Reconstructing with Even Less: Amplifying Leakage and Drawing Graphs

Evangelia Anna Markatou
TU Delft
Delft, Netherlands
Brown University
Providence, RI, USA
e.a.markatou@tudelft.nl

Roberto Tamassia
Brown University
Providence, RI, USA
roberto@tamassia.net

ABSTRACT

Leakage-abuse attacks using access pattern leakage from range queries have been shown to reconstruct encrypted databases. However, prior work is either restricted to one-dimensional databases or requires access to all possible responses in two-dimensions. In this paper, we explore what an adversary can achieve with minimal leakage, focusing on denser databases, and present a leakage abuse attack from access pattern of range queries in multiple dimensions. Our attack employs a novel technique to systematically amplify access pattern leakage, inferring a large number of new query responses that have not been requested by the user. Let m be the size of the database domain. Our attack works on d-dimensional databases and achieves approximate reconstruction. For dense databases and a parameter $0 < \lambda < 1$, our attack fully reconstructs an inner portion of size λm of the database (referred to as the λ-core) after observing $O(m \log m)$ queries, uniformly at random. These are significant improvements over previous attacks that require the full set of responses, which has size $O(m^2)$. We are the first to leverage graph drawing techniques for database reconstruction attacks. We implement our attack and evaluate it with experiments on real-world databases, achieving accurate reconstructions after observing a small percentage of the responses.

CCS CONCEPTS

• **Security and privacy → Cryptanalysis and other attacks**; **Database and storage security**.

KEYWORDS

Encrypted Database; Database Reconstruction; Leakage-Abuse Attack

ACM Reference Format:
Evangelia Anna Markatou and Roberto Tamassia. 2024. Reconstructing with Even Less: Amplifying Leakage and Drawing Graphs. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3658644.3670313

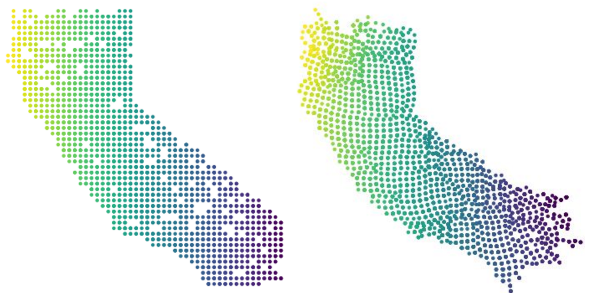

Figure 1: (Left) The (50×50) Cali dataset. (Right) Our reconstruction of the dataset after observing $13,005$ queries selected uniformly at random, or 0.8% of the possible queries, using the Kamada Kawai graph drawing algorithm, achieving mean error of 2.5. In comparison, Falzon et al. [23] require $\approx$ 1.6 million queries and Markatou et al. [54] require at least 0.5 million queries.

1 INTRODUCTION

In the past few decades, cloud computing has been changing the landscape of where data is stored and processed. People and companies choose to export their computational needs to cloud service providers. Along with the many benefits of cloud computing (e.g., efficiency, robustness and small set-up times), come some significant privacy drawbacks. Cloud service providers have access not only to plaintext data, but also to any computation the client wishes to perform. This highlights the need for efficient privacy-preserving technologies. In this work, we focus on the problem of private range queries on outsourced databases over an arbitrary number of attributes (dimensions).

There exist strong cryptographic primitives that allow for private range queries, like fully-homomorphic encryption (FHE) [26] or ORAM [30]. While offering very strong security, these solutions have prohibitive complexity that limits their use in practical applications. ***Searchable symmetric encryption*** (***SSE***) is a promising solution to the problem of searching encrypted databases. SSE achieves a practical tradeoff between efficiency and security and has attracted significant attention (see, e.g., [6, 7, 9–11, 13, 14, 17, 27, 28, 39, 41, 43, 44, 57, 60, 62]). SSE can be used to answer a variety of complex queries including boolean queries [10], range queries [24] and graph queries [22, 29]. These techniques have a lot of possible applications, including being used as a building block in creating a secure national gun registry [42] or for public policy analytics [19].

SSE schemes are much more efficient than those based on stronger cryptographic primitives such as FHE and ORAM, however they achieve this efficiency at the expense of some information leakage. A common type of leakage in SSE schemes is ***access pattern*** leakage. A scheme leaks the access pattern, when an adversary can observe individually encrypted records in each query response. This information is useful as it may allow the adversary to analyze co-occurrences of records across different responses. Other types of leakage from SSE schemes include volume pattern, where an adversary can observe the number of encrypted records in each response, and search pattern, where an adversary can tell if a query has been issued before. Exploiting leakage from SSE schemes to reconstruct the original plaintext database is the subject of leakage abuse attacks, an active area of research. The leakage abuse attacks presented in this paper assume that only access pattern leakage is available to the adversary. We do not use search pattern leakage nor assume specific query distributions.

Leakage abuse attacks on SSE schemes using leakage from range queries have been extensively studied. In the following, a range query on d attributes is referred to as a d-dimensional, or dD, range query. Also, an encrypted database on which dD range queries are issued is referred to as a d-dimensional database, or d D database. We assume that the d queried attributes take values from a finite ***domain*** $\mathcal{D} = [m_1] \times \cdots \times [m_d]$, for positive integers m_i, $i \in [1, d]$.

Prior Work. In their seminal work, Kellaris et al. [45] were the first to show how to reconstruct a 1D database from access pattern leakage, assuming a uniform query distribution. Subsequently, Grubbs et al. [33] achieved optimal approximate reconstruction of 1D databases, assuming access pattern leakage and a model of the database distribution. Lacharité et al. [51] considered the common scenario of ***dense*** databases, where there exists a record for every value of the domain of queried attributes using access pattern. They showed that dense 1D databases are more vulnerable to reconstruction attacks than general databases, and can be attacked using only access pattern leakage (other attacks also used an assumption on the query [45] or data distribution [33]).

Attacks on 2D databases were more recently investigated. Leakage from higher-dimensional range queries is harder to exploit than leakage from one-dimensional queries. After the publication of the first attacks on 1D queries [45], it took almost 5 years for the first attacks on 2D queries to be developed [23]. The leakage considered in our paper stems from range queries of an arbitrary dimension and thus is fully general. Attacks on one-dimensional queries are not effective against higher dimensional queries since the adversary does not have access to the query ranges in each dimension. A fundamental reason that makes higher dimensional queries harder to attack is that as the dimensions increase there are more possible spatial relationships between the records. For example, given distinct one-dimensional points a and b, either $a < b$ or $b < a$. Now, given two distinct two-dimensional points there are four possible relationships between them in terms of their order in the two dimensions. The number of spatial relationships increases exponentially in the number of dimensions. Correspondingly, the techniques we developed for attacks on higher dimensional queries are fundamentally different from techniques previously used in 1D.

Falzon et al. [23] achieved the first reconstruction attack on 2D databases using access and search pattern leakage, which requires that the adversary observes all possible queries and their responses. Markatou et al. [54] followed up with an approximate database reconstruction attack on 2D databases using access pattern leakage and partial search pattern leakage, which requires that the adversary observes all possible responses, but not all possible queries (two or more queries can have the same response). Note these attacks on 2D databases require the *entire set of responses* from all queries to be available to the adversary. In addition, the first attack assumes the adversary has further observed the *entire multiset of responses* as the same response can be returned by multiple queries. Little is known beyond two dimensions. Markatou et al. [53] present attacks on d-dimensional databases utilizing search pattern leakage as well as a different form of leakage, called *structure pattern*. These attacks work against specific schemes. For range queries on more than two attributes, no reconstruction attacks have been published against generic leakage (e.g. using access pattern).

Dense databases are an important class of databases as they arise in a variety of real-world applications including medical datasets (e.g., attributes "patient age," "gender," "blood type" and "hospitalization year"), census datasets (e.g., attributes "household size" and "dwelling type"), and student datasets (e.g., attributes "graduation year", "school district", and "course grade"). We recall that a database is dense, when it has at least one record with every possible domain value. A number of papers on reconstruction attacks specifically consider dense databases [32, 35, 51, 54].

Contributions. In this work, we present a database reconstruction attack on multidimensional databases and answer the following questions:

(1) Are multi-dimensional dense databases more vulnerable to reconstruction attacks than non-dense ones?
(2) Is it possible to reconstruct a multi-dimensional database after observing only a small subset of the query responses?

We consider a client who uses an SSE scheme that leaks access patterns to outsource a database over a domain of size m with an arbitrary number, d, of attributes, and a passive persistent adversary who observes the leakage from d-dimensional range queries and wishes to launch a reconstruction attack.

- ***Reconstruction Space of Dense Multidimensional Databases (Section 4).*** We show that in two and higher dimensions, a dense database has a reconstruction space of size exponentially smaller (in the number of records) than a general database, and thus leakage-abuse attacks are more effective against it. Previously, the higher vulnerability of dense databases had been known only in one dimension [51].
- ***Leakage Amplification (Section 5).*** We show how to systematically amplify the access pattern leakage by augmenting the responses observed with inferred responses, thus allowing the adversary to exploit responses that the client never asked for or received. We are the first to propose an amplification method for leakage abuse attacks that can be applied as a general preprocessing step to any attack based on access pattern leakage.
- ***Approximate Reconstruction for Multidimensional Databases (Section 6).*** We present an attack that uses only access patterns to achieve approximate database reconstruction. Leveraging our leakage amplification method, we achieve a high level of accuracy after observing a number of queries that is one to two orders of magnitude smaller than in previous attacks. Additionally, for dense databases, we are able to reconstruct the

	Assumptions						Attack			Reconstruction Space
	Database Dimension	Query Distribution	Data Distribution	Knowledge of Domain	AP	SP	Query Complexity	ADR	Leakage Amplification	Size
Kellaris et al. [45]	1	Uniform			✓		$m^4 \log m$			2
Lacharité et al. [51]	1		Dense		✓		$m \log m$	✓		2
Grubbs et al. [33]	1	Uniform			✓		$m^4 \log m$	✓		2
Markatou et al. [55]	1			✓	✓	✓	$m^2 \log m$			2
Kornaropoulos et al. [49]	1			✓	✓	✓	-	✓		2
Falzon et al. [23]	2			✓	✓	✓	$m^2 \log m$			2^n
Markatou et al. [54]	2			✓	✓	✓	$m^4 \log m$	✓		2^n
Our Attack	d				✓		-	✓	✓	d^n
Our Attack (FDR)	d		Dense		✓		$m^2 \log m$	✓	✓	d^n
Our Attack (λ-CFDR)	d		Dense		✓		$\left(\frac{2}{1-\sqrt[d]{\lambda}}\right)^d m \log m$	✓	✓	d^n

Table 1: Comparison of our attack with selected prior reconstruction attacks from range queries on schemes that leak the access pattern. We give the number of dimensions (attributes) over which range queries are issued, the data distribution of the database (dense or not), the query distribution (uniform or arbitrary), and whether the domain (interval of values of each attribute) is known by the adversary. We also show the leakage observed by the adversary, where AP refers to access pattern leakage and SP refers to search pattern leakage. We show the query complexity of each attack, i.e., the asymptotic number of uniformly distributed queries required by the attack to achieve either full database reconstruction (FDR) or λ-core full database reconstruction (λ-CFDR). Some attacks achieve approximate database reconstruction (ADR) when fewer queries are observed. We also indicate when leakage amplification is employed. Finally, we indicate the size of the reconstruction space. For simplicity, we omit big O notation. We denote with d the number of dimensions (attributes over which range queries are issued), with m the size of the database domain (product of the sizes of the intervals of each attribute), and with n the number of database records.

λ-core of the database (inner portion of size λm) after observing $O\left(\left(\frac{2}{1-\sqrt[d]{\lambda}}\right)^d m \log m\right)$ queries, uniformly at random.

- ***Graph Drawing (Section 6.1).*** When our attack achieves approximate database reconstruction, we introduce the use of graph drawing methods to complete the reconstruction. Specifically, we leverage techniques from force-directed graph drawing and orthogonal grid drawing.
- ***Precision and Recall for Reconstruction Attack Evaluation (Section 7).*** We introduce the notions of precision and recall for the quantitative evaluation of the accuracy of reconstructions. We are the first to adapt these classic information retrieval concepts for use in the analysis of leakage abuse attacks.
- ***Implementation and Experimental Evaluation (Section 7).*** We implement our attack and evaluate our results on real-world databases, showing that our attack is effective in practice as it relies on much less observed leakage than previous attacks. (See, e.g., Figure 1.)

A comparison of our work with selected related prior work on reconstruction attacks from encrypted range query schemes that leak the access pattern is summarized in Table 1.

2 RELATED WORK

In this work, we consider a client who outsources an encrypted database to a server and performs d-dimensional range queries on it. We do not consider a system that includes false positives.

Schemes. In order to achieve an effective trade-off between security and efficiency, we assume the client uses Searchable Symmetric Encryption (SSE) and builds an encrypted index that stores the responses to all possible range queries. A query response is then obtained by looking up into this index. Falzon et al. [23] propose such an approach for two-dimensional databases. This approach can be easily extended to work with databases of arbitrary dimensions. Demetrzis et al. [15, 16] and Faber et al. [21] have presented schemes for performing 1D range queries based on the classic range tree data structure. Wang and Chow [70] improve upon this work, presenting forward and backward-secure range search schemes. Falzon et al. [24] propose a number of range-reporting schemes for d-dimensional databases that offer different security and efficiency trade-offs. Espiritu et al. [20] present a framework for creating cryptographic schemes using SSE that handle aggregate range queries. Leveraging public key encryption, the MRQED scheme by Shi et al. [61] and the Maple scheme by Wang et al. [69] support multi-dimensional range queries, both leaking the access pattern. Order-revealing encryption [1, 5] can also support range queries. However, its leakage is unacceptably high (see, e.g., [4, 18, 34]).

Attacks. Kellaris et al. [45] initiated the study of access and volume pattern leakage from 1D range queries and presented the first full database reconstruction attacks. This work requires access to all possible leakage and an assumption on how the queries are issued. Lacharité et al. [51] followed up on this work, and showed that dense 1D databases are much more vulnerable than general ones, showing an optimal attack that requires only access pattern leakage. Grubbs et al. [33] further improved upon the attack, showing an optimal approximate database reconstruction attack for general 1D databases. Kornaropoulos et al. [49] explored approximate 1D database reconstruction attacks that do not assume knowledge of the query distribution. Access pattern leakage from k-nearest neighbor queries in one-dimensional databases has also been shown to be vulnerable to reconstruction attacks [48].

One-dimensional databases are also vulnerable to attacks that use volume pattern leakage, as shown in [32, 35, 45, 50]. The first volume-based attack was by Kellaris et al. [45], which was then improved in terms of query complexity by Grubbs et al. [32]. Gui et al. [35] do not depend on a uniform query distribution and are able to achieve partial reconstructions. Kornaropoulos et al. [50] attack specific practical response-hiding schemes.

Our attack is also applicable to 1D datasets. The leakage amplification technique in 1D is related to using PQ trees, a classic data structure used in 1D attacks [33, 55]. However, in multiple dimensions, no data structure similar to PQ-trees has been used in reconstruction attacks.

Falzon et al. [23] are the first to present a reconstruction attack on two-dimensional databases. This attack achieves a full database reconstruction attack when given complete access pattern and search pattern leakage. Markatou et al. [54] further improve upon this work by presenting an approximate database reconstruction attack that requires complete access pattern leakage but only partial search pattern leakage. Markatou et al. [53] present attacks on databases of arbitrary dimensions by exploiting volume and search pattern leakage, as well as structure pattern, a type of leakage that is inherent in all efficient range search SSE schemes. Their attacks focus on leakage of specific schemes, as opposed to more generic leakage. A more thorough comparison can be found in Section 8.

Given the proliferation of leakage abuse attacks against SSE schemes, mitigation techniques have also emerged. Examples include SEAL [14], which uses ORAM, Pancake [31], which adds fictitious queries using a client-side preprocessor, and a related method by Markatou and Tamassia [56]. Kornaropoulos et al. [47] present a technique to quantify the privacy of searchable encryption schemes using leakage inversion. Kamara et al. [40] created LEAKER, a framework for evaluating leakage attacks.

3 PRELIMINARIES

Basic Concepts. A d-***dimensional database***, D, maps each point of a ***domain*** $\mathcal{D} = [m_1] \times \cdots \times [m_d]$ to a set of ***records***. Let $m = m_1 \times \cdots \times m_d$, which we call the ***domain size***. Each database record r has a domain ***value*** $x = (x_1, \ldots, x_d) \in \mathcal{D}$, denoted as $D[r] = x$, which comprises the tuple of its ***attributes***. A d-***dimensional range query*** requests all records with domain value inside a query hyperrectangle $[a_1, b_1] \times \cdots \times [a_d, b_d]$, where $[a_i, b_i] \subseteq [1, m_i]$ denotes the query range in the i-th dimension. A hyperrectangle is the generalization of a 2D rectangle to higher dimensions.

We say that a point $p = (p_1, \ldots p_d)$ of domain $\mathcal{D}$ dominates domain point $q = (q_1, \ldots, q_d)$ if $p_i \geq q_i, \forall i \in [1, d]$. Two records of the database are *collocated* if they have the same value. Two records of the database are *neighbors* if their values are points at euclidean distance 1. For example, records with values $(0, 0, 0)$ and $(0, 0, 1)$ are neighbors, but records with values $(0, 0, 0)$ and $(1, 1, 1)$ are not neighbors. An event happens with high probability if it occurs with probability greater than $1 - \frac{1}{m}$.

A ***one-dimensional slice*** of the database is a range that only extends in a single dimension. A ***one-dimensional section*** of the database is a range that only extends in a single dimension and covers it fully (e.g. a row or column in 2D). We say that a database D over domain $\mathcal{D}$ is ***dense*** when for each domain point $p \in \mathcal{D}$, there exists at least one record $r \in D$ with value p, i.e., such that $D[r] = p$.

We define the ***response*** (or access pattern) of a query to be the set of encrypted record identifiers returned by that query. For example, suppose the client wishes to query range $[1, 5] \times [4, 9]$, they generate query token q and send it to the server, which then processes q and sends back the appropriate (encrypted) records $\{r_1, r_2, \ldots, r_k\}$. The client can then decrypt the results. We call ***response set***, denoted with RS, the set of responses observed by the adversary to the queries issued by the client. The ***full response*** set of a database D, denoted $\mathrm{RS}(D)$, is the response set associated with all possible queries on D. In most practical scenarios, the response set is a small subset of the full response set.

Threat model. We study the security of SSE database schemes that support d-dimensional range queries and leak the access pattern. We consider a passive persistent adversary who can observe the leakage over an extended period of time. Unlike all previous attacks in dimensions two and higher, we do not assume knowledge of the database domain by the adversary, i.e., the adversary does not know the intervals of the database attributes.

Reconstruction Space. As shown in [23, 54] it may be information theoretically impossible for an adversary to distinguish between two databases from the observed leakage of the scheme. Such databases are called ***equivalent***.

Definition 1. *Databases D and D' over the same domain are equivalent if* $\mathrm{RS}(D) = \mathrm{RS}(D')$.

Definition 2. *The reconstruction space of D is the set of all databases equivalent to D.*

No adversary can distinguish between any two databases in the reconstruction space, unless they have access to auxiliary information.

Definition 3. *Full Database Reconstruction (FDR) of database D is the problem of reconstructing the values of all the records of the database D.*

Definition 4. *Let $\mathcal{D}$ be a d-dimensional domain of size m. For $0 < \lambda \leq 1$, the λ-**core** of $\mathcal{D}$ is the hyperrectangular section of domain $\mathcal{D}$ of size $\sqrt[d]{\lambda} m_1 \times \cdots \times \sqrt[d]{\lambda} m_d = \lambda m$ centred around the midpoint of $\mathcal{D}$. The λ-Core Full Database Reconstruction (λ-CFDR) is the problem of reconstructing the values of a database in the λ-core of $\mathcal{D}$.*

Range Search Scheme. We consider a range search scheme based on SSE. The client precomputes the answers to all range queries and stores them in an index. Then, an SSE scheme for multimaps like the one by Cash et al. [10] can be used to encrypt the index. A version of this scheme has been used by Falzon et al. [23] and Markatou et al. [54] in their 2D database reconstruction attacks. Additionally, Demertzis et al. [15] and Falzon et al. [24] present a version of this scheme and call it the quadratic scheme.

Graph Drawing. Our attack algorithm produces a graph whose nodes represent domain points with a nonempty set of records and whose edges identify records with values at euclidean distance ≈ 1 in the domain. In order to turn these combinatorial graphs into geometric graphs where nodes are assigned values, we turn to the field of graph drawing. We focus on two different approaches: (i) ***force-directed drawings*** of general graphs and (ii) ***orthogonal grid drawings*** of planar graphs. The first approach works in arbitrary dimensions while the second approach only works in 2D.

Force-directed drawings of graphs are obtained by simulating the evolution of a physical system where forces are exerted on the nodes of the graph. The output drawing is a stable configuration of the system, i.e., achieving local minimum energy. There is a large literature on force-directed drawings. Kamada and Kawai [38] is a classic force-directed graph drawing algorithm whose goal is to place the nodes such that the Euclidean distance between any two nodes is about the same as the shortest path distance between them in the graph. The algorithm runs in $O(n^3)$ time, where n is the number of nodes. After experimenting also with other force-directed algorithms such as Fruchterman and Reingold [25], we found that the Kamada Kawai algorithm best suits our attacks, since in our graphs, two neighboring nodes correspond to records that generally have values at distance 1 in the domain.

Force-directed drawing algorithms can produce drawings in arbitrary dimensions, and thus are suitable for our attacks on general d-dimensional databases. However, they yield drawings where the values of the nodes have real values (approximated as floating point values). An orthogonal drawing of a graph is a 2D drawing such that the edges of the graph are drawn as polygonal chains comprising horizontal and vertical segments. An orthogonal grid drawing places all the nodes at points with integer coordinates. Thus, orthogonal grid drawings overcome the aforementioned limitation of force-directed methods. Many algorithms have been developed for orthogonal drawings, with special attention given to planar orthogonal grid drawings, where no two edges cross. For our 2D attacks, we evaluate the classic orthogonal grid drawing algorithm by Tamassia et al. [64, 66].

For an introduction to force-directed and orthogonal graph drawing algorithms, see the relevant chapters in the book by Di Battista el al. [3] and the handbook edited by Tamassia [65]. Force-directed drawings are also discussed in the survey by Brandes [8].

4 RECONSTRUCTION SPACE

The size of the reconstruction space (Definition 2) is an important metric in leakage abuse attacks since any two databases in the reconstruction space "look the same" to the adversary. Thus, the size of the reconstruction space provides an information-theoretic bound on the effectiveness of any reconstruction attack. The reconstruction space of dense databases is much smaller than the one for general databases. In [23, 54], it is shown that the reconstruction space of a two-dimensional database could be exponential in the number of records. This is a huge reconstruction space, which makes it much harder for an adversary to pinpoint the original data after a reconstruction attack. In this section, we show that the result of [23, 54] generalizes in arbitrary dimensions (Theorem 1), and further show that dense databases have a reconstruction space that scales with the number of dimensions and is independent of the domain size and the number of records of D (Theorem 2).

THEOREM 1. *For every positive integer n, there exists a d-dimensional database D with n records such that encrypting D using an SSE scheme that leaks the the full response set (access pattern) from d-dimensional range queries yields a reconstruction space of size $O(d^n)$.*

PROOF. Consider the d-dimensional database D with n records, numbered from 1 to n, where record i has value $[2i, 2i-1, ..., 2i-1]$. We generate a set of databases, E_D, that we will show are all equivalent to D. A database D' is in E_D if it can be generated by copying D and then taking a subset of its records and replacing their value with a permutation of it (a permutation of $[2i, 2i-1, ..., 2i-1]$). In order to show that D and such D' are equivalent, we have to show that $\mathsf{RS}(D) = \mathsf{RS}(D')$.

Let us say response $resp \in \mathsf{RS}(D)$ corresponds to a range starting at domain point $s = [s_1, ...s_d]$ and ends at domain point $e = [e_1, ...e_d]$. We need to show that there exists a range such that $resp \in \mathsf{RS}(D')$. We note that for each pair of records (i_l, i_{l+1}), i_{l+1} dominates i_l. Suppose that the response in D contains records $[i_0, ..., i_k]$. The same query on D' has to contain records numbered $i_1, ..., i_{k-1}$, as s must be dominated by records $i_1, ..., i_k$ since i_1 dominates i_0. Similarly, e dominates points $i_0, i_1, ..., i_{k-1}$. If record i_0 in D' is different than the one in D, then its value is a permutation π of the one's in D. We can apply this permutation on s and get $s' = \pi(s)$. Similarly for i_k, we can apply permutation π' to get $e' = \pi'(e)$. Range $[s', e']$ on D' contains all records $i_0, i_1, ..., i_k$.

Thus, any $resp \in \mathsf{RS}(D)$ exists in $\mathsf{RS}(D')$. We can make a similar argument showing that any $resp \in \mathsf{RS}(D')$ exists in $\mathsf{RS}(D)$. Thus, we can conclude that D and D' are equivalent databases. Since there are n records in D and there are d unique ways to permute each record's values, we can generate $O(d^n)$ distinct databases that belong to E_D. □

THEOREM 2. *The size of the reconstruction space of any d-dimensional dense database D encrypted using an SSE scheme that leaks the full response set (access pattern) is $O(2^d d!)$.*

PROOF. We consider an adversary that has observed the access pattern from all possible responses. We show that using this leakage the adversary is able to reconstruct the order of all records and thus the database grid. However, it is impossible for the adversary to determine the orientation of the database grid (e.g. the adversary can tell that record r_2's value is between the values of records r_1 and r_3 in the first dimension, but cannot determine if r_1 has the smallest or largest value).

First, the adversary can find the smallest response s_r that contains each record r. Since the adversary has seen all possible responses, then all points in s_r must be collocated. The adversary can at this point create a graph G and add a node for each set of collocated points. Then, the adversary can look for responses that contain exactly two of these sets. If a response exists $\{s_1, s_2\}$, then s_1 and s_2 correspond to neighboring domain points. The adversary can then add an edge between the corresponding nodes for s_1 and s_2 on the graph. At this point, the adversary has constructed a perfect grid graph. However, without access to auxiliary information, the adversary cannot determine the orientation of this grid (e.g. which grid corner corresponds to the smallest value). All manipulations that generate an equivalent database to D must maintain the grid intact. Note that this is in contrast to the general case, where the empty space allows for more manipulations. Thus, the only possible ways to generate an equivalent database is by applying the symmetries of a hypercube that can be found in the hyperoctahedral group. We conclude that there are $O(2^d d!)$ equivalent dense databases. □

Reconstruction attacks can be way more effective under a smaller reconstruction space. Thus, we conclude that leakage-abuse attacks

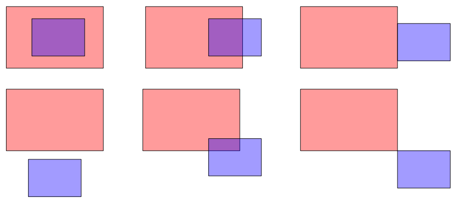
Figure 2: We see that no matter how the rectangles overlap, their intersection forms a rectangle, potentially of lower dimension (i.e., a line segment or a point).

are more effective against dense multi-dimensional databases than general multi-dimensional databases, due an exponential gap in the sizes of the corresponding reconstruction spaces.

5 LEAKAGE AMPLIFICATION

In our preprocessing step, we show how to amplify the observed response set and then how to identify and group together any collocated records. This process does not produce new information but allows us to explicitly derive information that was implicit in the data, and improves the efficiency of our attack.

Amplify RS. The first step of our attack is to generate more responses from the ones we have observed. The heart of this algorithm comes from the following observation: The non-empty intersection of two rectangles is a rectangle, a line segment or a point (Figure 2). We generalize our observation to arbitrary dimensions in Lemma 1.

Lemma 1. *The non-empty intersection of two hyperrectangles is always a hyperrectangle itself.*

From Lemma 1, it follows that the non-empty intersection of two responses must be a possible response. In Algorithm 1, we take the intersection of every pair of responses, and if it is non-empty we add it to our set of responses. This technique allows the adversary to exploit responses that the user has not asked for or even seen.

Algorithm 1: *AmplifyResponses*(RS)

```
1: Let set responses contain the unique responses in RS.
2: Let $new\_responses$ be an empty list.
3: for each $r_1$ in responses do
4:   for each $r_2$ in responses do
5:     Let $r$ be the intersection of $r_1$ and $r_2$
6:     if $r$ is non-empty and not in responses then
7:       Add $r$ to $new\_responses$.
8: Add $new\_responses$ to responses
9: return responses
```

Theorem 3. *Let D be a d-dimensional dense database with n records over domain $\mathcal{D} = [m_1] \times \cdots \times [m_d]$ of size m. Given the response set* RS *of size ℓ observed by the adversary from access pattern leakage of d-dimensional range queries, Algorithm 1 produces an amplified response set of size $O(\ell^2)$ in time $O(n\ell^2)$.*

Proof. Algorithm 1 takes the intersection of each pair of responses, and if it is not empty, adds it to the response set. It follows from Lemma 1 that this intersection indeed corresponds to a real response. It takes ℓ^2 iterations to look at each pair, and their intersection takes $O(n)$ time. Thus, Algorithm 1 takes $O(n\ell^2)$ time and produces $O(\ell^2)$ additional responses. □

It is possible to further augment the responses, but for efficiency reasons, we intersect each pair of queries once. It is worth noting that mitigation techniques (e.g. [56]) which work by restricting the query space could be countered using Algorithm 1, since the adversary is able to generate responses that were never even requested.

Identify and Group Collocated Records. Once we have augmented the response set, we move to identify which records are collocated (have the same domain value). Intuitively, records with the same domain value appear in exactly the same responses. Any records that have the same domain value are called collocated.

A simple approach to finding the collocated nodes would be to identify the smallest response that contains each record r, say $s_r = \{r, r'\}$, and denote all records in s_r as collocated. However, it is possible that the smallest response that contains r' is actually $s_{r'} = \{r'\}$. In this case, we need to modify s_r and remove r' from it.

Algorithm 2 follows: We first identify the smallest response s_r that contains each record r. The nodes in s_r are all the candidates for the collocated C records of r. Then, for each record r, we find all responses R_r that contain it, and ensure that each element in its collocated set C is also in all the responses in R_r (Algorithm 3). Any element that does not satisfy this is removed from C. Now, we look for records whose collocated set is not consistent. For example, record r_1, could have collocated set $C_1 = \{r_1, r_2\}$, but record r_2 has collocated set $C_2 = \{r_2\}$. This means that r_1 and r_2 are not collocated. We take the intersection of C_1 and C_2, and identify the two complements: $\{r_1\}, \{r_2\}, \{\}$. Each record is assigned the set that contains it: r_1 is assigned $\{r_1\}$ and r_2 is assigned $\{r_2\}$. This procedure ensures consistency in the sets. Once we have identified the collocated sets (Algorithm 4), we translate the responses, such that each collocated set is replaced by an ID in each response.

Algorithm 2 takes as input a set of responses and outputs a map, whose keys are records and values are all the records collocated with the key record. Algorithm 3 takes as input a set of responses and the map output by Algorithm 2 and outputs a set of responses where each response contains IDs corresponding to domain points (as opposed to records).

Algorithm 2: *FindColocatedRecords*(*responses*)

```
1: Initialize dictionary $Colocated$.
2: for each record $r$ do
3:   Find $s_r$, the smallest response that contains $r$
4:   $Colocated[r] = s_r$
5: for each record $r$ in $Colocated$ do
6:   Find any records in $Colocated[r]$ that do not appear in all the same
     responses as $r$ and remove them from $Colocated[r]$
7: return $EnsureConsistency(Colocated)$
```

Lemma 2. *Algorithm 2 identifies as many collocated records of database D as allowed by the observed leakage, in time $O(mn^2 + n\ell)$.*

Proof. Given two non-collocated records i, j, one can only distinguish between them if there exists at least one response that

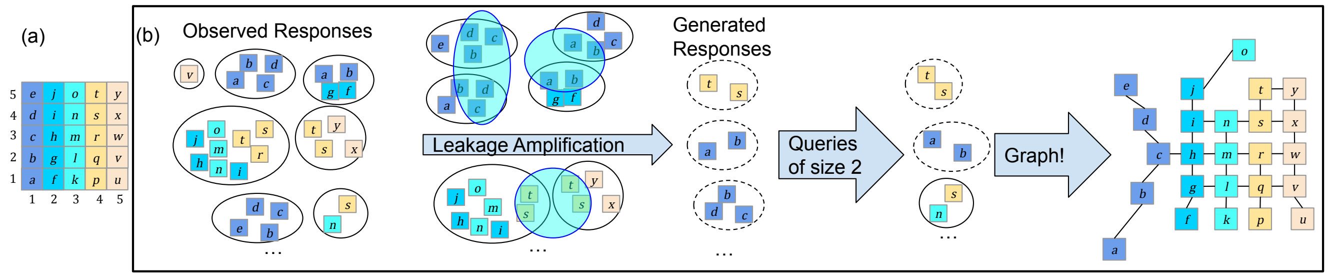

Figure 3: Algorithm 5. (a) A 5 × 5 database. (b) The first step of the attack is the leakage amplification process, where we intersect the observed responses and generate new responses (in dashed lines). Then, Algorithm 5 finds all responses of size 2 (Line 4), both observed and generated. We then generate a graph representation of the database. (Line 8).

Algorithm 3: *EnsureConsistency*(*Colocated*)

```
1: Let $todo$ be the a set of the keys of $Colocated$
2: while $todo$ is not empty do
3:   Pop an element from $todo$ and set $r$
4:   for each record $r'$ in $Colocated[r]$ do
5:     if $Colocated[r] \neq Colocated[r']$ then
6:       $a = Colocated[r] - Colocated[r']$
7:       $b = Colocated[r'] - Colocated[r]$
8:       $c = Colocated[r] \cap Colocated[r']$
9:       Assign $Colocated[r]$ to $a, b$ or $c$ based on which contains $r$
10:      Assign $Colocated[r']$ to $a, b$ or $c$ based on which contains $r'$
11:      Add the elements of $a, b, c$ to $todo$
12: return $Colocated$
```

Algorithm 4: *Translate*(*Colocated*, *responses*)

```
1: Let $new\text{-}responses$ be an empty list
2: Assign each value of $Colocated$ an ID, such that values that are
   identical have the same ID.
3: for $resp$ in $responses$ do
4:   Let $new\text{-}resp$ be an empty set.
5:   for $r$ in $resp$ do
6:     Add the ID of $Colocated[r]$ to $new\text{-}resp$
7:   Add $new\text{-}resp$ to $new\text{-}responses$
8: return $new\text{-}responses$
```

contains i and not j. Additionally, both i and j must appear in some response, otherwise we do not know they exist. Algorithm 2 first finds an approximation of which records are collocated by determining the smallest response that contains each record. Given all available leakage, the smallest response s_r that contains a record r, corresponds to a range that covers one domain point, r's value. Thus, any collocated records would be in the same smallest response. However, our algorithm works with less leakage as well. Thus, for each record r, we go through all the responses that contain it, and if we find a record in s_r that is not in any of the responses, we remove it from s_r, as the records are not collocated. At this point, if we have observed i and j and a response that contains i and not j, we have successfully determined that i is not collocated with j and removed j from s_i. However, we need to propagate this knowledge to j. This is where Algorithm 3 comes. It ensures consistency, such that if i is not collocated with j, i is removed from s_j (as j is not in s_i).

The for loop in line 2 of Algorithm 2 takes $O(n\ell)$ time, as we find the smallest response that contains each record. Algorithm 3 executes $O(n + m^2)$ loops, as a range can only be split until a point range. In each loop, $O(n)$ work is done. Thus, Algorithm 2 takes $O(mn^2 + n\ell)$ time. □

6 DATABASE RECONSTRUCTION

We present our attack (Algorithm 5) which achieves approximate database reconstruction given the access pattern leakage of range queries on a multi-dimensional database. See Figure 3 for a 2D example of Algorithm 5. Note that for simplicity in Figure 3, we assume that there is exactly one record on each domain point, which is formally addressed in Algorithm 2. We highlight that our attack works on any database, but performs better the more dense the database is. One of our goals with this attack is to see what an adversary can achieve with very little information. We focus on local information and do not build upon relationships we have discovered, as they might be wrong and errors propagate.

Algorithm 5: *DatabaseReconstruction*(RS)

```
1: $responses = AmplifyResponses(\mathrm{RS})$ (Algorithm 1)
2: Let $Collocated = FindCollocatedPoints(responses)$ (Algorithm 2)
3: $new\text{-}responses = Translate(Colocated, responses)$ (Algorithm 4)
4: Let $pairs$ be all elements of $new\text{-}responses$ of size 2
5: Initialize empty graph $G$
6: for each $(a, b) \in pairs$ do
7:   Add edge $(a, b)$ to graph $G$
8: return $G$
```

After preprocessing the responses, we find all responses of size 2. Then, we construct a graph G with nodes the responses' identifiers, and an edge between two nodes if there exists a response containing just those two nodes.

Lemma 3. *Let D be a d-dimensional database with n records over domain $\mathcal{D} = [m_1] \times \cdots \times [m_d]$ of size m. Given the response set* RS *of size ℓ observed by the adversary from access pattern leakage of d-dimensional range queries, Algorithm 5 runs in time $O(n\ell + mn^2)$.*

Amplifying the leakage takes $O(mn^2 + n\ell)$ time and creating the graph $O(n^2)$ time. Thus, Algorithm 5 runs in time $O(n\ell + mn^2)$.

6.1 Graph Drawing

The graph we construct contains information about neighboring nodes of the database, but may not be a grid graph as (i) we may not have observed all records or (ii) the database could be not dense. In order to approximately reconstruct the database based on this graph, we turn to techniques from the field of graph drawing. We mainly consider two classic graph drawing methods: (i) an algorithm by Tamassia et al. [64, 66] which consists of three steps: Planarization, Orthogonalization, and Compaction (POC) and (ii) a force directed graph drawing method by Kamada and Kawai [38]. We note that graph drawing is an integral part of the database reconstruction attack. The graph we have constructed gives us spatial closeness information of the record values; we use a graph drawing algorithm to turn that information into reconstructed values.

Although we found that these two methods perform best for database reconstruction, in Figure 4, we compare how a number of methods perform on drawing a square grid graph (25×25). The first plot (on the top right) shows a random placement of nodes as a baseline. In the top left, we display the Chrobak & Payne [12] method which depends on planar graph triangulation. Both these methods fail to show the grid-nature of our graph. In the middle left, we display the Fruchterman Reingold method [25], one of the first force-directed graph drawing techniques, where "vertices behave as atomic particles or celestial bodies, exerting attractive and repulsive forces on one another." This is an improvement upon the previous two methods and it does showcase the grid structure, however the drawing is convoluted. In the middle right, we use the spectral approach [46], which uses eigenvectors of the graph Laplacian to compute the drawing. Here, we again see an improvement on the grid layout, but the distance between the points is inconsistent. Finally, in the bottom row, we showcase the Kamada Kawai [38] and Tamassia et al. [64, 66] methods, which draw a perfect grid.

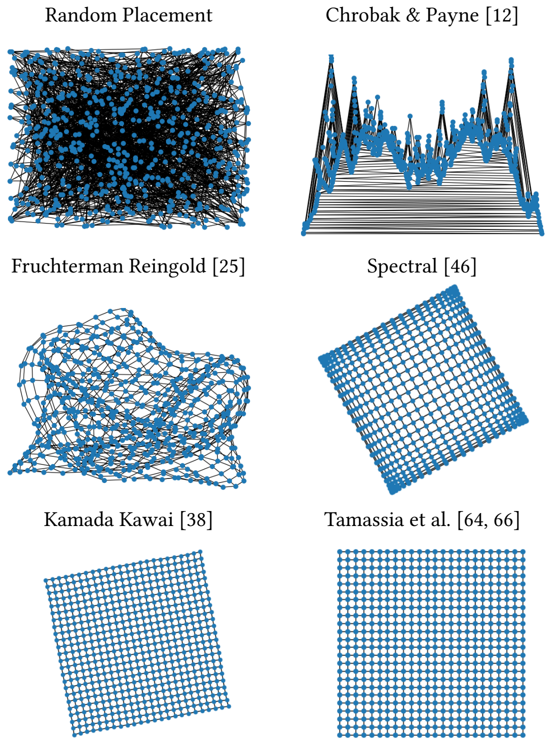

Figure 4: Graph drawing of a 25×25 grid with different techniques

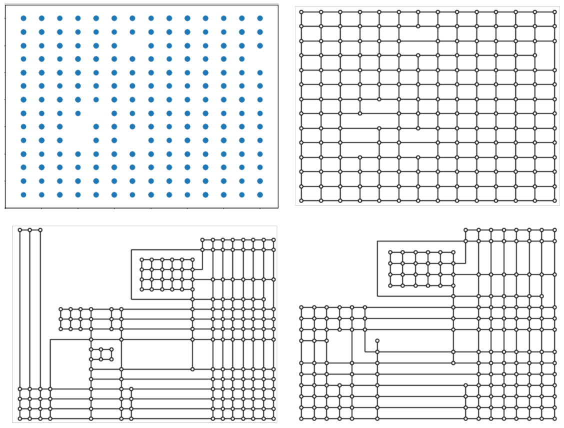

Figure 5: The original database is depicted on the top left and our most compact reconstruction on the top right. On the bottom, we can see less compact (and less accurate) possible reconstructions.

Note that only Tamassia et al. [64, 66] strive to draw edges along horizontal and vertical lines.

In case Algorithm 5 returns a perfect grid graph, we do not need to use a graph drawing technique. Instead, we can traverse the grid assigning values to records. For example, the following algorithm works for two-dimensional databases: First, identify the four corners of the database, these are the elements with the smallest number of edges. By finding the shortest path between two pairs of corners, we can determine the "top" and "bottom" edges of the database, and assign values to the records. Afterwards, we can align the "top" and "bottom" edges, by finding the shortest path from an element of the "top" edge to some element in the bottom edge. Then, we assign values to all elements in that shortest path.

6.2 Orthogonal Layout Approach

The first step of the Tamassia et al. algorithm [64, 66] is planarization. This step extracts a planar embedding from the graph. Then, the algorithm moves on the orthogonalization step. The goal is to minimize the number of bends on the layout, which is achieved by using network flow techniques. The final step is compaction, where the layout area is minimized. We note that this approach only works on planar graphs, and G may be non-planar. Since planarization is an NP-hard problem, we take a heuristic approach to create a maximal planar subgraph of G. We repeatedly compute the Kuratowski subgraph and remove it from G until G is planar. Then, we add back as many edges as we can from the ones we removed, maintaining G's planarity. Then, we run the planarization, orthogonalization, and compaction steps. Our heuristic planarization can make different choices on which edges to remove/keep, resulting in different graph drawings (Figure 5).

6.3 Force-Directed Approach

We utilize the Kamada-Kawai graph drawing algorithm [38] from the force-directed graph literature to reconstruct a database of an arbitrary number of dimensions. For simplicity we describe the algorithm for the 2D case following the explanation of [65]. Let $d_{i,j}$ denote the shortest path in G between nodes i and j. Then, the ideal length of spring between these nodes is $l_{i,j} = d_{i,j}$. The strength of the spring between i and j is $k_{i,j} = \frac{K}{d_{i,j}^2}$, for some constant K. If the coordinates of node i are given by (x_i, y_i), then minimizing the following energy function E gives us the desired layout.

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{k_{i,j}}{2} ((x_i - x_j)^2 + (y_i - y_j)^2 + l_{i,j}^2 - 2l_{i,j}\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2})$$

6.4 Information Theoretic Limitations

Recent work by Alegría et al. [2] has interesting applications on our work for two-dimensional databases. The authors show that if a graph can be drawn such that its outer face is a rectangle, then the problem of testing for the existence of a unit-length rectangular drawing, can be solved in polynomial time. Otherwise, it is NP-complete. Thus, the reconstruction of certain databases (whose outer face is rectangular) can be done very efficiently. If the database is not rectangular, an optimal reconstruction becomes much harder.

6.5 Dense Database Reconstruction

Although our attack focuses on approximate database reconstruction, if the database is dense, we can achieve FDR and λ-CFDR as well. In our formal analysis, we assume a uniform query distribution, consistent with prior work [23, 32, 33, 48, 51, 54]. However, our attack works under different query distributions, as we experimentally show in Section 7.

Theorem 4. *Let D be a d-dimensional dense database with n records over domain $\mathcal{D} = [m_1] \times \cdots \times [m_d]$ of size m. Given the response set* RS *observed by the adversary from access pattern leakage of d-dimensional range queries, Algorithm 5 achieves Full Database Reconstruction (FDR) after observing $O(m^2 \log m)$ queries issued uniformly at random with high probability.*

We show a sketch proof for Theorem 4. Algorithm 5 can achieve FDR after observing all the responses. Using a coupon collector argument, we conclude that after observing $O(m^2 \log m)$ queries uniformly at random, Algorithm 5 achieves FDR (since for all record pairs p, q, we can observe a response that contains both of them and two responses that each contain one of them, we can identify all collocated records and their neighbors, which gives us a grid representation of the original database). Given fewer responses, Algorithm 5 achieves ADR as shown in Section 7.

Recall that the λ-core of domain $\mathcal{D}$ is the hyperrectangular section of domain $\mathcal{D}$ of size $\sqrt[d]{\lambda}m_1 \times \cdots \times \sqrt[d]{\lambda}m_d = \lambda m$ centered around the midpoint of $\mathcal{D}$ (Definition 4). We show that Algorithm 5 can reconstruct the value of all records in the λ-core of the domain of dense database D much faster than achieving FDR.

Theorem 5. *Let D be a d-dimensional dense database with n records over domain $\mathcal{D} = [m_1] \times \cdots \times [m_d]$ of size m. Given the response set* RS *observed by the adversary from access pattern leakage of d-dimensional range queries, Algorithm 5 achieves λ-Core Full Database Reconstruction (λ-CFDR) after observing*

$$O\left(\frac{2^d}{(1 - \sqrt[d]{\lambda})^d}\, m \log m\right)$$

queries issued uniformly at random, with high probability.

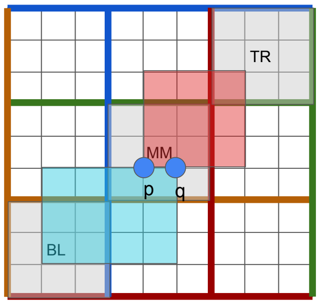

Figure 6: (A 2D example) Algorithm 5 reconstructs the λ-core of the database (MM) after observing $O(\frac{2^d}{(1-\sqrt[d]{\lambda})^d} m \log m)$ queries uniformly at random. We show that for each pair p, q, with high probability, we observe a blue response, which starts at BL and ends at q, and a red response, which starts at p and ends at TR. The intersection of the two, gives us a response that contains only the neighbors p, q.

Proof. Let us split the database in sections. Let MM be the λ-core of the database we want to reconstruct, BL be the section that starts at the origin $(1, ..., 1)$ and ends where MM starts and TR be the section that starts at the end of MM and ends at the end of the domain $(m_1, ..., m_d)$. In Figure 6, we see a two dimensional database, with the sections highlighted in gray.

If, for every neighboring pair p, q in the λ-core (MM) (say q dominates p), we observe or generate a response that contains just the two of them and two responses that contain just q or just p, then Algorithm 5 will be able to connect them in G, and fully reconstruct MM. In order to generate a response p, q, we need to intersect two responses. One response r_1 must start in BL and end at q and the other response r_2 must start at p and end in TR. MM's domain size is $\sqrt[d]{\lambda}m_1 \times ... \times \sqrt[d]{\lambda}m_d$. Then, BL's domain size is $\frac{1-\sqrt[d]{\lambda}}{2}m_1 \times ... \times \frac{1-\sqrt[d]{\lambda}}{2}m_d$. There are $(\frac{1-\sqrt[d]{\lambda}}{2})^d m$ possible queries that start at BL and end at q. The probability that a query picked uniformly at random starts in BL and ends at q is smaller than

$$\frac{(\frac{1-\sqrt[d]{\lambda}}{2})^d m}{m^2} = \left(\frac{1-\sqrt[d]{\lambda}}{2}\right)^d \frac{1}{m},$$

Let $c = (\frac{1-\sqrt[d]{\lambda}}{2})^d$. Suppose we observe $\frac{8}{c} m \log m$ queries. The probability that none of them satisfies the condition is greater than

$$(1 - \frac{c}{m})^{\frac{8}{c} m \log m} \approx \frac{1}{e^{8 \log m}} \approx \frac{1}{m^8}.$$

Similarly, for a response that starts in TR and ends in q. We can also use the same argument to find the probability that we can generate a response that contains just p and a response that contains just q. There are at most dm^2 pairs of points p, q. Thus, by union bound, after observing $O\left(\frac{2^d}{(1-\sqrt[d]{\lambda})^d} m \log m\right)$ queries uniformly at random, Algorithm 5 can find all neighbors in MM with probability greater than $1 - \frac{6dm^2}{m^8}$. Then, we can extract a maximal graph of the database, reconstruct the λ-core (MM) and achieve λ-CFDR, with high probability for $m > d$. □

7 EVALUATION

We implemented our attack and conducted experiments on real-world 2D and 3D datasets. Our experimental results show that

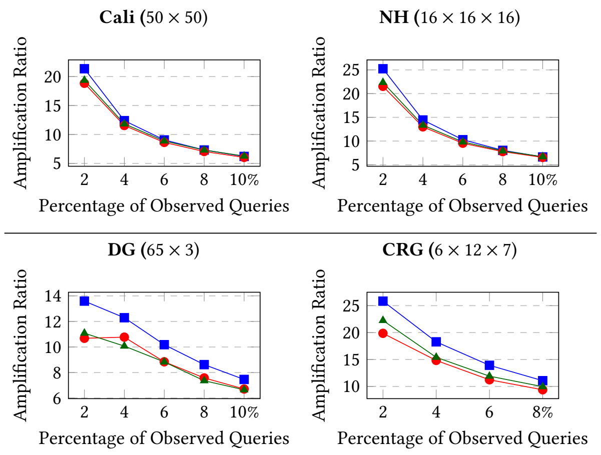

Figure 7: Leakage amplification ratio for Uniform(—■—), Beta(2,1) (—●—) and Gaussian (—▲—) distributions.

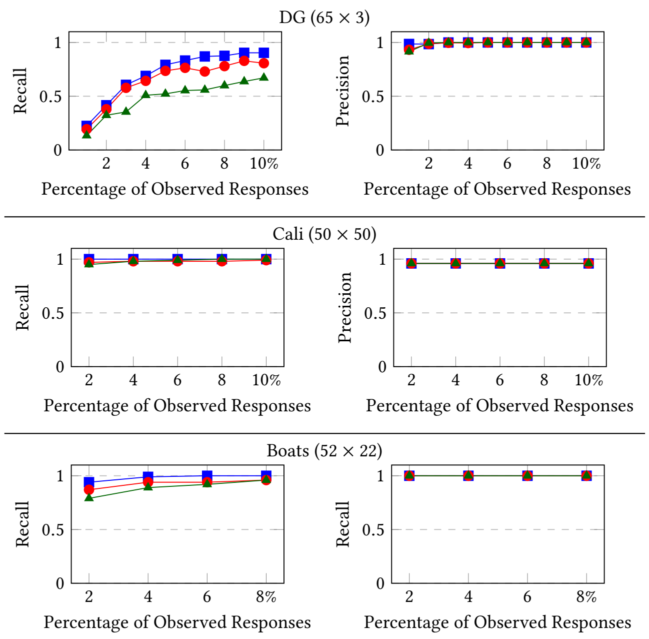

Figure 8: Recall and precision of our attack on 2D datasets as a function of the percentage of all responses observed by the adversary, under different distributions (Uniform (—■—), Beta(2,1) (—●—), Gaussian (—▲—))

our attacks significantly outperform prior work in terms of query complexity, requiring one to two orders of magnitude fewer query responses (Table 2).

Evaluation Metrics. In order to evaluate our attack, we use a variety of metrics. First, we introduce the usage of precision and recall as accuracy metrics in reconstruction attacks that produce an intermediate graph representation of the database. Our attack creates a graph G, where an edge represents our belief that the two graph nodes correspond to neighboring domain points Thus, we define ***precision*** as the fraction of edges in our graph that correctly identify neighboring domain points. Conversely, ***recall*** is defined as the ration of the number of correct edges we have identified to the total number of correct edges.

While precision and recall measure the accuracy of the intermediate graph representation, we further evaluate the final database reconstruction obtained from the graph drawing techniques using two metrics also used in previous work (e.g. [49, 54]): the mean error, also known as the mean absolute error, and the directed Hausdorff distance. The ***mean error*** (ME) is the average distance of our approximation of the value of a record to the original value. The ***Directed Hausdorff distance*** (DH) measures the similarity between the set of original database points, R, and the set of reconstructed points, R. DH is defined as $\max(dh(R, E), dh(E, R))$, where $dh(R, E) = \max_{p \in R}(\min_{q \in E} dist(p, q))$. Note that after computing the graph drawing further processing is needed to scale the drawing to the correct domain size and rotate/reflect it, in order to pick the correct member of the reconstruction space to compare against.

Datasets. We evaluate the effectiveness and complexity of our attack with five real-world datasets and a synthetic one.

DG [58]. The 2D (65×3) DG dataset stores school enrollment data for Massachusetts districts classifying students by gender (male, female, and non-binary). The dataset is dense and has 6,362 student records, each corresponding to 10 students (rounded up).

CGR [59]. The 3D ($6 \times 12 \times 7$) CGR dataset stores school enrollment data from 6 counties of Massachusetts classifying students by grade (1-12) and race. The dataset is dense and has 46,203 student records, each one corresponding to 10 students (rounded up).

Cali [52]. The 2D California dataset contains 21,047 latitude-longitude points of road intersections in California. This dataset has been used in prior work (e.g. [53, 54]). We normalize this dataset to domain (50×50).

Filled-Cali. A variation of the Cali dataset, where we added records filling any "holes" inside the California shape.

NH [63]. This is a real-world 3D elevation dataset containing 4096 elevation points sampled from the United States Geological Survey's Elevation Data of the White Mountains of New Hampshire and normalized to a $16 \times 16 \times 16$ domain.

Boats. This 2D dense dataset with domain 52×22 contains the number of recreational boating accidents (in the hundreds) per US state per year (from 2000-2021). The reports come from the Bureau of Transportation Statistics.

Grid. This 2D database is a synthetic dataset that contains exactly one record per domain value.

Implementation. We have implemented our attack in Python, relying extensively on the Numpy [37], Scipy [68] and NetworkX [36] libraries. We used the tsmpy library [67] for Tamassia et al. [64, 66]. Our experiments ran on a computing grid.

Leakage Amplification. In Figure 7, we evaluate our leakage amplification method (Algorithm 1) against four datasets. We plot the amplification ratio which is the number of unique observed and generated responses over the number of unique responses observed. We note that the method is very successful in practice. In all four cases, after observing a small percentage of queries, we increase the number of responses by an order of magnitude.

Graph Reconstruction. We evaluate our attack on reconstructing our datasets under a Uniform, Beta(2,1) and Gaussian query

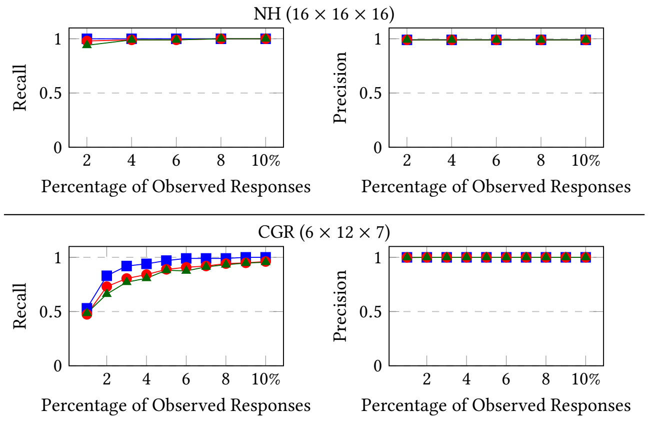

Figure 9: Recall and precision of our attack on the 3D datasets as a function of the percentage of all responses observed by the adversary under three query distributions (Uniform(—■—), Beta(2,1) (—●—), Gaussian (—▲—)).

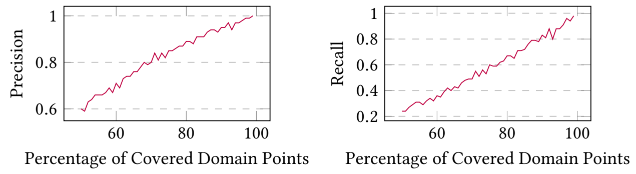

Figure 10: We use the Grid(25×25) dataset to show the effectiveness of our attack. We randomly remove records from the Grid database (horizontal axis) and measure the precision, and recall of Algorithm 5 on these databases of different density. The adversary observes 20% (uniformly at random) of the responses on all runs.

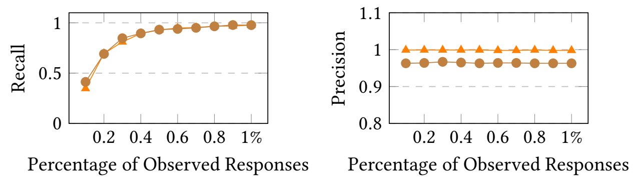

Figure 11: Recall and precision of our attack as a function of the percentage of all responses observed by the adversary (Filled Cali —▲—, Cali —●—), under a uniform query distribution.

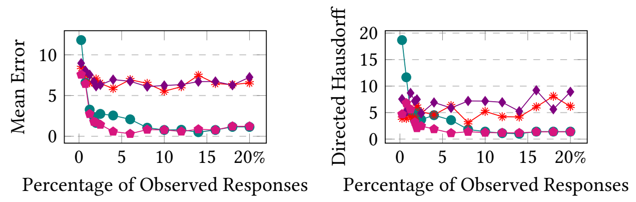

Figure 12: Mean Error and DH for Grid Database(20×20) under a uniform query distribution (Tamassia et al. [64, 66](—●—), Kamada Kawai [38] (—⬟—), Chrobak & Payne [12] (—◆—), Fruchterman Reingold [25] (—✱—)).

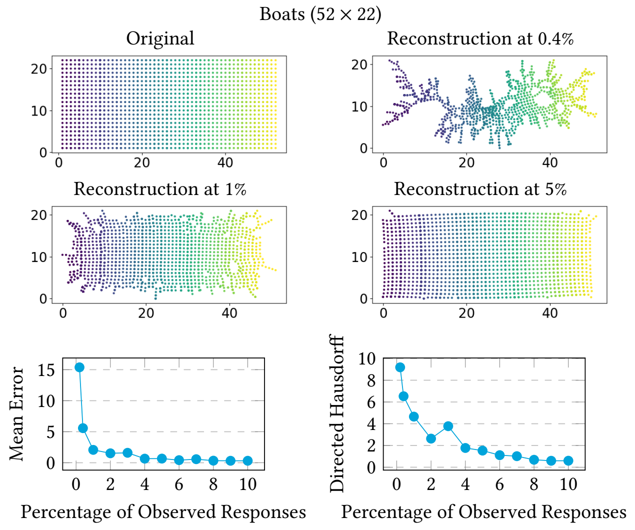

Figure 13: Reconstruction for the Boats Database under a uniform query distribution. We plot the original dataset and our reconstruction using the Kamada Kawai graph drawing technique.

distribution. In Figure 8, we see the precision and recall of our attack against the 2D datasets. As expected, the attack works best under a uniform query distribution and worst under the Gaussian distribution. This is due to the fact that the Gaussian distribution samples queries that cover the edge of the domain less frequently, and thus the attack does not have enough information to recover the neighboring relationships towards the edges of the domain. We observe similar results in the 3D datasets in Figure 9. In Figure 11, we evaluate our attack when reconstructing the Cali dataset, and a variant of it, where we added records filling any "holes" inside the California shape, which we call Filled-Cali. We note that our attack performs approximately the same with regards to recall, but we can only achieve 100% precision on the filled version of Cali. This is expected, as our attack cannot identify such holes. In order to further examine the effectiveness of our database reconstruction attack on databases of different density, we use the Grid dataset. We evaluate the effectiveness of our attack after removing points uniformly at random from the dataset. We observe that our recall deteriorates much faster than the precision as the percentage of empty domain points increases (Figure 10).

Graph Drawing. In Figure 12, we reconstruct the grid database (20×20) and then evaluate four different graph drawing techniques based on mean error and directed Hausdorff distance. We observe that the Tamassia et al. [64, 66] and Kamada Kawai [38] methods outperform the Chrobak & Payne [12] and Fruchterman Reingold [25] methods. Since the Kamada Kawai works in higher than two dimension and does outperform the Tamassia et al. [64, 66] method for smaller percentages of observed responses, we used it for the rest of our evaluation. In Figure 13, we evaluate our reconstruction of the Boats dataset. We observe that with as little as 1% of the observed responses (under a uniform query distribution) we achieve a mean error of change≈ 2 and below. Visually inspecting the reconstruction, we see that the core of the database clearly shows the underlying grid structure and is much better reconstructed than the records towards the edges. In Figure 14, we show our reconstruction of the Boats Dataset under three different query distributions. As expected, our attack performs best under the uniform distribution.

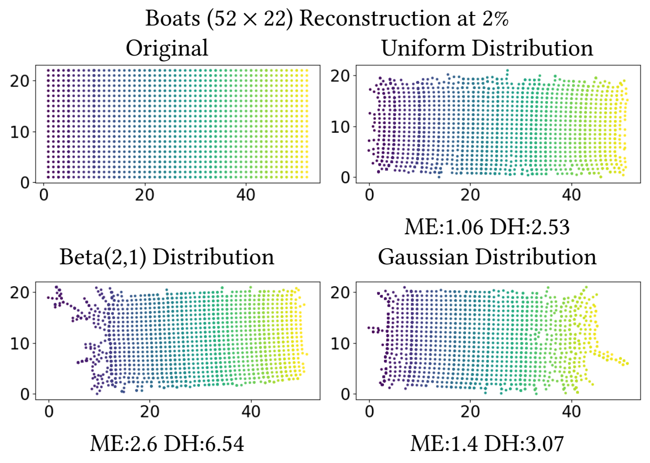

Figure 14: Reconstruction of our Boats Dataset after observing 2% of the responses under three different distributions using the Kamada Kawai graph drawing technique.

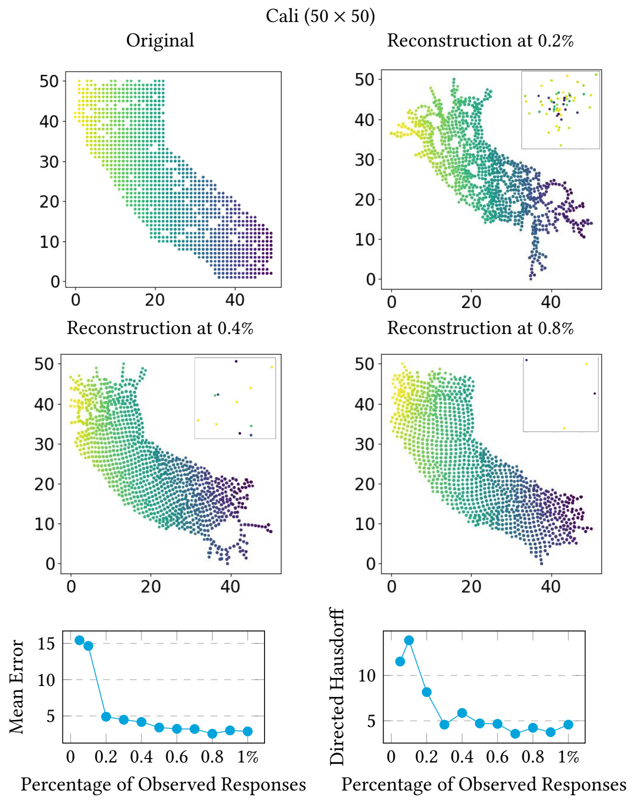

Figure 15: Reconstruction for the Cali (50×50) Database under a uniform query distribution. We plot the original dataset and our reconstruction using the Kamada Kawai graph drawing technique.

The main difference between the three distributions is the reconstruction of the neighboring relationships between nodes in the edges of the database. In Figure 15, we evaluate our reconstruction of the Cali (50 × 50) dataset. We observe that with as little as 0.8% of the responses, we can achieve a mean error of 2.5. In Figure 16 we can see our reconstruction of the NH dataset after observing 2.5% of the responses, achieving a mean error of 1.86. In Figure 17 we show the runtime in seconds of our attack under the three query distributions. Note that depending on the query distribution and the number of queries observed, the graph we construct may be disconnected, often comprising one large connected component plus several small components (typically, single-vertex). In Figure 15, we display these small components separately from the main one. Instead, for clarity, these small components are omitted from Figures 13 and 14.

Comparison with Prior Work. In Table 2, we compare the performance of our attack against prior work. First, for the 2D dense dataset, Boats, both Falzon et al. [23] and Markatou et al. [54] require all possible queries for their attack to work. However, they both return a perfect reconstruction. Our work can achieve a database reconstruction with a mean error of 2.2 after observing 1% of the queries (uniformly sampled). After observing 10% of the queries, our attack can achieve perfect precision and recall, leading to a perfect reconstruction. On the 2D non-dense dataset Cali, Falzon et al. [23] still need all possible queries for the attack to work. However, Markatou et al. [54] achieve database reconstruction with a mean error of 0.03 after observing 33% of the queries. Our attack can achieve a mean error of 2.5% after observing only 0.8% of the queries. Our attack does not significantly improve after observing more responses, achieving at best a mean error of 2.2. Our attack is the only one that can reconstruct the 3D datasets CRG and NH.

Overall, our approach is suitable for practical scenarios where the attacker observes a small fraction of the possible responses. In situations where a large fraction of the queries is available, the attacker can switch to one of the other methods to achieve improved reconstruction.

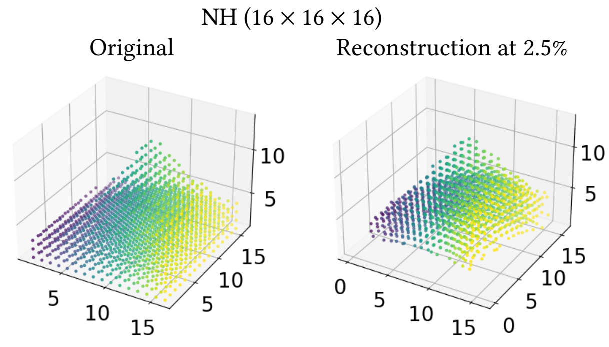

Figure 16: Reconstruction of our 3D Datasets: (Top) NH Reconstruction after observing 2.5% of the observed responses (ME=1.86, DH=5.76). We plot the original dataset and our reconstruction using the Kamada Kawai graph drawing technique.

8 TAKEAWAYS

Dense(r) Databases. In this work, we looked at access pattern leakage in multidimensional databases. Lacharité et al. [51]'s result that dense 1D databases are more vulnerable to leakage-abuse attacks than sparse ones, generalizes in multiple dimensions. Theorem 1 shows the a d-dimensional database could have a reconstruction space exponential to the number of records in it. However, a

Dataset Details		Boats			Cali				CRG	NH
Possible Queries		348,634			1,625,625				45,864	1,942,080
Domain		52×22			50×50				6×12×7	16×16×16
% Queries (uniformly sampled)		1%	10%	100%	0.8%	10%	33%	100%	3%	2.5%
Falzon et al. [23]	ME	-	-	0	-	-	-	0	-	-
Full	DH	-	-	0	-	-	-	0	-	-
Markatou et al. [54]	ME	-	-	0	-	-	0.03	0	-	-
Approximate	DH	-	-	0	-	-	0.3	0	-	-
Our Work	ME	2.2	0	0	2.5	2.2	2.2	2.2	1.2	1.9
Approximate	DH	4.5	0	0	4.2	3.6	3.6	3.6	2.2	5.7

Table 2: Comparison of our reconstruction attack with [23, 54], assuming access and search pattern leakage.

dense d-dimensional database's reconstruction space is independent of the number of records and only depends on d, making dense databases more prone to successful reconstruction attacks. Our attack is much more effective against denser databases, as range queries on dense databases reveal neighboring relationships. For example, given a record in a dense 2D database (i, j), it has four neighbors at $(i - 1, j)$, $(i, j - 1)$, $(i + 1, j)$ and $(i, j + 1)$. In this case there are four range query responses that contain (i, j) and have size 2. Each one of them reveals a neighbor. Now, if the database was not dense, we could observe way more responses of size 2 that contain (i, j). These responses would reveal local information, but not these direct neighboring relations.

Our attack is heuristic in nature. We assume that if we observe a response with two records (r_0, r_1), these records' values are "close", but we cannot tell how close they are. Thus, local information, like the "holes" in the original Cali dataset are lost in our reconstruction. This is why we can achieve full database reconstruction against dense datasets, but there is a limit to how well we can reconstruct non-dense ones. For example, even with all possible queries, we still have a mean error of 2.2 on the Cali dataset. Generally, any response (r_0, r_1) that does not correspond to neighboring records (at distance 1) lowers the maximum accuracy our attack can achieve.

Leakage Amplification. In this work, we show that observed leakage can be amplified to create new leakage. We are able to exploit geometric information about the nature of the queries, allowing us to generate responses that the user has never asked for or even seen. Recall that we don't create new information, but we derive these responses from what we have observed already. Unless a user is aware of the possible attacks to their system, they have no way of knowing what leakage their adversary has access to. The leakage amplification methods we developed can be quite effective, especially when little leakage is available, sometimes generating 20× more responses than observed.

Graph Drawing. Our attack generates a graph to represent neighboring relations between records of the database. Many decades of research have been invested in graph drawing techniques, finding the best ways to embed a graph. These techniques allowed us to elevate our graph representation to a database reconstruction. It is important to note that this is only a starting point, graph drawing algorithms specifically tailored to database reconstructions, could lead to even better reconstructions.

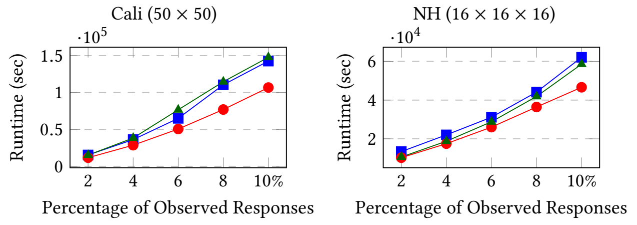

Figure 17: Attack runtimes in seconds under three query distributions (Uniform(—■—), Beta(2,1) (—●—), Gaussian (—▲—)).

Graph drawing techniques could be applicable to leakage-abuse attacks from other types of geometric queries, e.g. k-nearest neighbor queries or radius queries. This is because we can use graphs to represent relations between the records, given by geometric queries (e.g. these two record values are in the same circle), and then graph drawing can transform these relations into values. Additionally, a promising topic of further research is using graph drawing techniques for attacks on schemes that leak the structure pattern, e.g., the schemes of [24], by drawing graphs representing relations between search tokens. The attacks of [53] build and analyze such graphs but do not employ graph drawing algorithms. Graph drawing techniques could lead to general methods that transform a full database reconstruction attack into an approximate one.

Precision and Recall. We introduce the use of precision and recall as metrics to evaluate database reconstruction attacks. In our attack, these metrics provide us with important insights of the effectiveness of our attack at a preliminary step. They allow us to determine how well we deduced relations between record values early, before these relations are used to reconstruct the actual values. An interesting future direction is determining how precision and recall could be used on other leakage abuse attacks. For example, some of the attacks in Markatou et al. [53] construct a graph representation of the underlying data structure used by the scheme. Similarly, Markatou et al. [54] construct a graph representation of the dominance relations between the record values in order to reconstruct the order of the records.

Leakage Profiles. Markatou et al. [53] attack a number of schemes from Falzon et al. [24], achieving full database reconstruction against all of them, and approximate database reconstruction against the Linear scheme. The leakage of the schemes from [24] depends on the instantiation of the underlying searchable encryption scheme. The concrete instantiations presented in [24] employ SSEs that hide the access pattern and leak the search and volume pattern (see Table 1 in [24]). Thus, these instantiations are not vulnerable to our attack. However, different concrete instantiations of the schemes of [24] based on SSEs leaking the access pattern would be vulnerable to our attack, albeit more efficient in practice than the concrete instantiations of [24].

Comparing our attack with the ones from [53], we note that the attacks in [53] (with the exception of the Linear attack) require access to all possible queries and yield full database reconstruction, whereas our attack works after observing much fewer queries and yields approximate database reconstruction. Additionally, the attacks in [53] are tailored to the class of range schemes built from range-supporting data structures and to a specific concrete instantiation of them (e.g., range tree with best range cover), while our attack is general-purpose and can be launched on any range scheme without customization. This means that in addition to being able to exploit the geometric nature of the queries, Markatou et al. [53] can also exploit knowledge of the data structure and its querying algorithm. Finally, we believe that our leakage amplification technique could accelerate the attacks from [53], especially the Linear attack. Their Linear attack achieves approximate database reconstruction by exploiting range queries of prime size. Our leakage amplification technique could generate more prime-sized queries for the attack by intersecting the observed queries. The effectiveness of this technique will of course depend on the distribution of the queries.

Mitigation Techniques. Mitigating our attack can be tricky. Our attack depends intrinsically in exploiting range queries of size two. Hence, a natural mitigation technique would be to not respond to any queries of size two. Applying this restriction naively, would still allow us to deploy our attack by using the leakage amplification technique and generating such queries ourselves. However, the mitigation could be more sophisticated.

Following [24], we can use a tree-based construction where we stop the tree earlier than the leaf nodes. For example, stop before any ranges have size smaller than 4. In this case, any responses must correspond to ranges greater than 4. Additionally, any intersections of responses also must correspond to ranges greater than 4. With this approach, more granular information is not stored in the encrypted database. This approach is equivalent to making the domain coarser. However, in this case, the attacker can still apply our technique to the coarser domain and reconstruct a coarser database.

9 CONCLUSION

Understanding access pattern leakage from range queries and how it can lead to database reconstruction attacks is an important problem with a long line of research. The first papers in the area showed how to reconstruct one-dimensional databases. Then, came attacks on two-dimensional databases, which are much more difficult to reconstruct. We are the first to reconstruct a d-dimensional database solely from access pattern leakage. We also show that the practically relevant class of dense databases are especially vulnerable to attacks. Our work introduces a systematic technique for expanding leakage by inferring responses to queries not issued. We introduce using graph drawing methods for database reconstruction.

There are a number of open problems prompted by our work. It would be interesting to further explore attacks against sparse d-dimensional databases, as our attacks work best against dense ones. There is strong need for mitigation techniques. Our attacks exploit local information (responses of size 2). It is unclear how to restrict access to these responses, accounting for the leakage amplification, while maintaining usability. A systematic exploration of other graph drawing methods in addition to the development of new graph drawing techniques for leakage-abuse attacks is another open problem. Finally, it is important to explore how exploitable volume leakage is from multidimensional range queries.

10 ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation. Some experiments were conducted using the computational resources of the Center for Computation and Visualization at Brown University. We are grateful to Giuseppe Di Battista for enlightening discussions on graph drawing techniques and for sharing a preliminary version of his work [2]. We would also like to thank Francesca Falzon and Kenny Paterson for insightful discussions on SSE schemes and leakage abuse attacks. This work was done in part while Evangelia Anna Markatou was at Brown University.

REFERENCES

[1] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. 2004. Order preserving encryption for numeric data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD).*

[2] Carlos Alegría, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Fabrizio Grosso, and Maurizio Patrignani. 2023. Unit-length Rectangular Drawings of Graphs. In *Graph Drawing and Network Visualization.*

[3] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. 1998. *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall, USA.

[4] Vincent Bindschaedler, Paul Grubbs, David Cash, Thomas Ristenpart, and Vitaly Shmatikov. 2018. The Tao of Inference in Privacy-Protected Databases. *Proceedings of the VLDB Endowment.*

[5] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. 2009. Order-Preserving Symmetric Encryption. In *Advances in Cryptology–EUROCRYPT.*

[6] Raphael Bost. 2016. Sophos: Forward Secure Searchable Encryption. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security.*

[7] Raphaël Bost, Brice Minaud, and Olga Ohrimenko. 2017. Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS).*

[8] Ulrik Brandes. 2001. *Drawing on Physical Analogies.* Springer-Verlag, Berlin, Heidelberg, 71–86.

[9] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Cătălin Roşu, 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.*

[10] David Cash, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Cătălin Roşu, and Michael Steiner. 2013. Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries. In *Advances in Cryptology–CRYPTO.*

[11] Melissa Chase and Seny Kamara. 2010. Structured Encryption and Controlled Disclosure. In *Advances in Cryptology–ASIACRYPT, International Conference on the Theory and Application of Cryptology and Information Security (Lecture Notes in Computer Science).*

[12] Marek Chrobak and Thomas H Payne. 1995. A linear-time algorithm for drawing a planar graph on a grid. *Inform. Process. Lett.* 54, 4 (1995), 241–246.

[13] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. 2006. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security.*

[14] Ioannis Demertzis, Dimitrios Papadopoulos, Charalampos Papamanthou, and Saurabh Shintre. 2020. SEAL: Attack Mitigation for Encrypted Databases via Adjustable Leakage. In *USENIX Security Symposium.*

[15] Ioannis Demertzis, Stavros Papadopoulos, Odysseas Papapetrou, Antonios Deligiannakis, and Minos Garofalakis. 2016. Practical private range search revisited. In *Proc. ACM Int. Conf. on Management of Data (SIGMOD).*

[16] 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.

[17] Ioannis Demertzis, Charalampos Papamanthou, and Rajdeep Talapatra. 2018. Efficient Searchable Encryption Through Compression. *Proceedings of the VLDB Endowment.*

[18] F. Betül Durak, Thomas M. DuBuisson, and David Cash. 2016. What Else is Revealed by Order-Revealing Encryption?. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS).*

[19] Zachary Espiritu, Marilyn George, Seny Kamara, and Lucy Qin. 2024. Synq: Public Policy Analytics Over Encrypted Data. In *2024 IEEE Symposium on Security and Privacy (S&P).* IEEE Computer Society, 85–85.

[20] Zachary Espiritu, Evangelia Anna Markatou, and Roberto Tamassia. 2022. Time- and Space-Efficient Aggregate Range Queries over Encrypted Databases. *Proceedings on Privacy Enhancing Technologies.*

[21] 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*.
[22] Francesca Falzon, Esha Ghosh, Kenneth G. Paterson, and Roberto Tamassia. 2024. Leakage Inversion: Towards Quantifying Privacy in Searchable Encryption. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
[23] 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 ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
[24] Francesca Falzon, Evangelia Anna Markatou, Zachary Espiritu, and Roberto Tamassia. 2022. Range Search over Encrypted Multi-Attribute Data. *Proceedings of the VLDB Endowment.*
[25] Thomas M.J. Fruchterman and Edward M. Reingold. 1991. Graph drawing by force-directed placement. *Software: Practice and experience* 21, 11, 1129–1164.
[26] Craig Gentry. 2009. *A Fully Homomorphic Encryption Scheme.* Ph. D. Dissertation. Stanford University, Stanford, CA, USA. Advisor(s) Boneh, Dan.
[27] Marilyn George, Seny Kamara, and Tarik Moataz. 2021. Structured Encryption and Dynamic Leakage Suppression. In *Advances in Cryptology – EUROCRYPT*.
[28] Javad Ghareh Chamani, Dimitrios Papadopoulos, Charalampos Papamanthou, and Rasool Jalili. 2018. New Constructions for Forward and Backward Private Symmetric Searchable Encryption. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*.
[29] Esha Ghosh, Seny Kamara, and Roberto Tamassia. 2021. Efficient Graph Encryption Scheme for Shortest Path Queries. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIA CCS)*.
[30] Oded Goldreich and Rafail Ostrovsky. 1996. Software Protection and Simulation on Oblivious RAMs. *Journal of the ACM (JACM)* 43, 3.
[31] Paul Grubbs, Anurag Khandelwal, Marie-Sarah Lacharité, Lloyd Brown, Lucy Li, Rachit Agarwal, and Thomas Ristenpart. 2020. Pancake: Frequency Smoothing for Encrypted Data Stores. In *USENIX Security Symposium*.
[32] Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson. 2018. Pump Up the Volume: Practical Database Reconstruction from Volume Leakage on Range Queries. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
[33] Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson. 2019. Learning to Reconstruct: Statistical Learning Theory and Encrypted Database Attacks. In *Proc. IEEE Symp. on Security and Privacy (S&P)*.
[34] Paul Grubbs, Kevin Sekniqi, Vincent Bindschaedler, Muhammad Naveed, and Thomas Ristenpart. 2017. Leakage-Abuse Attacks against Order-Revealing Encryption. In *IEEE Symposium on Security and Privacy (S&P)*.
[35] Zichen Gui, Oliver Johnson, and Bogdan Warinschi. 2019. Encrypted Databases: New Volume Attacks against Range Queries. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, CCS*.
[36] Aric Hagberg, Pieter Swart, and Daniel S Chult. 2008. *Exploring network structure, dynamics, and function using NetworkX.* Technical Report. Los Alamos National Lab.(LANL), Los Alamos, NM (United States).
[37] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825, 357–362.
[38] Tomihisa Kamada and Satoru Kawai. 1989. An algorithm for drawing general undirected graphs. In *Information Processing Letters*, Vol. 31. 7–15.
[39] Seny Kamara, Abdelkarim Kati, Tarik Moataz, Jamie DeMaria, Andrew Park, and Amos Treiber. 2024. MAPLE: MArkov Process Leakage attacks on Encrypted Search. In *Proceedings on Privacy Enhancing Technologies*.
[40] 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 *IEEE European Symposium on Security and Privacy (EuroS&P)*.
[41] Seny Kamara and Tarik Moataz. 2019. Computationally Volume-Hiding Structured Encryption. In *Advances in Cryptology - EUROCRYPT (Lecture Notes in Computer Science)*.
[42] Seny Kamara, Tarik Moataz, Andrew Park, and Lucy Qin. 2021. A Decentralized and Encrypted National Gun Registry. In *IEEE Symposium on Security and Privacy*.
[43] Seny Kamara and Charalampos Papamanthou. 2013. Parallel and Dynamic Searchable Symmetric Encryption. In *Financial Cryptography and Data Security*.
[44] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. 2012. Dynamic Searchable Symmetric Encryption. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
[45] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. 2016. Generic Attacks on Secure Outsourced Databases. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
[46] Yehuda Koren. 2005. Drawing graphs by eigenvectors: theory and practice. *Computers & Mathematics with Applications* 49, 11.
[47] Evgenios M. Kornaropoulos, Nathaniel Moyer, Charalampos Papamanthou, and Alexandros Psomas. 2022. Leakage Inversion: Towards Quantifying Privacy in Searchable Encryption. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
[48] Evgenios M. Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. 2019. Data Recovery on Encrypted Databases with k-Nearest Neighbor Query Leakage. In *Proc. IEEE Symp. on Security and Privacy (S&P)*.
[49] Evgenios M. Kornaropoulos, Charalampos Papamanthou, and Roberto 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)*.
[50] 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)*.
[51] Marie-Sarah Lacharité, Brice Minaud, and Kenneth G Paterson. 2018. Improved reconstruction attacks on encrypted data using range query leakage. In *Proc. IEEE Symp. on Security and Privacy (S&P)*.
[52] Feifei Li, Dihan Cheng, Marios Hadjieleftheriou, George Kollios, and Shang-Hua Teng. 2005. California Road Network Dataset. Downloaded from http://www.cs.utah.edu/~lifeifei/SpatialDataset.htm.
[53] Evangelia Anna Markatou, Francesca Falzon, Zachary Espiritu, and Roberto Tamassia. 2023. Attacks on Encrypted Response-Hiding Range Search Schemes in Multiple Dimensions. In *Proceedings on Privacy Enhancing Technologies*.
[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 ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
[55] Evangelia Anna Markatou and Roberto Tamassia. 2019. Full Database Reconstruction with Access and Search Pattern Leakage. In *Information Security - 22nd International Conference, ISC (Lecture Notes in Computer Science)*.
[56] Evangelia Anna Markatou and Roberto Tamassia. 2019. Mitigation Techniques for Attacks on 1-Dimensional Databases that Support Range Queries. In *Information Security - 22nd International Conference, ISC (Lecture Notes in Computer Science)*.
[57] Muhammad Naveed, Manoj Prabhakaran, and Carl A. Gunter. 2014. Dynamic Searchable Encryption via Blind Storage. In *2014 IEEE Symposium on Security and Privacy (S&P)*.
[58] Massachusetts Department of Elementary and Secondary Education. 2022. Enrollment by District/Grade/Gender. https://www.doe.mass.edu/infoservices/reports/enroll/default.html?yr=2022.
[59] Massachusetts Department of Elementary and Secondary Education. 2022. Enrollment by School/Grade/Race. https://www.doe.mass.edu/infoservices/reports/enroll/default.html?yr=2022.
[60] 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 ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
[61] Elaine Shi, John Bethencourt, T.H. Hubert Chan, Dawn Song, and Adrian Perrig. 2007. Multi-Dimensional Range Query over Encrypted Data. In *IEEE Symposium on Security and Privacy (S&P)*.
[62] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. 2000. Practical techniques for searches on encrypted data. In *Proceeding IEEE Symposium on Security and Privacy (S&P)*.
[63] United States Geological Survey. Downloaded from https://apps.nationalmap.gov. Elevation Products (3DEP).
[64] Roberto Tamassia. 1987. On Embedding a Graph in the Grid with the Minimum Number of Bends. *SIAM J. Comput.* 16, 3, 421–444.
[65] Roberto Tamassia (Ed.). 2013. *Handbook of Graph Drawing and Visualization (Discrete Mathematics and Its Applications).* Chapman & Hall/CRC.
[66] Roberto Tamassia, Giuseppe Di Battista, and Carlo Batini. 1988. Automatic graph drawing and readability of diagrams. *IEEE Transactions on Systems, Man, and Cybernetics*.
[67] uknfire. 2021. tsmpy: An implementation of orthogonal drawing algorithm in Python. https://github.com/uknfire/tsmpy. Accessed: 2023-01-19.
[68] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17, 261–272.
[69] 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 ACM Symposium on Information, Computer and Communications Security (ASIA CCS)*.
[70] Jiafan Wang and Sherman S. M. Chow. 2022. Forward and Backward-Secure Range-Searchable Symmetric Encryption. In *Proceedings on Privacy Enhancing Technologies*.