# Reconstructing with Less: Leakage Abuse Attacks in Two-Dimensions

Francesca Falzon*
*University of Chicago*
ffalzon@uchicago.edu

Evangelia Anna Markatou*
*Brown University*
markatou@brown.edu

William Schor
*Brown University*
william_schor@brown.edu

Roberto Tamassia
*Brown University*
rt@cs.brown.edu

***Abstract*—Access and search pattern leakage from range queries are detrimental to the security of encrypted databases, as evidenced by a large body of work on efficient attacks that reconstruct one-dimensional databases. Recently, the first attack in two-dimensions showed that higher-dimensional databases are also in danger. This attack requires complete information for reconstruction. In this paper, we develop reconstructions that require less information. We present an order reconstruction attack that only depends on access pattern leakage, and empirically show that the order allows the attacker to infer the geometry of the underlying data. Notably, this attack achieves full database reconstruction when the 1D horizontal and vertical projections of the points are dense. We also give an approximate database reconstruction attack that is distribution-agnostic and works with any subset of the possible responses, given the order of the database. We support our results with experiments on real-world databases with queries drawn from various distributions. Our attack is effective, e.g. we achieve good reconstructions with 15% percent of the queries under a Gaussian distribution.**

***Index Terms*—Cryptography, Encrypted databases, Searchable Encryption, Attacks**

## I. Introduction

The growing adoption of cloud computing and storage in the past two decades has been accompanied by a corresponding increase of data breaches, as data is often stored in plaintext in the cloud. Encrypted cloud storage reduces the risk of such breaches. However, encrypted data has been traditionally hard to search., Searchable encryption provides a practical solution for processing range queries over encrypted data without the need for decrypting the data or the queries (see, e.g., [4], [5], [14], [15] and the survey by Fuller et al. [10]). For the sake of efficiency, searchable encryption schemes sacrifice full security by leaking some information about the queries and their responses. While the security proofs of these schemes prove that nothing is leaked beyond the given "leakage", the underlying data is still vulnerable to inferences from this leakage (see, e.g., [8], [12], [16]–[18], [20]).

The following standard types of leakage occur in searchable encryption schemes. A scheme leaks the ***access pattern*** if the adversary observes the encrypted records returned in response to queries. The ***search pattern*** is leaked if the adversary can distinguish if a query has been previously issued.

*FF and EAM are co-first authors who contributed equally, listed alphabetically.

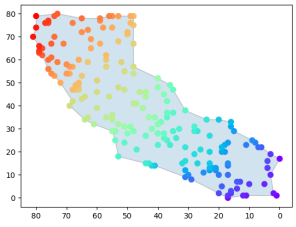



Fig. 1: Reconstruction of 200 randomly chosen points from the California Road Network dataset using only access patterns. We can clearly see that the original location data was in California.

This work considers an encrypted database with two attributes, referred to as a ***two-dimensional (2D) database*** to which ***range queries*** are issued. We assume a passive persistent adversary who observes the ***entire access pattern leakage***, i.e., all possible responses of queries, and a ***subset of the search pattern leakage***. Our adversary aims to reconstruct the order of the database records in the two dimensions (attributes) using solely the access pattern, a problem called ***order reconstruction***. After achieving this goal, the adversary attempts to further perform an approximate reconstruction of the (attribute) values of the database records by using the partial search pattern observed, a problem called ***approximate database reconstruction***.

### A. Contributions

Previous work on reconstruction attacks from range queries on 2D databases assumes that the adversary has knowledge of both the entire access pattern leakage and the entire search pattern leakage [8] and uses both forms of leakage to perform an attack that reconstructs in polynomial time the database record values, up to inherent information theoretic limitations. A natural question left open by this attack is what information is recoverable when given less leakage. In this work, we make progress on this question with the following contributions:

1) We show that order reconstruction faces additional information theoretic limitations when given only access pattern leakage. We incorporate these limitations into a complete formal ***characterization*** of the family of databases that have the same access pattern leakage.
2) We present an ***order reconstruction attack*** that allows an adversary who has observed the entire access pattern leakage to build a linear-space representation of the family of databases in polynomial time.
3) We design a ***distribution-agnostic approximate database reconstruction attack*** that reconstructs record values given the order of the records (e.g., obtained with the

above attack) and partial search pattern leakage from queries issued according to an unknown distribution.

4) We ***experimentally analyze*** the practical effectiveness of our attacks by fully implementing them and deploying them on real-world and synthetic datasets using a variety of range query distributions.
5) We develop new combinatorial and geometric concepts and algorithms related to point reconstruction from range queries that may be of independent interest.

Our work provides the ***first order reconstruction attack in 2D from access pattern leakage*** and the ***first approximate reconstruction attack in 2D from partial search pattern leakage and an unknown query distribution***. Our order reconstruction attack is also a ***full database reconstruction attack*** for the case when the 1D horizontal and vertical ***projections of the points are dense***, i.e., have a record for every domain value.

Our work presents significant improvements over the Full Database Reconstruction (FDR) of [8]. The FDR attack presented in [8] assumes a stronger adversary that has observed all possible queries in the database. Additionally, the FDR attack fails if even one response is missing. In contrast, we demonstrate that an adversary can still infer much about the underlying data values with fewer (and in some cases, significantly fewer) queries observed. In this work, we also tackle a different problem, the problem of order reconstruction using only access pattern and with no knowledge of the query distribution or search pattern.

Our Approximate Database Reconstruction attack can be viewed as the two-dimensional analogue of the work on attacks on 1D databases reported in [17]. To apply previous approximation approaches that assume knowledge of the order to 2D databases, we must completely characterize order reconstruction in 2D. However, much like FDR does not trivially extend from the 1D to 2D setting, our order reconstruction method demonstrates an exponential increase in the number of indistinguishable point configurations in the 2D setting. Thus, we cannot simply generalize 1D techniques to 2D. We re-examine a number of support-size estimators to better suit our problem and build a complex nonlinear system of equations to model the problem instead of the linear system of [17].

### B. Prior and related work

In their seminal work [15], Kellaris et al. show that for a 1D database with domain size $[N]$, one can reconstruct the values of the database records from access pattern leakage of range queries using $O(N^4 \log N)$ queries issued uniformly at random. Since then, a number of works have explored the problem in 1D (e.g. [12], [16]–[18], [20]), and in 2D [8].

Order reconstruction in 1D was first introduced in [15], as the first step of their FDR attack. Grubbs et al. [12] generalize the attack to one that achieves sacrificial $\epsilon$-approximate order reconstruction ($\epsilon$-AOR); the goal of $\epsilon$-AOR is to recover the order of all records, except for records that are either within $\epsilon N$ of each other or within $\epsilon N$ of the endpoints. Their attack achieves sacrificial $\epsilon$-AOR with probability $1-\delta$ given $O(\epsilon^{-1}\log\epsilon^{-1} + \epsilon^{-1}\log\delta^{-1})$ uniform queries.

TABLE I: Comparison of our work with related reconstruction attacks. Dense1D denotes a 2D database whose horizontal and vertical projections are 1D dense databases.

| | Queries | | Assumptions | | Leakage | | Attack | |
|---|---|---|---|---|---|---|---|---|
| | 1D range | 2D range | Query distrib. | Data distrib. | Search pattern | OR | FDR | **ADR** |
| Kellaris+ [15] | ✓ | | Uniform | Any | | ✓ | ✓ | ✓ |
| Lacharité+ [18] | ✓ | | Agnostic | Dense | | ✓ | ✓ | |
| Grubbs+ [12] | ✓ | | Uniform | Any | | ✓ | ✓ | ✓ |
| Grubbs+ [12] | ✓ | | Uniform | Minor | | ✓ | ✓ | ✓ |
| Markatou+ [20] | ✓ | | Agnostic | Any | | ✓ | | |
| Markatou+ [20] | ✓ | | Agnostic | Any | ✓ | | ✓ | |
| Kornaropoulos+ [17] | ✓ | | Agnostic | Any | ✓ | | | ✓ |
| Falzon+ [8] | | ✓ | Agnostic | Any | ✓ | | ✓ | |
| Falzon+ [8] | | ✓ | Known | Any | | | ✓ | |
| **This Work** | | ✓ | **Agnostic** | **Any** | | ✓ | | |
| **This Work** | | ✓ | **Agnostic** | **Any** | ✓ | | | ✓ |
| **This Work** | | ✓ | **Agnostic** | **Dense1D** | | ✓ | ✓ | |

Approximate database reconstruction from access pattern of range queries in 1D has been addressed in [12], [17], [18]. In [18], Lacharité et al. introduce $\epsilon$-approximate database reconstruction ($\epsilon$-ADR) as the reconstruction of each record value up to $\epsilon N$ error; they then give an attack that achieves $\epsilon$-ADR with $O(N\log\epsilon^{-1})$ uniform queries. In [12], the authors further introduce sacrificial $\epsilon$-ADR. The goal of $\epsilon$-ADR is to recover all values up to and error of $\epsilon N$, while "sacrificing" recovery of points within $\epsilon N$ of the domain end points. Concepts from statistical learning theory are applied to achieve a scale-free attack that succeeds with $O(\epsilon^{-2}\log\epsilon^{-1})$ queries.

In a different vein, Kornaropoulos et al. [17] reconstruct a 1D database without knowledge of the underlying query distribution and without having observed all possible queries by employing statistical estimators to approximate the support size of the conditional distribution of search tokens given a particular response. Their agnostic reconstruction attack achieves reconstruction with good accuracy in a variety of settings including and beyond the uniform query distribution.

Full database reconstruction in two-dimensions was first described in [8]. In this work, Falzon et al. describe the symmetries of databases in two dimensions, prove that the set of databases compatible with a given access pattern leakage may be exponential, and give a polynomial-time algorithm for computing a polynomial-sized encoding of the (potentially exponential) solution set.

Table I compares our results with previous work.

## II. Preliminaries

We recall combinatorial and geometric concepts using the terminology and notation introduced in [8].

**Notation.** We denote with $[N]$ the set of integers $\{1,\ldots,N\}$. We define the domain of a two-dimensional (2D) database to be $\mathcal{D} = N_0 \times N_1$ for positive integers $N_0$ and $N_1$. We refer to the points on the segment from $(0,0)$ to $(N_0+1, N_1+1)$ as the ***main diagonal***. Given a point $w \in \mathcal{D}$ we denote $w_0 \in [N_0]$ as its first coordinate and $w_1 \in [N_1]$ as its second coordinate i.e. $w = (w_0, w_1)$. We say that $w$ ***dominates*** $x$ if $x_0 \le w_0$ and $x_1 \le w_1$. This is denoted as $x \preceq w$. Similarly, we say that $w$ ***anti-dominates*** $x$ if $w_0 \le x_0$ and $x_1 \le w_1$. This is denoted as

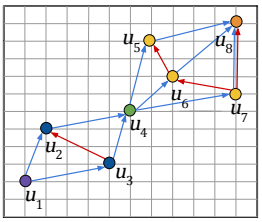



Fig. 2: Example of dominance graph (blue) and anti-dominance graph (red) for a point set with components $\{u_1\}$, $\{u_2, u_3\}$, $\{u_4\}$, $\{u_5, u_6, u_7\}$, and $\{u_8\}$.

$x \preceq_a w$. The dominance or anti-dominance is said to be ***strict*** if the above inequalities are strict. We say that $w$ ***minimally (anti-) dominates*** $x$ if there is no point $v \neq w, x$ such that $w$ (anti-) dominates $v$ and $v$ (anti-) dominates $x$.

A 2D database, $D$, over a domain $\mathcal{D}$ with $R \geq 1$ records is defined as an $R$-tuple of points in $\mathcal{D}$ i.e. $D \in \mathcal{D}^R$. Each tuple in $D$ is referred to as a ***record*** and each record is associated with a unique ***identifier*** (or ID) in $[R]$ that refers to its index in the tuple. We let $D[j]$ for some $j \in [R]$ denote the domain value associated with the record ID $j$. When clear from context, we may refer to records as points. We denote a digraph as $G = (V, E)$ such that $V$ is the vertex set and $E$ is the directed edge set. For any two vertices $u, v \in V$ we denote a directed edge from $u$ to $v$ as the pair $(u, v)$ (Figure 2).

The following definitions are illustrated in Figure 2.

**Definition 1.** *The **dominance graph**, $G = (V, E)$, of a set of points $S$, is the digraph where $V = S$ and $(a, b) \in E$ if $b$ minimally dominates $a$, $a, b \in V$.*

**Definition 2.** *The **anti-dominance graph**, $G' = (V', E')$, of a set of points $S$, is the digraph where $V' = S$ and $(a, b) \in E'$ if $b$ minimally anti-dominates $a$, $a, b \in V'$.*

**Definition 3.** *We define a **component**, $C$, of database $D$ to be a minimal non-empty subset of $D$ such that for any points $p \in C$ and $q \notin C$ either $p$ and its reflection along the main diagonal both dominate $q$ or are both dominated by $q$.*

**Range queries and leakage.** A ***range query*** is defined by a pair of domain points $q = (c, d) \in \mathcal{D}^2$ such that $c \preceq d$. The ***response*** or ***access pattern*** of a range query is the set of identifiers of records with values that fall within the range of the query. The response of a query $q = (c, d)$ is defined to be

$$\mathsf{Resp}(D, q) = \{j \in [R] \ : \ c \preceq D[j] \preceq d\}.$$

We similarly define the ***response multiset of a database*** $D$, denoted $\mathsf{RM}(D)$, as the ***multiset*** of all access patterns of $D$:

$$\mathsf{RM}(D) = \{\{\mathsf{Resp}(D, q) : q = (c, d) \in \mathcal{D}^2, c \preceq d\}\}.$$

We use the double bracket notation to emphasise that this is a multiset since distinct queries $q, q'$ may produce the same response, $\mathsf{Resp}(D, q) = \mathsf{Resp}(D, q')$. We define the ***response set*** of $D$, denoted $\mathsf{RS}(D)$, to be the corresponding set in which each response appears exactly once. The ***search pattern*** of a query $q = (c, d)$ is defined to be a query-specific token $\mathsf{SP}(D, q) = t$ where $t \in \left[\binom{N_0+1}{2}\binom{N_1+1}{2}\right]$. We assume a one-to-one correspondence between queries and tokens.

**Threat model.** We study the security of encrypted database schemes that support two-dimensional range queries and which leak the access pattern and search pattern of each query. We consider an ***honest-but-curious***, ***persistent*** adversary that has compromised the database management system or the client-server communication channel, and can observe the leakage over an extended period of time. Our order reconstruction attack considers an adversary that takes $\mathsf{RS}(\mathcal{D})$ as input and wishes to compute the order of all records. Our other attack considers an adversary that knows the order and some subset of the possible search tokens and wishes to approximate the domain value of each record.

**Assumptions and reconstruction attacks.** We explore reconstruction under a few different assumptions. In Section V we assume the adversary knows the size of the domain $N_0 \times N_1$, and the full response set $\mathsf{RS}(D)$. In Section VII we make no assumption about the number of queries that an adversary may have observed. We make no assumption about the distribution from which queries are drawn; we consider an adversary that has no knowledge of the distribution.

We define the ***Order Reconstruction*** (OR) problem as follows:

**Definition 4.** ***OR:*** *Given a set $\mathsf{RS}(D)$ of some database $D$, compute all pairs of dominance and anti-dominance graphs $(G, G')$ such that any database $D'$ with record relationships defined by $(G, G')$ is equivalent to $D$ with respect to the response set, i.e. $\mathsf{RS}(D) = \mathsf{RS}(D')$.*

Computing $(G, G')$ is the information theoretic best that an adversary can do without additional information (e.g without knowing the multiplicities of each response, or knowing the distribution of the data). We define the problem of ***Approximate Database Reconstruction*** (ADR) as follows:

**Definition 5.** ***ADR:*** *Given the order of points in $D$ and some subset of $\mathsf{RM}(D)$, approximate the values of the records.*

### A. Query Densities

We use the generalized notion of computing query densities in two-dimensions as presented in [8]. Their work extends the methods in [15] for computing the number of unique queries that a given set of points must match. By observing sufficiently many query responses of iid uniformly random queries, one may recover the value of a point $x$ by computing the proportion of responses that the identifier of $x$ appears in. In one-dimension, the function that computes the query density is quadratic and symmetric around the midpoint of $[N]$, and thus a given domain value $x$ can be computed up to two possible solutions: $x$ and its reflection across the midpoint [15]. We present the generalized definition of query density of one and multiple points in two-dimensions below.

**Definition 6** ( [8] ). *Let $\mathcal{D} = [N_0] \times [N_1]$, and $x \in \mathcal{D}$, define*

$$\rho_x = \left|\{(c, d) \in \mathcal{D}^2 : c \preceq x \preceq d\}\right|$$

*and for a set of points $S \subseteq \mathcal{D}$ define*

$$\rho_S = \left|\{(c, d) \in \mathcal{D}^2 : \forall x \in S, \ c \preceq x \preceq d\}\right|.$$

*Thus, these are the number of queries that contain $x$ or all points in $S$ (respectively).*

Given a point $x = (x_0, x_1) \in \mathcal{D}$, the formula for computing $\rho_x$ is as follows.

$$\rho_x = x_0 \cdot x_1 \cdot (N_0 + 1 - x_0) \cdot (N_1 + 1 - x_1) \quad (1)$$

More generally, the query density $\rho_S$ of a set of points $S \subseteq \mathcal{D}$ is as follows.

$$\rho_S = (\min_{x \in S} x_0) \cdot (\min_{y \in S} y_1) \cdot (N_0 + 1 - \max_{z \in S} z_0) \cdot (N_1 + 1 - \max_{w \in S} w_1) \quad (2)$$

## III. Order and Equivalent Databases

Before developing our attacks, we present our results on the information-theoretic limitations of order reconstruction.

### A. Equivalent Databases

**Definition 7.** *Databases $D$ and $D'$ are **equivalent with respect to the response multiset** if $\mathsf{RM}(D) = \mathsf{RM}(D')$ and **equivalent with respect to the response set** if $\mathsf{RS}(D) = \mathsf{RS}(D')$.*

As shown in [8], given some database $D$ we can generate a database $D'$ that is equivalent with respect to the response multiset by rotating/reflecting $D$ according to the symmetries of the square and by independently flipping the reflectable components across the main diagonal.

**Proposition 1.** [8] *Let $D$ be a database from domain $\mathcal{D}$ that contains components $C_1$ and $C_2$. Let $D'$ be a database containing $C_1$ and $C_2'$, where each point $p \in C_2'$ is the reflection of some point $p' \in C_2$ along the diagonal. We have that databases $D$ and $D'$ are equivalent with respect to the response set, i.e., $\mathsf{RS}(D) = \mathsf{RS}(D')$.*

Note that if $D$ and $D'$ are equivalent with respect to the response multiset, then they are equivalent with respect to the response set. However, the converse is not necessarily true. We show in Propositions 2 and 3 (Figure 3) that there are two additional symmetries that produce equivalent databases with respect to the response set as described

**Definition 8.** *A pair of points $(p, q)$ of a database $D$ is an **antipodal pair** if for every point $r \in D - \{p, q\}$ we have (1) $q_1 < r_1 < p_1$ and (2) either $r_0 < \min(p_0, q_0)$ or $r_0 > \max(p_0, q_0)$. See Figure 3b.*

**Definition 9.** *A pair $(p, q)$ of points of a database $D$ are said to be a **close pair** if $q$ minimally dominates $p$, and there exists no point $r \in D - \{p, q\}$ such that $r$ anti-dominates $p$ or $r$ is anti-dominated by $q$ or $r$ is between $p$ and $q$. See Figure 3c.*

The following proposition, illustrated in Figure 3b, shows that one cannot infer the horizontal ordering of an antipodal pair from the response set.

**Proposition 2.** *Let $D$ be a database from domain $\mathcal{D}$ that contains an antipodal pair $(p, q)$. Let $V$ be the widest vertical strip of points of $\mathcal{D}$ that contains $p$ and $q$, and let $P$ and $Q$ be the tallest horizontal strips of $V$ containing $p$ and $q$, respectively, but no other point of $D$. Let $D'$ be the database obtained from $D$ by replacing $p$ with another point, $p'$, of $P$ and $q$ with another point, $q'$, of $Q$. We have that databases $D$ and $D'$ are equivalent with respect to the response set, i.e., $\mathsf{RS}(D) = \mathsf{RS}(D')$.*

The proof of Proposition 2 can be found in Appendix C.By Proposition 2, the two points of the antipodal pair $(p, q)$ of $D$ and of the corresponding antipodal pair $(p', q')$ of $D'$ can be ordered, reverse ordered, or collinear in the horizontal dimension and these three distinct orderings cannot be distinguished from the response set $\mathsf{RS}(D)$.

**Proposition 3.** *Let $D$ be a database from domain $\mathcal{D}$ that contains a close pair $(p, q)$. Let $D'$ be the database obtained from $D$ by replacing $q$ with any point $q'$ such that $q'_0 = q_0$ and $p_1 \leq q'_1 \leq q_1$. Then $D$ and $D'$ are equivalent with respect to the response set, i.e., $\mathsf{RS}(D) = \mathsf{RS}(D')$.*

The proof of Proposition 3 may be found in Appendix C.

**Definition 10.** *Let $D$ be a database and let $G$ and $G'$ be the dominance and anti-dominance graphs of $D$, respectively. We define $\mathsf{E_o}(D)$ as the set of all possible point orderings of databases equivalent to $D$ with respect to response set, $\mathsf{RS}(D)$.*

Combining Propositions 1, 2 and 3, we capture all the information-theoretic limitations of order reconstruction.

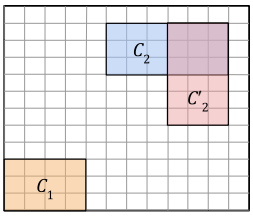
**Theorem 1.** *Let $D$ be a two-dimensional database. The set of point orderings $\mathsf{E_o}(D)$ can be obtained from the dominance graph $G$, the anti-dominance graph $G'$, the antipodal pair (if it exists), and the set of close pairs of $D$ by means of the following transformations:*

1) *Flipping the direction of $G$ and/or a subset of components of $G'$ according to Proposition 1.*
2) *If $D$ contains an antipodal pair, add or remove one or two edges from $G$ or $G'$ to make the pair collinear or switch their relationship from strict dominance to strict anti-dominance or vice versa.*
3) *For each close pair in $D$, add or remove one or two edges from $G$ or $G'$ to make them collinear or put them in a strict dominance relationship.*

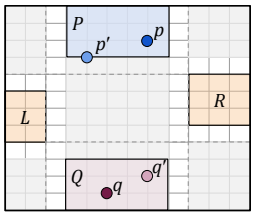
We prove Theorem 1 in Section IV-F. Note that the equivalent configurations of Propositions 2 and 3 arise only with respect to the response set. The multiplicity information from the response multiset provided by the search pattern leakage resolves them. Indeed, Theorem 1 adds transformations (2) and (3) to transformation (1) given in [8].

### B. Chains and Antichains

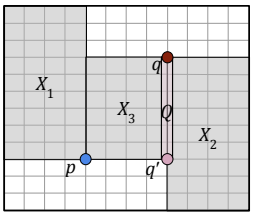
Our order reconstruction algorithm uses the concepts of chains and antichains of the dominance and anti-dominance relations for points in the plane [9], [25]. A set of points $S \subseteq \mathcal{D}$ is a ***chain*** if any two points $x, w \in S$ are in a dominance relationship i.e. $x \preceq w$ or $w \preceq x$. A subset of points $A \subseteq \mathcal{D}$ is an ***antichain*** if for any two points $x, w \in A$ neither $x \preceq w$ nor $w \preceq x$. Let $D \subseteq \mathcal{D}$ be a set of points. The ***height*** of a point $x \in D$ is the length of the longest chain in $D$ with $x$ as the maximal element. Note that two points of the same height cannot have a dominance relation. Thus, the set of all points in $D$ with the same height yields a partition $\mathcal{A}$ of $D$




(a) Illustration of Definition 3 and Proposition 1. $C_1$ and $C_2$ are components of $D$. Flipping $C_2$ along the diagonal yields an equivalent database with respect to the response multi-set.




(b) Illustration of Definition 8 and Proposition 2. Points $p$ and $q$ are an antipodal pair. Each remaining point is in $L$ or $R$. Replacing $p$ with $p' \in P$ and $q$ with $q' \in Q$ gives an equivalent database with respect to the response set.




(c) Illustration of Definition 9 and Proposition 3. Points $p$ and $q$ are a close pair. There are no points in regions $X_1$, $X_2$ or $X_3$. Replacing $q$ with any $q' \in Q$ yields an equivalent database with respect to the response set.

Fig. 3: Examples of transformations that yield equivalent databases with respect to the response set (Definition 7).

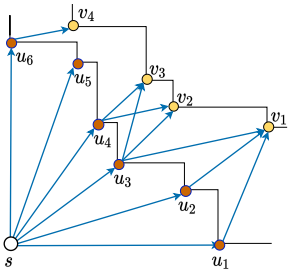



Fig. 4: An example of a dominance graph and its associated canonical antichain partition comprising antichains $A_0 = \{s\}$, $A_1 = \{u_1, \cdots u_6\}$, and $A_2 = \{v_1, \cdots v_4\}$.

into antichains, namely the ***canonical antichain partition***. We denote the canonical antichain partition by $(A_0, A_1, \ldots, A_L)$ where $A_i$ is the set of points at height $i$.

Let $D$ be a database and let $(G, G')$ be the dominance and anti-dominance graphs of $D$. Now note that the paths in the dominance graph correspond to chains in $D$. Formally, if $(u_1, u_2, \ldots, u_\ell)$ is a path of record IDs in $G$, then

$$D[u_1] \preceq D[u_2] \preceq \cdots \preceq D[u_\ell]$$

and $\{D[u_1], D[u_2], \ldots, D[u_\ell]\}$ forms a chain in $D$. By definition the edges of $G$ represent the minimal dominance relations of the points in $D$ and thus determining the length of a longest possible path in $G$ from a source to $u \in [R]$ is equivalent to determining the height of point $D[u]$ in the database. This gives us a nice way of partitioning the IDs such that the partition corresponds to the canonical antichain partition. Formally, if $s$ is a source of $G$ then $D[s]$ has height 0. And if $S_i \subseteq [R]$ is the set of IDs in $G$ that have a maximum distance of $i$ from any sink, then the canonical antichain partition of $D$ is given by

$$A_i = \{D[a] \ : \ a \in S_i\}.$$

For an example, see Fig. 4. Since $G$ is acyclic we can compute these longest paths efficiently. For convenience we may use $A_i$ to instead refer to the IDs of points within each partition of the canonical antichain.

These observations are crucial in the design of our OR algorithm. E.g., we construct the dominance graph starting at the IDs of points with height 0. We then compute the partition on IDs that corresponds to the canonical antichain partition and use the partition to construct the antidominance graph.

## IV. Overview of Order Reconstruction

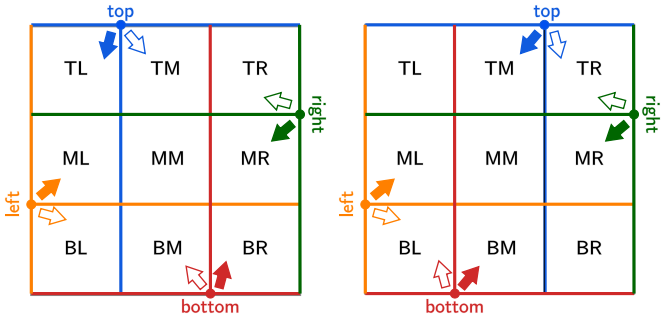
A high-level intuitive explanation for our order reconstruction algorithm is schematically illustrated in Figure 5, where we show a database that has distinct extreme points left, right, top and bottom. We assume, without loss of generality, that right dominates left. The two parts of the figure distinguish the cases where top is to the left or right of bottom, respectively. By symmetry, these two cases cover all the possible configurations of the extreme points. For simplicity, we assume that none of the remaining points are horizontally or vertically aligned with each other or the extreme points. Thus, only the four extreme points are on the boundary of the rectangle occupied by the database points. The order reconstruction algorithm presented in the next section will remove all these simplifying assumptions and reconstruct an arbitrary database. A first building block of our order reconstruction algorithm finds such extreme points from the response set.

### A. Partition of the Database into Regions

By drawing horizontal and vertical lines through the extreme points, we partition the database points into nine regions labeled XY for $\mathsf{X} \in \{\mathsf{T}, \mathsf{M}, \mathsf{B}\}$ and $\mathsf{Y} \in \{\mathsf{L}, \mathsf{M}, \mathsf{R}\}$, where T, B, L, R, and M stand for top, bottom, left, right, and middle, respectively. Note that some of these regions may be empty. We can compute the points in each region from the response set by finding minimal responses that contain certain pairs and triplets of extreme points and performing intersections and differences of such responses with each other and the entire database. We show how to compute the rows and columns, from which a region can be computed by intersecting its row with its column. The middle row and column are the minimal response containing left and bottom and the minimal response containing top and bottom, respectively. The other rows and columns are obtained by computing the minimal response containing the triplet of extreme points opposite to the column and subtracting this response from the database. For example, the left column is obtained by subtracting from the database the minimal response containing top, right, and bottom.

### B. Dominance and Anti-Dominance With a Corner

Consider a subset $S$ of the database containing a dominance corner, $s$, defined as a point that dominates or is dominated by all other points of $S$. For example, point left is a dominance corner for the points in region ML in Figure 5a. Another building block of our algorithm is a method that given $S$ and $s$, computes all pairs of points of $S$ that have a dominance




(a) top to the left of bottom (b) top to the right of bottom

Fig. 5: Partition of the database points into nine regions induced by the four extreme points.

relation. By symmetry, the same methods computes the anti-dominance relation pairs for a subset of points that admits a similarly defined anti-dominance corner. Let $s$ be a dominance corner for $S$ and assume $s$ is dominated by all the other points. The method considers for each point $v$ of $S$, the smallest response containing points $s$ and $v$. We have the the points of $S$ in this response are the points of $S$ dominated by $v$. For example, in the point set of Figure 4, we have that point $s$ is a dominance corner. Also, the smallest response containing $s$ and $v_3$ is $\{s, u_3, u_4, v_3\}$, which implies that the points dominated by $v_3$ are $s$, $u_3$ and $u_4$.

### C. Points in Different Rows and Columns

Consider two points, $p$, and $q$. For some placement of these points into regions, namely when they are in regions in different rows and columns, we can immediately decide their horizontal and vertical order and thus whether they are in a dominance or anti-dominance relation. For example, if $p$ is in BL and $q$ is in MM, MR, TM, or TR, then we have that $q$ is above and to the right of $p$ and thus dominates $p$. Also, if $p$ is in BM and $q$ is ML or TL, then we have that $q$ is above and to the left of $p$ and thus $q$ anti-dominates $p$. Similar considerations hold for other placements of $p$ and $q$ in different rows and columns.

### D. Points in Different Regions in Same Row or Column

Consider now the case when $p$ and $q$ are in different regions that share the same row or column. In this case, we know one of the horizontal or vertical ordering of the points, but not the other. Let $p$ be in TL and $q$ be in TR. We have that $p$ is to the left of $q$. We can use our building block method applied to the points in the top row and their anti-dominance corner right to determine whether $p$ and $q$ are in anti-dominance relation. If they are not, given that $p$ is to the left of $q$, we conclude that $q$ dominates $p$. The same reasoning holds when $p$ is in TL and $q$ is in TM and, more generally, by symmetry, for $p$ and $q$ in contiguous regions of the same row or column.

### E. Points in Same Region

We now turn to the case when $p$ and $q$ are in the same region. Here, we need to take into account the configurations of the extreme points. We distinguish the cases when top is to the left bottom (Figure 5a) and top is to the right of bottom (Figure 5b). It is worth noting that we can distinguish these two cases from the response set only if there is at least a point in the middle column. Otherwise, top and bottom are an antipodal pair (Definition 8 and Proposition 2).

In the case of Figure 5a, each region is included in a group of regions that has a dominance corner and another group of regions that has an anti-dominance corner. For example, suppose $p$ and $q$ are in TL, TM, ML, or MM. We have that left is a dominance corner for the top two rows and bottom in an anti-dominance corner for the left two rows. Applying our building block method to these two groups of regions, we determine whether $p$ and $q$ are in dominance or anti-dominance relation. In the case of Figure 5a, we can use the same approach for all regions except MM.

To deal with the remaining case of $p$ and $q$ within region MM in the configuration of Figure 5b, we observe that using dominance corner top or bottom, we can determine if $p$ and $q$ are in dominance relation. If so, we are done, else, we find the extreme points of MM and apply the order reconstruction algorithm recursively to the points within this region.

### F. Proof of Theorem 1

*Proof.* Let $D$ be a database and let left, right, top, and bottom be its four extreme points. Without loss of generality, these points must take one of the two configurations pictured in Figure 5. Note, any point's relative order can be determined if it is in a dominance relation with one point and in an anti-dominance relation with another point. If a point is not in such a relation, then we argue that the three transformations yield all databases equivalent to $D$ with respect to the response set.
*Case 1:* If top and bottom are antipodal, we have the configuration of Figure 5a or Figure 5b with an empty middle column and the ordering of all pairs of points is determined with the exception of the antipodal pair (Transformation 2).
*Case 2:* If top and bottom are *not* antipodal then we have two subcases.
*Case 2a:* If top anti-dominates bottom, we have the configuration of Figure 5a where the ordering of all pairs of points is determined.
*Case 2b:* Else, top dominates bottom and we have the configuration of Figure 5b, where the ordering of all pairs of points is determined except for pairs in MM. If $\mathsf{MM} = \emptyset$ or has a single point, we are done. Else, let $C$ be the subset of points of MM are not in anti-dominance relation with a point of $D$ not in MM. We have that all the remaining points of MM have their ordering determined. Also, $C$ comprises one or more components and/or close pairs whose ordering can be changed by means of Transformations 1 and 3.
Now, let us show that there are no other possible transformations that change the order of some pair of points $a, b$ in $C$, while leaving $\mathsf{RS}(D)$ the same. If $b$ minimally dominates $a$, there exists no response in $\mathsf{RS}(D)$ that contains right and $a$ without $b$. Any such transformation would result in one of the following changes: (i) $a$ dominates $b$, (ii) $a$ anti-dominates $b$, (iii) $b$ anti-dominates $a$ and (iv) $a$ and $b$ are collinear. In

(i), (ii) or (iii), then there would exist a response in $\mathsf{RS}(D)$ that contains right and $a$, but not $b$, which would result in a different response set. Thus, the transformation would make $a$ and $b$ be collinear. This is possible only if the corresponding sets $X_1$, $X_2$ and $X_3$ shown in Figure 3c are empty. As $b$ minimally dominates $a$, $X_3$ must be empty. Suppose there is some point $c \in X_1$, then there is a response that contains $a$ and $c$ without $b$ and a response that contains $b$ and $c$ without $a$. If $a$ and $b$ were collinear, one of those responses becomes impossible, modifying the response set. A similar argument can be made about $X_2$ so we have that $X_1$, $X_2$, and $X_3$ are empty. We conclude that $a$ and $b$ form a close pair and that we are applying Transformation 3 to make them collinear.
Alternatively, if $b$ minimally strictly anti-dominates $a$, there exists a response $r_1$ that contains right and $a$ without $b$ and a response $r_2$ that contains right and $b$ without $a$. The transformations would result in one of the following changes: (i) $a$ dominates $b$, (ii) $b$ dominates $a$, (iii) $a$ anti-dominates $b$ and (iv) $a$ and $b$ are collinear. In cases (i), (ii), or (iv) one of responses $r_1$ or $r_2$ would not exist, resulting in a different response set. What is left is case (iii), which means the anti-dominance relationship is flipped by applying Transformation 1. □

## V. Order Reconstruction

The adversary using the response set can reconstruct the order of all records in the database (up to equivalent orders). The order reconstruction algorithm follows the following steps:

(1) Find the extreme points of the database. (Algorithm 9)
(2) Find the first antichain of the database, which contains all points that do not dominate any point and generate the dominance graph of the database. (Algorithm 1)
(3) Find all antichains in the dominance graph. (Algorithm 2)
(4) Use antichains to generate the anti-dominance graph. (Algorithm 3)
(5) Use the dominance and anti-dominance graphs to find any antipodal pairs (Proposition 2), close pairs (Proposition 3) and flippable components. (Proposition 1). (Algorithm 4)

This attack achieves FDR when the horizontal and vertical projections of the points are dense.

### A. Preliminaries

Given some point $a$ in the minimal antichain $A_0$, our order reconstruction attack requires computing the IDs of all points that dominate $D[a]$. We developed Algorithm 8 (DominanceID) that takes as input the response set $\mathsf{RS}(D)$ of a database $D$ and an ID $a$ of some point with height 0 in $D$, and outputs the set of identifiers in $[R]$ of points that dominate $D[a]$. The full explanation of our approach and the pseudocode for Algorithm 8 can be found in Appendix A.

### B. Find Extreme Points

The first step is to identify at most four identifiers of points with extreme coordinate values. Specifically, we wish to find identifiers of points $\mathsf{left}, \mathsf{right}, \mathsf{top}$ and $\mathsf{bottom}$ such that for all $p \in D$ the following hold: (1) $\mathsf{left}_0 \le p_0 \le \mathsf{right}_0$ and $\mathsf{bottom}_1 \le p_1 \le \mathsf{top}_1$, and (2) $p \not\preceq \mathsf{left}, \mathsf{bottom}$ and $\mathsf{top}, \mathsf{right} \not\preceq p$. Note that since no points in $D$ are dominated by left and bottom, then their height is 0 and are thus a subset of $A_0$ in the canonical antichain partition of $D$. These points give a starting point for computing the rest of $A_0$. We recover these extremal points by calling Algorithm 7.

Our approach for finding such a subset of identifiers is as follows. Let $L$ and $S_1$ be the first and second largest responses in $\mathsf{RS}(D)$, respectively. Then $E_1 = L - S_1$ must correspond to the IDs of points that are extreme in some coordinate. To find the IDs of points that are extreme in some other coordinate, find the second largest response $S_2$ that contains $E_1$, and then compute $E_2 = L - S_2$. By extending this process, we find all points with extremal coordinates. It remains to find the correct point within each set $E_i$. Suppose $E_1$ and $E_2$ are the left and bottom edges, respectively. By finding $a, b \in [R]$ such that the smallest response containing $a$ and $b$ contains no other edge points, then $D[a]$ and $D[b]$ must not be dominating any other points in $D$. Hence $\mathsf{left} = D[a]$ and $\mathsf{bottom} = D[b]$. Similarly for the identifiers of top and right.

Without loss of generality, we assume that right dominates left. If not, simply reflect the database along dimension 0 to achieve this orientation. Algorithm 9 is inspired by [8].

**Lemma 1.** *Let $D$ be a database and let $\mathsf{RS}(D)$ be its response set. Algorithm 9 (FindExtremePairs) returns all configurations of extreme points $(\mathsf{left}, \mathsf{right}, \mathsf{top}, \mathsf{bottom})$ such that no points are dominated by left and bottom, and no points dominate right and top in $O(R^2|\mathsf{RS}(D)|)$ time.*

The pseudocode for Algorithm 9 is in Appendix B and the proof of Lemma 1 is in Appendix C.

### C. Generate Dominance Graph

This step takes as input the response set $\mathsf{RS}(D)$ and some configuration config given by running Algorithm 9 on $\mathsf{RS}(D)$, and outputs a dominance graph $G$ of $D$. We first start by computing all IDs of points with height 0. These must be the sinks of $G$. Let left, right, and bottom be given by config. By assumption, all points not dominated by left and bottom must be contained in the minimal query containing them.

Then for each $a \in A_0$ we build a subgraph of the dominance graph on $a$ and all IDs that dominate $a$. We use Algorithm 8, described in Appendix A, to compute this set of IDs. We initialize subgraph $G_a = \{a\}$ and then extend the graph by finding the next smallest response resp containing $a$, that also contains some ID $v$ not yet added to the graph. Since resp is minimal, then $v$ must dominate everything in the response. Moreover, $v$ must minimally dominate all IDs that are sinks in the current $G_a$ *and* are contained in resp. Thus we add $(t, v)$ to $G_a$ for all sinks $t$ of $G_a$ contained in resp.

Once subgraphs $G_a$ for all $a \in A_0$ have been computed, we simply take their union, $G = \cup_a G_a$, as the dominance graph and return $G$ and $A_0$.

**Lemma 2.** *Let $D$ be a database, $\mathsf{RS}(DB)$ be its response set, and config the correct configuration output by Algorithm 9 on $\mathsf{RS}(DB)$. Given $\mathsf{RS}(D)$ and config, Algorithm 1*

**Algorithm 1:** DomGraph(RS($D$), config)

**Input:** Response set RS($D$) of database $D$; a dictionary config mapping left, right, top, bottom to IDs.

// Find antichain-0. We assume right dominates left.
Let small be the smallest response containing left and bottom.
Let $A_0 =$ small
**for** $p \in$ small **do**
  Let $S$ be the smallest response that contains right and $p$.
  $Q = (S \cap \text{small}) - \{p\}$
  $A_0 = A_0 - Q$
// Find dominance graph.
Let $G$ be an empty graph
**for** each $a \in A_0$ **do**
  $G_a = (V, E)$ such that $V_a = \{a\}$ and $E_a = \emptyset$.
  $S = \text{DominanceID}(a, \text{top}, \text{left}, \text{right}, \text{RS}(D))$
  Let $R_S \subseteq \text{RS}(D)$ comprise the responses of size at least 2 that contain $a$ and only other IDs in $S$.
  **for** resp $\in R_S$ by increasing size **do**
    **if** $\exists v \in$ resp such that $v \notin G_a$ **then**
      Add vertex $v$ to $G_a$
      **for** each $t$ of resp such that $t$ is a sink of subgraph of $G_a$ that contains only points in resp **do**
        Add edge $(t, v)$ to $G_a$.
$G = \cup_{a \in A_0} G_a$, and remove any transitive edges
**return** $G$, $A_0$

(DomGraph) *returns the dominance graph of the points in $D$ in $O(R^3|\text{RS}(D)|)$ time.*

The proof of Lemma 2 can be found in Appendix C.

### D. Construct Antichains

Given $A_0$, we now wish to compute the entire canonical antichain partition of $D$. Here, we explain how to find the partition $\mathcal{A} = (A_0, \ldots, A_L)$ such that $L$ is the maximum height of any element in $D$. Computing each $A_i$ is equivalent to finding the set of elements whose maximum length path in $G$ from any $a \in A_0$ has length $i$. Thus, for each $p \in G$ we compute the longest path in $G$ from any $a \in A_0$ to $p$ and then add $p$ to the correct partition in $\mathcal{A}$. Lastly, order the elements in each antichain $A \in \mathcal{A}$ such that, without loss of generality, for any pair of ordered elements $c$ and $c'$, $c \preceq_a c'$. If $|A| \leq 2$ we are done. Else we compute all responses that contain exactly two elements in $A$. If such a response exists for a pair $c, c' \in A$ then we can infer that there exists no $c'' \in A$ such that $c \preceq_a c'' \preceq c'$. Thus we may use these responses to determine the ordering of the elements in $A$ such that any element must anti-dominate all previous elements in the ordering.

**Lemma 3.** *Let $D$ be a database and* RS($DB$) *be its response set. Given* RS($D$)*, a dominance graph $G$ of $D$, and the minimal antichain $A_0$, Algorithm 2 (FindAntichains) returns a dictionary* Antichains *such that* Antichains[$i$] *contains an ordered list of all IDs at height $i$ in $O(R^2|\text{RS}(D)|)$ time.*

The proof of Lemma 3 can be found in Appendix C.

### E. Generate Anti-Dominance Graph

The next step is to take the response set RS($D$), the dominance graph $G$, and the canonical antichain partition Antichains and construct the corresponding anti-dominance graph. There are three major steps that we must take: (1) fix the antichain orientations so that they are lined up correctly, (2) add any edges between IDs of different antichains that are in an anti-dominance relationship, and (3) identify all colinearities.

**Algorithm 2:** FindAntichains(RS($D$), $G$, $A_0$)

// Find antichains.
$(V, E) = G$, Antichains $= \{\}$, Antichains[0] $= A_0$
Compute longest paths$\in G$ from all $a \in A_0$ to all points in $D$.
$L = 0$
**for** each $p \in V$ **do**
  Let $\ell$ be the length of the longest path to $p$ from any $a \in A_0$.
  Add $p$ to Antichains[$\ell$]
  $L = \max(L, \ell)$
// Order the points of Antichains[$i$].
**for** $i = 0, \cdots, L$ **do**
  **if** |Antichains[$i$]| $> 3$ **then**
    Let $S$ be all responses in RS($D$) that contain exactly two elements of Antichains[$i$] (and perhaps other points)
    Remove all $p \notin$ Antichains[$i$] from $S$ and make $S$ a set.
    Order Antichains[$i$] such that pairs of consecutive points are responses in $S$.
**return** Antichains

First we iterate through Antichains; At iteration $i$, we look at Antichains[$j$] for all $j < i$ until we find an edge $(c_1, c_2)$ in $G$ such that $c_1 \in$ Antichains[$j$] and $c_2 \in$ Antichains[$i$]. If there is another edge $(c_1', c_2')$ in $G$ with $c_1' \in$ Antichains[$j$] and $c_2' \in$ Antichains[$i$], then we check if the edges in the antichains $i$ and $j$ are consistent. For example, if the orderings are $(c_1, c_1')$ and $(c_2', c_2)$ in Antichains[$j$] and Antichains[$i$], respectively, then we flip Antichains[$i$].

Once the chains are fixed, we add edges for anti-dominance relationships. We iterate through Antichains[$i$] and Antichains[$j$] for $i < j$ and look at each pair of elements $a_i, a_j$ such that $a_i \in$ Antichains[$i$] and $a_j \in$ Antichains[$j$]. For each $a_i$ and $a_j$ we compute all their successors and all predecessors in $G$. If there exists a path from some successor of $a_j$ to some predecessor of $a_i$, then we add $(a_j, a_i)$ to $G'$. Similarly, if there exists a path from some predessor of $a_j$ to some successor of $a_i$, then we add $(a_i, a_j)$ to $G'$.

The last thing that remains is to identify colinearities. For each edge $(q, p)$ in $G'$ find the smallest response $S$ containing $q$ and $p$. If there exists some $k \in S$ such that $k$ and $p$ are not connected in $G'$, then they must be colinear and so we add $(k, p)$ to $G'$. We similarly check if there exists a colinearity between $k$ and $q$ and add those edges to $G'$. The final step is to remove all transitive edges in $G'$ (if they exist) to keep only minimal anti-dominance relationship and return the anti-dominance graph $G'$.

**Lemma 4.** *Let $D$ be a database and* RS($DB$) *be its response set. Given* RS($D$)*, the dominance graph $G$ of $D$, and the ordered antichains of $D$ Algorithm 3 returns the anti-dominance graph of $D$ in $O(R^3|\text{RS}(D)|)$.*

The proof of Lemma 4 can be found in Appendix C.

### F. Order Reconstruction

We have already described the algorithms for computing the extreme points, the dominance graph, the antichains, and the

**Algorithm 3:** AntiDomGraph($\mathsf{RS}(D), G, \text{Antichains}$)

Initialize empty graph $G'$
// Fix chain orientation
**for** $i \in [1, |\text{Antichains}|]$ **do**
  Add an edge in $G'$ between consecutive points in Antichains$[i-1]$
  Find $(c_1, c_2) \in G$, where $c_1$ is the first point in Antichains$[k], k < i$ in an edge with a point from Antichains$[i]$. If there are multiple options for $c_2$, pick the smallest one in order.
  **if** $\exists (c_1', c_2') \in G$, for a point $c_1' \in$ Antichains$[k], k < i$, which is after $c_1$ in order, and $c_2' \in$ Antichains$[i]$, which is before $c_2$ in order, and there is no path from $c_1'$ to $c_2$ in $G$ **then**
    Flip the order of Antichains$[i]$
Add an edge in $G'$ between consecutive points in the last antichain
// All chains are fixed; Now add edges between them.
**for** $A_i =$ Antichains$[i]$ and $A_j =$ Antichains$[j]$, such that $i, j \in [|\text{Antichains}|]$ and $i < j$ **do**
  **for** $a_i \in A_i$ and $a_j \in A_j$ **do**
    **if** $a_i$ and $a_j$ not connected in $G$ **then**
      Find successors of $a_j$, $S_j \subseteq A_j$, and all predecessors of $a_j$, $P_j \subseteq A_j$. Add $a_j$ to $S_j, P_j$.
      Find successors of $a_i$, $S_i \subseteq A_i$, and all predecessors of $a_i$, $P_i \subseteq A_i$. Add $a_i$ to $S_i, P_i$.
      **if** $\exists$ path from $p$ to $q$ in $G$, s.t. $p \in S_j$, $q \in P_i$ **then**
        Add edge $(a_j, a_i)$ to $G'$
      **else if** $\exists$ path from $p$ to $a_j$ in $G$, s.t. $p \in P_i$ **then**
        Add edge $(a_j, a_i)$ to $G'$
      **else if** $\exists$ path from $p$ to $q$ in $G$, s.t. $p \in P_j$, $q \in S_i$ **then**
        Add edge $(a_i, a_j)$ to $G'$
      **else if** $\exists$ path from $p$ to $a_j$ in $G$, s.t. $p \in P_i$ **then**
        Add edge $(a_j, a_i)$ to $G'$
// Find any collinearities.
Let $E$ be an empty list.
**for** $(q, p) \in G'$ **do**
  $P_{q,p}, S_{p,q}, P_{p,q} = \mathsf{Boxes}(p, q)$
  Let $S = \cup P_{q,p} \cup S_{p,q} \cup P_{p,q}$
  **if** $\exists k \in S$, where there is no path from $k$ to $p$ in $G'$ **then**
    Add an appropriate edge between $k$ and $p$ to $G'$
  **if** $\exists k \in S$, where there is no path from $k$ to $q$ to $E$ **then**
    Add an appropriate edge between $k$ and $q$ to $E$
Add all edges in $E$ to $G'$
Remove transitive edges from $G'$
Return $G'$

**Algorithm 4:** OrderReconstruction($\mathsf{RS}(D)$)

PossibleConfigs = FindExtremePairs($\mathsf{RS}(D)$)
**for** config $\in$ PossibleConfigs **do**
  $G =$ DomGraph($\mathsf{RS}(D)$, config)
  $G' =$ AntiDomGraph($\mathsf{RS}(D), G,$ Antichains($\mathsf{RS}(D), G$))
  Let closePairs and antipodalPairs be empty lists.
  Find the smallest response that contains top and bottom. If it contains no other points, then add (top, bottom) to antipodalPairs.
  Find the smallest response that contains left and right. If it contains no other points, then add (left, right) to antipodalPairs.
  **for** each edge $(b, a) \in G$ **do**
    **if** $(b, a)$ satisfy Definition 9 **then**
      Add $(b, a)$ to closePairs
  **if** response set of points with orders $(G, G')$ is $\mathsf{RS}(D)$ **then**
    Return $(G, G',$ antipodalPairs, closePairs)

anti-dominance graph. We now put all these pieces together to design an algorithm that achieves OR of a target database $D$ given its response set $\mathsf{RS}(D)$. Algorithm 4 succeeds at OR by taking the following steps. First it runs Algorithm 9 (FindExtremePairs) to compute all candidate configurations of the extreme points; by computing all constant number of the configurations, we guarantee that at least one of those configurations must correspond to a correct arrangement of the extreme points in $D$ (up to rotation/reflection).

For each possible configuration computed above, it then computes the dominance graph using Algorithm 1 (DomGraph) and the anti-dominance graph using Algorithm 3 (AntiDomGraph). Incorrect configurations, result in graphs that are either of an incorrect form or result in a pair of dominance and anti-dominance graphs $(G, G')$ such that databases with orders described by $(G, G')$ are not compatible with $\mathsf{RS}(D)$. Algorithm 4 continues to iterate through the configurations until a correct pair of graphs $(G, G')$ is found and returned. Given a response set $\mathsf{RS}(D)$ of some database $D$ as input, Algorithm 4 (OrderReconstruction) is guaranteed to terminate and output a correct graph pair.

**Theorem 2.** *Given the response set* $\mathsf{RS}(D)$ *of a 2D database* $D$ *with* $R$ *records, Algorithm 4 (*OrderReconstruction*) returns an* $O(R)$*-space representation of the set* $\mathsf{E_o}(D)$ *of all possible orderings of the points of databases equivalent to* $D$ *with respect to the response set. The algorithm runs in time* $O(R^3|\mathsf{RS}(D)|)$*, which is* $O(R^7)$*.*

*Proof.* By Lemma 1, PossibleConfigs has all possible configurations of a given set of extreme points. Thus, at some point we pick the correct config. By Lemmas 2 and 4, we know that $G$ and $G'$ return correct weak dominance and anti-dominance graphs. By Proposition 2, we know that if the smallest response that contains top and bottom is empty, then they are an antipodal pair. Similarly for left and right. We find all such pairs. We iterate though pairs of points and find any that satisfy the close pair requirements from Definition 9, constructing the closePairs set. The anti-dominance graph encodes in it any components, as the separate connected components of anti-dominance graph form the different flippable components.

By Theorem 1, given $(G, G',$ antipodalPairs, closePairs$)$ output by the algorithm, we can construct all members of set $\mathsf{E_o}(D)$. The first graph we return is sufficient as any other extreme point configurations whose response set matches $\mathsf{RS}(D)$ are either rotations/reflections or contain antipodal pairs. This Algorithm takes $O(R^3|\mathsf{RS}(D)|)$ time, as it takes $O(R^3|\mathsf{RS}(D)|)$ time to run Algorithms 9, 1, 2 and 3. Finding antipodal pairs takes $O(|\mathsf{RS}(D)|)$ and finding close pairs $O(R^3)$. Finally, it takes $O(R^4)$ time to generate and compare the leakage.

We can encode graphs $G$ and $G'$ by their linear extensions in linear space, and the sets antipodalPairs and closePairs contain at most $O(R)$ points.

□

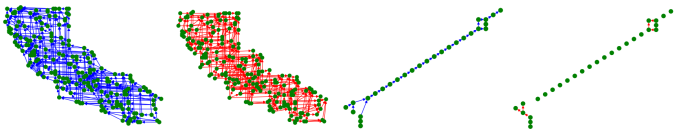


Fig. 6: (Left) Reconstructed dominance graph (blue edges) and anti-dominance graph (red edges) of 200 randomly chosen points from the California Road Network dataset. (Right) The dominance and anti-dominance graphs for the order reconstruction of 08/31/2009 from the Malte Spitz dataset.

### G. Experimental Results

In the previous subsections, we discussed the limitations of OR and described an algorithm that succeeds at OR when given the response set of a database. We now support our theoretical results with experimental results. We have deployed our OR attack on two real-world location datasets.

The California Road Network dataset [19] comprises $21,047$ road network intersections indexed by longitude and latitude. The longitude coordinates range between $-124.389°$ and $-114.294°$ and the latitude coordinates range between $32.541°$ and $42.017°$. We took a random sample of 200 points, truncated the coordinates to one decimal place, and scaled by a factor of 10. The resulting domain had size $[98] \times [93]$. We generated the response set for those points and then ran our OR attack (Algorithm 4) on that response set.

We chose to use this dataset because of its prior use in the order-revealing encryption (ORE) literature. Although the leakage we consider in this work is strictly less than the leakage produced by even the most secure ORE schemes (e,g,. Ideal), the goal of our OR attack parallels the goals of the ***sort attacks*** on ORE-schemes [21] [6] in many ways. Unlike those attacks, we allow the points to be co-linear and we make no assumption about the density of the data. Similar to the 2D-sort attack, though, we aim to recover the relative orders of the records corresponding to the correlated encrypted columns in two-dimensional space.

In Figure 1, we depict the result of our reconstruction of 200 random points from this dataset. One can observe that although, in theory, we only recover the relative orders of all the points, the actual reconstruction leaks additional information about the overall "shape" of the data. For our reconstruction, after finding the order of the points, each point is assigned coordinates corresponding to its index in each dimension's ordering. The figure shows each antichain in a different color, illustrating the height increase. The reconstructed dominance graph (blue edges) and anti-dominance graph (red edges) of the points are shown in Figure 6. Note that in both Figure 1 and Figure 6, the $x$-axis is inverted. This is because, in the Western hemisphere, the longitude axis is inverted.

Malte Spitz is a German politician who published six months of his phone location data. The Spitz data includes location data of Spitz's whereabouts from 166 days, between 8/31/2009 and 2/21/2010. We took longitude and latitude data from the first day of the Spitz data, truncated to one decimal place, and scaled by a factor of 10. In Figure 6, we include the reconstructed dominance and anti-dominance graphs. We picked this data as it has been used in previous work on reconstruction attacks [8], [17].

Order reconstruction in two-dimensions is significantly more enlightening than in one-dimensions. We conjectured that the geometry of the data is more observable when data is (close to) dense in one or both of the domains. Our results from the California dataset support this: we can clearly see that this location data comes from the state of California. In the Spitz case, we can still recover the shape of the dataset and see that it's a deeply diagonal database with a number of collinearities and reflectable components.

## VI. Estimating the Rho Functions

One of the challenges of reconstructing a database $D$ with only partial knowledge of $\mathsf{RM}(D)$, is that the adversary can no longer compute the exact $\rho$ values by simply looking at $\mathsf{RM}(D)$ and counting the number of unique queries that contain the corresponding identifier(s). As such, the two-dimensional FDR attack [8] can no longer be applied. To overcome the challenge of reconstruction with missing queries, we draw inspiration from [17] and use statistical estimators to obtain estimates of the $\rho$ values. In this section, we show how to estimate the $\rho$ values for two dimensional range queries; we provide an overview of a number of ***non-parametric*** estimators that make no assumptions about the underlying query distribution. Then, in Section VII we show how these $\rho$ estimates can be used to construct a system of non-linear equations whose solution corresponds to an approximate reconstruction of the target database.

Given a sample (multiset) $M$ of $m$ token-response pairs, we show how one may compute the appropriate sub-multisets $L \subseteq M$ that correspond to the $\rho$ functions of interest. Each of these sub-multisets is used to approximate the value of its respective $\rho$ value. We extend [17] by experimentally evaluating a number of statistical estimators beyond Jacknknife and Valiant-Valiant, and presenting our findings.

Formally, let $D$ be a database of $R$ records and let

$$M = \{\{(t_1, A_1), \ldots, (t_m, A_m) : A_i \in \mathsf{RS}(D)\}\}$$

be a sample (i.e. multiset) of $m$ token-response pairs that are leaked when queries are issued according to an arbitrary distribution. Define $L_i$ for ID $i \in [R]$ to be the multiset of token-response pairs in $M$ in which the response contains $i$.

$$L_i = \{\{t_k : i \in A_k \text{ and } (t_k, A_k) \in M\}\}$$

We can similarly define a subset $L_{i,j}$ for $i, j \in [R]$ as follows.

$$L_{i,j} = \{\{t_k : \{i, j\} \in A_k \text{ and } (t_k, A_k) \in M\}\}$$

Note that when $M$ contains all unique token-response pairs of $D$, then $\rho_i = |\mathsf{set}(L_i)|$ and $\rho_{i,j} = |\mathsf{set}(L_{i,j})|$ for all $i, j \in [R]$.

We introduce some basic notation and definitions from prior work on estimators. To remain consistent with prior estimator literature, in this section, $N$ and $D$ ***do not*** refer to the domain or database, respectively. Let $N$ be the size of sample $L$ and let $n$ denote the size of a subsample $L \subseteq M$. Denote by $D$ the number of distinct tokens in $M$ and $d$ the number of distinct tokens in a sub-sample $L \subseteq M$.

**Definition 11.** *[24] Let $L$ be a sample and let $f_i$ be the number of search tokens that are observed $i$ times in $L$. The **fingerprint** of a sample $L$ is the vector $F = (f_1, f_2, ..., f_n)$, where $|L| = n$. We can express the total number of token-response pairs in $L$ as $n = \sum_{i=1}^{n} i f_i$ and the number of observed distinct search tokens as $d = \sum_{i=1}^{n} f_i$.*

### A. Non-parametric Estimators

Sampling-based estimators have been used across various domains for goals ranging from estimating the number of unique attributes in a database of [13], the number of unique species in a population sample (e.g. [1], [2]), even the number of words that Shakespeare must have known [7]. Recently, non-parametric estimators have been used for database reconstruction setting to estimate the support size of the given conditional probability distribution of a particular record identifier.

In this section, let $M$ be the sample of observed token-response pairs from database $D \in \mathcal{D}^R$. Let $L$ be a sub-multiset of $M$ comprised of all token-response pairs that contain the identifiers of the points whose $\rho$ value we wish to compute. The goal is to estimate $\widehat{\rho} \approx \rho$ for use in reconstruction. We employ non-parametric estimators to estimate $\rho$.

**Chao-Lee.** Chao and Lee proposed an estimator that utilizes ***sample coverage*** [3]. Let $p_i$ be the probability that a query sampled from the distribution matches the $i$-th token-response pair, of the possible $Q = \binom{N_0+1}{2}\binom{N_1+1}{2}$ token-response pairs. Let $\mathbb{1}_L(i)$ be the following indicator function.

$$\mathbb{1}_L(i) = \begin{cases} 1 & \text{if } i\text{-th token-response pair is in } L \\ 0 & \text{otherwise.} \end{cases}$$

The sample coverage $C$ of a sample $L$ is the sum of the probabilities of the the token-response pairs that appear in $L$: $C = \sum_{i=1}^{Q} p_i \cdot \mathbb{1}_L(i)$. Note that $\hat{C} = 1 - f_1/n$ is a natural estimate for $C$, which can then be used to estimate $\widehat{\rho} \approx d/\hat{C}$. Chao and Lee use this approximation in combination with an additive term to correct estimates of data drawn from skew distributions. Let $\widehat{\rho}_{\text{ChaoLee}} = \frac{d}{\hat{C}} + \frac{n(1-\hat{C})}{\hat{C}} \cdot \hat{\gamma}^2$, where $\hat{\gamma}$ is an estimate of the coefficient of variation $\gamma = (\sum_i (p_i - p_{mean})^2/Q)^{1/2}/p_{mean}$ and $p_{mean}$ is the mean of the probabilities $p_1, \ldots, p_Q$.

**Shlosser**. In [23], Shlosser derived the estimator

$$\widehat{\rho}_{\text{Schloss}} = d + \frac{f_1 \sum_{i=1}^{n} (1-q)^i \cdot f_i}{\sum_{i=1}^{n} i \cdot (1-q)^{i-1} \cdot f_i},$$

where $q$ is the probability with which a pair is included in the sample. This estimator rests on the assumption that $q = n/Q$. As [13] notes, the Shlosser estimator further rests on the assumption that $\mathbb{E}[f_i]/\mathbb{E}[f_1] \approx F_i/F_1$ where $F_i$ is the number of tokens that appear $i$ times in entire database; This assumption isn't often satisfied in our setting, but our experiments demonstrate that Shlosser did comparable to Jackknife in various cases.

**Jackknife.** The jackknife method was introduced by Quenouille as a technique for correcting the bias of an estimator [22]. We use the jackknife estimators described in [1] [2], which have been used extensively in biology for the related problem of species estimation, as well as used in database reconstruction [17] and the problem of estimating the number of unique attributes in a relational database [13].

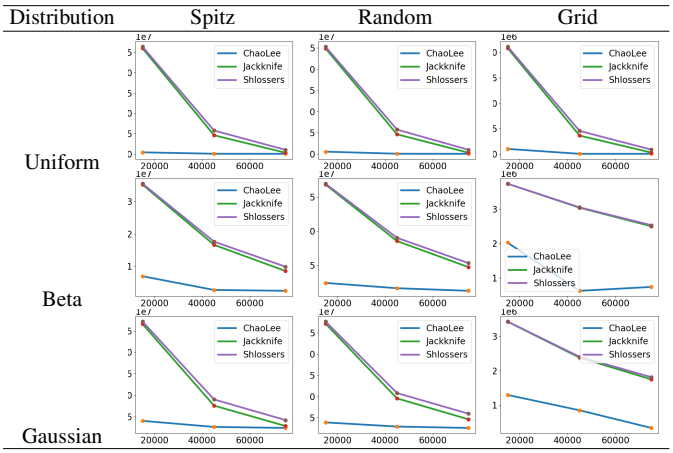



TABLE II: Effectiveness of the estimators.

One can view $d$ as a biased estimate of the true $\rho$. Thus, given a biased estimate $d$, jackknife estimators use sampling with replacement to estimate the bias $bias_{jack}$, and obtain $\widehat{\rho}_{\text{jack}} = d - bias_{jack}$. Let $d_n$ denote the number of unique tokens in $L$ and let $d_{n-1}(k)$ denote the number of unique tokens in $L$ when the $k$-th token-response removed. Note that $d_{n-1}(k) = d_n - 1$ if and only if the $k$-th pair is unique in $L$. Let $d_{n-1} = (1/n)\sum_{k=1}^{n} d_{(n-1)}(k)$. The first order jackknife estimator is $\widehat{\rho}_{\text{jack}} = d - (n-1)(d_{(n-1)} - d)$. The second order jackknife consequently consider all $n$ samples generated by leaving one pair out, in addition to all $\binom{n}{2}$ generated by leaving two pairs out. This method can be extended to an $k$-th order jackknife estimators that generates $\sum_{i=1}^{k}\binom{n}{i}$ samples and has bias $O(n^{-k+1})$.

### B. Experiments

We ran our estimators against three databases with domain size $[24] \times [24]$. The first one is the first day of the Spitz dataset (described Section V-G), a dataset deeply diagonal exhibiting numerous colinearities and flippable components. The second database contains points in a grid formation in one corner of the database. The last database contains a random placement of points in the domain. We call these databases Spitz, Grid and Random. Note that the same day from the Spitz database showcased the FDR algorithm in [8]. These databases can be found in tables III, V, and VI. They were chosen as they represent three fairly different point distributions.

We tested the estimation robustness of each estimator under the (i) Uniform distribution, (ii) Beta(2,1) distribution and (iii) Gaussian$(1/2, 1/5)$ distribution. Recall that our goal is to estimate the $\rho$ value for each ID and for all pairs of IDs. Thus, for each estimator, we computed $\widehat{\rho}_i$ and $\widehat{\rho}_{i,j}$ for all $i, j \in [R]$ under different query distributions and then computed the mean squared error. Our empirical results demonstrate that the Chao-Lee estimator performed best (Table II). We initially considered the Valiant-Valiant estimator that was use in [17], but it did not perform as well in our case.

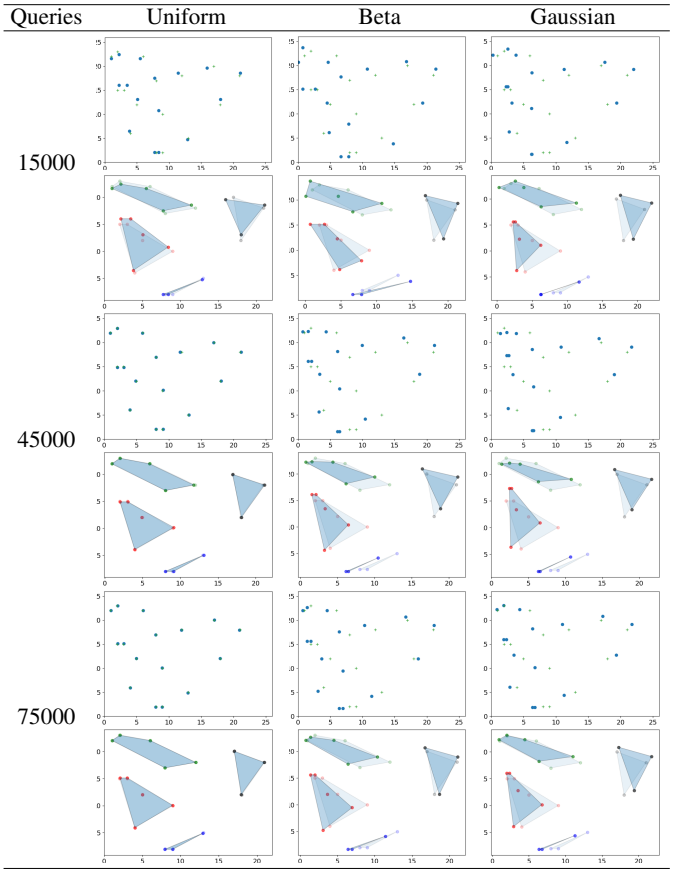



TABLE III: Reconstruction of the Random database under different distributions and number of observed queries. The blue points denote our reconstruction and the green the original. The blue shapes represent clusters of points from original database (transparent) mapped to the reconstruction (darker).

## VII. Approximate Database Reconstruction

Our distribution-agnostic attack for approximate database reconstruction depends on the order reconstruction algorithm and consists of two parts. First, we use support size estimator, Chao-Lee, to estimate how many query responses contain a point or a set of points. Then, we use these estimates to construct a system of equations, whose solution gives an approximate reconstruction of the database.

### A. Algorithm

We assume knowledge of the ordering of the database as given by the Algorithm 4. The first step of ADR is to build a system of equations. We know that point $p$ with coordinates $p_0, p_1$ will be included in $\rho_p = p_0 p_1 (N_0 - p_0)(N_1 - p_1)$ unique responses. The Chao-Lee estimator will give us an estimate of $\rho'_p$. We can then construct an equation with unknowns $x_p, y_p$.

$$x_p y_p (N_0 - x_p)(N_1 - y_p) = \rho'_p \tag{3}$$

Given a pair of point $p, q$, where $p$ dominates $q$, we know that both points are included in $\rho_{p,q} = q_0 q_1 (N_0 - p_0)(N_1 - p_1)$ unique responses. We estimate $\rho'_{p,q}$, and construct an equation with unknowns $x_p, y_p, x_q, y_q$.

$$x_q y_q (N_0 - x_p)(N_1 - y_p) = \rho'_{p,q} \tag{4}$$

We build a similar equation from any ordering of $p$ and $q$.

For any two points, that are both in a dominance and and anti-dominance relationship, we know that they must be collinear. We add this constraint to our system. We use the Chao-Lee estimator to approximate the $\rho$ values $(\rho_p, \rho_{p,q})$ from the subset of responses we have seen. We then construct a first guess for the values of the points using their ordering. Each point $p$ is given coordinates corresponding to its indexes in the first and second dimension. Finally, we find an approximation of the database's point values using a least-squares approach.

Algorithm 5 takes as input a subset $S$ of the response multiset $\mathsf{RM}(D)$, the ordering $G, G'$ and the domain size $(N_0, N_1)$. It returns a reconstructed point set.

**Algorithm 5:** $\mathsf{ADR}(S \subseteq \mathsf{RM}(D), G, G', N_0, N_1)$

1: Let $g$ be a reconstruction of the point values using $G$ and $G'$
2: Create a system of $\rho$ equations for all single points and pairs, including any collinearities.
3: Using the subset of responses we have observed $S$ and the Chao-Lee estimator approximate the $\rho$ value of each equation.
4: **return** the least-squares solution to the system of equations initializing at $g$

### B. Experiments

Algorithm 5 gives us an approximation of the database points. We test our attack on three the Spitz, Grid and Random datasets as before. We performed experiments with three different distributions by sampling queries according to the (i) uniform distribution, (ii) Beta(2,1) distribution and (iii) Gaussian$(1/2, 1/5)$ distribution. In Tables III, V and VI, we present the reconstruction of the Random, Grid, and Spitz datasets, respectively. The reconstructions are obtained after having observed 15000, 45000 and 75000 queries. Since the domain of the database is $[24] \times [24]$, there are 90000 unique queries. In our reconstructions we observe between 15% and 57% of the unique queries. For each distribution and number of observed queries, we show two plots. In the first, our reconstruction is plotted with blue circles and the original points are green crosses. In the second plot, we use agglomerative clustering to cluster the points of the original dataset. We draw the alpha shape of each cluster both in the original data and the reconstructed to visualize the similarities between the two.

We observe that even with a small number of queries, the reconstructed points look "similar" to the original ones. We quantitatively analyze the similarity by matching the clusters of the original points to the reconstruction. In Table IV, we show how well our reconstruction did by calculating the ***mean error***, which is the averaged sum of the euclidean distance of the reconstructed points to the original. We also calculate the ***mean squared error***, similarly to the mean error, but we square the euclidean distance before summing. Finally, to measure similarity between the reconstructed database and the original, we use ***Procrustes analysis*** [11], which scales/dilates, rotates,

| Observed Queries | | 15000 | | | | 45000 | | | | 75000 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Error Metric | | Mean Squared Error | Mean Error | Procrustes Disparity | Query Ratio | Mean Squared Error | Mean Error | Procrustes Disparity | Query Ratio | Mean Squared Error | Mean Error | Procrustes Disparity | Query Ratio |
| Spitz | Uniform | 2.967 | 1.492 | 0.018 | 15% | 0.037 | 0.176 | 0.0 | 39% | 0.032 | 0.162 | 0.0 | 56% |
| | Beta | 12.471 | 3.264 | 0.095 | 15% | 13.428 | 3.325 | 0.03 | 37% | 9.45 | 2.878 | 0.028 | 51% |
| | Gaussian | 8.606 | 2.55 | 0.066 | 15% | 9.329 | 2.665 | 0.011 | 36% | 9.032 | 2.674 | 0.005 | 50% |
| Grid | Uniform | 0.091 | 0.298 | 0.026 | 15% | 0.005 | 0.06 | 0.001 | 39% | 0.002 | 0.042 | 0.0 | 56% |
| | Beta | 4.995 | 1.849 | 0.718 | 15% | 7.232 | 2.493 | 0.336 | 37% | 0.699 | 0.8 | 0.042 | 51% |
| | Gaussian | 7.302 | 2.431 | 0.372 | 15% | 37.344 | 5.433 | 0.497 | 36% | 6.948 | 2.478 | 0.313 | 50% |
| Random | Uniform | 0.62 | 0.725 | 0.006 | 15% | 0.017 | 0.118 | 0.0 | 39% | 0.008 | 0.082 | 0.0 | 57% |
| | Beta | 2.214 | 1.393 | 0.015 | 15% | 3.361 | 1.697 | 0.017 | 37% | 1.927 | 1.28 | 0.008 | 51% |
| | Gaussian | 2.974 | 1.585 | 0.017 | 15% | 3.617 | 1.794 | 0.024 | 36% | 2.303 | 1.396 | 0.012 | 49% |

TABLE IV: Performance of our Approximate Reconstruction Attack.

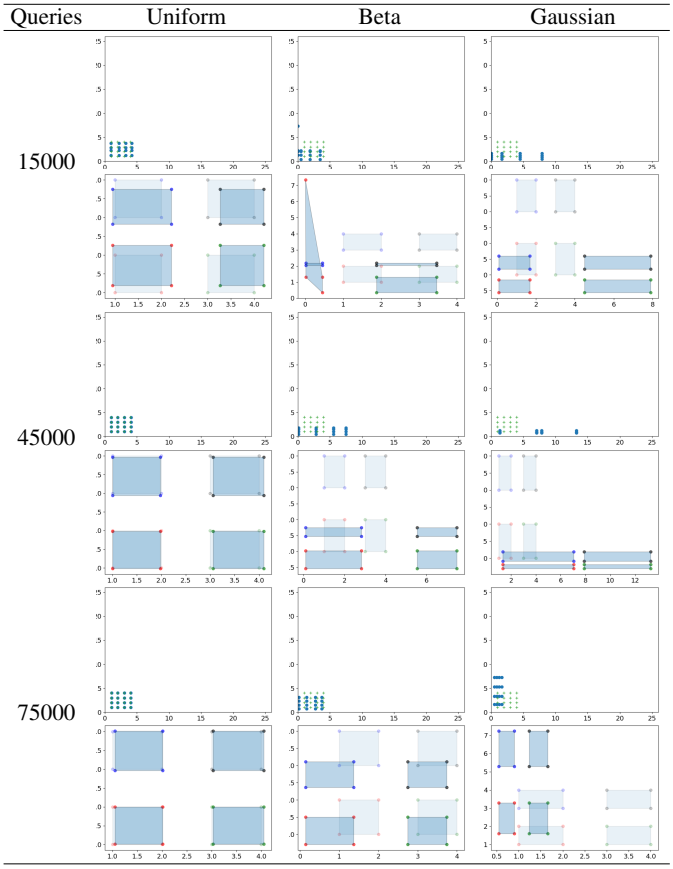



TABLE V: Reconstruction of the Grid database under different distributions and numbers of observed queries. The blue points denote our reconstruction and the green the original. The blue shapes represent clusters of points from original database (transparent) mapped to the reconstruction (darker).

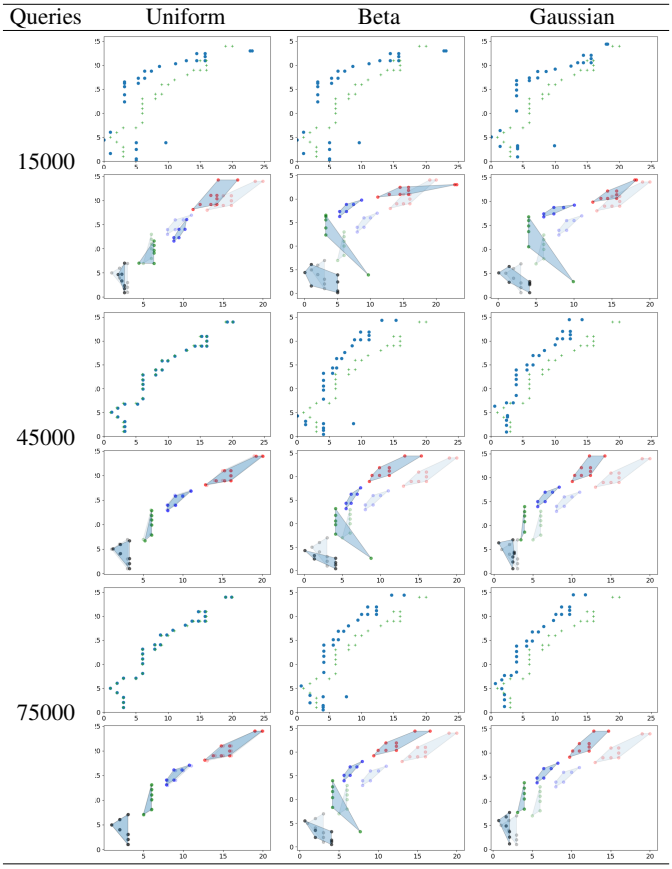



TABLE VI: Reconstruction of the Spitz database under different distributions and number of observed queries. The blue points denote our reconstruction and the green the original. The blue shapes represent clusters of points from original database (transparent) mapped to the reconstruction (darker).

reflects and translates the reconstructed points attempting to minimize the sum of the squares of the pointwise distances of the reconstructed and original sets. The resulting ***Procrustes disparity*** varies from 0 to 1, with 0 indicating optimal similarity. As expected, the reconstruction does best under a uniform distribution.

We observe that under few responses, the reconstruction struggles with very diagonal datasets (like Spitz), but does very well when the database contains points covering all parts of the domain (like Random). We observe that our reconstruction can effectively find the "shape" of the database under almost all our experimental settings with few queries (15%). A notable exception is Grid under non-uniform distributions, where our reconstruction underperforms. In the Spitz data, we observe a number of points that are reflected along the diagonal, a manifestation of reflecting components, which are an intrinsic limitation of the attack (Theorem 1 and [8]).

## VIII. Future Work Directions

This paper makes progress toward the development of practical reconstruction attacks from range queries in encrypted 2D databases and the understanding of the intrinsic limitations of such attacks. A first future direction is to optimize the theoretical and practical performance of our order reconstruction and

approximate reconstruction algorithms by deploying advanced data structures and specialized nonlinear solvers. Additionally, it would be interesting to experiment on a larger suite of real-world datasets and query distributions. Another open problem is to explore partial order reconstruction when only a subset of the access pattern leakage is available to the adversary. Some of our techniques extend to higher dimensions so our work lays a foundation for the development of reconstruction attacks and understanding of their limitations for databases where range searches are performed on multiple attributes.

## References


[1] K. P. Burnham and W. S. Overton, "Estimation of the size of a closed population when capture probabilities vary among animals," ***Biometrika***, vol. 65, no. 3, pp. 625–633, 1978.

[2] ——, "Robust estimation of population size when capture probabilities vary among animals," ***Ecology***, vol. 60, no. 5, pp. 927–936, 1979.

[3] A. Chao and S.-M. Lee, "Estimating the number of classes via sample coverage," ***Journal of the American Statistical Association***, vol. 87, no. 417, pp. 210–217, 1992.

[4] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," ***Journal of Computer Security***, vol. 19, no. 5, pp. 895–934, 2011.

[5] Dawn Xiaoding Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in ***Proc. IEEE Symp. on Security and Privacy***, ser. SP, 2000.

[6] F. B. Durak, T. M. DuBuisson, and D. Cash, "What else is revealed by order-revealing encryption?" in ***Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security***, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1155–1166.

[7] B. Efron and R. Thisted, "Estimating the number of unseen species: How many words did shakespeare know?" ***Biometrika***, vol. 63, no. 3, pp. 435–447, 1976.

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

[9] S. Felsner and L. Wernisch, "Maximum k-chains in planar point sets: Combinatorial structure and algorithms," ***SIAM J. Comput.***, vol. 28, no. 1, pp. 192–209, 1998. [Online]. Available: https://doi.org/10.1137/S0097539794266171

[10] B. Fuller, M. Varia, A. Yerukhimovich, E. Shen, A. Hamlin, V. Gadepally, R. Shay, J. D. Mitchell, and R. K. Cunningham, "SoK: Cryptographically protected database search," in ***Proc. IEEE Symposium on Security and Privacy 2017***, ser. S&P 2017, 2017.

[11] J. C. Gower, "Generalized procrustes analysis," ***Psychometrika***, vol. 40, no. 1, pp. 33–51, 1975.

[12] P. Grubbs, M. Lacharité, B. Minaud, and K. G. Paterson, "Learning to reconstruct: Statistical learning theory and encrypted database attacks," in ***Proc. IEEE Symp. on Security and Privacy 2019***, ser. S&P 2019, 2019.

[13] P. Haas, J. Naughton, S. Seshadri, and L. Stokes, "Sampling-based estimation of the number of distinct values of an attribute," in ***VLDB***, 1995.

[14] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in ***Proc. ACM Conf. on Computer and Communications Security***, ser. CCS. ACM, 2012.

[15] G. Kellaris, G. Kollios, K. Nissim, and A. O'Neill, "Generic attacks on secure outsourced databases," in ***Proc. ACM Conf. on Computer and Communications Security 2016***, ser. CCS 2016, 2016.

[16] E. M. Kornaropoulos, C. Papamanthou, and R. Tamassia, "Data recovery on encrypted databases with $k$-nearest neighbor query leakage," in ***Proc. IEEE Symp. on Security and Privacy 2019***, ser. S&P 2019, 2019.

[17] ——, "The state of the uniform: Attacks on encrypted databases beyond the uniform query distribution," in ***Proc. IEEE Symp.on Security and Privacy 2020***, ser. S&P 2020, 2020.

[18] M.-S. Lacharité, B. Minaud, and K. G. Paterson, "Improved reconstruction attacks on encrypted data using range query leakage," in ***Proc. IEEE Symp. on Security and Privacy 2018***, ser. S&P 2018, 2018.

[19] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng, "On trip planning queries in spatial databases," in ***Advances in Spatial and Temporal Databases***, C. Bauzer Medeiros, M. J. Egenhofer, and E. Bertino, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 273–290.

[20] E. A. Markatou and R. Tamassia, "Full database reconstruction with access and search pattern leakage," in ***Proc. Int. Conf on Information Security 2019***, ser. ISC 2019, 2019.

[21] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in ***Proc. ACM Conf. on Computer and Communications Security 2015***, ser. CCS 2015, 2015.

[22] M. H. Quenouille, "Approximate tests of correlation in time-series," ***Journal of the Royal Statistical Society. Series B (Methodological)***, vol. 11, no. 1, pp. 68–84, 1949.

[23] A. Shlosser, "On estimation of the size of the dictionary of a long text on the basis of a sample," ***Engineering Cybernetics***, no. 19, p. 97–102, 1981.

[24] P. Valiant and G. Valiant, "Estimating the unseen: Improved estimators for entropy and other properties," in ***Advances in Neural Information Processing Systems***, vol. 26, 2013, pp. 2157–2165.

[25] G. Viennot, "Chain and antichain families grids and young tableaux," in ***Orders: Description and Roles Ordres: Description et Rôles***, ser. North-Holland Mathematics Studies, M. Pouzet and D. Richard, Eds. North-Holland, 1984, vol. 99, pp. 409 – 463.


## Appendix A
## Description of Algorithm 8

In this section we describe in full detail how given the response set $\mathsf{RS}(D)$ and the ID $a$ of a point with height 0, one can compute the full set of IDs of points that dominate $D[a]$. We start by describing a helper function called Boxes.

Let $a, b \in [R]$ be the IDs of two points in $D$. Algorithm Boxes, takes as input a pair $(a, b)$ and returns the following responses of $\mathsf{RS}(D)$ (see Figure 7):

- $S_{a,b}$: minimal response containing $a$ and $b$.
- $P_{a,b}$: $D$ minus the maximal responses containing $b$ but not $a$; i.e., set of points $p$ such that every response containing $b$ and $p$ contains also $a$.
- $P_{b,a}$: $D$ minus the maximal responses containing $a$ but not $b$; i.e., set of points $p$ such that every response containing $a$ and $p$ contains also $b$.

**Algorithm 6:** $\mathsf{Boxes}(a, b)$

Let $S_{a,b}$ be the smallest response in $\mathsf{RS}(D)$ containing $a$ and $b$
Let $L = D$
Let $P_{b,a}$ and $P_{a,b}$ be empty lists
**for** $p \in L$ **do**
  **if** $\nexists r \in \mathsf{RS}(D)$, s.t. $p, b \in r$ and $a \notin r$ **then**
    Add $p$ to $P_{a,b}$
  **if** $\nexists r \in \mathsf{RS}(D)$, s.t. $p, a \in r$ and $b \notin r$ **then**
    Add $p$ to $P_{b,a}$
**return** $P_{b,a}$, $S_{a,b}$ and $P_{a,b}$

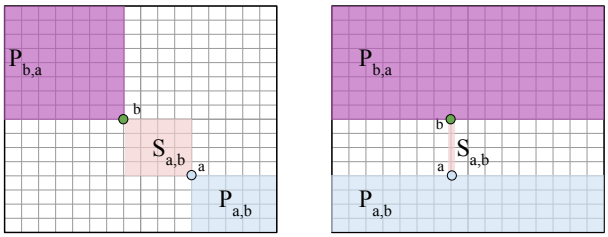



Fig. 7: Illustrating the sets output by Algorithm 6 for points $a$ and $b$, when $b$ strictly anti-dominates $a$ (left) and when $b$ and $a$ are collinear (right).

Note that given a pair of IDs $(a, b)$, there are at most two distinct maximal responses containing $a$ but not $b$ (or $b$ but not $a$). These responses comprise the points in the maximal horizontal and vertical strips of the domain that contain $a$ but not $b$ (or $b$ but not $a$). Note that if $a$ and $b$ share the same horizontal or vertical coordinate, only one of the above strips is nonempty.

Algorithm 8 (DominanceID) leverages Boxes to determine if top dominates $a$. If yes, then we return the minimal response containing $a$, top and right. Else top must strictly antidominate $a$. Let $S$ be the smallest response containing $a$, top and right and let $M$ be the smallest response containing $a$ and top. It is clear that $S - M$ contains all IDs of points that strictly dominate $a$. To find the IDs of points that are colinear with $a$, we run Edges with $M - \{a\}$ as input; the IDs of points that are colinear with $a$ must be one of the edges in the output. In particular, the colinear points must be $p \in E$ such that $E$ is the edge not containing top, left, or any element of $A_0$. And so the algorithm outputs $(S - M) \cup E$.

**Algorithm 7:** Edges$(S, \mathsf{RS}(D))$

Let $RS'$ be the set of responses that contain only points in $S$
Let L be the largest response in $RS'$
Let $S_1$ be the $2^{nd}$ largest response in $RS'$. $E_1 = L - S_1$.
Let $S_2$ be the $2^{nd}$ largest response containing $E_1$. $E_2 = L - S_2$.
Let $S_3$ be the $2^{nd}$ largest response containing $E_1$ and $E_2$. If $S_3$ exists, $E_3 = L - S_3$.
Let $S_4$ be the $2^{nd}$ largest set containing $E_1$, $E_2$, and $E_3$. If $S_4$ exists, $E_4 = L - S_4$.
**return** $E_1$, $E_2$, $E_3$, $E_4$

## APPENDIX B
## PSEUDOCODE

### A. *Psuedocode for Algorithm 9*

## APPENDIX C
## PROOFS

### A. *Proof of Proposition 2*

*Proof.* Let $D[i] = p$, $D[j] = q$, $D'[i] = p'$, and $D'[j] = q'$. We first show that $\mathsf{RS}(D) \subseteq \mathsf{RS}(D')$. Consider a response $A$ in $\mathsf{RS}(D)$ that contains $i$ and not $j$. We will exemplify a query to $D'$ with response $A$. Consider the set $B = (A - \{i\})$. Since $D[i]$ has a unique maximal value in the second coordinate the set $B$ must be an element of $\mathsf{RS}(D)$. By assumption, $\mathsf{RS}(D - \{p, q\}) = \mathsf{RS}(D' - \{p', q'\})$ and so we have that $B \in \mathsf{RS}(D')$. Let $(c, d) \in \mathcal{D}^2$ be a query that generates the response $B$ in $D'$. Now consider the query $((min_0, 1), (max_0, d_1))$ where $min_0 = \min(c_0, p_0, p'_0)$ and $max_0 = \max(d_0, p_0, p'_0)$. Since the only additional identifier contained in this region is $i$, then the response generated by this query is $A = B \cup \{i\}$ which implies $A \in \mathsf{RS}(D')$.

A similar argument holds for queries that contain $j$ and not $i$, as well as queries that contain both $i$ and $j$, which concludes the forward direction of the proof. One can also extend this reasoning to show that $\mathsf{RS}(D') \subseteq \mathsf{RS}(D)$. □

### B. *Proof of Proposition 3*

*Proof.* Let $D[i] = q$ and $D'[i] = q'$. By assumption $\mathsf{RS}(D - \{q\}) = \mathsf{RS}(D' - \{q'\})$. We first show that $\mathsf{RS}(D) \subseteq \mathsf{RS}(D')$. We claim that for any response $A \cup \{i\}$ in $\mathsf{RS}(D)$ there exists a response $A \cup \{i\} \in \mathsf{RS}(D')$. Let $A \cup \{i\}$ be a response in $\mathsf{RS}(D)$ and let $(c, d) \in \mathcal{D}^2$ be a query to $D$ that produces such a response. We will consider two possible cases and in

**Algorithm 8:** DominanceID$(a, \mathsf{top}, \mathsf{left}, \mathsf{right}, \mathsf{RS}(D))$

Let $S_1$ be the smallest response that contains left, top and right.
Let $S_2$ be the smallest response that contains $s_1$, top and right.
Let $M$ be the smallest response that includes $s_1$ and top
**for** $p \in M$ **do**
  **if** $p \in M - S_2$ **then**
    $P_{p,\mathsf{top}}, S_{p,\mathsf{top}}, P_{\mathsf{top},p} = \mathsf{Boxes}(\mathsf{top}, p)$
    $S = P_{p,\mathsf{top}} \cup S_{p,\mathsf{top}} \cup P_{\mathsf{top},p}$
    **if** left, right $\in S$ **then**
      // $a$ and top are collinear
      **return** $S_2$
    **else if** left $\in S$ **then**
      // top dominates $a$
      **return** $S_2$
    **else if** right $\in S$ **then**
      // top anti-dominates $a$
      $E = \mathsf{Edges}(M - \{a\}, \mathsf{RS}(D))$
      $S_2 = S_2 - M$
      Add all $p$ in an edge in $E$ not containing top or $a' \in A_0$ to $S_2$.
      **return** $S_2$
  **else if** $p \in M - S_1$ **then**
    $P_{p,a}, S_{a,p}, P_{a,p}\mathsf{Boxes}(a, p)$
    $S = P_{p,a} \cup S_{a,p} \cup P_{a,p}$
    **if** left, right $\in S$ **then**
      // $a$ and top are collinear
      **return** $S_2$
    **else if** right $\in S$ **then**
      // top dominates $a$
      **return** $S_2$
    **else if** left $\in S$ **then**
      // top anti-dominates $a$
      $E = \mathsf{Edges}(M - \{a\}, \mathsf{RS}(D))$
      $S_2 = S_2 - M$
      Add all $p$ in an edge in $E$ not containing top or $a' \in A_0$ to $S_2$.
      **return** $S_2$
**return** $S_2$

**Algorithm 9:** FindExtremePairs$(\mathsf{RS}(D))$

**Input:** Response set $\mathsf{RS}(D)$ of database $D$
$E_1, E_2, E_3, E_4 = \mathsf{Edges}(D, \mathsf{RS}(D))$
Let PossibleConfigs be all possible combinations of $E_1$, $E_2$, $E_3$ and $E_4$ into LeftE, RightE, TopE, BottomE.
Initialize empty dictionary config.
**for** LeftE, RightE, TopE, BottomE in PossibleConfigs **do**
  **for** $E_1, E_2 \in \{\mathsf{LeftE}, \mathsf{BottomE}\}, \{\mathsf{RightE}, \mathsf{TopE}\}$ **do**
    **for** $a, b \in E_1 \times E_2$ **do**
      **if** the smallest response in $\mathsf{RS}(D)$ that contains $a$ and $b$ does not contain any other element of $E_1$ or $E_2$ **then**
        Add $a, b$ to config under their corresponding key left, right, top, or bottom.
        Return to line 5.
    Add config to PosExtremes.
Return PosExtremes

each case explicitly give a query to $D'$ that must result in the response $A \cup \{i\}$.

*Case 1*: $p_0 < c_0$. Consider the query $((c_0, min_1), d)$ issued to $D'$ such that $min_1 = \min(q'_1, c_1)$. If $min_1 = c_1$ then $\mathsf{Resp}(D', ((c_0, min_1), d)) = \mathsf{Resp}(D', (c, d)) = A \cup \{i\}$ since all points $r \in A$ are identical in both $D$ and $D'$ and $q'$ is contained in this query. Else if $min_1 = q'_1$ then by definition of close pair, $q, q'$ must minimally dominate $p$. So no additional points beside $q'$ are contained in the response generated by $((c_0, min_1), d)$ thus $\mathsf{Resp}(D', ((c_0, min_1), d)) = A \cup \{i\}$.

*Case 2*: $c_0 \leq p_0$. Since the query $(c, d)$ contains $q$ then we have $c \preceq p$ and $q \preceq d$. Moreover $p \preceq q' \preceq q$ and so we have $\mathsf{Resp}(D', (c, d)) = A \cup \{i\}$.

That proves the forward direction of the proof. A similar argument holds for the backward direction and we conclude that $\mathsf{RS}(D) = \mathsf{RS}(D')$. □

### C. Proof of Lemma 1

*Proof.* We first show that Algorithm 7 returns the correct edges i.e. the sets $E_i$ for $i \leq 4$ contain IDs of all points with an extreme coordinate value. Note that the second largest response in $\mathsf{RS}(D)$ must exclude the ID of some extreme point $p$. For a contradiction, suppose $p$ is not extreme. Then we could minimally extend the query to include $p$ and the resulting query would have a response strictly larger than the original query and strictly smaller than $[R]$ since it is not extreme, hence a contradiction. Now consider the second largest response containing the ID of $p$. The remaining ID(s) must correspond to points with an extreme coordinate value in another direction, else we could minimally extend the query to include the non-extreme point(s). By extending this reasoning, we recover the IDs of all points with an extreme coordinate.

In Algorithm 9, line 2 stores the at most 4! assignments of the $E_i$ to LeftE, RightE, TopE, and BottomE. The for loop on line 4 then iterates through each possible assignment to identify the correct IDs within each edge set. We want to find the IDs for the left-most point, $a$, and bottom-most point, $b$, such that no points are dominated by $D[a]$ or $D[b]$. This corresponds to finding $a \in$ LeftE and $b \in$ BottomE such that the minimal response containing them contains no other extreme points. Suppose for a contradiction that some edge point $c$ was dominated by either $a$ or $b$, then the minimal query must also contain $c$. A similar argument holds for the top-most and right-most points.

The algorithm terminates in $O(R|\mathsf{RS}(D)|)$ time. It takes $O(R^2|\mathsf{RS}(D)|)$ time to find the edges. Then, we iterate through pairs of edges and look through $\mathsf{RS}(D)$ to find a smallest response. □

### D. Proof of Lemma 2

*Proof.* Let left, right, top and bottom be the points defined by config. WLOG assume that right dominates left and bottom. We first show that lines 2 to 7 find a set of IDs of points that are not dominating any point in $D$ (i.e. a minimal antichain $A_0$ of $D$ up to rotation/reflection). By correctness of Algorithm 9, no point is dominated by either left or bottom. Let $S$ be the smallest response in $\mathsf{RS}(D)$ containing left and bottom. All points *not* dominated by left and bottom must be in $S$, and thus we initialize $A_0 = S$.

By assumption, right must dominate all points with IDs in $S$. Let $p$ be a point with ID in $S$ and consider the response $T$ of query $(p, \text{right})$. If there is a point $q$ with ID in $S$ such that $p \preceq q$, then its ID must also be in response $T$. In line 6 we find the set $Q$ of all such IDs and delete $Q$ from $A_0$. Since the for loop on line 4 iterates through all IDs in $S$, and deletes the IDs of all points that must dominate at least one other point in $S$, then at the end of the loop $A_0$ must be the set of all points not dominating any other point.

On lines 10 to 18, we construct the dominance graph. Let $S$ be the IDs output by $\mathsf{DominanceID}(\mathsf{RS}(D), a)$ for some $a \in A_0$. Note that $S - \{a\}$ corresponds to the IDs of all records that dominate $D[a]$. The for loop starting on line 14 correctly builds the dominance subgraph on all IDs in $S$. We show that the following loop invariant is maintained: at the end of iteration $\ell$ (1) no point with ID in $S \setminus V(G_a)$ is dominated by a point with a vertex in $G_a$ and (2) if $i$ and $j$ are in $V(G_a)$ and $D[j]$ minimally dominates $D[j]$, then edge $(i, j)$ is in $G_a$. At the start $G_a = \{a\}$; this is correct since $a \in A_0$ and $A_0$ is the set of IDs of points that do not dominate any other point. Assume that at iteration $\ell$ the invariant holds. Find the next smallest response $T$ that contains $a$ and only other IDs in $S$. If $T$ contains $v$ not in $G_a$ then add it to $G_a$. (1) holds since no point in $S \setminus V(G_a)$ dominates $D[v]$, otherwise it would be contained in $T$ and we could form a strictly smaller response contradicting the minimality of $T$. For each sink $t \in G_a$ such that $t \in T$ we add $(t, v)$ to $G_a$. (2) holds since $D[v]$ must dominate all points with IDs in $T \cap V(G_a)$ and must minimally dominate all sinks $t$ in $G_a$ that are contained in $T$. Suppose there is some ID $j$ in $V(G_a)$ that is minimally dominated by $v$ but is not a sink. Then this would violate the correctness of $G_a$ at the end of iteration $\ell$ and hence this cannot happen.

Putting it all together, we want to show that taking the union of all $G_a$ gives us the complete dominance graph $G$. Let $p, q \in D$ be any points such that $p \preceq q$. By correctness of $A_0$, there exists some $a \in A_0$ such that $D[a] \preceq p, q$, and thus $p$ and $q$ are contained in the minimal query of $a$, right, and top. By the correctness of $G_a$, then an edge from the IDs of $p$ to $q$ must be added when constructing $G_a$. Since every dominance edge is added to a graph $G_a$ of some $a$, then taking the union over all $G_a$ gives the complete dominance graph of $D$.

The Algorithm terminates in $O(R^3|\mathsf{RS}(D)|)$ time. It takes $O(R|\mathsf{RS}(D)|)$ time to find the first antichain. Then, Algorithm 8 takes $O(R^2|\mathsf{RS}(D)|)$ and may be run $R$ times. □

### E. Proof of Lemma 3

*Proof.* Let $A_0$ be the set of IDs of points with height 0. We argue that the height of $p \in V$ is given by the maximum length of a path from $a$ to $p$ over all $a \in A_0$. Fix some $p \in V$ and suppose that the maximum length of any path from the vertices in $A_0$ to $p$ is $\ell$, and let there be such a maximal path from some $a \in A_0$ to $p$. By correctness of Algorithm 1, the path from $a$ to $p$ in $G$ corresponds to a chain in database $D$.

Thus the height of $p$ is $\geq \ell$. Suppose for a contradiction that $p$ has height $\ell' > \ell$; By definition of height there must exist a chain $C \subseteq D$ of size $\ell'$ with $p$ as the maximal element. Let $c_1 \preceq c_2 \preceq \cdots \preceq c_{\ell'}$ be the elements of $C$. We have that $c_{i+1}$ must minimally dominate $c_i$, otherwise we could could extend the chain from $a$ to $p$ to have length greater than $\ell'$. By correctness of $G$, each edge $(c_i, c_{i+1})$ must be in $G$. Hence the length of the longest path from $a$ to $p$ in $G$ is $\ell' > \ell$, a contradiction. Thus the height of $p$ is given by the length of the longest path from $a$ to $p$ over all $a \in A_0$.

Let $L$ be the number of partitions in the canonical antichain partition of $D$. We have shown that Algorithm computes the partition $\mathcal{A} = (A_0, \ldots, A_L)$ correctly. Let $a_1, \ldots, a_m$ be elements of a partition $A \in \mathcal{A}$. We show that Algorithm 2 correctly computes an ordering of $a_1, \ldots, a_m$ i.e. a $a_{\gamma_1}, \ldots, a_{\gamma_m}$ such that $\gamma_i = 1, \ldots, m$ and for all $j$ either $a_{\gamma_j} \preceq_a a_{\gamma_{j+1}}$ or $a_{\gamma_{j+1}} \preceq_a a_{\gamma_j}$. If $|A| < 3$ then we are done. $|A| \geq 3$ then on line 12 we compute all responses in $\mathsf{RS}(D)$ that contain exactly two elements in $A$ and denote this set as $S$. A response containing exactly two elements $a, a' \in A$ exists only if $a$ minimally anti-dominates $a'$ (or vice versa). Next we delete all $p \in D - A$ from responses in $S$ and make it a set. Let $\{a, a'\}$ be an element of the resulting set $S$. WLOG suppose $a'$ minimally anti-dominates $a$. Suppose that there exists another set $\{a', a''\} \in S$. Then by transitivity $a''$ must minimally anti-dominate $a'$. We can thus "order" the elements in $A$ by finding consecutive pairs of points in the responses.

This Algorithm terminates in $O(R^2|\mathsf{RS}(D)|)$ time, as it takes $O(R^2)$ time to find the longest paths in $G$ and $O(R^2|\mathsf{RS}(D)|)$ to order the antichains. □

### F. Proof of Lemma 4

*Proof.* The antichains returned by Algorithm 2 may have inconsistent direction. The first step of Algorithm 3 is to fix their orientation. We assume that the first antichain, $A_0$, has the correct orientation. Then, we find the first element of $A_0$ that has a dominance edge to a point in $A_1$, the second antichain. Let that edge be $(c_1, c_2), c_1 \in A_0, c_2 \in A_1$. If there are multiple options for $c_2$, we pick the smallest one in order. Note that each member $p$ of antichain $i$ must have a dominance edge with some member $q$ of antichain $j$, $j < i$. Otherwise, $p$ would be part of some previous antichain.

If the order of antichain 1 is wrong, then a point $c_1' \in A_0$ in order before $c_1$ must have an edge with point $c_2' \in A_1$, in order after $c_2$. If the chains were correctly ordered that would be impossible as $c_2'$ anti-dominates $c_1$ and $c_1$ anti-dominates $c_1'$. Thus, $c_2'$ cannot dominate $c_1'$. Thus, Algorithm 2 can correctly orient the second chain given the order of the previous antichains. Maintaining this invariant, Algorithm 2 correctly orients all antichains.

We begin constructing the anti-dominance graph by adding anti-dominance edges between consecutive pairs of points in each antichain.

It remains to add anti-dominance edges between points in different antichains. The algorithm iterates through pairs of chains, and finds points $a_i$ and $a_j$ that are not connected in $G$ and $a_i \in A_i$, $a_j \in A_j, i < j$. Point $a_i$ either anti-dominates $a_j$ or $a_j$ anti-dominates $a_i$. In order to determine their relationship, we look for a dominance edge between the antichains. If $a_j$ anti-dominates $a_i$, then all predecessors of $a_i$ are also anti-dominated by $a_j$ and its successors. So, if a predecessor of $a_j$ dominates a successor of $a_i$. Then $a_j$ must anti-dominate $a_i$. Similarly, if a successor of $a_j$ dominates a predecessor of $a_i$, then $a_i$ anti-dominates $a_j$.

Note that this technique finds only strict anti-dominance edges. It remains to find any collinear anti-dominance edges as well. Given a pair of points $p$ and $p$, such that $q$ anti-dominates $p$, and a point $k$ that is in $\mathsf{Boxes}(p, q)$, $k$ must have an anti-dominance relationship with both. If no such path exists in $G'$, we add appropriate edges depending on which of the Boxes $k$ is in. Note that in some cases, as explained by Proposition 3, it's impossible to determine all collinearities.

Our definition of the anti-dominance graph is that it contains minimal anti-dominance edges. Thus, after we remove any transitive edges, we have generated $D$'s anti-dominance graph.

The Algorithm terminates in $O(R^2|\mathsf{RS}(D)|)$ time, as it takes $O(R^2)$ to fix the antichains and add edges between them and $O(R^3|\mathsf{RS}(D)|)$ to run the Boxes algorithm for any anti-dominance pair. □