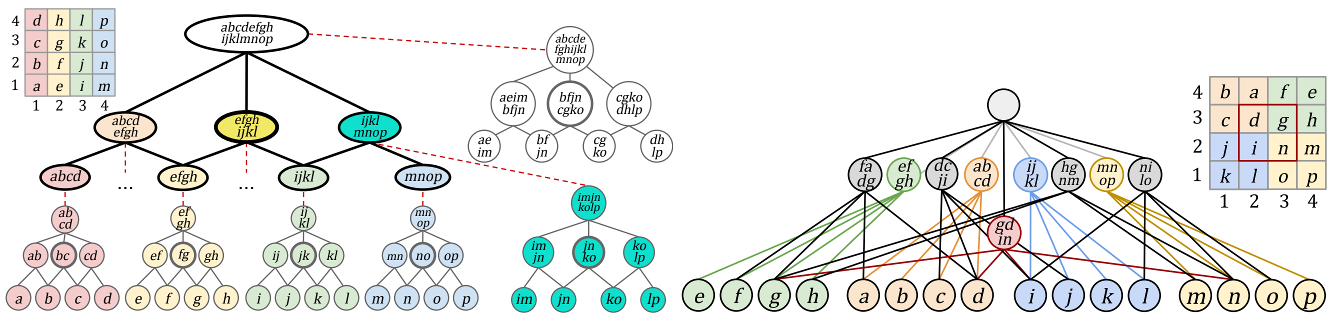
5 PERFORMANCE AND SECURITY

5.1 Performance

Parallelization. Our schemes are highly parallelizable both at setup and at query time. Recall, that our framework decomposes the domain into subranges and then constructs a mapping from each subrange to its corresponding records. This map can be encrypted using an EMM scheme that supports parallel setup by partitioning the label-value pairs and encrypting each batch on a different core. Search can also be parallelized in two distinct ways. Given a collection of search tokens, we can process the search for each token using a different core. Furthermore, since each search token may correspond to multiple records, the search for each individual search token can also be parallelized, e.g. by selecting an SSE scheme that supports concurrent access requests [9, 35].

Complexity. Table 1 compares the schemes in this section. The Range-SRC and Quad-SRC schemes have optimal query size but allow false positives. The other schemes avoid false positives but incur query size overhead. Notably, all the schemes have the same asymptotic storage and search time as their corresponding non-encrypted data structures. To achieve efficient client query time, the range cover algorithm builds the tokenset without instantiating the scheme's DAG, which is implicitly defined by the parameters of the domain. Thus, we can assign IDs to the nodes of the DAG so that the ID of each node in the cover is computed in $O(1)$ amortized time and space. Hence, the query time and space at the client is proportional to the query size (to generate the tokenset) plus the response size (to decrypt the received response). The query execution time at the server is also proportional to the query size plus the response size, since accessing the partial response associated with a token takes $O(1)$ expected time with an efficient multimap implementation. The client space is $O(1)$, plus $O(\log^d m)$ temporary space for the Range-BRC/URC schemes and $O(m^{\frac{d-1}{d}})$ temporary space for the Quad-BRC scheme when a query is issued, and temporary space proportional to the response size when the response is received.

Extensions. Our work specifically addresses range queries, which can be viewed as conjunctions of predicates, each associated with a 1D range query. However, our techniques support a broader class of queries involving a combination of conjunctions, disjunctions, and negations of 1D queries over different subsets of attributes. For example, the client can represent the disjunction of two 1D range queries (or the negation of a 2D range query) as the disjunction of four 2D range queries, which can be executed by taking the union of the responses to four 2D range queries. The client can use our query algorithm to find the search tokens for such queries.



(a) Illustration of a 2D TDAG.

(b) Illustration of a 2D QDAG

Figure 4: Examples of (a) a TDAG and (b) a QDAG for a $[4] \times [4]$ database. The inserted nodes are dark gray and the inserted edges are black.

Using a classic result from Boolean algebra, an arbitrary query on a d-dimensional domain involving conjunctions, disjunctions, and negations of 1D ranges can be reduced to disjunctive normal form and performed by taking the union of the responses to a collection range queries on d attributes, each computed with our methods. This collection can be identified by the client without accessing the server. An interesting line of future work is to find an optimal query collection that minimizes the overall number of search tokens, a problem related to spatial query optimization (see, e.g., [32]). Prior work has explored SSE schemes with support for Boolean queries [10, 52]. Building such an index over each dimension to support Boolean queries over multi-attributes would result in a different leakage profile compared to that of our approach.

5.2 Security

The following theorem summarizes the leakage of our schemes.

Theorem 12. *Let Σ be an EMM scheme that leaks search and volume pattern. The range encrypted multimap schemes linear, Range-*BRC/URC*, Quad-*BRC*, Range-*SRC*, and Quad-*SRC *instantiated with Σ leak each the search, volume and structure pattern.*

We now discuss the relative security of our schemes. We specifically focus on the co-occurrence of search tokens of each scheme, since co-occurrence stemming from access pattern has been leveraged in prior attacks [19, 47]. In Table 1, we depict the number of token co-occurrences associated with our schemes and those of prior work. In this section, we assume that all schemes are instantiated using an EMM scheme that hides access pattern and leaks search pattern and volume pattern. We consider a passive persistent adversary, e.g. an eavesdropper that has compromised the communication channel. Below, we extend previous approaches from attacks in 1D and 2D that leverage co-occurrences from the responses; these insights inform the security ranking of our schemes

Linear Scheme. When a client issues a query of size R, the adversary observes R distinct search tokens; each associated with a domain point of the queried range. The structure pattern leaks the exact size of each queried range *and* the number of matching records. In contrast, given a scheme that leaks only access pattern, the adversary would be able to infer the number of records matching the query, but not the size of the query. Over multiple queries, the access pattern leaks the co-occurrence of encrypted records that match each query. The structure pattern also leaks co-occurrence information, as over multiple query response pairs, the same search token for a specific domain value can be observed. The search tokens in this scheme are 1-1 with the domain points. Suppose we have a 2D database; If we see a response r corresponding to four search tokens, then the queried range must be of size 4. Moreover, r can either correspond to a range of size 2×2, 4×1, or 1×4. If two ranges of size 4 have three overlapping search tokens, then both ranges must be of dimension 4×1 (or 1×4), because a 2×2 range cannot intersect any size 4 range in 3 domain points.

Range-BRC/URC Scheme. Each time the client makes a query of size R, the adversary observes $O(\log^d R)$ distinct search tokens, each token corresponding to a canonical range. The relationship between these tokens is determined by the range covering technique used. For any given range query, URC requires at least as many search tokens as BRC. For example, a range of size 2×1 under URC always results in two search tokens. Under BRC, the range may result in either one or two search tokens. This choice reveals some information about the location of the range. Specifically, if only one search token is issued, the requested range must be a canonical range. Furthermore, canonical ranges corresponding to domain points in the inner part of the domain correspond to more queries than canonical ranges at the perimeter of the domain. A similar observation was leveraged in the attack of [19]. We conjecture that similar techniques may be applied to URC/BRC schemes.

Quad-BRC Scheme. The quadtree has fewer nodes than the range tree and uses less storage. A larger number of search tokens is required for each range, leaking slightly more co-occurrence information. However, the leakage is similar and thus we consider the Quad-BRC of similar security as the Range-BRC/URC scheme.

SRC Schemes. In the Range/Quad-SRC schemes, only a single search token is sent to the server at query time. Thus they do not leak the co-occurrence of search tokens. These two schemes also allow for false positives, which make reconstruction more difficult. In fact, false positives have been introduced as a mitigation technique (e.g. [25, 49]). Note that each canonical range corresponds to a certain number of possible queries. Thus, a persistent adversary could observe a number of search tokens and exploit the frequency with which a token is observed along with knowledge of the underlying DAG structure and then try to guess the token's corresponding node. Nevertheless, we support the claim by Demertzis et al. [14] that SRC schemes are the most schemes.

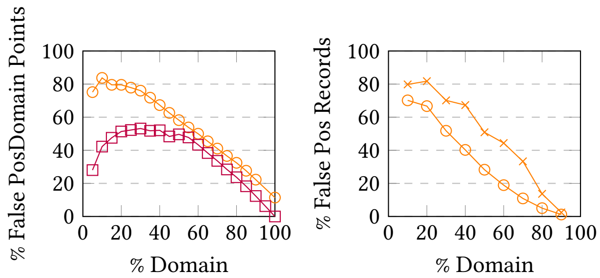


Figure 5: (Left) The number of *false positive domain points* returned with each query for Range-SRC (—□—) and Quad-SRC (—○—), averaged over all ranges of a given size. For both schemes, the false positive rate drops off steeply for queries larger than 30% of the domain. (Right) The percent of *false positive records* returned for Quad-SRC given the Spitz (—×—) and Cali (—○—) datasets. Since Cali is sparser than Spitz, we observe that queries to Cali return fewer false positives as a percentage of the total number of records.

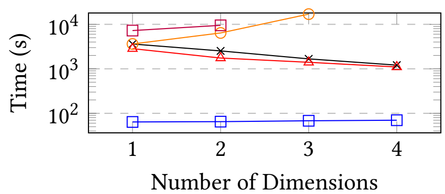


Figure 6: We compare the construction time of the Gowalla dataset with 1 million records and 2^{24} domain points, given a different number of dimensions with the Linear (—□—), Quad-BRC (—△—), Quad-SRC (—○—), Range-BRC/URC (—×—), and Range-SRC (—□—) schemes.

Leakage Abuse Attacks. We believe that it is possible to achieve (possibly partial) database reconstruction under the *linear* scheme after observing a small number of queries, since the queries leak local information. For the tree-based BRC/URC schemes, we believe that a reconstruction attack would need more queries in order to infer the tree structure, since the leakage is less local. Finally for the SRC schemes, we believe that an attack would require a significantly larger amount of queries observed accompanied by auxiliary information. This is due to the fact that SRC does not leak co-occurrences. It would be interesting to characterize the set of databases that produce equivalent leakage and develop attacks to better understand structure pattern.

6 EXPERIMENTS

In this section, we experimentally evaluate the performance of our Linear, Range-BRC/URC, Quad-BRC, Range-SRC, and Quad-SRC schemes using the following real-world datasets:

Gowalla [12]: A 4D dataset consisting of 6,442,892 latitude-longitude points of check-ins from users of the Gowalla social networking website between 2009 and 2010, a dataset used in the experiments by Demertzis et al. [14]. We further replicate Demertzis et al.'s Gowalla experiments by randomly partitioning the dataset into 10 sets, each consisting of 500,000 records. We then measured the indexing time and cost of our schemes by increasing the size of the domain by a new set of 500,000 tuples.

Spitz [60]: A 2D dataset of 28,837 latitude-longitude points of phone location data of politician Malte Spitz from Aug 2009 to Feb 2010 and previously used in prior attack work [19, 41, 47].

NH [61]: A 3D dataset comprised of 4096 elevation points on domain $[2^6] \times [2^6] \times [2^6]$ sampled from the United States Geological Survey's Elevation Data from the White Mountains of New Hampshire. We change the domain size by keeping exactly one aggregated elevation value per latitude and longitude value. **Cali [45]**: A 2D dataset of 21,047 latitude-longitude points of road network intersections in California, a dataset used in a 2D attack [47].

Implementation Details. We implemented our schemes in Python 3.9.2. We ran all of our experiments on a compute cluster. For simplicity, we used the same compute node for the client and the server so our results do not include any latency that would be incurred due to network transmission.

For cryptographic primitives, we used version 3.4.7 of the Python cryptography library [54]. To match the evaluation of Demertzis et al. [14], we use SHA-512 for PRFs and AES-CBC (with 128-bit block size) for encryption. We advise practitioners to follow the latest NIST standards when choosing which algorithms to use for PRFs and encryption. For our underlying EMM scheme, we used our own implementation of the Π_{bas} construction [9].

Results. For the two-dimensional datasets, we used the latitude and longitude as query attributes and 16-byte random strings as records. For our scheme experiments, we normalized the domain of Cali and Spitz to $[2^{10}] \times [2^{10}]$ and the domain of Gowalla to $[2^{16}] \times [2^{16}]$ to have the same number of domain points as in the 32-bit domain used by Demertzis et al. [14].

In Figure 5, we depict our results of the false positives of the Range-SRC and Quad-SRC schemes. We report both the percent of *false positive domain values* and the *false positive records* returned. Our experiments show that queries on sparse datasets generally return fewer false positive records.

Figure 6 shows the construction times for each scheme from one to four dimensions. The Linear scheme remains constant since the number of nodes across the varying dimensions remains the same. In contrast, the number of (auxiliary) canonical ranges for the SRC schemes increase as the dimension increases (in addition to the number of encryptions), and we thus see longer constructions times. For BRC/URC, since the domain size is fixed, the trees become more shallow with increasing dimension; We conjecture that construction time decreases since less time is needed to traverse the trees.

Table 2 shows the performance of our schemes. The first row gives the storage size (space), which grows with the number of database records and domain size consistently with the asymptotic space complexities of the schemes, with differences due to constant factors. The second row gives the construction time, which also grows with the number of database records and domain size consistently with our theoretical complexity analysis, with the exception of NH dataset, where large constant factors in the Quad-SRC scheme appear to dominate the logarithmic factors of the range-based schemes for the given domain sizes. As expected, the Linear scheme has the best storage size and construction time, however, its query complexity can be prohibitive. Schemes based on the quadtree have better storage size and construction time (except for the NH dataset) than the schemes based on the range tree.

The last row of Table 2 presents the query time, which depends on the size of range covers and the number of records that are

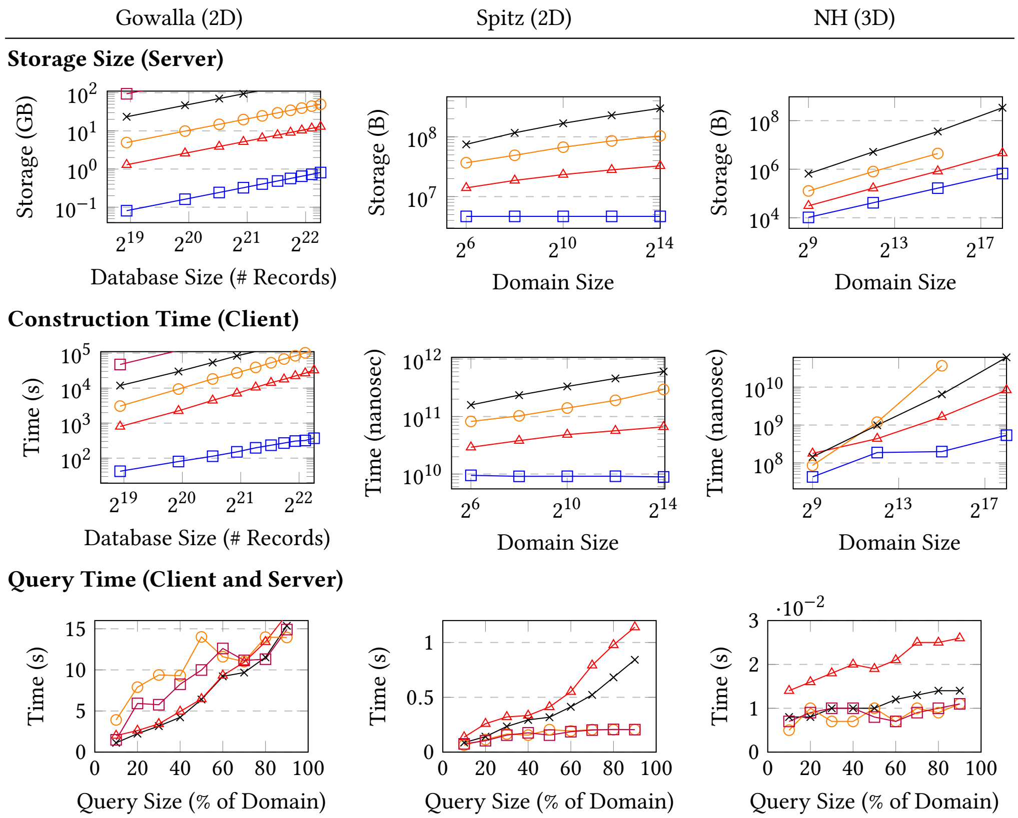


Table 2: Table of figures: Linear (—□—), Quad-BRC (—△—), Quad-SRC (—○—), Range-BRC/URC (—×—), and Range-SRC (—□—) schemes. For the Index Size and Construction time, we fix the domain size of the Gowalla dataset to $[2^{16}] \times [2^{16}]$ and vary the number of records, for the Spitz dataset we fix the number of records (28,837) and vary the domain size, for the NH dataset the number of records scales with the domain size ($n = m^{2/3}$). For the Query Time, we fix domain sizes $[2^{16}] \times [2^{16}]$ for Gowalla with 1 million records, $[2^{10}] \times [2^{10}]$ for Spitz with 28,837 records, and $[2^6] \times [2^6] \times [2^6]$ for NH with ≈ 4000 records. (The Linear scheme is not included in the query benchmarks due to its prohibitive complexity.)

returned. The reported query time is the sum of the time to execute methods Query, Eval, and Result (Figure 1) averaged over 500 queries of similar size sampled uniformly at random. In the Gowalla dataset, which has 1 million records, method Result (i.e., decrypting the records) dominates the query time. The SRC schemes return false positives, and thus have a higher query time than the BRC/URC ones. Also, the Quad-SRC scheme returns more false positives than the Range-SRC (Figure 5), and thus takes the longest time. The Spitz and NH datasets have way fewer records. Hence, method Query (determining the range cover) dominates the query time. Here, the SRC schemes require the least amount of query time as their range covers have size 1 and the number of false positive records is never more than four times the response size (Figure 5). Also, Quad-BRC has larger range covers than Range-BRC/URC (Table 1), thus Query takes longer to compute them. Note that also for query times, the reported experimental results are consistent with the asymptotic complexity. In particular, the difference in the relative performance of the SRC-based schemes vs. the range-based schemes in the Gowalla dataset and the Spitz and NH datasets is explained by the above detailed analysis.

Range-tree-based schemes exhibit better query complexity whereas quadtree-based schemes exhibit smaller storage size and construction times. Generally, schemes with more canonical ranges have longer build times. For example, Range-BRC/URC have more nodes than Quad-SRC, and consequently larger storage size.

7 CONCLUSION

We introduce a framework for designing schemes that support range queries over encrypted data in multiple dimensions. In particular, we describe how to turn a broad class of DAG-based spatial range search data structures into parallelizable encrypted databases that support range queries. We demonstrate the effectiveness of this framework by developing six schemes that offer trade-offs for space-complexity, query bandwidth, response size, and leakage to suit the needs of a wide variety of applications.

Several aspects of our work are novel extensions of the 1D schemes in prior work: We introduce a new scheme based on the quadtree – which previously had no analogue in 1D. We introduce a new data structure called the QDAG that helps to reduce the bandwidth of the Quad-BRC scheme to $O(1)$ while maintaining the same storage complexity. We adapt URC and BRC to work on the multi-level structure of the multi-dimensional range tree, while preserving their properties in 1D. The strength of our schemes lies in the fact that they are rooted in classic data structures which are efficient, flexible, and easy to implement.

REFERENCES

[1] Lars Arge, Mark de Berg, Herman J. Haverkort, and Ke Yi. 2004. The Priority R-Tree: A Practically Efficient and Worst-Case Optimal R-Tree. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data* (Paris, France) *(SIGMOD '04)*. Association for Computing Machinery, New York, NY, USA, 347–358.

[2] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. 1990. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data* (Atlantic City, New Jersey, USA) *(SIGMOD '90)*. Association for Computing Machinery, New York, NY, USA, 322–331.

[3] J. L. Bentley and J. H. Friedman. 1979. Data Structures for Range Searching. *ACM Comput. Surv.* 11, 4 (Dec. 1979), 13 pages.

[4] V. Bindschaedler, P. Grubbs, D. Cash, T. Ristenpart, and V. Shmatikov. 2018. The Tao of Inference in Privacy-Protected Databases. *Proc. VLDB Endow.* 11, 11 (July 2018), 14 pages.

[5] Dmytro Bogatov, Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. 2021. ϵpsolute: Efficiently Querying Databases While Providing Differential Privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, Republic of Korea) *(CCS '21)*. Association for Computing Machinery, New York, NY, USA, 2262–2276.

[6] D. Bogatov, G. Kollios, and L. Reyzin. 2019. A Comparative Evaluation of Order-Revealing Encryption Schemes and Secure Range-Query Protocols. *Proc. VLDB Endow.* 12, 8 (April 2019), 933–947.

[7] R. Bost. 2016. Sophos: Forward Secure Searchable Encryption. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA, 12 pages.

[8] R. Bost, B. Minaud, and O. Ohrimenko. 2017. Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives. In *Proceedings of th ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA). New York, NY, USA, 18 pages.

[9] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M.C. Rosu, and M. Steiner. 2014. Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*.

[10] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.C. Roşu, and M. Steiner. 2013. Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries. In *Advances in Cryptology – CRYPTO 2013*. Berlin, Heidelberg.

[11] M. Chase and S. Kamara. 2010. Structured Encryption and Controlled Disclosure. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings (Lecture Notes in Computer Science, Vol. 6477)*.

[12] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. 2011. Friendship and Mobility: User Movement in Location-Based Social Networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Diego, California, USA) *(KDD '11)*. New York, NY, USA, 1082–1090.

[13] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. 2006. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In *Proc. ACM Conf. on Computer and Communications Security*.

[14] I. Demertzis, S. Papadopoulos, O. Papapetrou, A. Deligiannakis, and M. Garofalakis. 2016. Practical private range search revisited. In *Proc. ACM Int. Conf. on Management of Data (SIGMOD)*.

[15] Ioannis Demertzis, Stavros Papadopoulos, Odysseas Papapetrou, Antonios Deligiannakis, Minos Garofalakis, and Charalampos Papamanthou. 2018. Practical Private Range Search in Depth. *ACM Trans. Database Syst.* 43, 1, Article 2 (2018), 52 pages. https://doi.org/10.1145/3167971

[16] Sabrina De Capitani di Vimercati, Dario Facchinetti, Sara Foresti, Gianluca Oldani, Stefano Paraboschi, Matthew Rossi, and Pierangela Samarati. 2021. Multidimensional indexes for point and range queries on outsourced encrypted data. *Proceedings of the GLOBECOM* (2021), 1–1.

[17] F. B. Durak, T. M. DuBuisson, and D. Cash. 2016. What Else is Revealed by Order-Revealing Encryption?. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS '16)*. New York, NY, USA, 12 pages.

[18] S. Faber, S. Jarecki, H. Krawczyk, Q. Nguyen, M. Rosu, and M. Steiner. 2015. Rich Queries on Encrypted Data: Beyond Exact Matches. In *Computer Security – ESORICS 2015*. Cham.

[19] F. Falzon, E. A. Markatou, Akshima, D. Cash, A. Rivkin, J. Stern, and R. Tamassia. 2020. Full Database Reconstruction in Two Dimensions. In *Proc. ACM Conf. on Computer and Communications Security (CCS)*.

[20] Francesca Falzon, Evangelia Anna Markatou, Zachary Espiritu, and Roberto Tamassia. 2022. Attacks on Encrypted Range Search Schemes in Multiple Dimensions. Cryptology ePrint Archive, Report 2022/090. https://ia.cr/2022/090.

[21] R. A. Finkel and J. L. Bentley. 1974. Quad Trees a Data Structure for Retrieval on Composite Keys. 4, 1 (mar 1974), 1–9.

[22] C. Gentry. 2009. *A Fully Homomorphic Encryption Scheme*. Ph.D. Dissertation. Stanford, CA, USA. Advisor(s) Boneh, D.

[23] J. Ghareh Chamani, D. Papadopoulos, C. Papamanthou, and R. Jalili. 2018. New Constructions for Forward and Backward Private Symmetric Searchable Encryption *(CCS '18)*. New York, NY, USA, 18 pages.

[24] O. Goldreich and R. Ostrovsky. 1996. Software Protection and Simulation on Oblivious RAMs. *J. ACM* 43, 3 (May 1996), 43 pages.

[25] Paul Grubbs, Anurag Khandelwal, Marie-Sarah Lacharité, Lloyd Brown, Lucy Li, Rachit Agarwal, and Thomas Ristenpart. 2020. Pancake: Frequency Smoothing for Encrypted Data Stores. In *USENIX Security Symposium*. 2451–2468.

[26] P. Grubbs, M.S. Lacharité, B. Minaud, and K.G. Paterson. 2018. Pump Up the Volume: Practical Database Reconstruction from Volume Leakage on Range Queries. In *Proc. ACM Conf. on Computer and Communications Security (CCS)*.

[27] P. Grubbs, M. Lacharité, B. Minaud, and K. G. Paterson. 2019. Learning to Reconstruct: Statistical Learning Theory and Encrypted Database Attacks. In *Proc. IEEE Symp. on Security and Privacy (S&P)*.

[28] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart. 2017. Leakage-Abuse Attacks against Order-Revealing Encryption. In *2017 IEEE Symposium on Security and Privacy (SP)*.

[29] Z. Gui, O. Johnson, and B. Warinschi. 2019. Encrypted Databases: New Volume Attacks against Range Queries. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*.

[30] Antonin Guttman. 1984. R-Trees: A Dynamic Index Structure for Spatial Searching. *SIGMOD Rec.* 14, 2 (jun 1984), 47–57. https://doi.org/10.1145/971697.602266

[31] F. Hahn and F. Kerschbaum. 2016. Poly-Logarithmic Range Queries on Encrypted Data with Small Leakage. In *Proceedings of the 2016 ACM on Cloud Computing Security Workshop* (Vienna, Austria) *(CCSW '16)*. New York, NY, USA, 12 pages.

[32] R. Helm, K. Marriott, and M. Odersky. 1995. Spatial Query Optimization: From Boolean Constraints to Range Queries. *J. Comput. System Sci.* 51, 2 (1995), 197–210. https://doi.org/10.1006/jcss.1995.1061

[33] S. Kamara and T. Moataz. 2018. SQL on Structurally-Encrypted Databases. In *Advances in Cryptology – ASIACRYPT 2018*. Cham.

[34] Seny Kamara and Tarik Moataz. 2019. Computationally Volume-Hiding Structured Encryption. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 11477)*. Springer, 183–213.

[35] S. Kamara and C. Papamanthou. 2013. Parallel and Dynamic Searchable Symmetric Encryption. In *Financial Cryptography and Data Security*. Berlin, Heidelberg.

[36] S. Kamara, C. Papamanthou, and T. Roeder. 2012. Dynamic Searchable Symmetric Encryption. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security* (Raleigh, North Carolina, USA) *(CCS '12)*. New York, NY, USA, 12 pages.

[37] Ibrahim Kamel and Christos Faloutsos. 1992. Parallel R-Trees. In *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data* (San Diego, California, USA) *(SIGMOD '92)*. Association for Computing Machinery, New York, NY, USA, 195–204.

[38] G. Kellaris, G. Kollios, K. Nissim, and A. O'Neill. 2016. Generic Attacks on Secure Outsourced Databases. In *Proc. ACM Conf. on Computer and Communications Security*.

[39] Shabnam Kasra Kermanshahi, Shi-Feng Sun, Joseph K. Liu, Ron Steinfeld, Surya Nepal, Wang Fat Lau, and Man Au. 2020. Geometric range search on encrypted data with Forward/Backward security. *IEEE Transactions on Dependable and Secure Computing* (2020), 1–1. https://doi.org/10.1109/TDSC.2020.2982389

[40] A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias. 2013. Delegatable Pseudorandom Functions and Applications *(CCS '13)*. New York, NY, USA, 669–684.

[41] E. M. Kornaropoulos, C. Papamanthou, and R. Tamassia. 2020. The State of the Uniform: Attacks on Encrypted Databases Beyond the Uniform Query Distribution. In *Proc. IEEE Symp.on Security and Privacy (S&P)*.

[42] Evgenios M. Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. 2021. Response-Hiding Encrypted Ranges: Revisiting Security via Parametrized Leakage-Abuse Attacks. In *Proc. IEEE Symp. on Security and Privacy (S&P)*.

[43] Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. 2018. The Case for Learned Index Structures. In *Proceedings of the 2018 International Conference on Management of Data* (Houston, TX, USA) *(SIGMOD '18)*. Association for Computing Machinery, New York, NY, USA, 489–504.

[44] M.S. Lacharité, B. Minaud, and K.G. Paterson. 2018. Improved reconstruction attacks on encrypted data using range query leakage. In *Proc. IEEE Symp. on Security and Privacy 2018 (S&P 2018)*.

[45] Feifei Li, Dihan Cheng, Marios Hadjieleftheriou, George Kollios, and Shang-Hua Teng. 2005. On Trip Planning Queries in Spatial Databases. In *Advances in Spatial and Temporal Databases*. Berlin, Heidelberg, 273–290.

[46] Ryan Marcus, Andreas Kipf, Alexander van Renen, Mihail Stoian, Sanchit Misra, Alfons Kemper, Thomas Neumann, and Tim Kraska. 2020. Benchmarking Learned Indexes. *Proc. VLDB Endow.* 14, 1 (sep 2020), 1–13.

[47] Evangelia Anna Markatou, Francesca Falzon, Roberto Tamassia, and William Schor. 2021. Reconstructing with Less: Leakage Abuse Attacks in Two Dimensions. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, Republic of Korea) *(CCS '21)*. Association for Computing Machinery, New York, NY, USA, 2243–2261.

[48] Evangelia Anna Markatou and Roberto Tamassia. 2019. Full Database Reconstruction with Access and Search Pattern Leakage. In *Proc. Int. Conf on Information Security (ISC)*.

[49] Evangelia Anna Markatou and Roberto Tamassia. 2019. Mitigation Techniques for Attacks on 1-Dimensional Databases that Support Range Queries. In *Information Security - 22nd International Conference, ISC 2019, New York City, NY, USA, September 16-18, 2019, Proceedings (Lecture Notes in Computer Science, Vol. 11723)*.

[50] M. Naveed, S. Kamara, and C. V. Wright. 2015. Inference Attacks on Property-Preserving Encrypted Databases. In *Proc. ACM Conf. on Computer and Communications Security (CCS)*. 12 pages.

[51] M. Naveed, M. Prabhakaran, and C. A. Gunter. 2014. Dynamic Searchable Encryption via Blind Storage. In *2014 IEEE Symposium on Security and Privacy*.

[52] V. Pappas, F. Krell, B. Vo, V. Kolesnikov, T. Malkin, S.G. Choi, W. George, A. Keromytis, and S. Bellovin. 2014. Blind Seer: A Scalable Private DBMS. In *2014 IEEE Symposium on Security and Privacy*.

[53] S. Patel, G. Persiano, K. Yeo, and M. Yung. 2019. Mitigating Leakage in Secure Cloud-Hosted Data Structures: Volume-Hiding for Multi-Maps via Hashing. In *Proc. ACM Conf. on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. New York, NY, USA, 15 pages.

[54] Python Cryptographic Authority. 2018. pyca/cryptography. https://cryptography.io/ version 3.4.7.

[55] Jianzhong Qi, Guanli Liu, Christian S. Jensen, and Lars Kulik. 2020. Effectively Learning Spatial Indices. *Proc. VLDB Endow.* 13, 12 (jul 2020), 2341–2354.

[56] Jianzhong Qi, Yufei Tao, Yanchuan Chang, and Rui Zhang. 2018. Theoretically Optimal and Empirically Efficient R-Trees with Strong Parallelizability. *Proc. VLDB Endow.* 11, 5 (jan 2018), 621–634.

[57] P. Rizomiliotis, E. Molla, and S. Gritzalis. 2017. REX: A Searchable Symmetric Encryption Scheme Supporting Range Queries *(CCSW '17)*. New York, NY, USA.

[58] E. Shi, J. Bethencourt, T-H. H. Chan, D. Song, and A. Perrig. 2007. Multi-Dimensional Range Query over Encrypted Data *(SP '07)*. USA, 15 pages.

[59] D. Song, D. Wagner, and A. Perrig. 2000. Practical techniques for searches on encrypted data. In *Proceeding IEEE Symposium on Security and Privacy (S&P)*.

[60] Malte Spitz. 2011. CRAWDAD dataset spitz/cellular (v. 2011-05-04). Downloaded from https://crawdad.org/spitz/cellular/20110504.

[61] United States Geological Survey. [n.d.]. Elevation Products (3DEP). Downloaded from apps.nationalmap.gov.

[62] B. Wang, Y. Hou, M. Li, H. Wang, and H. Li. 2014. Maple: Scalable Multi-Dimensional Range Search over Encrypted Cloud Data with Tree-Based Index. In *Proc. of the 9th ACM Symposium on Information, Computer and Communications Security (ASIA CCS '14)*.

[63] Jiafan Wang and Sherman S. M. Chow. 2022. Forward and Backward-Secure Range-Searchable Symmetric Encryption. *Proc. Priv. Enhancing Technol.* 2022, 1 (2022), 28–48. https://doi.org/10.2478/popets-2022-0003

[64] Cong Zuo, Shi-Feng Sun, Joseph K Liu, Jun Shao, and Josef Pieprzyk. 2018. Dynamic searchable symmetric encryption schemes supporting range queries with forward (and backward) security. In *European Symposium on Research in Computer Security (ESORICS) (LNCS)*. Springer, 228–246.