Attacks on Encrypted Response-Hiding Range Search Schemes in Multiple Dimensions

Evangelia Anna Markatou*
Brown University
markatou@brown.edu

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

Zachary Espiritu
Brown University
zesp@brown.edu

Roberto Tamassia
Brown University
roberto@tamassia.net

ABSTRACT

In this work, we present the first database reconstruction attacks against response-hiding private range search schemes on encrypted databases of arbitrary dimensions. Falzon et al. (VLDB 2022) present a number of range-supporting schemes on arbitrary dimensions exhibiting different security and efficiency trade-offs. Additionally, they characterize a form of leakage, structure pattern leakage, also present in many one-dimensional schemes e.g., Demertzis et al. (SIGMOD 2016) and Faber et al. (ESORICS 2015). We present the first systematic study of this leakage and attack a broad collection of schemes, including schemes that allow the responses to contain false-positives (often considered the gold standard in security). We characterize the information theoretic limitations of a passive persistent adversary. Our work shows that for range queries, structure pattern leakage can be as vulnerable to attacks as access pattern leakage. We give a comprehensive evaluation of our attacks with a complexity analysis, a prototype implementation, and an experimental assessment on real-world datasets.

KEYWORDS

leakage-abuse attacks, encrypted databases

1 INTRODUCTION

An *encrypted database* (EDB) allows a client to outsource sensitive data to an untrusted cloud server and then privately query this data. In the last decade, we have seen an increased demand for EDBs that support expressive queries. For example, MongoDB's "Queryable Encryption" enables clients to execute expressive queries over encrypted data [9]. With the deployment of EDBs in the wild, it is more important than ever to understand the security of such schemes. In this work, we study classes of schemes that support private *range queries* over multi-attribute (multi-dimensional) data. Concretely, a range query over two attributes takes the form of:

```
SELECT * FROM T WHERE (years BETWEEN 2010 AND 2020)
AND (avg_temp BETWEEN 15 AND 18)
```

where T is a table and `years` and `avg_temp` are attributes. Range queries are fundamental and support for such query expressiveness is a requisite for practical EDBs. It is thus important that we understand the security provided by such EDBs e.g. [26].

*Both authors contributed equally to this research.

Proceedings on Privacy Enhancing Technologies 2023(4), 204–223

https://doi.org/10.56553/popets-2023-0106

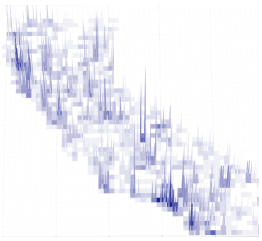

Figure 1: Reconstruction by our attack on the range tree scheme with uniform range cover (Section 4) for the Cali dataset over a domain with 1024×1024 points. The bar heights represent the number of records at each domain point. The attack succeeds in 68s.

One common way to construct EDBs that support range queries is to use *searchable symmetric encryption* (SSE) (e.g. [11, 31, 46]) to build an encrypted index that maps ranges to corresponding records. A number of such schemes have been proposed for both one attribute data (e.g. [19, 24, 41, 68]) and multi-attribute data [26]. These schemes are efficient in practice which make them a primary candidate for real-world applications. This efficiency, however, comes at the cost of "leaking" a small amount of well-defined information about the underlying database or queries.

Leakage typically occurring in SSE schemes includes one or more of the following: *search pattern* (whether two queries are the same); *volume pattern* (the number of records in the query response); and *access pattern* (which individually and deterministically encrypted records are returned with each query). Though this leakage may seem benign, a number of database reconstruction attacks leveraging the leakage of range queries in one [36, 37, 39, 47, 50–52] and two [25, 54] dimensions have been described. Existing attacks in 2D are more theoretical and do not attack existing constructions. Our work goes beyond 2D and is the first to attack concrete range schemes. Falzon et al. [26] note that "structure pattern leakage is inherent in schemes derived from standard multidimensional search data structures," however the full extent to which this leakage can be exploited is not explored. We initiate the first cryptanalysis of structure pattern, present new techniques for exploiting the leakage and answer the following open question in the affirmative:

> *Can a passive persistent adversary attack existing multidimensional range schemes even in the absence of access pattern leakage?*

Our attacks work against non-interactive schemes presented in multiple works i.e., [19, 20, 24, 26]. These works use SSE and take a similar approach to scheme design which can be summarized as follows: Each scheme is associated with an underlying *range data structure* that can be represented as a graph; each node of the graph is associated with a range over the domain. The client then defines an index that maps each node's range to the set of matching records, and encrypts this map using the underlying SSE scheme. To issue a query, the client computes the set of nodes that cover the range,

Attack	Assumptions				Leakage				Attack			Rec. Space
	Dim	Query	Data	Scheme	AP	VP	EP	SP	# Queries for FDR	Runtime	Space	Size
Kellaris et al. [47]	1	Uniform		Quadratic [19]	✓				$m^4 \log m$			2
Lacharité et al. [52]	1		Dense	Quadratic [19]	✓				$m \log m$	$m \log m(m+n)$		2
Grubbs et al. [36]	1	Uniform		Quadratic [19]	✓				$m^4 \log m$			2
Markatou et al. [55]	1			Quadratic [19]		✓			$m^2 \log m$			2
Kornaropoulos et al. [50]	1			Quadratic [19]	✓		✓		–			2
Kellaris et al. [47]	1	Uniform		Quadratic [19]		✓			$m^4 \log m$			2
Grubbs et al. [37]	1			Quadratic [19]		✓			$m^2 \log m$			2
Gui et al. [39]	1	Unknown	Dense	Quadratic [19]		✓			–			
Kornaropoulos et al. [51]	1			Regular [19, 24, 51]		✓	✓	✓	–			
Falzon et al. [25]	2			Quadratic [25]	✓		✓		$m^2 \log m$	$m(nm^2 + n \log n)$		2^n
Markatou et al. [54]	2			Quadratic [25]	✓		✓		$m^2 \log m$			2^n
Linear Attack	d			Naive [19], Linear [26]		✓	✓	✓	$m^{2-\frac{1}{d}} \log^2 m$	m^5	m^3	$2^d(d!)$
Token Pair Attack	d			Range-Universal [24], Log-URC [19], Range-URC [26]		✓	✓	✓	$m^2 \log m$	$m^2 \log^d m$	$m^2 \log^d m$	$2^d(d!)$
	d			Quad-BRC [26]		✓	✓	✓	$m^2 \log m$	$m^{2+\frac{d-1}{d}}$	$m^{2+\frac{d-1}{d}}$	$2^d(d!)$
Range-BRC Attack	d			Range [19], Range-BRC [26]		✓	✓	✓	$m^2 \log m$	m^4	m^2	$2^d(d!)$
SRC Attack	d			TDAG [19], QDAG-SRC [26]		✓	✓	✓	$m^4 \log m$	–	$\Omega(dm)$	$\geq 2^{2^{d-1}}(d!)$
	d			Quadratic [19, 25]		✓	✓	✓	$m^4 \log m$	–	$\Omega(dm^2)$	$\geq 2^n$

Table 1: Comparison of our work with selected prior attacks on schemes for encrypted range search. AP, VP, EP, and SP refer to access pattern, volume pattern, equality pattern (also known as search pattern), and structure pattern, respectively. The time and space complexity of an attack are shown when reported by the authors. We note that previous works have typically focused on query complexity (number of queries sampled under a uniform query distribution needed to achieve full database reconstruction), often omitting the analysis of time and space complexity. Reconstruction space sizes are asymptotic lower bounds achieved in the worst case. m refers to the domain size, n is the number of records and d is the number of dimensions. We omit big-O notation.

generates a search token for each node, and sends these tokens to the server for look up.

Our attacks demonstrate insecurities of these schemes and highlight the importance of implementing additional mitigation techniques [35] when using SSE-based schemes. We show that volume and search pattern—when combined with the structural information of the underlying range search data structure—can be as detrimental to security as access and search pattern.

Our first attack works against the linear scheme, which boasts the smallest storage overhead. Our second attack works against a class of schemes that includes the quad-tree and a variation of the range-tree previously designed to reduce leakage. Our third attack works against another variation of the range-tree data structure. Our fourth attack works against a wide class of range schemes that achieve efficiency by allowing for false positives in responses and are regarded as the most secure [19]. We evaluate our attacks using real-world datasets.

1.1 Related Work

Schemes. We consider range search schemes that are built on searchable encryption primitives, which relax the security (compared to strong primitives like ORAM [33] and fully homomorphic encryption [29]) by leaking some well-defined information and achieve practical runtimes (see, e.g., [7, 8, 11–14, 16–18, 21, 30, 31, 44–46, 57, 59, 64]).

Demetrzis et al. [19, 20] present searchable encryption schemes for 1D databases. They present multiple schemes that trade-off security and efficiency. Faber et al. [24] also present range search schemes that support 1D range queries. Wang and Chow [67] support forward and backward secure range search for 1D databases. Falzon et al. [26] present the first range search schemes in multiple dimensions, and offer a variety of security and efficiency tradeoffs. Range search can also be achieved using other primitives. For example, Shi et al.'s MRQED [63] scheme and Maple [66] support range queries on databases of arbitrary dimensions using public key cryptography and leak at least the access pattern. Order-revealing encryption [1, 5] also supports range queries, however it leaks a lot more information [3, 22, 38]. In addition to range-reporting schemes, schemes have been presented on other types of queries, like aggregate range queries (e.g., Espiritu et al. [23]) and shortest path queries on graphs (e.g., Ghosh et al. [32]).

Attacks. Leakage analysis of SSE schemes has been studied in a passive adversarial setting e.g. [4, 10, 42, 58, 61]. Kellaris et al. [47] showed the first attack that leverages access and volume pattern leakage from 1D range queries to achieve full database reconstruction. Lacharité et al. [52] improved upon this work achieving efficient database reconstruction attacks for dense 1D databases. Grubbs et al. [36] follow up with an optimal approximate database reconstruction attack for any 1D database. The above works assume knowledge of the query distribution. Kornaropoulos et al. [50] and Markatou and Tamassia [55] show one-dimensional database reconstruction attacks that utilize search pattern leakage while assuming no knowledge of the query distribution.

Volume pattern leakage has also been shown to be exploitable in 1D databases [37, 51]. Recent attention has been devoted to exploiting volume and search pattern e.g. [4, 58]. Kamara et al.[43] present a framework for evaluating leakage attacks. Kornaropoulos

et al. [48] quantify the privacy of searchable encryption schemes using leakage inversion techniques. Two attacks most related to our Linear attack are the generic 2D database reconstruction attacks in [25, 54]. Unlike these works, our attacks are on concrete range schemes and work on databases of *two and higher* dimensions.

The closest prior attack to our SRC attack is by Kornaropoulos, Papamanthou, and Tamassia [51], who attack a class of 1D response hiding schemes, called *regular schemes*, including [20, 24]. Demertzis et al. [20] note their schemes are susceptible to attacks but do not give a full description or analysis. Demertzis et al. [18] propose a potential attack on the Logarithmic-SRC scheme [20] but also do not give a full description or analysis.

In addition to range-reporting schemes, prior attacks have targeted other types of rich queries, like the attack by Kornaropoulos et al. [49] against k-nearest neighbor queries and the attack by Falzon and Paterson [27] against shortest path queries.

1.2 Contributions

Our contributions are summarized as follows:

- We present the ***first attacks*** against the response-hiding non-interactive 1D range search schemes in [19, 24] and the response-hiding non-interactive multi-dimensional range search schemes in [26]. Previous attacks were limited to schemes supporting 1D or 2D queries.
- We leverage ***structure pattern*** to carry out database reconstruction in arbitrary dimensions. Our work shows that structure pattern can be as exploitable as access pattern. (Sections 3–6)
- We introduce ***new techniques*** for reconstruction attacks. We develop a ***number theoretic approach*** to reduce the number of observed responses needed to attack the linear scheme. We describe methods that ***exploit graphs*** built from search and structure patterns. We further present a framework based on ***integer linear programming*** to attack a broad class of SRC schemes, considered the gold standard. (Sections 3–6)
- We describe the ***information theoretic limitations*** of a passive-persistent adversary.
- We implement our attacks and ***experimentally evaluate*** them on real-world datasets. (Section 7)

In Section 8, we describe the techniques we develop in more detail and explain how they can be extended to new schemes. Table 1 compares our attacks with related work. Our work demonstrates pitfalls of basing private range search schemes on range-search data structures and informs future research on expressive queries.

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 $d \geq 1$. 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 n 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.

We say that points p and p' are ***neighbors*** if they share every coordinate but one, and in the remaining coordinate, their values differ by one. We call a set of contiguous points of D that only differ in the same single coordinate a ***one-dimensional section***.

We say that a and b are the extreme values of range $[a, b]$. We define the ***core*** of the database domain $\mathcal{D}$ as the set of points of $\mathcal{D}$ that do not have an extreme value in any dimension. We define the ***boundary*** of a database as the set of points of $\mathcal{D}$ with at least one extreme value in some dimension.

Let $\mathcal{S}$ be a set of range queries and let A and B be queries in $\mathcal{S}$. We say that A ***minimally contains*** B if A contains B and there is no other query C in $\mathcal{S}$ distinct from A and B such that A contains C and C contains B. We use double-brace notation to denote a multiset, e.g. {{1,1,4,5,7}}.

2.1 Range Trees and Quadtrees

The schemes we attack build upon the range tree and region quad tree data structures.

Range Tree [2] A ***range tree*** is a data structure that holds points. It allows for efficient range queries, especially in two and higher dimensions. In one dimension, a range tree is a binary tree. Each node corresponds to a range: the left child of the node corresponds to the first half of the range and the right child corresponds to the second half of the range. The root node covers the domain, and the leaf nodes each cover a single domain point.

For example, in Figure 3(a), we can see how the root node covers range [1,16], and its children cover ranges [1,8] and [9,16]. In two dimensions, the range tree is no longer a binary tree. Instead, there is a main tree that orders the points according to their first dimension, and each node of this main tree has a third child. This third child leads to its own copy of a binary tree that orders the points in the range of this node along the second dimension. The range tree is defined recursively for higher dimensions, having a separate tree that orders the points across each dimension. See Figure 2 for a 2D example.

Region Quad Tree [28]. A ***region quadtree*** on a d-dimensional square domain $\mathcal{D}$ comprises of a 2^d-ary tree whose nodes are associated with a subdomain. The root node is identified with the whole domain $\mathcal{D}$; The tree recursively sub-divides the square domain into quadrants (or *orthants* in dimensions greater than 2). Each internal node has 2^d children – each child corresponding to one of the 2^d quadrants associated with its parent.

Range Covers. There are numerous ways to query a range supporting data structure like a range tree or a quad-tree. In this paper, we consider three range covering techniques: ***Best Range Cover (BRC)***, ***Uniform Range Cover (URC)***, and ***Single Range Cover (SRC)***. BRC selects the smallest number of nodes that perfectly covers the range. With BRC, ranges of the same size may correspond to a different number of nodes. For example, in Figure 3, using BRC to query range $[1, 2]$ (nodes a and b) returns a single node (ab). In contrast, querying $[2, 3]$ (nodes b and c) returns two nodes (b and c). This discrepancy led to the development of URC [19], which ensures that ranges of the same size correspond to range covers of the same size. The final range cover we consider is SRC. This range cover returns a single node corresponding to the smallest range containing the query; its response may result in false positives.

2.2 Formalizing Leakage

Structured encryption is parameterized by different leakage functions, which output information about the underlying data structure and its contents. We define two common leakage functions of Encrypted MultiMap (EMM) schemes relevant to this work [12].

- The ***search pattern*** (also known as ***equality pattern***) reveals when two queries are equal. Without loss of generality, we can assume a 1-to-1 correspondence between range queries and query identifiers. Search pattern is a function EP that takes as input a multimap MM and a label $\ell \in \mathbb{L}$, and outputs an ID: $\mathsf{EP}(\mathsf{MM}, \ell) \mapsto i \in [|\mathbb{L}|]$.
- The ***volume pattern*** of a label ℓ in a multimap reveals the number of records associated with ℓ. Formally, the volume pattern is a function Vol that takes as input a label in the label space ℓ and outputs the number of records associated with the given label: $\mathsf{Vol}(\mathsf{MM}, \ell) \mapsto |\mathsf{MM}[\ell]|$.

Throughout this paper, we assume that the underlying EMM scheme is response-hiding, and leaks the multimap size at setup, and search and volume pattern at query time. The constructions considered use EMMs to support range queries over range search data structures. The EMM is used to encrypt a multimap that maps subqueries, also referred to as ***canonical ranges***, to records associated with each subquery. The result is an additional form of leakage called structure pattern that is a function of the data structure, the range cover, and the leakage of the EMM scheme [19, 26].

- The ***structure pattern*** leakage reveals if two queries have a common subquery. Let query q be associated with k labels $\ell_i \in \mathbb{L}$, $i \in [1, k]$. Then the structure pattern leakage is
  $\mathsf{SP}(\mathsf{MM}, q) = \{(\mathsf{EP}(\mathsf{MM}, \ell_i), \mathsf{Vol}(\mathsf{MM}, \ell_i))\}_{i \in [k]}$.

2.3 Attack Input

The ***tokenset*** $\mathbf{t}$ of a query q is the set of tokens associated with q. For each token t sent by the client, the server returns the encrypted set $C(t)$ retrieved from an encrypted multimap, from which the adversary determines the volume, vol_t, associated with token t. For each scheme, we present a reconstruction attack that takes as input a ***volume map***, denoted with VM, that for each tokenset $\mathbf{t}$, maps $\mathsf{VM}[\mathbf{t}] = \sum_{t \in \mathbf{t}} vol_t$, and for each token $t \in \mathbf{t}$, maps $\mathsf{VM}[t] = vol_t$.

Our attack on the SRC schemes also takes as input a ***frequency map*** FM, which associates each tokenset with the number of times it has been observed. Maps VM and FM take linear time to build on the size of the input and require less storage than the input, since their sizes are independent of n. We assume the adversary has knowledge of m, d, n, as well as of the range encrypted multimap scheme employed. Our attacks take as input VM and (in the SRC case) FM and return a grid comprising one node for each point of in domain $\mathcal{D}$, where each node is labeled with the number of database records at the corresponding point.

We show that VM and FM are an equivalent representation of the multiset of structure pattern. We assume that the queries are issued independently; we do not exploit the order of the queries, for example, by assuming that their order is correlated to their position. It is thus sufficient to consider a multiset of the structure pattern.

Algorithm 1: *LinearReconstruction*(VM)

```
// Find tokensets that correspond to one-dimensional queries.
Let primeTokensets store the tokensets of unit and prime size in VM.
Let 1dSlices be an empty map, mapping tokens (which share the same
  coordinates in all but one dimension) to a list of tokensets
// Group tokensets by one-dimensional section.
for t ∈ primeTokensets do
    Find all keys, K, in 1dSlices that intersect in ≥ 2 elements with t
    Add t to K and let V be a list of the values of K in 1dSlices + t
    Delete all keys in K from 1dSlices and add K → V to 1dSlices
// Order the elements of each one-dimensional section.
Create a PQ-tree for each key of 1dSlices with its values.
// Make a grid representing the domain value of each token.
Let G be a graph with nodes all the observed search tokens.
for each PQ-Tree T do
    Pick a frontier (a possible ordering of the search tokens) of T.
    Add an edge to G for every pair of neighbors in this frontier.
// Reconstruct the database.
Label the nodes of G with their volume in VM.
return G
```

Theorem 1. *Let Σ be an EMM scheme leaking search and volume pattern. Let D be a d-dimensional database over domain $\mathcal{D}$, MM the resulting multimap when encrypting with Σ, and $q^{(1)}, \ldots, q^{(k)}$ be range queries over $\mathcal{D}$. Then, there exists an invertible transformation between the multiset of leakage $\{\{\mathsf{SP}(\mathsf{MM}, q^{(i)})\}\}_{i \in [k]}$ and the corresponding volume map* VM *and frequency map* FM.

The proof of Theorem 1 (along with all our other proofs) can be found in the Appendix. Building FM requires observing each query once, which is a strong assumption. In Appendix B, we prove that these maps can be built after observing $m^4 \log m$ queries issued uniformly at random.

2.4 Equivalent Databases.

We generalize the notion of equivalent databases from [25, 54] below. Intuitively, two databases are $\mathcal{L}$-equivalent if they are indistinguishable from their leakage.

Definition 1. *Let D and D' be databases with domain $\mathcal{D}$ and the same record IDs. Let $\mathcal{L} = (\mathcal{L}_S, \mathcal{L}_Q)$ be a leakage function and $\mathcal{Q}$ be the set of range queries on $\mathcal{D}$. Databases D and D' are **$\mathcal{L}$-equivalent** if $\{\mathcal{L}(D, q)\}_{q \in \mathcal{Q}} = \{\mathcal{L}(D', q)\}_{q \in \mathcal{Q}}$. The set of equivalent databases is called the* reconstruction space.

2.5 Threat Model and Assumptions

Throughout this paper, we consider a passive, persistent, honest-but-curious adversary that has compromised either the communication channel or server. This adversary is able to observe the tokensets issued by the client and the number of records associated with each token the tokensets. The linear attack (Section 3) assumes that any non-empty subset of prime-sized range queries are issued. Our other attacks assume that all possible range queries are issued. For all attacks, we assume that the adversary is able to build the volume map VM from the observed queries. In Section 6, we make the additional assumption that the adversary can build the frequency map FM.

3 THE LINEAR ATTACK

The linear scheme comprises of $n = |\mathcal{D}|$ canonical ranges, which are one-to-one with the points in the domain. To query for a range $q \subseteq \mathcal{D}$, the client computes a token for each point in the range and sends the resulting tokenset to the server. The server can then use this tokenset to retrieve the matching records. To query for range $[1, 4]$, the client sends four tokens corresponding to four canonical ranges: $[1, 1]$, $[2, 2]$, $[3, 3]$ and $[4.4]$. Variants of this scheme have been proposed in both one [19] and multiple [26] dimensions.

3.1 Reconstruction Attack

Each domain point is associated with a single unique token. The linear scheme thus leaks both the size of the query range and information about the points in the range. For example, when querying a range of size 4, the adversary observes 4 tokens. The adversary can also infer information about the shape of the range e.g., if the database is two-dimensional, then a query of size 4 must correspond to a square range (2×2) or a one-dimensional slice (1×4). In particular, if the adversary observes a tokenset of prime size, p, then they can infer that the range has size p in one dimension and size 1 in the other dimensions. This leakage allows the adversary to extract useful 1D information. Our attack can thus leverage 1D techniques to reconstruct a multidimensional database. Our attack builds from the leakage a labeled graph whose vertices correspond to domain points and whose edges denote adjacent points. This resulting graph provides a reconstruction of the database up to symmetries and forms the basis of our attack.

Reconstruction Attack. Our attack finds queries of prime size, groups the search tokens into one-dimensional segments, and then orders them. Our attack (Algorithm 1) follows in five steps:

(1) **Find all tokensets (queries) of prime size.**
(2) **Group one-dimensional sections.** If the intersection of two tokensets of prime size has at least two search tokens, then these queries must be from the same 1D section (e.g., same row or column). We create map 1dSlices mapping search tokens to sets of tokensets, where all search tokens in a key of 1dSlices correspond to the same one-dimensional section.
(3) **Order one-dimensional sections.** Use PQ-trees [6] to get the partial order of the search tokens in each key of 1dSlices.
(4) **Order Reconstruction.** Construct a graph G whose nodes are the observed tokens. For each PQ-tree, find a frontier and add edges in G between neighboring tokens in each frontier.

Theorem 2. *Let D be a database over a d-dimensional domain $\mathcal{D} = [m_1] \times \cdots \times [m_d]$ of size m and let D be encrypted with the linear scheme. Given the volume map for a set of range queries on D comprising all queries of unit and prime size, Algorithm 1 achieves full database reconstruction of D by building in $O(m^5)$ time and $O(m^3)$ space an $O(m)$-size representation of the reconstruction space of D. The input to the algorithm is available with probability greater than $1 - \frac{1}{m^2}$ after observing*

$$O\left(\sum_{i=1}^{d} \frac{m^2}{m_i} \log m_i \cdot \log\left(\frac{m^2}{m_i} \log m_i\right)\right) \tag{1}$$

uniformly distributed queries, which is $O(m^{2-\frac{1}{d}} \log^2 m)$ queries when $m_i = m^{1/d}$ for $i = 1, \cdots, d$.

3.2 Reconstruction Space

The reconstruction space of the linear scheme comprises of the symmetries of a d-dimensional cube. In 2D, this means that we can reconstruct up to rotation and reflection of the rectangular domain. This is because each query contains one (deterministic) token for each point in the queried range. As a result, the server learns a map between search tokens and their corresponding records, a partition on the records with respect to their domain values, and which records belong to contiguous regions of the domain. If the server sees a sufficient number of queries it can piece the search tokens together into a d-dimensional grid.

Theorem 3. *Let D be a database on a d-dimensional domain and let $\mathcal{L}$ be the leakage of the linear scheme. The set of databases $\mathcal{L}$-equivalent to D, or reconstruction space of D, corresponds to the symmetries of a d-cube. (i.e. rotation/reflection across each axis).*

4 TOKEN PAIR ATTACK

Our next attack applies to a number of schemes, including the 1D Rangetree scheme with universal range cover [24] and the Logarithmic-URC scheme [19], as well as the Range-URC [26] and Quad-BRC [26] schemes in arbitrary dimensions. Range-URC can be viewed as a generalization of the Rangetree scheme with universal range cover and the Logarithmic-URC scheme.

Range-URC. This scheme uses a range-tree for the underlying range-supporting data structure and URC for computing the tokensets. Demertzis et al. [19, 20] and Falzon et al. [26] both construct schemes using a range tree with URC. This attack applies to both these constructions, and any constructions that use range trees with URC and leak volume and search pattern.

Quad-BRC. This scheme was first introduced by Falzon et al. [26] and uses a region quad-tree for the underlying range-supporting data structure and BRC for computing the tokensets. Recall that the

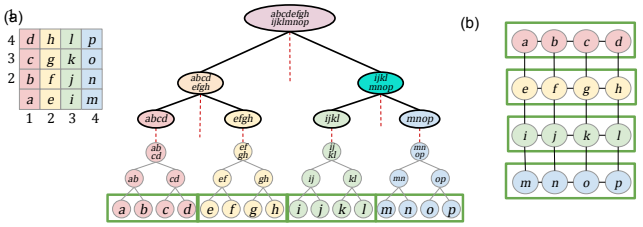

Figure 2: (a) A range tree scheme in 2D and (d) the graph constructed by Algorithm 2. The nodes in the green rectangles correspond to queries with one token under URC.

Algorithm 2: *TokenPairAttack*(VM)

1: Let Q_1 be the keys of VM of size 1.
2: Let Q_2 be the keys of VM of size 2 with only members of Q_1.
3: Construct graph G with nodes the elements of Q_1
4: **for** t = $\{t_0, t_1\} \in Q_2$ **do**
5: Add an edge between t_0 and t_1 in G.
6: Label the nodes of G with their volume in VM.
7: **return** the connected component of G of size m with the smallest total volume.

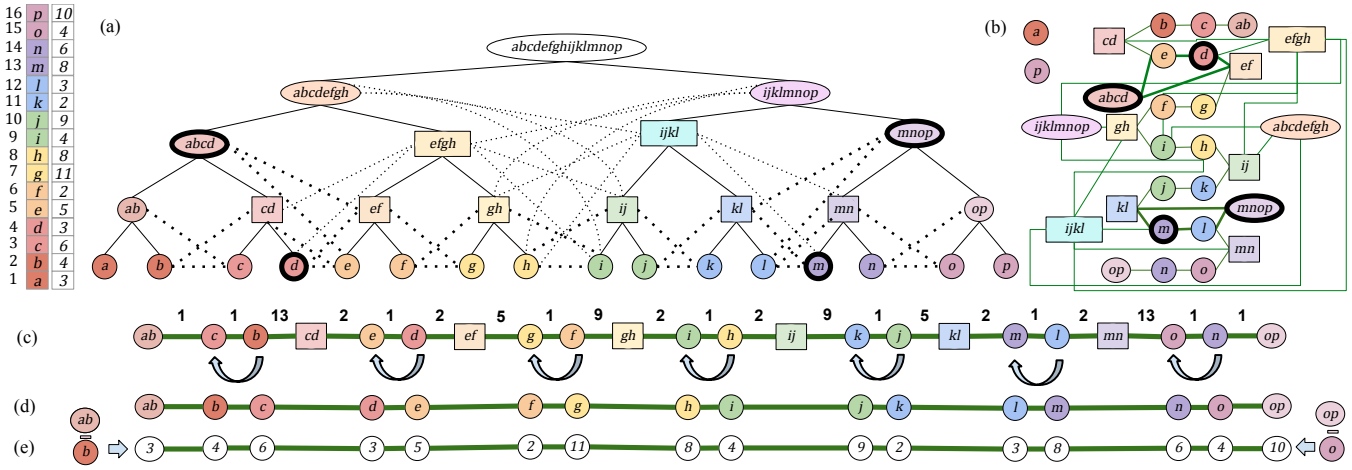

Figure 3: Attack on the range tree BRC scheme for a 1D domain (Algorithms 3 and 4). (a) Domain with volume of each point and range tree. We find the inner nodes of the range tree (rectangular) by relying on the property that tokensets form a continuous range (Algorithm 3, Line 15). E.g, tokensets $(d, \{ef\})$, $(\{ef\}, g)$ and $(d, \{ef\}, g)$, and the absence of tokenset (d, g) imply that $\{ef\}$ is an inner node. Thick dotted lines show the triangular structures identifying leaf nodes. (b) Co-occurrence graph G, whose edges (in green) join nodes of the range tree that form a tokenset, e.g., (b, c), (Algorithm 3, Line 4). (c) Construction of graph G_{trim}. We remove from G edges between inner nodes (Algorithm 3, Line 16). We use the times two tokens appear in a tokenset together (edgecounts) (Algorithm 3, Line 19) to remove edges with edgecount > 2. We identify most leaf nodes using G, but some nodes like $\{abcd\}$ and d appear identical in G after we trim. We distinguish them using graph G, e.g., $\{abcd\}$ has fewer edges than d (Algorithm 3, Line 27). Graph G_{trim} now contains all inner nodes from G with edgecount 2, and some non-inner neighbors. (d) We extract the inner nodes from the new graph, and swap every other pair of nodes (Algorithm 3, Line 43). (e) We assign volumes to all core domain points (Algorithm 3, line 45). Then, we find the volumes on the boundary domain points (a,p) by replacing the two nodes with only one edge ($\{ab\}$,$\{op\}$) with their volume minus the volume of their neighbor (Algorithm 4, line 2).

canonical ranges of a quadtree correspond to hypercubes whose side lengths are powers of two.

4.1 Reconstruction Attack

We leverage the fact that these schemes leak neighboring point search tokens. For example, in Figure 3, if a client queries for the range $[1, 2]$ using Range-URC, then she must compute search tokens corresponding to the canonical ranges $[1, 1]$ (token a) and $[2, 2]$ (token b). Our goal is to thus infer which search tokens correspond to neighboring domain points.

In order to build intuition, we make the following observations regarding the Range-URC scheme. Recall that in each dimension, the URC algorithm first computes the BRC and then recursively breaks each node into its children until there is at least one node at each level [19]. Thus, if a client queries a range q using a single token, then the range must be of size 1.

We now sketch the Token Pair attack (Algorithm 2).

(1) Let Q_1 be the set of tokensets of size 1. In Figure 2, these are the nodes in a box or highlighted.
(2) Let Q_2 be the set of tokensets of size 2, $\{t, t'\}$ such that $\{t\}, \{t'\} \in Q_1$.
(3) Initialize a graph G whose vertex set comprises of the tokens appearing in Q_1. For each $\{t, t'\} \in Q_2$, add the edge (t, t') to G.
(4) We complete the graph by mapping each search token of G to its corresponding volume. The connected component of size m in G corresponds to the ordered search tokens of the database.

Figure 2 depicts a 2D range tree and the resulting graph G from our attack. We now state a theorem summarizing the database reconstruction from the leakage of Range-URC and Quad-BRC.

Theorem 4. *Let D be a database over a d-dimensional domain of size m and let D be encrypted with the range tree scheme and uniform range cover (URC) (respectively, the quadtree and best range cover (BRC)). Given the volume map for all range queries on D, Algorithm 2 achieves full database reconstruction of D by building in $O(m^2 \log^d m)$ (respectively, $O(m^{2+\frac{d-1}{d}})$) time and space an $O(m)$-size representation of the reconstruction space of D. The input to the algorithm is available with probability greater than $1 - \frac{1}{m^2}$ after observing $O(m^2 \log m)$ uniformly distributed queries.*

4.2 Reconstruction Space

Theorem 5. *Let D be a database with domain $\mathcal{D} = [m_1] \ldots \times [m_d]$ and $\mathcal{L}$ be the leakage of the range tree scheme with URC (respectively, the quadtree scheme with BRC). The set of databases $\mathcal{L}$-equivalent to D corresponds to the symmetries of a d-cube.*

5 THE RANGE-BRC ATTACK

The canonical ranges of the Range-BRC scheme correspond to ranges in a range tree i.e., dyadic ranges. The client uses BRC to compute the tokensets. Demertzis et al [19, 20] and Falzon et al. [26] both describe schemes utilizing range trees with BRC in one and multiple dimensions, respectively. This attack applies to both constructions, in addition to any EMM constructions that use range trees with BRC and leak volume and search pattern leakage.

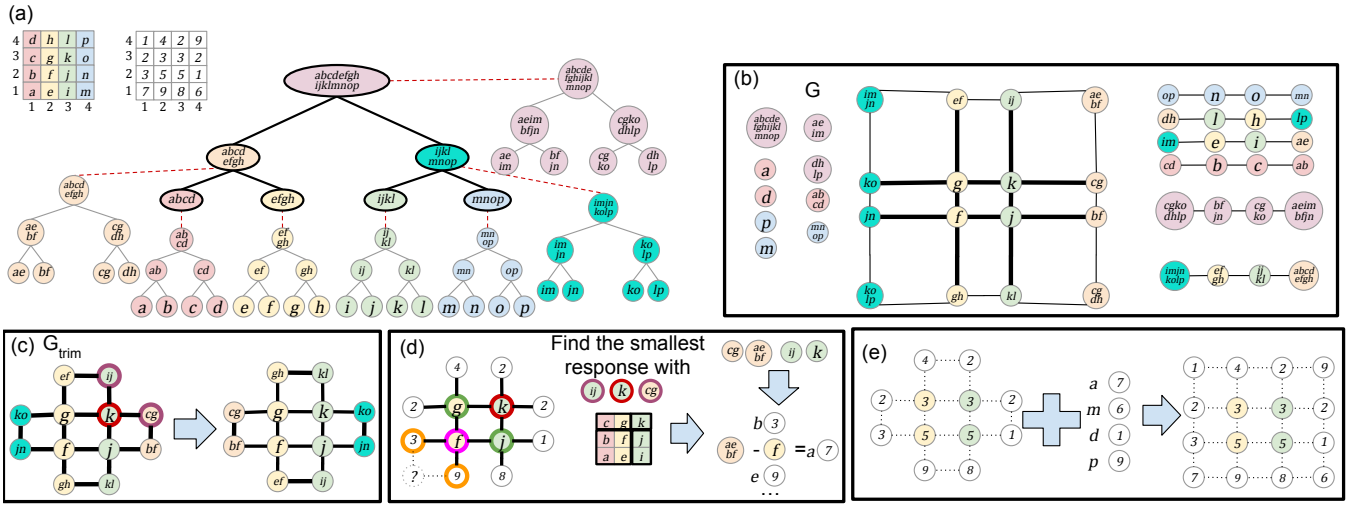

Figure 4: Attack on Range-BRC for a 2D domain (Algorithms 3 and 4). (a) Domain with volume of each point and range tree. (b) Similar to the 1D case, we create the co-occurrence graph G (Algorithm 3, Line 4). (c) We also create G_{trim} (Algorithm 3, Line 28). (d) Algorithm 4 extrapolates the volume at the boundary domain points. We can find the volumes of domain points that are extreme only in one dimension by replacing each such non-leaf node in G' with its volume minus the volume of its neighbor (Algorithm 4, Line 2). For example, we find the volume at e by subtracting the volume of f from ef. For each missing volume (domain values extreme in more than one dimensions), say the volume at a, we find node k, diagonal to a and 2 away in each dimension that a is extreme in. We then identify two neighbors of k in G_{trim}, $\{ij\}$, $\{cg\}$, such that the smallest tokenset (Algorithm 4, Line 13) containing k, $\{ij\}$ and $\{cg\}$ contains $\{aebf\}$ (which is the token corresponding to a 2x2 square that contains a). Since we know all the volumes but for a's, we can extrapolate the volume of a (Algorithm 4, Line 14). (e) We similarly identify corner node volumes (a, d, p, m), combine them with the augmented grid, and reconstruct the database.

5.1 Reconstruction Attack

We present an attack against Range-BRC that achieves polynomial run-time. This attack is more complex than the ones presented so far, as these previous attacks exploited co-occurrences of neighboring domain point tokens. However, these co-occurrences do not exist in Range-BRC, and instead, we exploit knowledge of the structure of the tree. For example, leaf node tokens appear in different patterns than non-leaf nodes. Our algorithm extracts the leaf nodes of each single-dimensional tree in the range tree, and then orders them to reconstruct the database.

Our attack is presented in Algorithms 3 and 4. Recall the definitions of core and boundary of the domain from Section 2. Algorithm 3 reconstructs the database records in the core of the domain. Notably, the reconstruction is based primarily on structure and search pattern leakage. For the core of the database, we are able to identify exactly which tokens correspond to each domain point. The volume pattern leakage is used only in the final step, to assign the number of records on each point of the core.

Reconstructing the records on the boundary of the domain requires different techniques. Algorithm 4 utilizes both volume leakage and structure pattern to determine the number of records on the boundary. This complication is due to information theoretic limitations specific to nodes with extreme values. For example, the tokens corresponding to the corners of the database never appear in a tokenset with other tokens. If they are requested by the client, they are always alone. Thus, there are no co-occurrences to exploit. Instead we use the volumes of parent nodes (in the range tree) of the corner nodes, along with their neighbors to infer the volumes in the corner nodes. However, if two corners of the database have the same volume, we cannot determine which token corresponds to which node. This is an interesting case, where we can fully reconstruct the database, but we cannot fully reconstruct the client queries i.e., determine which range corresponds to which token.

We define the following. A ***leaf node*** is a node that has no children. A ***boundary node*** corresponds to a query that covers at least one extreme domain value (nodes a and p are boundary nodes in Figure 3(a)). A node that is not a leaf or boundary node is a ***core node*** (rectangular in Figure 3(a)). The attack proceeds as follows:

(1) **Create the co-occurrence graph.** We find all distinct queries that are mapped to a tokenset of size 2 and compute a co-occurrence graph $G = (V, E)$ whose nodes V correspond to tree nodes and edges E to pairs of tokens that form a tokenset (Figure 3(b)).

(2) **Infer the core nodes.** We identify the core nodes in the range tree (rectangular) in Figure 3(a)). Given query tokensets (s_1, s_2, s_3), (s_1, s_2) and (s_2, s_3), and no query (s_1, s_3), s_2 is a core node. Identifying the core nodes helps us identify the leaf nodes, which are the nodes of the database grid.

(3) **Trim the co-occurrence graph.** Now, we want to distinguish between the boundary and leaf nodes. Observe that leaf nodes form triangular structures in G with their parent nodes (e.g. c-ab, b-c and b-cd in Figure 3(a)). We remove any edges between core nodes in G. Additionally, we use the number of times two tokens appear in a tokenset together i.e. the edgecounts, to remove any edges with edgecount more than two. This is because parents of leaf nodes have an edgecount of two with one of their children, but ancestors further up the tree have a higher edgecount. To distinguish between leaf and non-leaf nodes that look identical

in G, we use the original co-occurrence graph. After this step, we have identified all the leaf nodes, each of which correspond to a token of a single domain point. Next, we will order these leaf nodes in a grid structure corresponding to the core of the domain of the database.

(4) **Core Grid Reconstruction.** There is a component in G that contains a d-dimensional grid (Figure 3(d)) with nodes forming the dotted triangular structures. Since we know the relationship of the nodes in these structures, we can remove the core nodes (Figure 3(c)), re-order, and reconstruct the core of the database.

(5) **Inferring the extreme nodes' volumes.** We now extrapolate the volumes of the boundary domain points. In our grid structure above (Figure 3(d)), we see that there are some nodes that are core nodes. Replacing the core nodes with the core node's volume minus the volume of its neighbor, we can reconstruct the volume of these domain points. If the core nodes replace neighbors in the original co-occurrence graph G_o, then we add an edge between two such nodes. For each dimension $i \in [2, d]$ in increasing order, we identify all missing volumes on the grid that are on extreme domain values in i dimensions. For each such volume v (e.g. the corner represented by a in Fig. 4(c)), we identify the tokens that surround the i-cube of size 2^i whose corner is v. Then, we find the smallest tokenset that contains these tokens. It contains one more token corresponding to the i-cube. Since we know the volumes of all the points but v's, we can extrapolate v's volume. Once we identify all volumes of nodes on extreme domain values in i dimensions, we add the relevant grid edges, based on the new nodes' locations on the grid (if necessary).

Theorem 6. *Let D be a database over a d-dimensional domain of size m and D be encrypted with the range tree scheme and best range cover* (BRC). *Given the volume map for all range queries on D, Algorithm 3 achieves full database reconstruction of D by building in $O(m^4)$ time and $O(m^2 \log^d m)$ space an $O(m)$-size representation of the reconstruction space of D. The input is available with probability greater than $1-\frac{1}{m^2}$ after observing $O(m^2 \log m)$ uniformly distributed queries.*

5.2 Reconstruction Space

Theorem 7. *Let D be a database with domain $\mathcal{D} = [m_1] \times \cdots \times [m_d]$ and let $\mathcal{L}$ be the leakage of the range tree scheme with range covering algorithm* BRC. *The set of databases $\mathcal{L}$-equivalent to D corresponds to the symmetries of a d-cube.*

The proof for Theorem 7 is similar to the proofs of Theorems 3 and 5. The leakage can be used to determine all neighboring relationships between ranges corresponding to points of the domain, constructing a dense grid that covers the entire domain. The only possible transformations for the database correspond to the symmetries of a d-cube.

6 SRC SCHEMES ATTACK

Our SRC attack applies to a broad range of schemes that can be instantiated with the SRC algorithm – including the quadratic scheme, the TDAG-SRC scheme, and the QDAG-SRC scheme. We describe the QDAG-SRC and the quadratic scheme at the end.

Algorithm 3: *RangeTreeReconstructionBRC*(VM)

Let E, Q be the keys (tokensets) of VM of size 2 and ≥ 2, respectively.

(1) Create the co-occurrence graph.
Construct undirected graph G, whose nodes are the tokens observed, and there is an edge between any two tokens that appear as a pair in E.
Let $G_o = G$

(2) Infer the core nodes.
Initialize set core $\leftarrow \emptyset$.
Initialize table edgecounts with edgecounts$[e] = 0, \forall e \in E$.
for each tokenset $S \in Q$ **do**
  Construct subgraph G_S of G induced by the nodes of S.
  // Graph G_S is an i-dimensional grid ($1 \leq i \leq d$) (Lemma 3)
  Let C be the subset of nodes of G_S with the smallest degree in G_S.
  Let $I \leftarrow S - C$ // I is a subset of core nodes of S
  Add I to set core.
  Remove any edges $\in G_S$ from G not connected to a node in C.
  if $|C| = 2$ // We may be in a one dimensional slice. **then**
    for each edge $e \in G_S$ **do**
      edgecounts$[e] \leftarrow$ edgecounts$[e] + 1$

(3) Trim the co-occurrence graph.
// Disambiguate identical components of the graph
for all node $v \in$ core **do**
  if there is no edge e incident on v where edgecounts$[e] = 2$ **then**
    Remove node v from G
  else
    Find all neighbors of v in G with edgecounts$[(v, u)] = 2$ and remove them from G, but for one with the most edges in G_o.
Let G_{trim} be the largest component of G.

(4) Core Grid Reconstruction.
// Contract edges between remaining core nodes
for each vertex $u \in G$, where $u \in$ core **do**
  Let v, w be the neighbors of u in G.
  Add edge (v, w) to G, and remove node u from G.
// Re-order the nodes in G
for each connected subgraph H of G **do**
  // Ignore boundary nodes and make H a grid.
  Ignore any nodes with fewer than 2^d neighbors in H.
  Assign coordinates in $[2, m_1 - 1] \times \cdots \times [2, m_d - 1]$ to each vertex of H according to its position on the grid (e.g. one of the corners is assigned value $[2, \ldots, 2]$ and each remaining node is assigned the value $[a_1, a_2, \ldots, a_d]$ such that the node is at distance $a_i - 2$ from 2 in the i-th dimension.)
  for each dimension i of grid H **do**
    for one-dimensional section S of H along coordinate i **do**
      Construct subgraph G_S of H induced by the nodes of S
      Swap every other pair of nodes of G_S
    Apply any changes to G_S in G
Label the nodes of G with their volume in VM.
// We have reconstructed the core of the database

(5) Inferring the extreme nodes' volumes.
$G = FindExtremeVolumes(\text{VM}, G_o, G)$ (Algorithm 4)
return G

To generalize the attack, we leverage the notion of a range-supporting data structure from [26]. A ***range-supporting data structure*** for a domain $\mathcal{D}$ is a DAG G with a single source together

with a range covering algorithm RC. Each node of G is associated with a canonical range; we denote the canonical range of node v in G with $v.range$. The root node corresponds to the entire domain and the edges of G denote containment i.e. an edge from vertex u to v implies that $u.range$ contains $v.range$. For this attack, we require two additional assumptions on the scheme: (1) that the canonical ranges of the children partition the canonical range of the parent, and (2) that the leaves are one-to-one with the domain points.

6.1 Reconstruction Attack

SRC schemes are more difficult to attack than URC and BRC schemes, since SRC queries contain only a single (encrypted) range. This prevents us from making the same spatial connections between queries that enabled the prior attacks. In fact, Demetrzis et al. [20] conjecture that even novel attacks could not achieve full database reconstruction against their one-dimensional SRC schemes. Nevertheless, we show that we can indeed attack SRC schemes.

Let (G, SRC) be a range-supporting data structure satisfying the following two properties: (1) Every non-sink node v in G has a subset of children C, such that $\{c.range : c \in C\}$ partition $v.range$, and (2) sinks of G are 1-1 with the domain point values. Our SRC attack works on all schemes built with such a (G, RC) pair.

We construct and solve an integer linear program (ILP) whose constraints are based on the underlying DAG; The ILP is satisfied by any database in the reconstruction space, given that every possible range query has been issued exactly once. For every node v in the DAG (e.g. the QDAG) we associate a variable x_v that corresponds to the volume of $v.range$. We first write a constraint relating the volume of each non-leaf node to its children. For example, in a QDAG, the volume of a parent node v must sum to the volumes associated with the four children whose canonical ranges form quadrants of $v.range$. For each non-sink v in G we write the following constraint:

$$\boxed{x_v = \sum_{c \in C} x_c} \tag{2}$$

where C is the set of v's children whose canonical ranges partition $v.range$. Now suppose that every domain query has been issued exactly once. We can determine exactly how many unique queries correspond to each SRC node in G. We refer to this as the *frequency* of the node. Let F be the set of all frequencies. For a frequency $f \in F$, let X_f be the set of variables corresponding to nodes with frequency f, $n_f = |X_f|$, and V_f be the set of volumes with frequency f. For $f \in F$, we restrict the variables in X_f to values in V_f, since there should be a 1-1 correspondence between variables in X_f and volumes in V_f. We implement the correspondence as follows. For each $f \in F$ we define a $n_f \times n_f$ matrix of Boolean variables $b_{1,1}, b_{1,2}, \ldots, b_{n_f,n_f}$ such that each row corresponds to a variable in X_f and each column corresponds to a volume in V_f. For each $f \in F$, we then write the following constraints, where $x_s \in X_f$ and $v_t \in V_f$.

$$\boxed{x_s - \sum_{t=1}^{n_f} v_t b_{s,t} = 0; \quad \sum_{s=1}^{n_f} b_{s,t} = 1; \sum_{t=1}^{n_f} b_{s,t} = 1} \tag{3}$$

Our attack either needs to observe every range query exactly once or needs knowledge of the query distribution. Given the distribution, after observing enough queries, the adversary can deduce how many unique queries correspond to each tokenset. In the Appendix, we explain how an adversary can estimate the frequencies given a dictionary mapping each search token to the number of times it was observed and assuming that queries are issued uniformly at random. The adversary can then create constraints using Equations 2 and 3 and use a generic ILP solver to reconstruct the database.

Algorithm 4: *FindExtremeVolumes*(VM, G_o, G')

```
// Find volumes of extreme domain points
Replace any node with one neighbor in G' with its volume minus its neighbors' volume.
Add an edge between two new volume nodes, if the nodes they replaced were connected in G_o.
Let G' consist only of its largest component, a d-dimensional grid missing some nodes.
// We reconstruct the i-dimensional boundary sections in order
for i ∈ [2, d] do
    for nodes v in G' missing a volume, extreme in any i dimensions do
        Let N_v be the potential neighbors of v in G'.
        Let c be the common neighbor of N_v in G' (not v).
        Create N'_v by finding the other (leaf) neighbors of c in G' in the same dimension as each node in N_v.
        Find the other common neighbor of N'_v in G' that is not c, c'
        Create N''_v by finding the other (non-leaf) neighbors of c' in G_trim in the same dimension as each node in N'_v.
        Find the smallest key, k, in VM that contains c' and N''_v.
        Let v's volume be the sum of the volumes of all nodes in k minus the volumes of N_v, N''_v, c and c'.
    Add relevant edges for the new volume nodes based on their location on the grid in G'.
return G'.
```

Algorithm 5 takes as input VM and FM and returns grid graph G whose nodes are labeled with volumes.

Theorem 8. *Let (G, SRC) be a range-supporting data structure for a d-dimensional domain $\mathcal{D}$ such that:*

(1) each non-sink v in G has a subset of children C such that their canonical ranges, $\{c.range : c \in C\}$, are a partition of $v.range$;
(2) the sinks of G are one-to-one with the points in $\mathcal{D}$.

Let D be a database over $\mathcal{D}$ encrypted using the GenericRS *scheme from [26] with (G, SRC) and D as input and instantiated with an EMM scheme that leaks volume and search pattern. Given the volume map and frequency map for all range queries on D, where each query is issued exactly once, Algorithm 5 achieves full database reconstruction of D. The input to the algorithm is available with probability greater than $1 - \frac{1}{m^2}$ after observing $O(m^4 \log m)$ uniformly distributed queries.*

The effectiveness of Algorithm 5 depends on the size of the reconstruction space, which is determined by the underlying data structure (G, SRC) and the database D.

Our attack on SRC schemes is related to the attack by Kornaropoulos, Papamanthou, and Tamassia (KPT) [51], which approximately reconstructs a database from *one-dimensional* range queries. The KPT attack utilizes *counting functions* to determine the number of canonical ranges that return a given (encrypted) response. This information is used to build a system of equations that captures

the distance between consecutive records. In contrast, we build a system of equations representing how the volume of canonical ranges is distributed to its subranges, as given by the DAG. Our attack assumes a uniform query distribution to observe all possible queries with the same frequency and aims at full database reconstruction. The KPT attack does not assume knowledge of the query distribution and uses nonparametric estimators over a subset of the possible queries to achieve an approximate reconstruction.

6.2 QDAG-SRC

Unlike previous schemes, the QDAG-SRC and Quadratic-SRC schemes display symmetries other than those of the d-cube. We present a database that demonstrates these additional symmetries, which yields the following lower bound. The data structure underlying the QDAG-SRC scheme is a modified region quadtree called a ***quadtree-like DAG (QDAG)***. The QDAG was introduced by Falzon et al. [26] to minimize false positives when using SRC.

To build a quadtree over a square 2D domain $\mathcal{D}$, we start with a standard quadtree over $\mathcal{D}$. For any four canonical ranges $\mathcal{R}$ of the same size that form a square, we add an additional five canonical ranges of the same size: one between any two neighboring ranges in $\mathcal{R}$ and one centered at the point where the four ranges in $\mathcal{R}$ meet. In Figure 9 (a) we see an example of a set of canonical ranges $\mathcal{R}$ and in (b) we see the additional five ranges overlapping them.

Theorem 9. *Let D be a dense database with domain $\mathcal{D} = [m_1] \times \cdots \times [m_d]$ and let $\mathcal{L}$ be the leakage of* QDAG-SRC. *Let $S_{\mathcal{L}}$ be the set of databases $\mathcal{L}$-equivalent to D. We have $|S_{\mathcal{L}}| \geq 2^{d+2^{(d-1)}}(d)(d!)$.*

Before demonstrating a lower bound for the size of the reconstruction space of the QDAG-SRC scheme, we first build intuition with an example of a Quadtree-SRC scheme, i.e., a scheme whose underlying data structure is a quadtree and whose range cover algorithm is SRC. Recall that a *region quadtree* is a tree that recursively partitions a square domain into 2^2 quadrants. Each non-leaf node v has four children; each child of v is associated with a quadrant of $v.range$. Using SRC for a quadtree results in a false positive rate of $O(m)$. As we will see, the Quadtree-SRC scheme – in contrast to the QDAG-SRC scheme – is more secure at the expense of significantly more false positives. We now state a lemma about the complexity of the ILP needed to attack the QDAG-SRC scheme.

Lemma 1. *Let D be a database over domain $\mathcal{D}$ and* EDB *be the encrypted database resulting from encrypting D with the* QDAG-SRC *scheme. Let* VM *and* FM *be the volume map and frequency map constructed from the query leakage of* EDB. *On input of* VM *and* FM, *Algorithm 5 builds and solves an ILP of size $O(\Omega(dm))$.*

6.3 Quadratic-SRC Scheme

One notable SRC scheme is the quadratic scheme [19, 25], or the Quadratic-SRC scheme. This scheme stores a key-value pair for every possible range query. A quadratic scheme over a d-dimensional domain $\mathcal{D}$ can be represented as an range-supporting data structure, (G, SRC), where G is a DAG described as follows. Let Q be the set of all possible d-dimensional range queries over $\mathcal{D}$ and associated with each range a node of G; the nodes are one-to-one with the ranges in Q. For each pair of nodes u, v add an edge from u to v if and only if $u.range$ minimally contains $v.range$. Since Q contains all the single point ranges and no other range is minimally contained by the single point ranges it is straightforward to see that the leaves are indeed one-to-one with the domain points.

Algorithm 5: *GenericReconstructionSRC*(VM, FM)

1: Let max be the maximum volume in FM.
2: Let F be the set of frequencies in FM.
3: Let G be the underlying DAG and for each node $v \in G$, create integer ILP variable x_v with bounds $[0, \max]$.
4: **for** non-leaf node $v \in G$ **do** add Equation 2 to the ILP.
5: **for** $f \in F$ **do** add Equations 3 to the ILP.
6: Run the ILP solver to retrieve assignment A.
7: Let H be a grid corresponding to the tokens forming leaves of the tree.
8: Label the nodes of H with their volume in VM.
9: **return** H

We further claim that G is a unique DAG. To see that it is a DAG, suppose for a contradiction that G is not a DAG. Then it must contain a cycle $v_1, v_2, \ldots, v_k, v_1$. But this means that $v.range$ minimally contains $v_2.range$ and $v_2.range$ minimally contains $v_3.range$. Extending this logic, we see that $v_1.range$ must minimally contain $v_1.range$, which is a contradiction. To prove uniqueness, suppose that construction of the DAG for a domain $\mathcal{D}$ resulted in two distinct graphs G and G'. These graphs must differ in the existence of at least one edge; WLOG suppose that the edge (u, v) exists in G and not in G'. By construction, the fact that (u, v) is an edge in G implies that $u.range$ minimally contains $v.range$. But since an edge (u, v) exists if and only if $u.range$ minimally contains $v.range$ this means that we would have also added this edge in the construction of G', hence a contradiction.

Since G is a DAG satisfying all the conditions described in Theorem 8, it follows that a database encrypted with the quadratic scheme can be reconstructed with our SRC attack. This scheme has been attacked using access and search pattern leakage in one-dimensions [36, 47, 52] and in two-dimensions [25, 54]. Additionally, the quadratic scheme has been attacked using volume pattern in one-dimensions [37, 39, 47]. However, volume-based attacks in the multi-dimensional setting remain an open problem.

Our SRC attack can utilize volume and search pattern leakage to perform a database reconstruction attack on the quadratic scheme on databases of arbitrary dimensions. However, we note that to launch the attack, one would require computational resources that we do not have. Since our attack is based on solving an ILP, it can be fully parallelized; there are many tools available that solve ILPs and exploit parallelization [40, 60]. Thus, we conjecture that even the quadratic scheme in multiple dimensions is vulnerable to a powerful adversary. We now state the following lower bound about the reconstruction space of the quadratic-SRC scheme.

Theorem 10 ([25, 54]). *Let D be a database with domain $\mathcal{D} = [m_1] \times \cdots \times [m_d]$ containing n points and let $\mathcal{L}$ be the leakage of the quadratic-*SRC *scheme with range covering algorithm* SRC. *Let $S_{\mathcal{L}}$ be the set of databases $\mathcal{L}$-equivalent to D. We have $|S_{\mathcal{L}}| \geq 2^n$, for $d > 1$ and $|S_{\mathcal{L}}| = 2$ for $d = 1$.*

Theorem 10 follows from the reconstruction space presented in [25, 54] on two-dimensional databases. We conclude this section

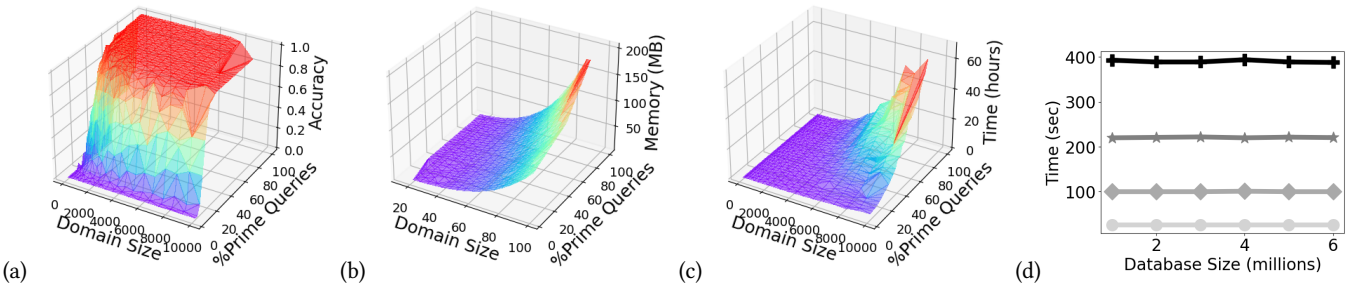

Figure 5: (a) Accuracy, (b) memory, and (c) runtime of our linear attack for 2D databases of different domain sizes, after observing different percentages of prime-size queries. (d) Runtime of the attack on Gowalla, varying the number of records (in millions) and percent of prime-size queries (25% (lightgray circle •), 50% (gray diamond ♦) , 75% (darkgray star ⋆), plus sign 100% (black +)).

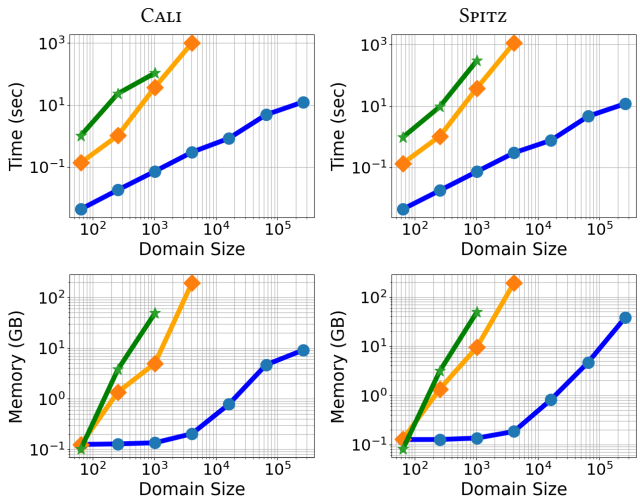

Figure 6: Median runtime in seconds (top) and median memory usage (GB) (bottom) of our attacks on the range-URC (blue circle •), range-BRC (orange diamond ♦), and QDAG-SRC (green star ⋆) schemes for Cali and Spitz on different domain sizes.

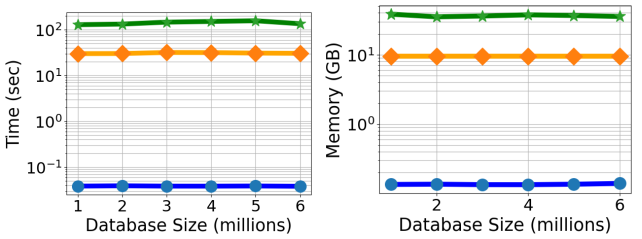

Figure 7: Runtime (left) in seconds and memory requirement (right) in GB of our attacks on the range-URC (blue circle •), range-BRC (orange diamond ♦), and QDAG-SRC (green star ⋆) schemes for the Gowalla $[2^5] \times [2^5]$ dataset varying the number of records (in millions).

by stating a lemma about the size of the ILP needed to attack the Quadratic-SRC scheme.

Lemma 2. *Let D be a database over domain $\mathcal{D}$ and* EDB *be the encrypted database resulting from encrypting D with the* Quadratic-SRC *scheme. Let* VM *and* FM *be the volume map and frequency map constructed from the query leakage of* EDB*. On input of* VM *and* FM*, Algorithm 5 builds and solves an ILP of size $O(\Omega(dm^2))$.*

7 EXPERIMENTS

We experimentally evaluate the performance of our attacks using the following real-world datasets:

Cali [53]: 21,047 lat-long points of California road intersections. It was used in a prior attack [54].

Spitz [65] 28,837 latitude-longitude points of phone location data of politician Malte Spitz between August 2009 and February 2010. It was used in several previous attacks [25, 50, 54].

Gowalla[15]: 6,442,892 latitude-longitude points from users of the Gowalla social networking website between 2009 and 2010, a dataset used in the experiments by Demertzis et al. [19]. We further replicate Demertzis et al.'s Gowalla experiments by randomly partitioning the dataset into 10 sets, each consisting of 500,000 records. We measure the indexing time and cost of our schemes by increasing the domain size by a new set of 500,000 tuples.

7.1 Results

We performed experiments on our attacks on the Linear, Range-URC, Range-BRC, and QDAG-SRC schemes. Our attacks always returned the original database up to the symmetries of a d-cube, when given the complete leakage as input including our reconstructions of databases encrypted with the QDAG-SRC. For simplicity, we considered domains with all dimensions of the same size (i.e., 2D square grids and 3D cube grids).

Figure 5 displays our results for the linear attack. In Figure 5(a) we show the median accuracy of our attack against 2D databases of different domain sizes, after observing different percentages of prime queries. We measure the accuracy of our attack as the percent of correctly reconstructed domain point volumes. This attack achieves great accuracy with a relatively small percent of prime queries. The attack requires little storage space (Figure 5(b)), but as the domain size increases so does the runtime (Figure 5(c)). We also ran our attack against a $[2^5] \times [2^5]$ Gowalla dataset, varying the number of records from 1 million to 6 million. The runtime is only affected by the domain size. For 2D ranges, we

could transform the leakage into access and search pattern leakage and input it to the approximate database reconstruction attack on 2D databases from [54]. However, this results in loss of information, making the reconstruction space potentially exponentially larger.

We also ran our attacks on the Range-URC, Range-BRC and QDAG-SRC schemes. We used Spitz and Cali normalized at different domain sizes to demonstrate our attacks in Figure 6. We observe elbows in our plots as we increase the domain size. We believe this is caused by variance in the speed of our computing grid between machines with higher versus lower memory. The QDAG-SRC attack took the longest time and generally required more memory, since it involved solving an ILP. The Range-URC attack was the most efficient. We also ran these attacks against the Gowalla dataset to observe how the number of records affects the runtime and memory (Figure 7). We observed that the attacks are generally not affected. The QDAG-SRC attack against the Gowalla dataset (Figure 7) shows some random variance in the runtime and memory needed. We believe this is due to randomness in the solver that causes it to take different search paths on each execution.

Our attacks, with the exception of the linear attack, require the adversary to observe all queries at least once. In Figure 8, we experimentally show how many queries the adversary needed to sample under the Uniform, Gaussian(1/2,1/5), and Beta(1,1.2) distributions over different database domain sizes until they observed each query at least once. We also plot the Baseline, which depicts the number of possible queries per domain size. The number of queries needed to perform each attack scales with the size of the domain. We note that even as the domain size increases, the adversary needs a similar number of queries across all distributions.

Implementation Details. We implemented our URC, BRC, and SRC attacks in Python 3.9.2; we implemented the linear attack in C++ using a library for PQ-trees [34]. We ran all of our experiments on a compute cluster. For simplicity, we used the same compute node for the client and the server; our results do not include any network transmission latency.

For cryptographic primitives, we used the Python cryptography library version 3.4.7 [62]. To match the evaluation of Demertzis et al. [19], we used SHA-512 for PRFs and AES-CBC (with 128-bit block size) for encryption. For our underlying EMM scheme, we implemented Π_{bas} from Cash et al. [11]. We used the CP-SAT solver from Google's ortools package [60] as our ILP solver.

8 TAKEAWAYS

8.1 General Techniques

We describe a number of new combinatorial and linear programming techniques that apply to the non-interactive 1D range search schemes in [19, 24] and the non-interactive multi-dimensional range search schemes in [26]. One key property that we leverage in the Linear, URC, and BRC attacks is token co-occurrences. Our general approach to capturing token co-occurrences is to encode the relationships using graphs; in these graphs, nodes correspond to tokens, and edges correspond to pairs of tokens that appear together in (certain) tokensets. These graphs capture important information about the underlying data structure, from which we can reconstruct the data. One important piece of information we are able to extract from such graphs is ***linear substructures***, the one-dimensional

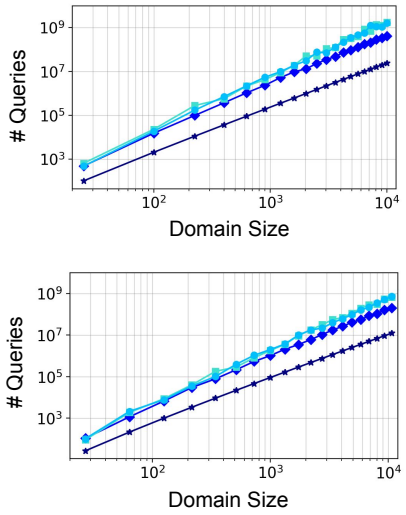

Figure 8: (Left) 2D and (Right) 3D number of queries sampled under the Uniform (blue diamond ♦), Beta(1,1.2) (turquoise square ■), and Gaussian(1/2,1/5) (skyblue circle •) distributions until all unique queries are observed vs. domain size. Also shown is the Baseline (navy star ⋆), i.e., the total number of possible queries.

substructures of the underlying data structure. Once the linear substructures have been extracted, we can piece them together to re-create the multi-dimensional database. In our Linear attack, this corresponds to constructing the one-dimensional sections (lines 1-10 of Algorithm 1). In our Range-BRC attack, this corresponds to identifying potential one-dimensional slices (lines 14-16 of Algorithm 3). SRC schemes, on the other hand, do not leak co-occurrence information and we thus describe a different approach for attacking SRC schemes. Our ILP attack is SRC scheme agnostic; it works effectively against all SRC schemes satisfying two very standard characteristics: (1) for each node v, the canonical ranges of (potentially a subset of) the children of v partition $v.range$; and (2) the sinks are one-to-one with the domain points.

8.2 Comparing our Attacks to Prior Work

Prior attacks (e.g., [25, 47, 52, 54]) consider a "generic" leakage that is implementation-independent. This leakage is a common-denominator leakage found in most efficient schemes and can concretely be attributed to the quadratic scheme – a theoretical scheme that stores one key-value pair for each range and which requires too much storage for most practical scenarios. Our attacks leverage implementation-specific leakage of concrete response-hiding schemes presented in the literature.

8.3 Extending our Attacks

The generality of our techniques enables us to apply our attacks to multiple schemes. All the schemes described by Demertzis et al. [19], Faber et al. [24] and Falzon et al. [26] leak either token co-occurrences or use SRC as a range cover; in fact, we are able to attack all six schemes in [26]. Our attack against Range-URC is applicable to schemes that leak information about neighboring

points, e.g., Quad-BRC. Similarly, our SRC attack is applicable to schemes that return a single range cover and satisfy two standard characteristics, such as QDAG-SRC and Quadratic-SRC. One could describe other schemes, such as the quadratic scheme with BRC. Since every possible range in the domain is itself a canonical range, then the best range cover always results in a single token. Generalizing the BRC attack to other data structures beyond range trees remains an open question. We conjecture that the techniques we developed, such as identification of the inner nodes of the tree, could be extended to other data structures.

8.4 Structure vs. Access Pattern

Access pattern is considered to leak more information than search or volume pattern, as it reveals co-occurrences in the responses, allowing an adversary to determine which (encrypted) records appear in a range together. In this paper, we show that structure pattern, along with volume pattern can be as (if not more) dangerous as access pattern. Consider a database that supports two-attribute range queries. If the database leaks access and search pattern, then the reconstruction space is exponential to the number of records $O(2^n)$ [25, 54]. We show that all concrete schemes from Falzon et al. [26] that do not have false positives have a reconstruction space of size at most 8 in two dimensions. Structure pattern can be thought of as access pattern for the nodes of the underlying data structure.

8.5 Mitigations

There are many techniques that could mitigate these attacks at the expense of some performance. The main technique used in these attacks (besides the SRC attack) is the exploitation of token co-occurrences. To reduce the effectiveness of this technique, the client could batch their queries, sending two or more queries at a time. This way, the adversary cannot be certain of which tokens correspond to the same query (unless more queries are observed).

Note, if we know that records a and b are neighbors and that records b and c are neighbors, we can infer that a and c are also close. The client could disallow such neighboring queries, thus restricting the adversary's ability to reconstruct local information.

Our linear attack specifically uses prime queries to extract one-dimensional information. To prevent this, the client could round up prime-sized queries. Generally, our attacks, with the exception of the SRC attack, depend on the existence of linear substructures (1D ranges) in the multi-dimensional space. Our linear attack finds all prime-sized queries because they all correspond to 1D sections of the database. The URC attack identifies 1D sections of size 2. The BRC attack also depends on utilizing 1D sections. A general mitigation technique would be to only return d-dimensional range queries i.e., ranges with at least two domain points in each dimension.

Toward mitigating the SRC attack, the client could take advantage of the large amount of queries required by the attack and periodically rebuild the EDB. This would hinder the adversary's progress, but not defend against the attack completely since the adversary may be able to store and reuse previously inferred information. This attack could also be mitigated by adding false records to the responses. Other mitigation techniques include frequency smoothing [35, 56] and oblivious data structures (e.g. [18]). We leave studying the effectiveness of these mitigations to future work.

9 CONCLUSION

We are the first to systematically explore structure pattern leakage from range queries, which is inherent to any efficient range search scheme for an encrypted database. We show that along with search and volume pattern leakage, structure pattern leakage can be as dangerous as access pattern leakage, as demonstrated by the relative sizes of the reconstruction spaces. We present the first attacks on range search schemes for databases with arbitrary dimensions. Our attacks achieve full database reconstruction even on SRC schemes, which were previously considered very secure. Our attacks prompt the exploration of mitigation techniques such as frequency smoothing [35], rounding the ranges to a specified integer multiple [56], batching queries, periodic rebuilding, avoiding one-dimensional queries, oblivious data-structures and alternate range decomposition approaches.

ACKNOWLEDGMENTS

Work supported in part by the National Science Foundation, the Kanellakis Fellowship at Brown University, and a gift from the NetApp University Research Fund, a corporate advised fund of Silicon Valley Community Foundation. The authors would also like to thank William Schor for his preliminary contributions to the implementation of the schemes. Part of this research was conducted using computational resources of the Center for Computation and Visualization at Brown University.

REFERENCES

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. 2004. Order Preserving Encryption for Numeric Data. In *Proc. ACM SIGMOD International Conference on Management of Data* (Paris, France) *(SIGMOD)*. 12 pages.

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

[3] 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.

[4] Laura Blackstone, Seny Kamara, and Tarik Moataz. 2020. Revisiting Leakage Abuse Attacks. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society.

[5] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. 2009. Order-Preserving Symmetric Encryption. In *Advances in Cryptology - EUROCRYPT 2009*. Berlin, Heidelberg.

[6] K.S. Booth and G. S. Lueker. 1976. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of computer and system sciences* 13, 3 (1976).

[7] Raphael Bost. 2016. Sophos: Forward Secure Searchable Encryption. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS '16)*. Association for Computing Machinery, New York, NY, USA, 1143–1154. https://doi.org/10.1145/2976749.2978303

[8] Raphaël Bost, Brice Minaud, and Olga Ohrimenko. 2017. Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 1465–1482. https://doi.org/10.1145/3133956.3133980

[9] Cynthia Braund and Pramod Borkar. 2022. MongoDB Releases Queryable Encryption Preview. https://www.mongodb.com/blog/post/mongodb-releases-queryable-encryption-preview.

[10] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. 2015. Leakage-Abuse Attacks Against Searchable Encryption. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (Denver, Colorado, USA) *(CCS '15)*. Association for Computing Machinery, New York, NY, USA, 668–679. https://doi.org/10.1145/2810103.2813700

[11] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael 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*. The Internet Society, Reston,

VA, USA. https://www.ndss-symposium.org/ndss2014/dynamic-searchable-encryption-very-large-databases-data-structures-and-implementation
[12] 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.
[13] Javad Ghareh Chamani, Dimitrios Papadopoulos, Mohammadamin Karbasforushan, and Ioannis Demertzis. 2022. Dynamic Searchable Encryption with Optimal Search in the Presence of Deletions. In *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, Kevin R. B. Butler and Kurt Thomas (Eds.). USENIX Association, 2425–2442.
[14] 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)*. Springer International Publishing, Cham.
[15] 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)*. Association for Computing Machinery, New York, NY, USA, 1082–1090. https://doi.org/10.1145/2020408.2020579
[16] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. 2006. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security* (Alexandria, Virginia, USA) *(CCS '06)*. Association for Computing Machinery, New York, NY, USA, 79–88. https://doi.org/10.1145/1180405.1180417
[17] Ioannis Demertzis, Javad Ghareh Chamani, Dimitrios Papadopoulos, and Charalampos Papamanthou. 2020. Dynamic Searchable Encryption with Small Client Storage. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society.
[18] Ioannis Demertzis, Dimitrios Papadopoulos, Charalampos Papamanthou, and Saurabh Shintre. 2020. SEAL: Attack Mitigation for Encrypted Databases via Adjustable Leakage. In *USENIX Security Symposium*. 2433–2450.
[19] Ioannis Demertzis, Stavros Papadopoulos, Odysseas Papapetrou, Antonios Deligiannakis, and Minos Garofalakis. 2016. Practical Private Range Search Revisited. In *Proceedings of the 2016 International Conference on Management of Data* (San Francisco, California, USA) *(SIGMOD '16)*. Association for Computing Machinery, New York, NY, USA, 185–198. https://doi.org/10.1145/2882903.2882911
[20] 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
[21] Ioannis Demertzis, Charalampos Papamanthou, and Rajdeep Talapatra. 2018. Efficient Searchable Encryption Through Compression. *Proc. VLDB Endow.* 11, 11 (2018), 1729–1741. https://doi.org/10.14778/3236187.3236218
[22] F. Betül Durak, Thomas M. DuBuisson, and David 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)*. Association for Computing Machinery, New York, NY, USA, 1155–1166. https://doi.org/10.1145/2976749.2978379
[23] Zachary Espiritu, Evangelia Anna Markatou, and Roberto Tamassia. 2022. Time- and Space-Efficient Aggregate Range Queries over Encrypted Databases. *Proc. Priv. Enhancing Technol.* 2022, 4 (2022), 684–704. https://doi.org/10.56553/popets-2022-0128
[24] Sky Faber, Stanislaw Jarecki, Hugo Krawczyk, Quan Nguyen, Marcel Rosu, and Michael Steiner. 2015. Rich Queries on Encrypted Data: Beyond Exact Matches. In *Computer Security – ESORICS 2015*, Günther Pernul, Peter Y A Ryan, and Edgar Weippl (Eds.). Springer International Publishing, Cham, 123–145.
[25] Francesca Falzon, Evangelia Anna Markatou, Akshima, David Cash, Adam Rivkin, Jesse Stern, and Roberto Tamassia. 2020. Full Database Reconstruction in Two Dimensions. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, USA) *(CCS '20)*. Association for Computing Machinery, New York, NY, USA, 443–460. https://doi.org/10.1145/3372297.3417275
[26] Francesca Falzon, Evangelia Anna Markatou, Zachary Espiritu, and Roberto Tamassia. 2022. Range Search over Encrypted Multi-Attribute Data. *Proc. VLDB Endow.* 16, 4, 587–600.
[27] Francesca Falzon and Kenneth G. Paterson. 2022. An Efficient Query Recovery Attack Against a Graph Encryption Scheme. In *Computer Security – ESORICS 2022*, Vijayalakshmi Atluri, Roberto Di Pietro, Christian D. Jensen, and Weizhi Meng (Eds.). Springer International Publishing, Cham, 325–345.
[28] R. A. Finkel and J. L. Bentley. 1974. Quad Trees a Data Structure for Retrieval on Composite Keys. *Acta Informatica* 4, 1 (mar 1974), 1–9.
[29] C. Gentry. 2009. *A Fully Homomorphic Encryption Scheme*. Ph.D. Dissertation. Stanford University, Stanford, CA, USA. Advisor(s) Boneh, D.
[30] Marilyn George, Seny Kamara, and Tarik Moataz. 2021. Structured Encryption and Dynamic Leakage Suppression. In *Advances in Cryptology – EUROCRYPT 2021*, Anne Canteaut and François-Xavier Standaert (Eds.).
[31] J. Ghareh Chamani, D. Papadopoulos, C. Papamanthou, and R. Jalili. 2018. New Constructions for Forward and Backward Private Symmetric Searchable Encryption. In *Proc. ACM Conf. on Computer and Communications Security* (Toronto, Canada) *(CCS '18)*. New York, NY, USA, 18 pages.
[32] Esha Ghosh, Seny Kamara, and Roberto Tamassia. 2021. Efficient Graph Encryption Scheme for Shortest Path Queries. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security* (Virtual Event, Hong Kong) *(ASIA CCS '21)*. Association for Computing Machinery, New York, NY, USA, 516–525. https://doi.org/10.1145/3433210.3453099
[33] O. Goldreich and R. Ostrovsky. 1996. Software Protection and Simulation on Oblivious RAMs. *J. ACM* 43, 3 (May 1996), 43 pages.
[34] Greg Grothaus. 2011. PQTrees. https://github.com/Gregable/pq-trees. Accessed: 2022-01-12.
[35] 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 *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Berkeley, CA, USA, 2451–2468. https://www.usenix.org/conference/usenixsecurity20/presentation/grubbs
[36] 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)*. New York, NY, USA.
[37] Paul Grubbs, Marie-Sarah Lacharite, Brice Minaud, and Kenneth G. Paterson. 2018. Pump up the Volume: Practical Database Reconstruction from Volume Leakage on Range Queries. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (Toronto, Canada) *(CCS '18)*. Association for Computing Machinery, New York, NY, USA, 315–331. https://doi.org/10.1145/3243734.3243864
[38] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart. 2017. Leakage-Abuse Attacks against Order-Revealing Encryption. In *Proc. IEEE Symp. on Security and Privacy (SP)*. New York, NY, USA.
[39] Z. Gui, O. Johnson, and B. Warinschi. 2019. Encrypted Databases: New Volume Attacks against Range Queries. In *Proc ACM Conf. on Computer and Communications Security (CCS)*.
[40] Gurobi Optimization, LLC. 2022. Gurobi Optimizer Reference Manual. https://www.gurobi.com
[41] F. Hahn and F. Kerschbaum. 2016. Poly-Logarithmic Range Queries on Encrypted Data with Small Leakage. In *Proc. ACM Cloud Computing Security Workshop* (Vienna, Austria) *(CCSW)*. 12 pages.
[42] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. 2012. Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation. In *Proc. Annual Network and Distributed System Security Symposium (NDSS)*. The Internet Society.
[43] Seny Kamara, Abdelkarim Kati, Tarik Moataz, Thomas Schneider, Amos Treiber, and Michael Yonli. 2022. SoK: Cryptanalysis of Encrypted Search with LEAKER – A framework for LEakage AttacK Evaluation on Real-world data. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*. 90–108. https://doi.org/10.1109/EuroSP53844.2022.00014
[44] Seny Kamara and Tarik Moataz. 2019. Computationally Volume-Hiding Structured Encryption. In *Advances in Cryptology - EUROCRYPT - Part II (Lecture Notes in Computer Science, Vol. 11477)*. Springer, 183–213.
[45] S. Kamara and C. Papamanthou. 2013. Parallel and Dynamic Searchable Symmetric Encryption. In *Financial Cryptography and Data Security*. Berlin, Heidelberg.
[46] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. 2012. Dynamic Searchable Symmetric Encryption. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security* (Raleigh, North Carolina, USA) *(CCS '12)*. Association for Computing Machinery, New York, NY, USA, 965–976. https://doi.org/10.1145/2382196.2382298
[47] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. 2016. Generic Attacks on Secure Outsourced Databases. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS '16)*. Association for Computing Machinery, New York, NY, USA, 1329–1340. https://doi.org/10.1145/2976749.2978386
[48] Evgenios M. Kornaropoulos, Nathaniel Moyer, Charalampos Papamanthou, and Alexandros Psomas. 2022. Leakage Inversion: Towards Quantifying Privacy in Searchable Encryption. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (Los Angeles, CA, USA) *(CCS '22)*. Association for Computing Machinery, New York, NY, USA, 1829–1842. https://doi.org/10.1145/3548606.3560593
[49] E. M. Kornaropoulos, C. Papamanthou, and R. Tamassia. 2019. Data Recovery on Encrypted Databases with k-Nearest Neighbor Query Leakage. In *Proc. IEEE Symp. on Security and Privacy (S&P)*.
[50] 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)*.
[51] 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)*.

[52] Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson. 2018. Improved Reconstruction Attacks on Encrypted Data Using Range Query Leakage. In *2018 IEEE Symposium on Security and Privacy (SP)*. 297–314. https://doi.org/10.1109/SP.2018.00002

[53] 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.

[54] 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.

[55] Evangelia Anna Markatou and Roberto Tamassia. 2019. Full Database Reconstruction with Access and Search Pattern Leakage. In *Proc. Int. Conf. on Information Security (ISC) (Lecture Notes in Computer Science)*. Springer.

[56] Evangelia Anna Markatou and Roberto Tamassia. 2019. Mitigation Techniques for Attacks on 1-Dimensional Databases that Support Range Queries. In *Proc. Int. Conf. on Information Security (ISC) (Lecture Notes in Computer Science, Vol. 11723)*. Springer.

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

[58] Simon Oya and Florian Kerschbaum. 2021. Hiding the Access Pattern is Not Enough: Exploiting Search Pattern Leakage in Searchable Encryption. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 127–142. https://www.usenix.org/conference/usenixsecurity21/presentation/oya

[59] Sarvar Patel, Giuseppe Persiano, Kevin Yeo, and Moti Yung. 2019. Mitigating Leakage in Secure Cloud-Hosted Data Structures: Volume-Hiding for Multi-Maps via Hashing. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. Association for Computing Machinery, New York, NY, USA, 79–93.

[60] Laurent Perron and Vincent Furnon. 2019. OR-Tools version 7.2. Google. https://developers.google.com/optimization/.

[61] David Pouliot and Charles V. Wright. 2016. The Shadow Nemesis: Inference Attacks on Efficiently Deployable, Efficiently Searchable Encryption. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS '16)*. Association for Computing Machinery, New York, NY, USA, 1341–1352.

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

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

[64] Dawn Xiaodong Song, David A. Wagner, and Adrian Perrig. 2000. Practical Techniques for Searches on Encrypted Data. In *2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 14-17, 2000*. IEEE Computer Society, New York, NY, USA, 44–55. https://doi.org/10.1109/SECPRI.2000.848445

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

[66] Boyang Wang, Yantian Hou, Ming Li, Haitao Wang, and Hui Li. 2014. Maple: Scalable Multi-Dimensional Range Search over Encrypted Cloud Data with Tree-Based Index. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security* (Kyoto, Japan) *(ASIA CCS '14)*. Association for Computing Machinery, New York, NY, USA, 111–122. https://doi.org/10.1145/2590296.2590305

[67] Jiafan Wang and Sherman SM Chow. 2022. Forward and Backward-Secure Range-Searchable Symmetric Encryption. *Proceedings on Privacy Enhancing Technologies* 1 (2022), 28–48.

[68] 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.

A PROOFS

A.1 Proof of Theorem 1

Proof. It is straightforward to see how one can build VM and FM from the multiset $\{\{\mathsf{SP}(\mathsf{MM}, q^{(i)})\}\}_{i\in[k]}$. To show the reverse, we construct the structure pattern using VM and FM. Initialize an empty multiset S. For each tokenset $\mathbf{t}^{(i)} \in \mathsf{VM}$:

(1) Initialize an empty dictionary M.
(2) For each $t \in \mathbf{t}^{(i)}$ set $M[t] \leftarrow \mathsf{VM}[t]$.
(3) $f \leftarrow \mathsf{FM}[\mathbf{t}^{(i)}]$
(4) Add f copies of M to the multiset S.

Each issued query $q^{(i)}$ corresponds to a tokenset $\mathbf{t}^{(i)}$ i.e. the search pattern of the canonical ranges that cover q. VM associates each tokenset with the observed volume i.e. the volume pattern of the response. Since each map M in S is added as many times as the corresponding tokenset has been observed, then the structure pattern multiset is in one-to-one correspondence with the multiset S. □

A.2 Proof of Theorem 2

Proof. The first step of Algorithm 1 is to find any queries that correspond to a set of search tokens of prime size, say set Q. The size of the range being queried is leaked, as it is the number of search tokens the client sends the server. If a range has prime size p, then the query covers p points in one dimension and one point in the remaining dimensions. Thus, all queries in Q query are 1D sections. The next step is to group queries that come from the same one-dimensional section. Note that if two queries' search token intersection contains two or more elements, then the queries must correspond to ranges along the same one-dimensional section. We thus group queries in their corresponding one-dimensional section, and create a PQ-tree for each one-dimensional section. The attack then generates a graph G that contains an edge between neighboring search tokens, representing a partial order reconstruction of the search tokens. Once we map the search tokens to their corresponding volumes, we achieve partial database reconstruction. If the adversary has observed enough queries for the order of the individual one-dimensional sections to be fully reconstructed, graph G is a d-dimensional grid fully ordering all search tokens, and thus achieving full database reconstruction.

To achieve FDR, every PQ-tree must have enough information to reconstruct the order of each one-dimensional section. Consider a one-dimensional section, e.g. a row R. The search tokens in R share all values but one, the one corresponding to the first dimension. Thus, their values span from 1 to m_1. Split the search tokens in two groups: A includes all search tokens with values less than $m_1/2$ in the first dimension and B contains the remaining points in R. The PQ-tree can order these search tokens if in its input there exists a range that starts before and a range that starts after every search token. Thus, if the PQ-tree observes a range that starts before every point in A or ends before every point in B or after the last point of B, it can fully order the search tokens in R. Let's count the number of range queries that start at a specific point in A, end anywhere in B and have prime length. There are more small prime numbers than larger. Thus, the worst case scenario is our starting point being in the beginning of A. Thus, the possible size of our range is between $m_1/2$ and m_1. We approximate the number of prime numbers between $m_1/2$ and N_1 to be around $\frac{m_1}{\log m_1} - \frac{m_1/2}{\log m_1/2} > \frac{m_1/6}{\log m_1/6}$, for $m_1 > 26$. Thus, the probability that a range query satisfies these constraints is $\frac{1}{m_1 \log m_1 m_2^2 \ldots m_d^2}$. Let $x = m_1 \log m_1 m_2^2 \ldots m_d^2$. After observing $10x \log x$ queries, then we will not have observed even one query satisfying the constraints with probability $(1 - 1/x)^{10x \log x} \approx \frac{1}{x^{10}}$. There are $m_1/2$ such queries from A and $m_1/2$ similar such queries from B. By union bound, the probability that even one of them is missing is approximately $\frac{m_1}{(m_1 \log m_1 m_2^2 \ldots m_d^2)^{10}} \leq \frac{1}{m^5}$. There are fewer than m rows, thus the

probability we missed one of them is $\leq \frac{1}{m^4}$. We can make a similar argument for each PQ tree, concluding that if the adversary observes $\sum_{i=1}^{d} \Omega\left(\frac{m^2}{m_i} \log m_i \cdot \log\left(\frac{m^2}{m_i} \log m_i\right)\right)$ queries, Algorithm 1 achieves FDR with probability greater than $1 - \frac{1}{m^2}$.

The volume map contains $O(m^2)$ entries and in the worst case the entries comprise of $O(m)$ tokens and volumes. The algorithm thus requires $O(m^3)$ storage. Algorithm 1 first identifies all tokensets of prime size which takes $O(m^3)$ time. The Algorithm then identifies which tokens correspond to the same one-dimensional slice. This requires a loop over all $O(m^2)$ tokensets and on each loop doing a set intersection between sets of size $O(m)$, $O(m^2)$ times. Thus, it takes $O(m^5)$ time. We then construct a PQ tree for each one-dimensional slice and create the augmented graph G, which takes $O(m^2)$ time. Thus, Algorithm 1 takes $O(m^5)$ time, $O(m^3)$ space and succeeds with probability greater than $1 - \frac{1}{m^2}$ after observing $\sum_{i=1}^{d} \Omega\left(\frac{m^2}{m_i} \log m_i \cdot \log\left(\frac{m^2}{m_i} \log m_i\right)\right)$ queries uniformly at random. □

A.3 Proof of Theorem 3

Proof. Consider a token t and its neighboring tokens $t_0, t'_0, t_1, t'_1, \ldots, t_d, t'_d$. There exists a range query that issues t with each one of its neighboring search tokens. There exists no query of size two with t that does not contain one of its neighbors as that query would not correspond to a valid range. Combining all these queries, we can construct a grid that covers the entire domain. This grid is dense and does not allow for reflectable components as in [25]. Thus the reconstruction space only includes transformations of D corresponding to the symmetries of a d-cube. □

A.4 Proof of Theorem 4

Proof. Rangetree with URC. Recall that URC "starts with the set of nodes output by BRC, and keeps on breaking certain nodes into their two children, until there is at least one node for each level $0, \ldots, max$, where max is the highest level of nodes in the result" [19]. The attack constructs a graph G that orders the search tokens of ranges of size 1. We now show that (i) the nodes of G are one-to-one with the domain points; (ii) between any two neighboring points in the domain, there is an edge between their respective search tokens in G; and (iii) no edge exists between search tokens whose corresponding canonical ranges are not neighbors.

(i) By definition of URC, only range queries of size 1 result in the client issuing a single token. On line 1 Algorithm 2, the attack computes the set Q_1 of all tokensets of size 1. The algorithm then defines graph $G = (V, E)$ on the vertex set $V = \{t : \{t\} \in Q_1\}$, therefore its nodes correspond to the domain points in $\mathcal{D}$.

(ii) By definition of URC, it also follows that all range queries of size 2 are covered with two canonical ranges. So to query a range of size 2, a client must issue two search tokens: one for each of the two points in the range. One line 2, the algorithm computes the set Q_2 of tokensets of size 2 whose tokens appear in Q_1. For each $\{t, t'\} \in Q_2$, the algorithm adds the edge (t, t') to G. Tokens t and t' must correspond to neighboring points in $\mathcal{D}$ and there is an edge (t, t') in G.

(iii) Let t and t' be tokens of non-neighboring points $p, p' \in \mathcal{D}$, respectively. However, $p \cup p'$ is not a valid range. Thus $\{t, t'\}$ is not a valid tokenset and the algorithm never adds edge (t, t') to G.

From these three properties, it follows that G contains a component of size m that fully orders the point-value search tokens and their volumes, thus resulting in full database reconstruction.

Quadtree with BRC. Recall that the Quad-BRC scheme is a scheme with exact cover and no false positives. As before, the algorithm constructs a graph G that orders the search tokens of range queries of size 1. We now prove that (i) the vertex set of G contains the search tokens corresponding to the domain points of $\mathcal{D}$ (ii) between any two neighboring points in the domain, there is an edge between their respective search tokens in G; and (iii) no edge exists between search tokens whose corresponding canonical ranges are not neighboring ranges of the same size.

(i) The leaves of the quadtree correspond to the individual domain points and thus, under BRC, a range of size 1 is covered by a single canonical range of size 1. Correspondingly, the client issues a single search token to query for a range of size 1. On line 1 Algorithm 2 the algorithm computes the set Q_1 of all tokensets of size 1. This set thus contains the tokensets of all range queries of size 1.

(ii) In the quadtree, all canonical ranges are hypercubes. Consider a range of size 2; under BRC this range is covered using two canonical ranges, one for each point . For all ranges of size two, the client must issue two search tokens $\{t, t'\}$ such that $\{t\}, \{t'\} \in Q_2$. Thus $\{t, t'\} \in Q_2$ and the algorithm adds edge (t, t') to G.

(iii) In a quadtree with BRC, a range query corresponds to two search tokens if and only if the range can be covered by two neighboring canonical ranges of the same dimension. For a contradiction, suppose that r and r' are canonical ranges of unequal size such that $r \not\subset r'$ and $r' \not\subset r$, and let t and t' be their corresponding search tokens. Then $r \cup r'$ is not a hyper-rectangle and thus is not a valid range query. This implies that the tokenset $\{t, t'\}$ cannot exist and so the edge (t, t') is not added to G.

Now suppose that r and r' are non-neighboring canonical ranges of the same size, and let t and t' be their corresponding search tokens. Once again, $r \cup r'$ is not a valid range, $\{t, t'\}$ is not a valid tokenset, and thus (t, t') is never added to G.

As before, full database reconstruction follows.

Complexity Analysis. For both schemes, the graph G has size $O(m)$. For the range scheme, the volume map has size $O(m^2 \log^d m)$, and thus it takes $O(m^2 \log^d m)$ time to go through VM and construct G. For the quadtree, the volume map has size $O(m^{2+d-1/d})$ – the total number of queries $O(m^2)$ multiplied by the worst case range cover $O(m^{d-1/d})$.

If the adversary has observed all possible queries, which happens with high probability after $\Omega(m^2 \log m)$ uniformly distributed queries by the coupon collector principle, graph G contains all search tokens corresponding to domain points and all edges corresponding to range queries of size 2. □

A.5 Proof of Theorem 5

Proof. Rangetree with URC. Under URC, the client sends a single search token only when it queries a canonical range of size 1,

i.e. a single point. Thus, all ranges of size 2 must be covered by two ranges: one range for each of the two domain values in the range.

Let Q_1 be the set of all tokensets of size one. And let Q_2 be the set of tokensets of size two consisting of only tokens appearing in Q_1. Observe that there is a bijection from $\mathcal{D}$ to search tokens appearing in Q_1. This implies that there is a bijection from ranges of size two to Q_2. Given Q_2, the client can thus construct a grid spanning the domain of the database, thereby fully ordering the search tokens and recovering their corresponding value. The reconstruction space thus corresponds to the symmetries of a d-cube.

Quadtree with BRC. Under BRC, the only time that the client sends a single search token is when it queries a canonical range, i.e., a quadrant of size 2^{id} where $i \in \left[0, \frac{1}{d}\log_2 m\right]$.

Let Q_1 and Q_2 be defined as before. In a quadtree with BRC, a range query corresponds to two search tokens if and only if the range can be covered by two neighboring canonical ranges of equal size. Thus, for any range of size two, the client must issue a search token for each point in the range.

Thus, there is an injective mapping from ranges of size two to tokensets in Q_2. Thus, given Q_2, the adversary can construct a multi-component graph whose components represent a partition of the canonical ranges of the same size. There thus exists a component of size m, whose nodes correspond to the m domain points. From this component, the adversary can extract a full ordering of the search tokens and recover their corresponding value. The reconstruction space is thus the symmetries of a d-cube. □

A.6 Proof of Theorem 6

Proof. First, we show that we can reconstruct the inner grid database $\mathcal{D}$ and then we show that we can reconstruct the volumes of the extreme points as well (up to the symmetries of the square). Finally, we show that our algorithm succeeds with probability greater than $1 - \frac{1}{m^2}$ after observing $\Omega(m^2 \log m)$ queries uniformly random queries in $O(m^4)$ time.

The first step of the algorithm is to construct a co-occurrence graph G with nodes search tokens (i.e. range tree nodes), and an edge between two nodes if there is a tokenset consisting of the two of them. Then, for each tokenset S of size greater than two, we observe a grid (Lemma 3) graph G_S of G induced by the nodes of S. In a one-dimensional query, this is a line graph. G_S is also a line graph when the nodes of G_S cover the same ranges in all dimensions but one. In case of a line graph, G_S only has two nodes that have the smallest degree. In all other cases, there are four or more nodes that have the smallest degree, as they are the corners of the grid.

We are able to identify all one-dimensional queries (including some higher dimensional ones) by noting which tokensets have only two nodes with the smallest degree in G_S. The next step is to identify the inner nodes that cover two domain points. We measure the *edgecount* of each edge $e = (s_1, s_2)$: the number of times s_1 and s_2 appear in our identified queries together. Note that a token corresponding to a query of size 2, s_1, (e.g. gh in Figure 3) is connected in G with another token s_2, such that their edgecount is exactly 2. There are only two possible tokensets that contain s_1 and s_2 together, (s_1, s_2) and (s_1, s_2, s_3), for some token s_3. Extending the range in either direction will replace either s_1 or s_2 with their ancestors. For example, there are only 2 tokensets containing gh and i together, (gh, i) and (gh, i, f). Extending the range in one direction replaces gh with $\{efgh\}$ and in the other direction replaces i with $\{ij\}$. Note that any other one-dimensional tokens that cover a larger range, have edges with higher edge counts as there are more possible tokensets. For example, there are three tokensets containing $efgh$ and i together,$(efgh, i)$, $(efgh, i, d)$ and $(efgh, i, cd)$. Extending the range in one direction replaces gh with $\{efgh\}$ and in the other direction replaces i with $\{ij\}$. It is possible that inner nodes that cover multi-dimensional ranges have *edgecount* of 2 with a non-inner node. However, these tokens cannot be connected to leaf tokens as they would not form valid ranges, and thus such edges do not exist in G. In the end, we only consider the largest component of the graph, which contains the leaf tokens, ignoring other smaller components that may contain these multi-dimensional tokens.

At this point, we have identified the inner nodes that cover pairs of points and most of the point-query tokens. We still have to distinguish between certain leaf tokens and certain boundary tokens. Specifically, some inner tokens have isomorphic edges with edgecount of two. One edge connects to a boundary node, and the other edge connects to an inner node. For example, in Figure 3, nodes d and $abcd$ are isomorphic. We are able to extract the correct token, by picking the one with the most edges in the original co-occurrence matrix. The leaf node will always have one more edge than the boundary node, as the leaf node can be in a tokenset with the boundary node's sibling.

Now, we have identified all the pair token nodes and the tokens (or volumes) of all non-extreme leaf nodes. However, due to the nature of BRC and graph G, they are not in order. By doing a series of swaps and contractions, removing the pair-token nodes and putting the leaf nodes in order, we can reconstruct the inner grid of the database.

Once we have reconstructed the inner grid of the database, we can now reconstruct the volumes at the extreme points. Note that for the inner grid, we were able to identify which search token corresponds to which domain point. However, we cannot do the same for the extreme points, we can only extrapolate the volumes. The reason is that due to BRC, the search tokens for extreme points of the database often appear alone. For example, in Figures 3 and 4, the corner search tokens appear identical in the co-occurrence graph (nodes a, p and a, d, p, m respectively).

In our d-dimensional grid, in place of each domain point p extreme in one dimension is a search token s covering p and its neighbor p_n, where s has exactly one neighbor in our graph G', the search token covering p_n. We can thus extrapolate the volume of p from the volumes of s and p_n. We are unable to generalize this technique to tokens extreme in multiple dimensions as there is no such token.

Let's assume we have extrapolated all volumes of domain points extreme in up to $i-1$ dimensions. For each domain value v (extreme in i dimensions) we have yet to extrapolate, we have to find a basic i-dimensional cube c with sides of size 2, that contains v. In order to identify this cube, we leverage the structure of BRC tokensets. The range query that covers an i-dimensional cube of side length 3, c_3, which includes v, consists of a tokenset of size i+2. These tokens correspond to an i-dimensional cube of side length 2, c_2, (containing v), one token corresponding to a point value t diagonal

to v right outside c_2, and i 2×1 rectangles, set R. Essentially, the rectangles extend c_2 by one value in each dimension. We need one more point to cover all of c_3, which is t. Using the co-occurrence graph and the inner grid, we are able to identify tokens from R and t. Then, the smallest response that contains all of them, must contain c_2, in order to form a valid range. Then, using the volume of c_2, we can extrapolate the volume of v. For example in Figure 4, to extrapolate the volume of a, the 2-dimensional cube of side length 2 is $\{abef\}$, the 2 2×1 rectangles are $\{ij\}$ and $\{cg\}$ and $t = k$. This way, we can extrapolate the volume of domain points extreme in i dimensions. Thus, by induction, our algorithm can find the volumes of all extreme domain points.

The first step of the algorithm is to create dictionaries E, Q, which takes $O(m^2 \log^d m)$, as we have to go through all possible queries and their responses. Then, we go through $O(m^2)$ tokensets and for each construct a graph with their tokens and remove relevant edges from G. This takes $O(m^4)$ time. Then, we disambiguate identical components of the graph and contract edges of the graph, which takes at most $O(m^2)$ time. Then, we swap one-dimensional section of the graph, which takes $O(m^2)$ time.

We then have to identify volumes of the extreme points. There are $O(m)$ such points, and for each point we find a constant number of neighbors in two graphs, which takes $O(m^2)$ time. Then, we look through the tokensets to identify the required tokenset and extract the extreme point's volume. Finding the extreme point values takes $O(m^3)$ time. Thus, in total, our attack takes $O(m^4)$ time.

Our attack needs to observe all possible queries and their responses to work. Using a coupon collector argument, we observe all possible queries after the $\Omega(m^2 \log m)$ queries have been issued under a uniform distribution with probability greater than $1 - \frac{1}{m^2}$.

The adversary needs $O(m^2 \log^d m)$ space to store the search tokens and their responses. Graph G requires $O(m^2)$ space, with $O(m)$ nodes and $O(m^2)$ edges, similar to the temporary storage required by the G_S's. Most work on the algorithm is done on these graphs and there is a constant number of additional data-structures used, all requiring less than $O(m^2)$ space. Thus, the algorithm requires $O(m^2 \log^d m)$ space.

□

Lemma 3. *Let D be a d-dimensional database, over domain $\mathcal{D} = [m_1] \times \ldots \times [m_d]$, with $m = m_1 \cdot \ldots \cdot m_d$, which is encrypted under the range tree scheme and leaks volume, search and structure pattern under* BRC. *Let G be the co-occurrence graph with nodes the tokens of the range tree, and an edge between two nodes if they compose a tokenset. Graph G_S induced by the nodes of a tokenset S on G forms an k-dimensional grid, where $1 \leq k \leq d$.*

Proof. We prove this lemma by induction on the number of dimensions of the range that corresponds to tokenset S. In case of a 1D range query (on range r), we show that the corresponding tokenset S forms a line graph G_S of G induced on the nodes of S. A token $s \in S$ has one or two edges in G_S (unless $|S| = 1$).

Let us say that s_1 covers one of the endpoints of r, range r_1. As S forms a valid range, some token must cover the domain point $p \in r$, right next to r_1. Let us say token s_2 covers r_2, with $p \in r_2$. As $r_1 + r_2$ form a valid range, there is an edge between s_1 and s_2 in G_S. Notably, s_1 cannot be connected to any other token in S, as it would not form a valid range. On the other hand, $\exists p' \in r$ right outside r_2 (if $|S| > 2$). Thus, some token $s_3 \in S$ must cover it (with range r_3). Since $r_2 + r_3$ forms a valid range there exists an edge between s_2 and s_3. Since this is a 1D query, s_2 cannot have any other connections. Similarly we can show that any token of S that does not correspond to range covering an endpoint of r has two edges and the remaining two have one edge. Thus, G_S is a line graph, which is a one-dimensional grid.

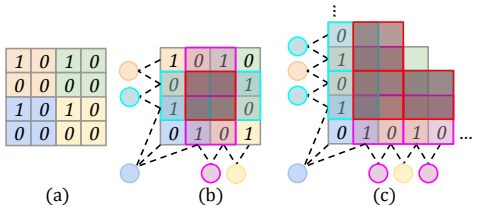

Figure 9: Here we demonstrate an assignment of volumes that gives a lower bound on the reconstruction space of (a) Quad-SRC **scheme and (b)-(c) the** QDAG-SRC **scheme.**

Suppose that any tokenset covering an $(i-1)$-dimensional range query forms a grid. Let S cover an i-dimensional range r. Let the last dimension of r be of size k. There are k $(i-1)$-dimensional ranges r_j, $j \in [1, k]$ that can be combined to cover r. Let r_j and r_{j+1} differ by one in the last dimension. Let tokens s_j (covering some part of r_j) and s_{j+1} (covering some part of r_{j+1}) cover the same ranges in the first $i-1$ dimensions. The combination of the ranges they cover is valid. Thus, there either exists an edge between them in G or there exists a token that covers their ranges combined.

Let T_i be the set of tokensets S_j covering each of the k $(i-1)$-dimensional ranges r_j. For each neighboring pair of tokensets in T_i, there either exists a new tokenset S' that covers their combined ranges (larger tokens apply) or the tokens are included in the BRC response (if no possible combination with neighbors exists). Note that it cannot be the case that only a subset of the neighboring tokensets can be combined, as the tokens are created in the same way for the different neighboring ranges. In case they are included in the BRC response, this pair of tokensets forms a grid (with their edges from G). Additionally, any new tokenset (replacing a previous pair) must also form a grid with its neighbors. Since all neighboring tokensets form grids, all of them together in G_S form a grid of dimension up to i, (potentially the $(i-1)$-dimensional ranges form a grid of dimension smaller than $i-1$).

□

A.7 Proof of Theorem 8

Proof. We show that each solution in $\mathcal{A}$ corresponds to a valid database. Let G be the underlying DAG over domain $\mathcal{D}$. For correctness we require that for all nodes v in G.

$$x_v = \sum_{\substack{w \text{ sink of } G, \\ w.range \subseteq v.range}} x_w. \tag{4}$$

That is, for every node v in G, v's volume must be the sum of the volumes assigned to the leaf-nodes that correspond to points in $v.range$. By Property (1), any non-sink node v of has a subset of children C such that $\{c.range : c \in C\}$ partition $v.range$. By

Equation 2 there exists a constraint of the form $x_v = \sum_{c \in C} x_c$. Each $c \in C$, must itself also have a subset of children C' whose canonical rages partition $c.range$. By recursively substituting the variables corresponding to the children that partition the canonical range of each variable, until we reach the sinks (which are 1-1 with the points in the domain by Property (2)), we end up with the Equation 4. Each of the substituted equations are constraints in the ILP that are satisfied, so Equation 4 must also be satisfied. We conclude that any solution in $\mathcal{A}$ must correspond to a real database.

Let $S_{\mathcal{L}}$ denote the reconstruction space and let $S_{\mathcal{A}}$ be the set of databases that correspond to solutions in $\mathcal{A}$. We will show that $S_{\mathcal{A}} = S_{\mathcal{L}}$. It is straightforward to see that $S_{\mathcal{L}} \subseteq S_{\mathcal{A}}$. In particular, since Equations 2 and 3 characterize the DAG G, then any database $D \in S_{\mathcal{L}}$ must satisfy the ILP.

Next we show that $S_{\mathcal{A}} \subseteq S_{\mathcal{L}}$. Since the databases in $S_{\mathcal{L}}$ are leakage-equivalent, then by Theorem 1 they result in the same volume map VM and frequency map FM, assuming each range query is issued exactly once. By Theorem 1 it is sufficient to show that the databases in $S_{\mathcal{A}}$ also result in VM and FM.

Let $\widehat{D} \in S_{\mathcal{A}}$ and let $\widehat{G}$ be its DAG of $\widehat{D}$ with volumes assigned to each node. Let $\widehat{\mathsf{VM}}$ and $\widehat{\mathsf{FM}}$ be the volume map and frequency map of $\widehat{G}$, respectively. From the leakage, we can build a map M mapping each observed tokenset $\mathbf{t}$ to its volume-frequency pair (vol, f). Equation 3 restricts each observed volume to be assigned to one node. The constraints impose a 1-to-1 correspondence between each volume with a given frequency f and each node in the DAG with frequency f. Since each observed tokenset has an associated volume-frequency pair, then the tokensets are 1-to-1 with the nodes in $\widehat{G}$; in particular each tokenset $\mathbf{t}$ such that $M[\mathbf{t}] = (vol, f)$ can be mapped to a node of $\widehat{G}$ with volume-frequency pair (vol, f).

Since each volume-frequency pair is 1-to-1 with the nodes in the $\widehat{G}$ of the same frequency then v must also have the same frequency and volume. Thus $\widehat{\mathsf{FM}}[\mathbf{t}] = \mathsf{FM}[\mathbf{t}]$. Also, we have that $\widehat{\mathsf{VM}}[\mathbf{t}] = (t, vol_{\mathbf{t}}) = \mathsf{VM}[\mathbf{t}]$. Since this holds true for all tokensets, then $\widehat{\mathsf{VM}} = \mathsf{VM}$ and $\widehat{\mathsf{FM}} = \mathsf{FM}$. This proves that Algorithm 5 achieves full database reconstruction.

The proof demonstrating that the input is available with probability $1 - \frac{1}{m^2}$ after the adversary observes $O(m^4 \log m)$ uniformly distributed queries follows from Lemma 4. □

A.8 Proof of Theorem 9

Proof. We demonstrate an assignment of volumes to the database resulting in a reconstruction space exponential in m. Each leaf node has 3 siblings. For each set of 4 sibling leaf nodes assign a volume of 1 to one leaf and 0 to the other siblings. Each leaf has a frequency of one and thus any set of volumes corresponding to the four siblings can be permuted; there are 2^2 unique permutations per set of 4 siblings, and $m/(2^2)$ such sets of siblings. More generally, we see that the reconstruction space of the quadtree is lower bounded by $(2^d)^{m/2^d} \gg 2^d(d!)$.

In contrast, the QDAG with the SRC range covering algorithm, offers a smaller false positive rate at the expense of a significantly smaller reconstruction space; we note however that the reconstruction is still $O(2^{d-1})$ greater than the symmetries of the hypercube. This is because the additional nodes and edges in the QDAG create a number of additional restrictions that the volume assignments must satisfy, hence reducing the total number of possible symmetries. In order to maintain the same volume assignments to the leaf nodes' parents, we cannot independently permute the volumes of the leaves.

Each QDAG node corresponding to each domain point not at the edge of the database is covered by an additional three nodes (compared to the quadtree). Assign each such domain point a unique value greater than 1. Each QDAG node corresponding to the domain points at the edge of the database is covered by 0 additional nodes (in the case of a corner) or 1 additional node otherwise. Assign each of these external domain points alternating bit volumes (See Figure 9). Since there is an even number of domain values along each edge this alternating bit assignment is always possible. Now observe, for a given edge, we can re-assign the bit-complement volumes to the domain points along the edge. The volumes associated with the nodes covering the edge nodes remain the same. In general, we can re-assign the bit-complement volumes to parallel edges independently of each other. In two dimensions this results in 2^2+2^2 additional symmetries. In d dimensions this results in $d \cdot 2^{2^{(d-1)}}$ additional symmetries. Composing them with the symmetries of the hypercube yields a lower bound of $2^{d+2^{(d-1)}}(d)(d!)$. □

B ESTIMATING FREQUENCIES

In the SRC attack (Algorithm 5) we assume that each query is issued exactly once. This is a strong assumption, so we now show how an adversary can correctly estimate the frequencies with inverse polynomial probability in $O(m)$.

Algorithm 6: GetFrequencies($\widehat{F}, G, \mathcal{D}$)

// $\widehat{F}$ is a dictionary mapping search tokens to the number of times each search token was observed, G is a DAG over domain $\mathcal{D}$.
Let N be the number of queries observed.
Let Q be the number of unique range queries over $\mathcal{D}$.
for st in $\hat{F}$ **do**
  $\widehat{F}[\text{st}] \leftarrow \widehat{F}[\text{st}]/(N/Q)$
return $\widehat{F}$

Lemma 4. *Let D be a d-dimensional database, over domain $\mathcal{D} = [m_1] \times \cdots \times [m_d]$, which is encrypted under the QDAG SRC scheme. Let $\widehat{F}$ be a dictionary mapping the observed search tokens to the number of times that search token was observed, G be the QDAG over $\mathcal{D}$. If the adversary observes N uniformly distributed queries, then the frequency of each search token* st *computed by Algorithm 6 (GetFrequencies) corresponds to the number of unique range queries that are associated with* st *happens with probability at least* $1 - 2|\widehat{F}| \exp(-2N/m^4)$, *where* $m = m_1 \times \cdots \times m_d$.

Proof. Suppose that the adversary has observed N queries being issued, and has constructed a dictionary $\widehat{F}$. For each search token st observed define the i.i.d. random variable

$$X_i = \begin{cases} 1, & \text{if the } ith \text{ search token is st} \\ 0, & \text{otherwise.} \end{cases}$$

Let Z_{st} be the number of unique range queries that correspond to search token st and let Q is the number of unique range queries over $\mathcal{D}$. Observe that we have

$$\mathbb{E}[X_i] = \frac{Z_{\text{st}}}{Q}.$$

Define variable $A_{\text{st}} = \sum_i^N X_i$. We thus have that GetFrequencies succeeds when for all st we have

$$\frac{A_{\text{st}}}{N} \in \left[\frac{Z_{\text{st}}}{Q} \pm \varepsilon\right]$$

for $\varepsilon = O(1/Q) = O(m^{-2})$. Applying a Chernoff bound argument we see that

$$\Pr\left[\frac{A_{\text{st}}}{N} \geq N\left(\frac{Z_{\text{st}}}{Q} - \varepsilon\right)\right] \leq \exp(-2N\varepsilon^2).$$

A similar argument holds for the upper bound. Taking a union bound over the $|\widehat{F}|$ times we must approximate frequencies gives us a total success probability of $1 - 2|\widehat{F}|\exp(-2N/m^4)$. □