Learning to Reconstruct: Statistical Learning Theory and Encrypted Database Attacks

Paul Grubbs*, Marie-Sarah Lacharité†, Brice Minaud‡§, Kenneth G. Paterson†

*Cornell University, pag225@cornell.edu
†Royal Holloway, University of London, {marie-sarah.lacharite.2015, kenny.paterson}@rhul.ac.uk
‡École Normale Supérieure, CNRS, PSL University
§Inria, brice.minaud@inria.fr

***Abstract*—We show that the problem of reconstructing encrypted databases from access pattern leakage is closely related to statistical learning theory. This new viewpoint enables us to develop broader attacks that are supported by streamlined performance analyses. As an introduction to this viewpoint, we first present a general reduction from reconstruction with known queries to PAC learning. Then, we directly address the problem of ϵ-approximate database reconstruction (ϵ-ADR) from range query leakage, giving attacks whose query cost scales only with the relative error ϵ, and is independent of the size of the database, or the number N of possible values of data items. This already goes significantly beyond the state-of-the-art for such attacks, as represented by Kellaris *et al.* (ACM CCS 2016) and Lacharité *et al.* (IEEE S&P 2018). We also study the new problem of ϵ-approximate order reconstruction (ϵ-AOR), where the adversary is tasked with reconstructing the order of records, except for records whose values are approximately equal. We show that as few as $\mathcal{O}(\epsilon^{-1} \log \epsilon^{-1})$ uniformly random range queries suffice. Our analysis relies on an application of learning theory to PQ-trees, special data structures tuned to compactly record certain ordering constraints. We then show that when an auxiliary distribution is available, ϵ-AOR can be enhanced to achieve ϵ-ADR; using real data, we show that devastatingly small numbers of queries are needed to attain very accurate database reconstruction. Finally, we generalize from ranges to consider what learning theory tells us about the impact of access pattern leakage for other classes of queries, focusing on prefix and suffix queries. We illustrate this with both concrete attacks for prefix queries and with a general lower bound for all query classes.**

I. Introduction

This article concerns the analysis of *leakage* from *encrypted databases*. The latter are cryptographic techniques that allow a client to outsource a database to an untrusted server while maintaining the ability to make queries on the data. Fuller *et al.* [FVY+17] give a comprehensive survey of this area. All known techniques represent a trade-off between security and efficiency, with various forms of leakage being intrinsic to these approaches. For example, without taking special precautions such as using oblivious memory techniques, the *access pattern*, that is, the set of records returned in response to queries, leaks to the server. While in many cases, formal security proofs are able to establish that nothing more than the access pattern leaks to the adversarial server, there still remains the question: what is the practical impact of this leakage? A more refined version of the question is:

If an encrypted database supports a certain class of queries, but leaks the access pattern, then how damaging is that leakage as a function of the query and data distribution and number of queries?

The setting of this article is one in which *only* access pattern is leaked to an adversarial server. Access pattern leakage is inherent to nearly all practical constructions of encrypted databases, and the survey by Fuller *et al.* [FVY+17] overviews a plethora of schemes to which such attacks apply. In the particular case of range queries, all known practical solutions leak this information [LMP18]. Nevertheless, the previous question is currently answered using *ad hoc* cryptanalysis, requiring cumbersome and laborious analyses to establish the impact of leakage as a function of the number of queries. The central aim of our work is to transform this situation by bringing statistical learning theory to bear on the problem.

A. Database Reconstruction: State of the Art

Range queries are fundamental to the operation of databases, and have rightfully received significant attention in the attack literature. The state-of-the-art for attacks based on leakage from range queries is represented by the work of Kellaris-Kollios-Nissim-O'Neill (KKNO) [KKNO16] and Lacharité-Minaud-Paterson (LMP) [LMP18]. KKNO gave attacks showing that $\mathcal{O}(N^4 \log N)$ queries suffice to achieve *Full Database Reconstruction (FDR)*, that is, to reconstruct the *exact* value for every record. Here, N is the number of different possible values, which we assume without loss of generality come from the interval $[1, N]$. For *dense* data, where every possible value is in at least one record, this was improved to $\mathcal{O}(N^2 \log N)$ queries by KKNO and then to $\mathcal{O}(N \log N)$ queries by LMP. All of these results assume the query distribution is uniform on ranges (though for the results in the dense setting, this assumption is needed only to facilitate analysis and not for the algorithms to succeed).

A typical value of N might be, say, 125 for data pertaining to age in years, making even an $\mathcal{O}(N^4 \log N)$ attack potentially worrisome. But for many data types, N can be much larger – think of discrete data such as numerical zip codes, timestamps, or salary data. For large N, especially when the data is sparse rather than dense (as is typically the case), FDR

is much too expensive (KKNO proved a general lower bound of $\Omega(N^4)$ on the number of range queries needed), and really too strong an attack goal.

For this reason, LMP introduced the notion of *Approximate Database Reconstruction (ADR)*, where the adversary's goal is to find the value of every record up to an (additive) error of ϵN rather than exactly. For small ϵ, such an attack is still extremely effective: imagine learning all salaries in a database up to an error of 1%. LMP gave the first algorithm for ADR, which achieves ϵ-ADR from access pattern leakage on only $\mathcal{O}(N \cdot \log \epsilon^{-1})$ queries. However, it still requires a density assumption and its analysis is highly complex.

B. Overview of Our Contributions

None of the aforementioned attacks exploiting range query leakage is fully satisfactory: FDR is too expensive for large N, while the only ADR algorithm we have (from LMP's work [LMP18]) relies on a density assumption and its query cost still scales with N. This presents a potentially misleading picture of the impact of leakage for range queries, one which may lead to underestimating the potential damage. Additionally, leakage from other kinds of queries has received little attention, nor have other attack settings of practical importance like known-query attacks.

In this work, we show how statistical learning theory effectively addresses the problem of database reconstruction, yielding new results across a range of settings. A common thread through all of our results is the analysis of *concept spaces* over the set of all queries. The results we apply from learning theory rely on the *VC dimension* of the concept space, intuitively a measure of how complex it is. (See Appendix A for a short primer on statistical learning theory.)

PAC learning and known-query attacks. In Section II, we show that database reconstruction given a set of *known* queries can be recast as an instance of Probably Approximately Correct (PAC) learning, and standard results from that field can predict how many queries are needed to achieve reconstruction. We present this reduction to PAC learning as an introduction to how we view database reconstruction as a learning problem, as well as an illustration of the power of this viewpoint. While the attack model here is rather powerful, it is considered realistic in some recent literature [ZKP16], [GSB+17], [GMN+16]; our analysis largely resolves the question of how damaging such attacks can be. In the remainder, we no longer assume queries are known, aligning our setting with most prior work.

Sacrificial ϵ-ADR. In Section III, we present two new ϵ-ADR algorithms for range queries. These attacks are *scale-free*: their query complexity depends not on the number of possible values N, but only the precision ϵ. They accommodate any number of queries, as opposed to the "all-or-nothing" attacks of KKNO and LMP. To obtain scale-freeness, we must *sacrifice* recovering some records near the endpoints. As we explain in Section III-A, scale-free ϵ-ADR is impossible in general – $\mathcal{O}(N)$ queries are necessary to recover values near the endpoints 1 and N, so we must sacrifice these.

The first algorithm (Section III-B), whose analysis is somewhat simpler, achieves sacrificial ϵ-ADR using $\mathcal{O}(\epsilon^{-4} \log \epsilon^{-1})$ uniformly random range queries. Setting $\epsilon = 1/N$ yields an FDR attack with the same complexity as KKNO's original FDR attack. Indeed, our algorithm can be seen as generalizing the ideas of KKNO to the ADR setting, and making it scale-free. In Section III-C, we introduce our second attack, the ApproxValue algorithm, which achieves sacrificial ϵ-ADR using only $\mathcal{O}(\epsilon^{-2} \log \epsilon^{-1})$ uniformly random queries, but under the additional, mild requirement that the database contains a record whose value is in the range $[0.2N, 0.3N]$ (or its reflection). Setting $\epsilon = 1/N$ in our algorithm again yields a FDR algorithm with complexity $\mathcal{O}(N^2 \log N)$ that works whether data is sparse or dense, assuming only a single favorably-located record. Our proof techniques for both attacks are rooted in learning theory, using VC dimension and the concept of ϵ-samples. Both attacks also come with general lower bounds showing that they are optimal in the number of queries within a $\log$ factor.

In order to assess the effectiveness of these algorithms and the tightness of our bounds, we implement our attacks and experiment on synthetic data (Section III-D). For example, if $N = 10^6$ and the condition of the ApproxValue algorithm is met, KKNO's FDR attack would require about 10^{26} queries. We found experimentally that only 500 queries (or 24 orders of magnitude fewer than KKNO) are needed to approximate almost all records to within 5% error.

Lifting requirements on the query distribution. In view of the previous attacks, it may seem that the topic of analyzing leakage from range queries is mostly closed, but these attacks still require that the adversary knows the query distribution, and that queries are independently and identically distributed (i.i.d.). This second requirement especially makes little sense for real-world queries, and we contend that practical attacks should not require it. In this regard, we view KKNO's work and our aforementioned results as important indicators of what is possible in principle, and valuable warnings regarding the power of range query leakage, but not as practical, ready-for-use attacks.

The question, then, is what an attacker can hope to learn in practice, given only the access pattern leakage of some range queries, without the (unrealistic) assumption of a known i.i.d. query distribution. We investigate this question in Section IV. The LMP results were a step in this direction: their algorithms are not distribution-dependent. However, they do require that the database is dense, whereas we would like to investigate this question in the general setting. First, we observe that what the adversary can learn in that setting is the *order* of the records' values, rather than their values directly.

Sacrificial ϵ-AOR. In Section IV, we introduce the attack target of *sacrificial ϵ-Approximate Order Reconstruction* (sacrificial ϵ-AOR). This asks that the order of all records should be recovered, except for records that are within ϵN of each

other (which the algorithm groups), and the sacrificed records whose values are within ϵN of 1 or N. Thus, save for sacrificed values, sacrificial ϵ-AOR reveals the order of any two records as soon as they are at least ϵN apart.

As our main result in Section IV, we introduce the scale-free ApproxOrder algorithm, which takes as input the access pattern leakage of some range queries, builds a *PQ-tree*, and extracts from it approximate order information. The algorithm does not use any knowledge of a query distribution. If, for the sake of analyzing the algorithm, we assume a uniform query distribution, ApproxOrder achieves sacrificial ϵ-AOR after only $\mathcal{O}(\epsilon^{-1} \log \epsilon^{-1})$ queries. Once again, our analysis relies on learning theory, more specifically the concept of an ϵ-net. We also prove that the query complexity of our algorithm is optimal within a constant factor.

For $\epsilon = 1/N$, ϵ-AOR yields exact order reconstruction for all records. If the database is dense, then recovering order directly implies recovering values, so we obtain full database reconstruction in $\mathcal{O}(N \log N)$ queries, recovering as a special case the main result of LMP.

The ApproxOrder algorithm is not merely theoretical, but highly practical. There is no barrier (such as the i.i.d. query assumption) to running it on real data. Our experiments in Section IV-C show that the attack behaves as predicted by the theory. As an example, for $N = 10^6$, after only 500 queries, the attack is able to fully order records, except for records whose difference in value is less than 2% of the support size.

ϵ-AOR to ϵ-ADR. A crucial question remains: what are the implications of the AOR attack? That is, what does learning approximate order reveal to the attacker? It is well known that leaking record order is highly damaging, if only because it can be closely correlated to record values using an auxiliary distribution [NKW15], [GSB+17]. In fact, the severe implications of order leakage is one of the main motivations behind the development of second-generation encrypted databases schemes that attempt to hide that leakage, as argued in [LMP18]. To concretize that point, in Section IV-D we present an attack showing how approximate database values can be reconstructed from approximate order information: we extend our sacrificial ϵ-AOR attack to a sacrificial ϵ-ADR attack using an auxiliary model of the database distribution. (As per [NKW15], [LMP18], such distributions are often available.)

We conduct experiments with real datasets of US ZIP codes and public sector salaries in Section IV-D. The resulting sacrificial ϵ-ADR attack is effective: with only 50 queries, we can learn the first two digits of a ZIP code (often identifying a city) for a majority of records in the target database. With 100 queries on salaries, we can predict a majority of salaries to within 10000 USD. The table in Figure 1 compares our different attacks on range queries.

Beyond range queries. As illustration of the power of the viewpoint we have taken, in Section V, we generalize approximate reconstruction to other query classes and analyze the resulting attacks using tools from learning theory. Using *generalization error* as a metric γ on the values in the database, we show that all query classes with finite VC dimension reveal the distance (in γ) between the underlying values of records, which allows an attacker to group records whose values are close. Further, we show how to use an ϵ-net to precisely analyze how many queries are needed to guarantee all groups of records have small diameter according to γ. We construct, analyze, and evaluate the first reconstruction attack on prefix queries. We conclude the section with a general lower bound, via a reduction to PAC learning, relating the query class's VC dimension, attack accuracy, and number of queries needed for any reconstruction attack using access pattern leakage. In addition to being of theoretical interest, this suggests VC dimension or similar concepts from learning theory could be a useful way to compare different techniques which leak access pattern.

Notation. Throughout $[n]$ denotes the set of integers $\{1, \ldots, n\}$; $[a, b]$ denotes the set of integers within the given interval; and open brackets such as $[a, b[$ denote that the corresponding endpoint is excluded. (If $b \leq a$, $[a, b[$ is empty.) We model a database as a set of R records where each record has a single attribute that takes an integer value in $[N]$. We let $\mathsf{val}(r) \in [N]$ denote the value of the record r.

Assumptions. We assume the adversary knows the number of possible values N, and the set of all possible queries. We do not assume that the adversary knows the set of all records in advance, or even their number. We do not assume that every value appears in at least one record (no density assumption).

The ApproxValue algorithm in Section III further assumes that queries are drawn i.i.d. and uniformly at random. The attack and its analysis can be generalized to other query distributions. As explained earlier, we then introduce the ApproxOrder algorithm in Section IV precisely to do away with assumptions on the query distribution. Likewise all algorithms in Section V function without that assumption. When it comes to analyzing the query complexity of those algorithms, we are forced to make a hypothesis on the query distribution. In that case, we choose an assumption about the query distribution that helps provide insight into a typical behavior of the algorithm. We stress that that hypothesis is in no way required for the algorithm to function and succeed.

C. Related Work

Dautrich Jr. and Ravishankar [JR13] introduced the use of PQ-trees in revealing the order of records in a database with access pattern leakage. They experimentally measured, in some special cases, how quickly the number of orders contained in the tree decreases as more queries are gathered. We use also use PQ-trees for revealing order from range queries, but otherwise our aims are distinct from theirs—their paper focuses primarily on heuristic measures of security after some ordering information is revealed.

Kellaris *et al.* (KKNO) [KKNO16] described the first exact reconstruction attack on range queries with access

pattern leakage; Lacharité *et al.* (LMP) [LMP18] improved the results of KKNO in the dense setting, obtaining an $\mathcal{O}(N \log N)$ exact reconstruction attack; see Section I-A for more detail. Kornaropoulous et al. [KPT18] gave an approximate reconstruction attack for access pattern leakage from k-nearest-neighbor queries. Other papers attacking encrypted databases include [GSB+17], [NKW15], [BGC+18], [GMN+16], [CGPR15]; these mostly analyze so-called "property-revealing encryption" schemes, which leak strictly more than what we assume.

II. PAC Learning and Database Reconstruction Attacks

In this section, we explore the connection between learning theory and database reconstruction attacks. Concretely, we demonstrate a connection between approximate database reconstruction and "Probably Approximately Correct" (PAC) learning [Val84] in the setting where the attacker has access pattern leakage from some known queries. For a brief introduction to PAC learning, see Appendix A-D.

Reconstruction via PAC learning. The attack setting we consider here is one in which an attacker has observed the access pattern leakage from a number of *known* queries drawn i.i.d. from a fixed query distribution (which the adversary does not need to know). The assumption of known queries is somewhat stronger than has been considered previously in this literature; however, some recent works have argued that it is realistic [GSB+17], [GMN+16] on the grounds that the adversary is able to make some queries or has compromised an honest user.

A crucial question is the relationship between the *number* of known queries and the amount of information the adversary can learn about the database itself, cf. Section I. We will see that this question is essentially resolved via a simple reduction to PAC learning in the known query setting.

We can think of a database DB with R records having values in $[N]$ as being a vector of length R with values in $[N]$; the value of record j is $DB[j]$. We construct a concept space $\mathcal{C} = (\mathcal{Q}, \mathbb{C})$ where the points in the ground set are the possible queries and $\mathbb{C} = \cup_{i \in [N]} C_i$ for $C_i = \{q \in \mathcal{Q} \,|\, q(i) = 1\}$. Here, $q(i) = 1$ means that value i matches query q. With this set-up, we have the following result.

Theorem II.1. *Let $\mathcal{Q}$ be a class of queries taking inputs in a set X and $\mathcal{C} = (\mathcal{Q}, \mathbb{C})$ be the concept space constructed as above. Let π_q be any distribution over $\mathcal{Q}$. Let d be the VC dimension of $\mathcal{C}$, and assume d is finite. Then, there is an adversary such that for any database DB, given as input $m \in \mathcal{O}(\frac{\mathsf{d}}{\epsilon} \log \frac{\mathsf{d}}{\epsilon\delta})$ queries sampled from π_q and their access pattern leakage on DB, the adversary outputs a database DB' such that $\Pr_{\pi_q}\left[\, q(DB[j]) \neq q(DB'[j]) \,\right] \leq \epsilon$ holds simultaneously for all $j \in [R]$, with probability at least $1 - R\delta$.*

This theorem requires some explanation. We chose to use the generalization error $\Pr_{\pi_q}\left[\, q(DB[j]) \neq q(DB'[j]) \,\right]$ as the accuracy measure in our result. This is intended to surface the core points without adding unnecessary detail, but it may also make the result hard to interpret. Section V studies in more detail how generalization error relates to traditional notions of attack accuracy.

The proof proceeds via a natural reduction to PAC learning. The adversary gets as input m known queries along with their access pattern leakage (i.e. which records match the query) for each of the R records in the database. The core observation is that the access pattern is a binary classification of each database element; further, each database element is a concept in $\mathbb{C}$. This means that the task of reconstructing each database element can be seen as R independent PAC learning experiments for the concept space $\mathcal{C}$ defined above. The adversary simply runs the PAC learner R times, invoking it once for each record j. For each invocation, the adversary gives the learner as input the m queries and their access patterns (i.e. the 0/1 labellings) for record j. Each time the learner is run, it outputs a hypothesis $H_{\mathsf{ind}} \in \mathbb{C}$ corresponding to an element of X. The adversary's complete output is then of the form $[H_1, H_2, \ldots, H_R]$, which we denote by DB'. Each independent invocation of the learner outputs a hypothesis H_j with $\Pr_{\pi_q}\left[\, q(H_j) \neq q(DB[j]) \,\right] > \epsilon$ with probability at most δ, and a union bound over the R elements completes the proof.

Note here that, while the linear dependence on R in the probability bound may look discouraging, the sample complexity of PAC learning is only logarithmic in δ, so the loss in tightness from the union bound over R events is small. Note also that the union bound over all R database elements can be avoided entirely with a more careful analysis—for example, observe that any learner will output the same hypothesis on any two database records with the exact same access pattern on all m queries, so the adversary need only run the learner once per group of records having the same access pattern leakage. The dependency on R can also be removed at the cost of replacing the concept set $\mathbb{C}$ by $\mathbb{C}^{\Delta}$ (as defined in section V). Since tightness is not the goal of this result, we favor simplicity in presentation. In the full version, we describe some possible extensions of this result.

Closing remark. In the encrypted database literature, it has become apparent that known- and chosen-query attacks are damaging. However, the *quantitative* question ("How severe a risk is a known- or chosen-plaintext attack?") has not been fully explored. We posit that extending the above result using techniques from learning theory will fully resolve this question. Rather than developing this theme further here, we leave it to future work and focus the remainder of this work on more challenging attack settings.

III. Sacrificial Approximate Database Reconstruction

In this section, we introduce *sacrificial ϵ-approximate database reconstruction* (sacrificial ϵ-ADR), which asks to successfully recover the value of every record in the database within ϵN, for some target precision ϵ, save for records whose value lies within ϵN of 1 or N. The term *ϵ-approximate* means

Attack	Goal	Data density	Query complexity		Source
			Proposed attack	Generic lower bound	
KKNO	FDR	any	$\mathcal{O}(N^4 \log N)$	$\Omega(N^4)$	[KKNO16]
KKNO	FDR	dense	$\mathcal{O}(N^2 \log N)$	–	[KKNO16]
LMP	FDR	dense	$N \log N + \mathcal{O}(N)$	$\frac{1}{2} N \log N - \mathcal{O}(N)$	[LMP18]
LMP	ϵ-ADR	dense	$\frac{5}{4} N \log \epsilon^{-1} + \mathcal{O}(N)$	$N \log \epsilon^{-1} - \mathcal{O}(N)$	[LMP18]
GeneralizedKKNO	sacrificial ϵ-ADR	any	$\mathcal{O}(\epsilon^{-4} \log \epsilon^{-1})$	$\Omega(\epsilon^{-4})$	Section III-B
ApproxValue	sacrificial ϵ-ADR	any$*$	$\mathcal{O}(\epsilon^{-2} \log \epsilon^{-1})$	$\Omega(\epsilon^{-2})$	Section III-C
ApproxOrder	sacrificial ϵ-AOR	any$*$	$\mathcal{O}(\epsilon^{-1} \log \epsilon^{-1})$	$\Omega(\epsilon^{-1} \log \epsilon^{-1})$	Section IV

Fig. 1. Comparison of database reconstruction attacks that use access pattern leakage from range queries chosen uniformly at random. N is the number of possible plaintext values. All attacks are up to global reflection. Generic lower bounds are for any attack targeting the same goal under the same assumptions. "$*$" denotes some additional but mild requirements on the existence of records with particular values.

that reconstruction is within an error of ϵN, as in [LMP18]. The term *sacrificial* means the attack "sacrifices" records whose value lies within ϵN of the endpoints. We explain the need for this and provide a full definition in Section III-A. The rest of the section presents two results.

In Section III-B, we extend KKNO's database reconstruction attack [KKNO16] to sacrificial ϵ-ADR. A direct application of the ϵ-sample theorem from learning theory shows the dependency on N vanishes: the required number of queries becomes $\mathcal{O}(\epsilon^{-4} \log \epsilon^{-1})$, making the attack *scale-free*.

In Section III-C, we introduce a new algorithm, ApproxValue, for sacrificial ϵ-ADR with a mild additional hypothesis h_1: the database contains at least one record whose value lies in $[0.2N, 0.3N] \cup [0.7N, 0.8N]$. Under this hypothesis, ApproxValue achieves sacrificial ϵ-ADR within only $\mathcal{O}(\epsilon^{-2} \log \epsilon^{-1})$ queries. The analysis also uses the ϵ-sample theorem, but is somewhat more involved. This attack shows the pathological nature of KKNO's lower bounds on query complexity for FDR. An experimental validation in Section III-D supports the analysis, and shows that the constants in the $\mathcal{O}$ notation are empirically very small.

As noted in the introduction, the previous two results also imply full database reconstruction within $\mathcal{O}(N^4 \log N)$ queries in general, and $\mathcal{O}(N^2 \log N)$ when h_1 is satisfied. In the full version, we also show that both attacks are optimal in data within a $\log$ factor—any adversary achieving sacrificial ϵ-ADR for all databases (resp. databases satisfying h_1) must require $\Omega(\epsilon^{-4})$ (resp. $\Omega(\epsilon^{-2})$) queries.

A. Definition of Sacrificial ϵ-ADR

We now formally define sacrificial ϵ-approximate database reconstruction (sacrificial ϵ-ADR). Let $\epsilon > 0$ be the desired precision. Let $\mathsf{est\text{-}val}(r)$ denote the value predicted by the algorithm for record r. Sacrificial ϵ-ADR is said to succeed iff one of the following two events occur:

1) For every record r such that $\epsilon N \leq \mathsf{val}(r) \leq N + 1 - \epsilon N$, $|\mathsf{est\text{-}val}(r) - \mathsf{val}(r)| < \epsilon N$.
2) For every record r such that $\epsilon N \leq \mathsf{val}(r) \leq N + 1 - \epsilon N$, $|\mathsf{est\text{-}val}(r) - (N + 1 - \mathsf{val}(r))| < \epsilon N$.

The fact that reconstruction is only possible up to reflection is inherent to this setting, as seen in [KKNO16], [LMP18]. It is required that for all values (except those within ϵN of the extrema), either the estimated value is within ϵN of the correct value, or its reflection. But whichever case it is holds *simultaneously* for all values. In other words, only one bit of information is missing *globally* regarding the reflection symmetry. Note that setting $\epsilon = 1/N$ yields full database reconstruction (FDR), i.e. exact value reconstruction for all records.

Finally, we come to explaining why our attack needs to be *sacrificial*. Sacrificing values that are close to 1 and N is inherent to a *scale-free* attack under a uniform query assumption. Intuitively, these values are harder to recover because fewer range queries touch them. The probability of hitting records with values 1 and N with a uniform range query is $2/(N+1) = \mathcal{O}(1/N)$. This remains true for any record whose value is within $\mathcal{O}(1)$ of 1 or N: hitting one of these records requires $\Omega(N)$ queries. If they are not hit, then it is impossible for the algorithm to differentiate them or determine which records are on the same side of $N/2$—reflection symmetry cannot be determined *globally* for those values. Note that if the set of all records is known our algorithms can infer that these records have values close to either 1 or N because they were not hit by a query.

If a query on some range $[1, x]$ for some $x \in [\epsilon N, N + 1 - \epsilon N]$ is ever issued, then the attacker is trivially able to break the reflection symmetry between the values within ϵN of 1 and N (since the query will hit records with values near one of the endpoints, but not the other). The problem is that with uniform queries, the probability of such a query is $\mathcal{O}(1/N)$, so requiring such a query to occur is not scale-free. In practice, though, a query of that form seems likely, since endpoints are generally "interesting" to query. For that reason, we view the sacrificial aspect of the attack as more of an artefact of the analysis than a practical issue. Nevertheless, it must be addressed in a formal treatment of the attack.

B. Generalizing the KKNO Attack

We now present our generalization of the KKNO attack to sacrificial ϵ-ADR. Our algorithm proceeds in two steps. The first step is to (approximately) recover the value of each record up to reflection individually: for each record, we recover an

approximation of its *symmetric value*, defined as $\mathsf{symval}(r) \stackrel{\text{def}}{=} \min\big(\mathsf{val}(r), N+1-\mathsf{val}(r)\big)$. The second step of the algorithm is to determine which values are on the same side of $N/2$ so that, in the end, the value of records is recovered up to reflection *globally*, as discussed above.

We focus here on the first step of the attack because it suffices to highlight the main ideas. For the second step and its analysis, we refer the reader to the full version. Note that the first step does not sacrifice any values: this is necessary only to break the reflection symmetry.

The idea underpinning the attack is natural: a given query distribution (in this case, the uniform distribution) induces a distribution on the probability that each value is hit by a range query. By measuring that probability empirically, the value of a record can be inferred. More precisely, for a value $k \in [1, N]$, let A_k denote the set of ranges in $[1, N]$ that contain k. Observe that there are $|[1, k] \times [k, N]| = k(N + 1 - k)$ such ranges.

We also assimilate A_k with the event that a uniform range falls within A_k, i.e. contains the value k. Since there are $N(N + 1)/2$ possible (non-empty) ranges in total, we have:

$$p(k) \stackrel{\text{def}}{=} \Pr(A_k) = \frac{2}{N(N+1)} k(N + 1 - k). \qquad (1)$$

We note that $x \mapsto p(x)$ is quadratic and reaches its maximum at $x = (N+1)/2$. It is symmetric about that value, as implied by the reflection symmetry of the setting.

The algorithm simply measures $p(x)$ empirically for each record by counting how many times that record is hit by a query, and dividing by the number of queries. It then infers the symmetric value of the record by choosing k such that $p(k)$ is as close as possible to the empirical measurement. Pseudo-code is provided in Algorithm 1 in Appendix D.

We now turn to the analysis of the algorithm: how many queries are required to achieve sacrificial ϵ-ADR? Because the function p used to infer record values is quadratic and flat around $(N + 1)/2$, getting an error of ϵ on the *input* of p, i.e., on record values, requires an error bounded by $\mathcal{O}(\epsilon^2)$ on the *output* of p. That is, for ϵ-ADR to succeed, the difference between the empirical estimate c/Q for a record r and its expectation $p(\mathsf{val}(r))$ should be $\mathcal{O}(\epsilon^2)$. See the full version for the formal proof.

Hence what we want is that the empirical probability of each event A_k should be within $\mathcal{O}(\epsilon^2)$ of its expected value, for all values k. If we were to naively apply a union bound, since there are N distinct values k, we would get a dependency on N. Instead, a direct application of VC theory shows that $\mathcal{O}(\epsilon^{-4} \log \epsilon^{-1})$ queries suffice, with no dependency on N. To see this, the idea is to define the ground set X as the set of all ranges in $[1, N]$, and the concept set $\mathbb{C}$ as the A_k's, i.e., each A_k is a concept. Then what we want is exactly a $\Omega(\epsilon^2)$-sample on that concept class.

Proposition III.1. *The growth function of* $(\mathsf{X}, \mathbb{C})$ *is* $2n$*, and its VC dimension is 2.*

The proof of Proposition III.1 is given in the full version. As a direct consequence, we can apply the ϵ-sample theorem (Theorem A.4), with $\Omega(\epsilon^2)$ playing the role of the ϵ in the statement of the theorem, to conclude that

$$\mathcal{O}\left(\epsilon^{-4} \log \epsilon^{-1} + \epsilon^{-4} \log \delta^{-1}\right)$$

queries suffice for Algorithm 1 to recover the symmetric value of all records within ϵN, except with probability at most δ. Thus for any fixed probability of success $\eta = 1 - \delta < 1$, Algorithm 1 succeeds within $\mathcal{O}(\epsilon^{-4} \log \epsilon^{-1})$ queries.

For the full attack, see the full version. The rest of the attack uses similar ideas and the first step presented here is representative of the techniques involved. The final query complexity remains $\mathcal{O}(\epsilon^{-4} \log \epsilon^{-1})$.

C. The APPROXVALUE Attack

Next we introduce the APPROXVALUE algorithm, which achieves sacrificial ϵ-ADR within only $\mathcal{O}(\epsilon^{-2} \log \epsilon^{-1})$ queries, saving roughly a square root factor over the generalization of KKNO's algorithm presented above. In particular, setting $\epsilon = 1/N$ yields full database reconstruction within $\mathcal{O}(N^2 \log N)$ queries, significantly improving on KKNO's original $\mathcal{O}(N^4 \log N)$ bound. This comes at the cost of the attack requiring a mild hypothesis about the database. The hypothesis h_1 asks that there exist at least one record in the database with a value in $[0.2N, 0.3N] \cup [0.7N, 0.8N]$. The constants here are for concreteness; others would work as well. The record does not need to be known to the adversary, it suffices that it exist in the database.

We note that an hypothesis such as h_1 is necessary to achieve a query complexity of $\mathcal{O}(\epsilon^{-2} \log \epsilon^{-1})$: in the full version, we prove that any algorithm achieving sacrificial ϵ-ADR in full generality requires $\Omega(\epsilon^{-4})$ queries. With h_1 we prove a similar lower bound of $\Omega(\epsilon^{-2})$, so APPROXVALUE is optimal in data within a log factor. We believe that $\mathcal{O}(\epsilon^{-2})$ (resp. N^2) better captures the actual cost of ϵ-ADR (resp. full reconstruction) on a real database, assuming a uniform query distribution: the $\mathcal{O}(\epsilon^{-4})$ cost of the original attack was because the bound had to account for pathological cases where h_1 is not satisfied. Such databases have all records concentrated around 1, $N/2$ and N. None of the real datasets used in our experiments had this property.

Next we explain how we gain a square factor over the previous attack, which required $\mathcal{O}(\epsilon^{-4} \log \epsilon^{-1})$ queries. Above we saw that the main ϵ^4 term comes as a result of ϵ being squared *twice*: first, to move from the estimate c/Q of $p(\mathsf{val}(r))$ to the symmetric value of the record, and second by applying the ϵ-sample theorem to uniformly approximate p across all values.

The second squaring is inherent: even if we wanted to approximate p on a single value, we would still need to approximate p within ϵ, which requires $\Omega(\epsilon^2)$ queries. (See the full version for more details.) In contrast, the first squaring ultimately comes from the fact that p is quadratic. We can avoid it if we use a linear function to approximate record values: this is exactly what happens in the APPROXVALUE algorithm. Pseudo-code of the APPROXVALUE algorithm is provided in Algorithm 2 in Appendix D. The first step is to identify a record whose (symmetric) value is as close as

possible to $N/4$. Hypothesis h_1 implies a suitable record exists. We call the resulting record the *anchor* record. Because its value is close enough to $N/4$, identifying such a record does not incur the quadratic cost of the function p as in the previous section, because that cost only occurs near the extremum of the function at $(N+1)/2$. We then determine the value of every other record up to symmetry around the anchor by measuring the empirical probability that a query hits both the target record and the anchor record. Let v_A denote the value of the anchor record and k denote the value of a record we are trying to approximate. The expectation of the previous probability is

$$d(v_A, k) \stackrel{\text{def}}{=} \frac{2}{N(N+1)} \cdot \begin{cases} k(N+1-v_A) & \text{if } k \le v_A \\ v_A(N+1-k) & \text{if } k > v_A. \end{cases}$$

The second step of the ApproxValue algorithm is to use the function $x \mapsto d(v_A, x)$ exactly as we used $x \mapsto p(x)$ in the previous section to estimate a record's value. The crucial point is that while $x \mapsto p(x)$ was quadratic, $x \mapsto d(v_A, x)$ is piecewise linear (with a single bend at v_A). This avoids the first squaring discussed earlier.

The third and final step of ApproxValue is to use p once again to break the symmetry around v_A inherent to the mapping $x \mapsto d(v_A, x)$ used in the previous step. However, this limited use of p does not incur a new square factor. In the end, we obtain the following result.

Theorem III.2. *Let $\epsilon < 1/4$. Assume the database under attack satisfies hypothesis h_1. Then after $\mathcal{O}(\epsilon^{-2}\log\epsilon^{-1} + \epsilon^{-2}\log\delta^{-1})$ queries,* ApproxValue *achieves sacrificial ϵ-ADR with probability of success at least $1-\delta$.*

A formal proof is given in the full version. Given any constant probability of success $\eta < 1$, ApproxValue achieves sacrificial ϵ-ADR within $\mathcal{O}(\epsilon^{-2}\log\epsilon^{-1})$ queries.

D. Experimental Results

The ApproxValue attack achieves ϵ-ADR within $\mathcal{O}(\epsilon^{-2}\log\epsilon^{-1})$ queries (for any given constant probability of success $\eta < 1$). We experimentally evaluate the tightness of this bound for a fixed number of records, R, and various numbers of possible values, N, so that we generate both dense and sparse databases. Record values are sampled uniformly at random, so hypothesis h_1 was satisfied with high probability. Our results are averaged over 500 databases, each with 500 randomly sampled queries.

For the attack to succeed, the difference $|\mathsf{est\text{-}val}(r) - \mathsf{val}(r)|$ should be at most ϵN for records at least ϵN away from the endpoints. The records whose values are near the endpoints may have been placed on the wrong side of $N/2$ relative to the anchor record. The bottom group of lines in Figure 2 shows, after every 10 queries, the maximum symmetric value of such misclassified records. As discussed in Section III-A, sacrificing reconstruction of some records is necessary. Nevertheless, we see that our practical results are even better than Theorem III.2 suggests: the upper bound on the maximum symmetric value of a sacrificed record still holds when we take it *with all constants set to 1 –*

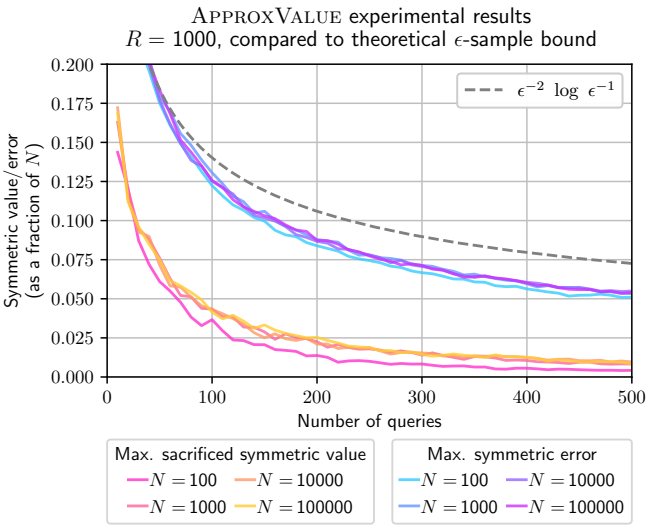

Fig. 2. Maximum symmetric errors of all records and maximum symmetric values of records that were sacrificed. Results averaged over 500 databases satisfying h_1 for each value of N.

in particular, taking the VC dimension to be 1, not taking into account the success probability, and taking any multiplicative constant hidden by the $\mathcal{O}()$ notation to be 1.

The primary reason this "no-constants" bound holds is because the bound in Theorem III.2 inherits the looseness of the ϵ-sample theorem (cf. Theorem A.4): while VC theory is a good predictor of asymptotic behavior, constants are notoriously loose. In particular, one point where loss of tightness arises in the proof of the ϵ-sample theorem (e.g., as in [MU17, Lemma 14.17]) is when using the growth function to upper bound the number of subsamples induced in the so-called double sample. Tightening this bound is possible, for instance, with sample-based growth functions [STW99]. A benefit of running experiments is that they allow us to estimate the constant in practice: in our experiments, simply setting all constants to 1 provided a reasonable estimation of the attack's success.

In addition to limiting the sacrificed values' distance from the endpoints, a successful ϵ-ADR attack must correctly estimate the other records' values within ϵN, up to global reflection. The upper group of lines in Figure 2 is the maximum error of the *symmetric* values, i.e., the maximum difference $|\min\{\mathsf{est\text{-}val}(r), N+1-\mathsf{est\text{-}val}(r)\} - \mathsf{symval}(r)|$ over all records r, as a fraction of N. The reason we plot the symmetric error rather than the absolute error is that it allows us to present results for all records at once—even sacrificed records. It also gives an upper bound on the errors $|\mathsf{est\text{-}val}(r) - \mathsf{val}(r)|$ for records that were not sacrificed. Overall, we see that experimental results behave in the manner predicted by the theory, including scale-freeness, and that the $\mathcal{O}()$ upper bound derived by the theory holds in practice, even when setting the hidden multiplicative constant to just 1.

IV. Approximate Order Reconstruction

In this section, to remove the requirements that the query distribution should be i.i.d. (which we view as unrealistic) and known to the adversary, we turn our attention to sacrificial ϵ-approximate order reconstruction (sacrificial ϵ-AOR), defined in Section IV-A. In Section IV-B, we explain our ApproxOrder algorithm for sacrificial ϵ-AOR, based on PQ-trees, and its analysis. In Section IV-C, we experimentally evaluate the bounds. In Section IV-D we show how the attack can be extended to recover approximate record values, rather than just their order, and present experimental results.

A. Definition of Sacrificial ϵ-AOR

Sacrificial ϵ-approximate order reconstruction (sacrificial ϵ-AOR) asks to recover the order of records, except for records that are within ϵN of each other ("approximate" recovery), and for records within ϵN of the endpoints 1 and N ("sacrificed" records).

We first introduce some notation: if A is a set of records, then the *diameter* of A is the largest difference between the values of any two records in A, i.e.: $\mathsf{diam}(A) \stackrel{\text{def}}{=} \max\{\mathsf{val}(b) - \mathsf{val}(a) : a, b \in A\}$. We let $<$ denote the order on records induced by their values, i.e. $r < s$ iff $\mathsf{val}(r) < \mathsf{val}(s)$. For two sets of records A and B, we write $A < B$ to denote $\forall a \in A, b \in B, a < b$.

An algorithm is said to achieve sacrificial ϵ-AOR iff it outputs disjoint subsets of records $A_1, \ldots, A_k$ such that:

1) $\forall i, \mathsf{diam}(A_i) < \epsilon N$.
2) $A_1 < \cdots < A_k$ holds up to reflection.
3) For all $r \notin \bigcup A_i$, $\mathsf{val}(r) \in [1, \epsilon N[\cup]N + 1 - \epsilon N, N]$.

The definition implies that the algorithm reveals the order of the values of any two records, as soon as they are at least ϵN apart; except possibly for records whose value is within ϵN of 1 or N. If we set $\epsilon = 1/N$, sacrificial ϵ-AOR is equivalent to recovering the exact order of all records.

B. The ApproxOrder Attack

Our attack makes use of *PQ-trees* [BL76], a special structure that makes it possible to represent the set of all orders on records compatible with the access pattern leakage. The leaves of a PQ-tree are labeled by records. Each leaf corresponds to a distinct record. Internal nodes constrain how their children may be ordered: P-nodes allow any ordering, while Q-nodes order their children up to reflection (i.e. only two orderings are possible). A more detailed presentation of PQ-trees is provided in Appendix B.

Pseudo-code for the ApproxOrder attack is given in Algorithm 3 in Appendix D. The idea is to first build the PQ-tree induced by the query access pattern leakage. The algorithm then locates the deepest node T in the tree such that the leaves below T contain a strict majority of all records. The algorithm returns as its output the set A_i of leaves below each child C_i of T, in the order of the children of T. Thus, the order between two records is learned by the adversary iff they appear below distinct children of T, and the order between the two records matches the order of the children of T below which they appear.

Analytically, our main result is as follows. The theorem assumes hypotheses h_2 and h_3, which will be presented below, and a uniform query distribution.

Theorem IV.1. *Let $\epsilon < 1/4$. Assume the database under attack satisfies hypotheses h_2 and h_3. Then after $\mathcal{O}(\epsilon^{-1}\log\epsilon^{-1} + \epsilon^{-1}\log\delta^{-1})$ queries,* ApproxOrder *achieves sacrificial ϵ-AOR with probability of success at least $1-\delta$.*

The proof of Theorem IV.1 is given in the full version. For any fixed constant probability of success, the algorithm succeeds using only $\mathcal{O}(\epsilon^{-1}\log\epsilon^{-1})$ queries. As a direct corollary (setting $\epsilon = 1/N$), the expected number of queries before the PQ-tree collapses into a single Q-node, thus completely revealing the order up to reflection, is $\mathcal{O}(N\log N)$.

The overall idea is that after that number of queries, with high probability there exist certain queries whose endpoints partition $[1, N]$ into sufficiently small buckets while revealing the order between these buckets. By properties of PQ-trees, this situation implies the existence of a node within the PQ-tree that essentially directly reveals ϵ-approximate order (and that node can be easily located as the deepest node covering a majority of records). Moreover, the existence of the aforementioned queries inducing the partition is implied by an ϵ-net, so that ultimately the query complexity required for those queries to exist is directly derived from the ϵ-net theorem of VC theory.

The result does require some assumptions about the existence of records having certain values in the database, namely hypotheses h_2 and h_3. Hypothesis h_2 requires that there exist two records with values a and b such that $a, b \in [N/4, 3N/4]$, and $b - a \geq N/3$ (what really matters for the proof to go through is that a, b should be $\Omega(N)$ away from 1, N and each other); and additionally that there exist at least three records with values within $[\epsilon N, N + 1 - \epsilon N]$ that are more than ϵN away from each other (note that a and b can be two of these values). Hypothesis h_3 requires that a strict majority of all records have a value within $[\epsilon N, N + 1 - \epsilon N]$, and that no range of length ϵN contains the values of a (strict) majority of all records. On the face of it, h_2 and h_3 seem like they are restrictive in that they make several requirements on the database. But those requirements are quite mild. Both hypotheses essentially ask that the database should not be too concentrated over a few values. We have not encountered a real-world dataset that failed to satisfy those requirements. Further, only h_2 is actually required for the T node to exist and leak sacrificial ϵ-approximate order as claimed. The only point of hypothesis h_3, which is more demanding, is to ensure that that node is the deepest node covering a majority of records, so that it can be easily located. But that is a theoretical concern: in our experiments, the desired T node was usually in the first two levels of the tree. Thus, the practically relevant hypothesis is h_2, which only requires that the database should not be entirely concentrated near the endpoints.

In the full verison, we prove that the query complexity

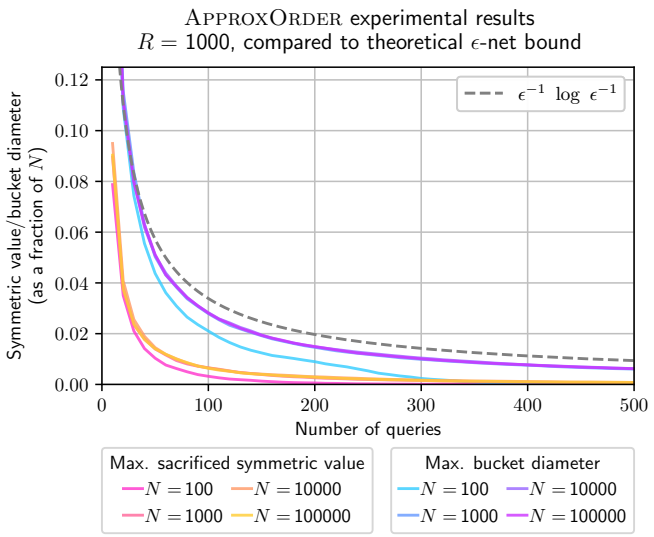

Fig. 3. Maximum symmetric values of records not in buckets and maximum bucket diameters. Results averaged over 500 databases for each value of N.

of our algorithm is optimal within a constant factor. More precisely, any (unbounded) adversary achieving sacrificial ϵ-AOR for all databases must require $\Omega(\epsilon^{-1} \log \epsilon^{-1})$ queries.

C. Experimental Results

Assuming uniform queries, the ApproxOrder attack succeeds within $\mathcal{O}(\epsilon^{-1} \log \epsilon^{-1})$ queries (for any given constant probability of success $\eta < 1$). We experimentally evaluate the tightness of this bound for a fixed number of records R, and various numbers of possible values, N, so that we generate both dense and sparse databases. Record values are sampled uniformly at random, so hypotheses h_2 and h_3 were satisfied with high probability. Our results are averaged over 500 databases, each with 500 randomly sampled queries. We measured the results after every 10 queries, and therefore sometimes needed a heuristic to identify a likely candidate for the Q node when the number of queries is very small. When the root node was not a Q node, our experiments chose the first child Q node that contained at least a third of the records. As our results indicate, this node usually contained an overwhelming majority of the records.

The bottom group of lines in Figure 3 shows the maximum symmetric value (as a fraction of N) of any record that was not in one of the Q node's children buckets. When the ApproxOrder attack succeeds, the only records that are not necessarily in buckets are those with values in $[1, \epsilon N[$ or $]N + 1 - \epsilon N, N]$. If all records have been placed into buckets below the Q node, the maximum excluded symmetric value is set to 0. These results show that the theoretical upper bound holds, even when taking it with *all constants set to 1*, like in Section III-D. The attack also behaves in the predicted *scale-free* way: changing N has little effect on empirical results.

The upper group of lines in Figure 3 shows the maximum diameter (as a fraction of N) of the Q node's child buckets. We compare this to the expected maximum diameter dictated by the ϵ-net bound, and see that convergence happens as quickly as predicted by the bound taken with *all constants set to 1*, as in the previous case. Again, results are scale-free.

Another way of interpreting these results is to ask, after a certain number of queries, for what ϵ have we achieved sacrificial ϵ-approximate order reconstruction? Our results indicate that the bottleneck is the maximum bucket diameter, not the sacrificed values, so the upper group of lines in Figure 3 could be interpreted in this way.

Although our theoretical analysis for the ApproxOrder attack assumes a uniform query distribution, this assumption was *only* for the analysis and the attacker does not need to know the query distribution to carry out the attack. We consider now another more realistic distribution on queries, namely fixed-width range queries. Such queries are widespread in practice: for example, the industry-standard TPC-H contains six explicit fixed-width range queries. For a given number of possible values N and width $W \leq N$, there are $N + 1 - W$ such ranges: $[1, W], [2, W + 1], \ldots, [N + 1 - W, N]$. We experimentally evaluate how well the ApproxOrder attack performs for a dataset of $R = 1000$ records, $N = 10000$ possible values, and range queries of different widths. The results are in Figure 4. Unlike the case of uniform range queries, the limiting factor here in attaining ϵ-AOR is initially the too-high symmetric values of the sacrificed records. For small range widths (relative to the domain size, N), these results are to be expected: when only a few queries have been observed, the total number of possible values that have matched any query so far is limited, and thus the maximum symmetric value of a record that is not in a bucket may be high. After this initial period, the attack's performance follows the results of the uniform range query case and reflects the behaviour of $\epsilon^{-1} \log \epsilon^{-1}$.

D. From AOR to ADR

We now show how our approximate ordering attack can be combined with a model of the database distribution π (commonly called an *auxiliary distribution*) to mount powerful ϵ-ADR attacks. That is, we leverage our approximate ordering attack above to achieve approximate database recovery. Our attack is somewhat reminiscent of the LMP auxiliary distribution attack, with two major differences: (1) it does not require the additional rank leakage used by LMP, and (2) we can study its performance analytically.

We implemented the resulting ϵ-ADR attack and conducted experiments with several datasets representative of practical use cases of encrypted databases. As in the analysis of approximate order reconstruction, we will assume here that the query distribution is uniform *only* to make the exposition simpler—no part of our attack requires queries to be uniformly distributed. Our attack takes as input the output of any algorithm achieving ϵ-AOR. It also takes a model of the database distribution π (which needs only to approximate the

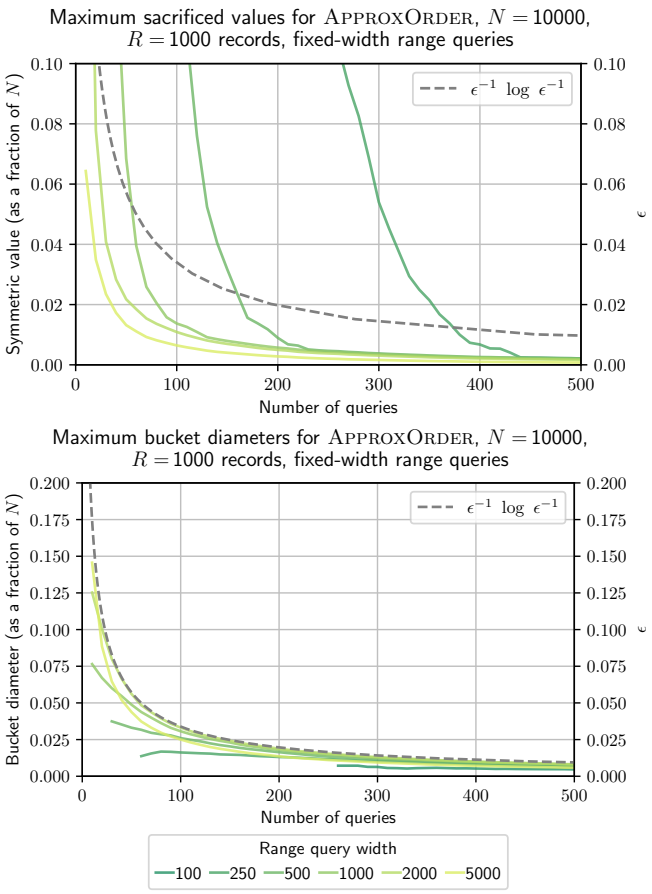

Fig. 4. (**Top**) Maximum symmetric values of records not in buckets. (**Bottom**) Maximum bucket diameters. Results for fixed-width queries averaged over 500 databases for each value of range query width.

true database distribution), the query distribution π_q, and the domain size N. It outputs an estimate for the underlying value of every record in the database. The pseudocode for the attack is given in Algorithm 4 in Appendix D.

Attack intuition. Briefly, the attack uses the observation that, for every disjoint subset of records A_i ($i \in [1, \ldots, k]$) given by ϵ-AOR, some information about the *ranks* of the records (i.e. their positions in a sorted list of all records) in the subset is revealed. Because each sacrificed record could be before A_1 or after A_k the exact ranks are unknown, but lower and upper bounds can be computed.

There are three main questions to answer in building an attack from this observation. **(1)** How should the attack orient the A_i? **(2)** How many sacrificed records should go before A_1? **(3)** How should record values be estimated? We will describe the steps of the attack at a high level here; a full description of the attack steps appears in Appendix C.

The first step is *record rank estimation*. This step has three tasks: choosing an orientation of the A_i, guessing how many of the sacrificed records are less than the first sorted group A_1, and computing a range of possible ranks for the records in each A_i. The attack uses a heuristic to orient the A_i that looks at the number of records above and below the median group. To estimate the number of sacrificed records to put below A_1, we model the number of sacrificed records to the left of A_1 as successes in a binomial distribution and use it to estimate the smallest rank of an element in A_1. After that, the range of ranks for each A_i can be computed as a running sum of the smallest rank and number of records in each group.

The second step is *partition estimation*. This step estimates a range of possible values for each group A_i, given the range of ranks for the group. We estimate the value for each rank using the expected value of the order statistic with that rank. The third step is *database estimation*, which estimates a value for each record in a group given a range of values for that group. Our estimate for each record is the median of the range.

Experiment setup and data. We implemented Algorithm 4 in Python 2.7 and ran all our experiments on an Ubuntu 16.04 desktop with an Intel Core i7-6700 CPU, clocked at 3.4GHz. We used an existing C++ implementation [Gro11] of the PQ-tree data structure and used SWIG [swi18] to call it from Python.

We evaluate the attack on two datasets. The first is a database of registered pilots from the US government Federal Aviation Administration (FAA) [faa17]. It contains the ZIP code of residence for over 61,000 pilots nationwide. ZIP codes are five-decimal-digit numbers. The most significant digits reveal increasingly precise information about location—for example, the first three digits identify a neighborhood in large cities. For more information see [Wik18]. For this experiment we use US Census data about ZIP code population as an auxiliary model of the distribution. Though there are some high-probability ZIP codes, the distribution is overall fairly uniform. Further, the FAA ZIP codes are not well-modeled by the census data - their statistical distance is about 0.51.

The second is a database of California (CA) state public employee salaries from 2016. Salary data is sensitive both for cultural reasons and because of the possibility of blackmail. The database contains over 248,000 numbers between 0 and 762,000 US dollars. Most are salaries (i.e. at least 25,000 US dollars), but a sizeable fraction are in the low hundreds of dollars. We did not remove the low dollar amounts (as doing so could bias experiments in our favor) but we did truncate the few outliers over 500,000 US dollars. We used a database of around 120,000 New York (NY) state public employee salaries from the same year as auxiliary data for this experiment. Both NY and CA salary datasets are roughly Gaussian with means 73,000 and 67,000 respectively. Their statistical distance is about 0.19. Rather than use the full CA salary database, in this experiment we subsampled random databases of 10,000 salaries and averaged the results to better understand how the attack performs on smaller databases.

Results and discussion. Our attacks will measure accuracy as percent error, that is, if the true value of a record is u and the attack guesses v, (for $u, v \in [N]$ the error for that record is $|u - v|/N$. The baseline guessing attack for this accuracy

	Percent Error					
	25%		50%		75%	
# Queries	ZC	SAL	ZC	SAL	ZC	SAL
10	4	2	7	4	11	7
25	2	1	4	2	7	4
50	1	1	3	2	6	3
100	1	1	2	2	5	3
BL	15	2	27	5	37	9

Fig. 5. Accuracy of Algorithm 4 on FAA ZIP codes ("ZC", $N = 9,999$) and CAL salaries ("SAL", $N = 500,000$): percentage of records recovered with error at most the listed percent of N. "BL" refers to baseline guessing. Error is computed as $|(\text{actual}) - (\text{guessed})|/N$).

measure is predicting the median of the database distribution for every record. Figure 5 shows the results of the ZIP code experiment averaged over 20 randomly-generated transcripts and the salary experiments averaged over 10 randomly subsampled databases each with 10 randomly-generated transcripts. We also show the baseline guessing accuracy. The variance was low in all our experiments with 25 or more queries. With only ten queries, the variance for the 75th percentile error is quite high, which we intuitively expect—with so few queries many groups of records will be large. The results in that table assume the A_i have been oriented correctly. Because ZIP codes have a fairly flat distribution our heuristic procedure OrientSubsets chose the wrong orientation in about half of the experiments. The A_i were oriented correctly in all runs of the salary experiment. Since there are only two ways to orient the A_i an incorrect guess is mostly inconsequential. We do not include trials for which the PQ tree does not have a Q node at the first level. This happened only a few times in all experiments for ZIP codes. For salary experiments with 10 queries about one-quarter of the trials did not have a Q node at the first level. With 100 queries, about one-tenth of the trials did not. (The attacker can tell when there is no Q node and choose to see more queries before running the attack.)

The attack on ZIP codes performed extremely well. With only ten queries, we are able to guess the first digit correctly for over half the records on average. Concretely, about half the database records would have their state of residence partially revealed with only ten queries. With only one hundred queries, we recover the first two digits (or a small window with only a few possibilities) for a majority of the records in the database.

For the attack on salaries, the accuracy of the baseline guessing attack is artificially low because of the skew of the distribution—the max value (which we use as the denominator to compute percent error) is much larger than all but a tiny fraction of salaries. Thus, the baseline guessing attack having 5% error translates to 25,000 USD, but most salaries are within 25,000 USD of the median, so baseline guessing performs very poorly on most salaries. In contrast, our attack predicts a majority of the records in the database to within 2% error (10,000 USD) with only fifty queries, and with only 100 queries predicts a quarter with 1% error (5,000 USD).

V. Generalizing Approximate Reconstruction

We have seen how ϵ-nets and ϵ-samples can be used to build and analyze approximate reconstruction attacks on range queries. In this section, we abstract a core technical idea from those attacks – that records accessed the same way by most queries must be "close" – and show how it extends beyond range queries. We explore this in three ways: (1) by using learning theory to define a natural and general notion of distance relevant to access pattern attacks, (2) by showing how ϵ-nets are the right technical tool for analyzing the meaning of this distance for particular query classes, and (3) by using this distance notion to prove a general lower bound on the query complexity of any attack with access pattern leakage. To the best of our knowledge, our general lower bound is the first such proof ever given for this setting and illustrates a core finding of this work: the security impact of access pattern leakage for any class of queries is related to its VC dimension.

Distances induced by range queries. Let C_i be the set of range queries matching value i, and C_j, j. Then, the set of queries matching i XOR j (exactly one of i and j) is $\Delta(C_i, C_j)$, where Δ is the symmetric difference operator on sets, and the number of such queries is $\gamma(i,j) \stackrel{\text{def}}{=} |\Delta(C_i, C_j)|$. We can make three interesting observations about $\gamma(i,j)$. First, it is related to the numerical distance metric $|i-j|$ (though, importantly, they are not identical). Second, $\gamma(\cdot,\cdot)$ is itself a metric on $[N]$. Third, distance in this metric is approximately revealed by the access pattern leakage of range queries: if every query accesses either both or neither records i and j, then $\gamma(i,j)$ is likely to be small. These three properties were used extensively in our attacks on range queries, but they are not specific to range queries: we can abstract them using ideas from learning theory.

Distance, generally. Consider any class of queries $\mathcal{Q}$ on $[N]$ and distribution π over those queries, and consider the concept space $(\mathcal{Q}, \mathbb{C})$ with concepts $\mathbb{C} \stackrel{\text{def}}{=} \{C_i\}_{i\in[N]}$, where each $C_i \stackrel{\text{def}}{=} \{q \in \mathcal{Q} \,|\, q(i) = 1\}$. Each query is a point in this concept space, and there is a set corresponding to each possible value in $[N]$ containing the queries that match it. Now, define the *symmetric difference* concept space $(\mathsf{X}, \mathbb{C})^\Delta \stackrel{\text{def}}{=} (\mathsf{X}, \mathbb{C}^\Delta)$, where $\mathbb{C}^\Delta \stackrel{\text{def}}{=} \{\Delta(C_i, C_j)\}_{i,j\in[N]}$ and $\Delta(\cdot,\cdot)$ is the symmetric difference of the input sets. This new concept space contains, for each pair i, j, the queries which return exactly one of i, j. Next define the function $\gamma_\pi(i,j) \stackrel{\text{def}}{=} \Pr_\pi[\,\Delta(C_i, C_j)\,]$. As above for range queries, where implicitly π was the uniform distribution, this defines a metric on $[N]$. To see that the triangle inequality holds, observe that for any i, j, and k, any query in $\Delta(C_i, C_j)$ is in $\Delta(C_i, C_k)$ or $\Delta(C_k, C_j)$. This allows us to generalize the use of ϵ-nets in ApproxOrder to *arbitrary* query classes. If the adversary observes a set of queries that is an ϵ-net for the symmetric difference concept space, then it must be the case that for any subset S of records with identical access pattern, the underlying values V of those records satisfy $\mathsf{diam}_\gamma(V) \stackrel{\text{def}}{=} \max_{i,j\in V} \gamma_\pi(i,j) \le \epsilon$.

Thus, if we simply group together records that have the same access pattern, then the existence of an ϵ-net provides an upper bound on the distance (with respect to the measure γ) of records in the same group. Essentially, access pattern leakage from *any* query class reveals a kind of approximate equality between the underlying values of the records in the database. This approximate equality depends both on the query class and the query distribution. For range queries, we used this approximate equality to build the ApproxOrder attack and reveal a great deal of information with few queries. However, closeness in the metric γ may not be practically interesting for all query classes and distributions: for example, access pattern leakage from the "query class" which is sampled uniformly at random from $2^{[N]}$ is unlikely to reveal anything interesting. Nevertheless, for many query classes used in practice, closeness in this distance metric can lead to serious privacy breaches. For example, for prefix queries, two values being close in this metric implies they have a common prefix. We will show a simple attack that allows an adversary to reveal which records in the database are approximately equal according to the distance metric γ_π.

Approximate equality attack. Consider a set of queries $\mathcal{Q}$, possible record values $[N]$, and resulting concept space $(\mathcal{Q}, \mathbb{C})$, whose VC dimension d we assume is finite and ≥ 2. Let π_q be any distribution over $\mathcal{Q}$. The attack takes as input records $\{r_1, r_2, \ldots, r_R\}$ along with a 0-1 matrix AP with R rows and Q columns, where $\mathsf{AP}_{ij} = 1$ iff query j returns record i. The attack views each row of the matrix as a number in $[0, 2^Q - 1]$ and outputs a partition by grouping all records with the same number. Let $g_i = \{r_1^i, \ldots, r_k^i\}$ be any such group, and let $V = \{v_1, \ldots, v_k\}$ be the underlying values of these records. An application of the ϵ-net theorem lets us immediately conclude that $\Pr_{\pi_q}[\,\mathsf{diam}_\gamma(V) \leq \epsilon\,] > 1 - (2Q)^d 2^{-\epsilon Q/2}$, and this bound holds for all groups simultaneously.

A. Prefix and Suffix Queries

Next, we show how to instantiate the approximate equality attack for a practically relevant query class. For a set $\Sigma^{\leq \ell}$ of all strings with length $\leq \ell$ from some alphabet Σ, define a *prefix query* q to be a string in the set $\cup_{j=1}^{\ell} \Sigma^j$. In text search, prefix queries are usually indicated by a trailing asterisk "*". For any element $j \in \Sigma^{\leq \ell}$, define the predicate $q(j)$ to be 1 if either $q = j$ or q is a prefix of j, and 0 otherwise. As an example, take the database $\{\mathsf{cat}, \mathsf{carbon}\}$. A prefix query "c*" on these two values would return both, but "carb*" would return only the second one.

Although prefix queries are technically a subset of range queries, there are three crucial differences which obviate the use of previous attacks on range queries: prefix queries do not reveal order, they cannot overlap without one query being contained in the other, and the number of queries matching any fixed string is constant. (Replacing "prefix" with "suffix" in the discussion above gives an identical query class that matches strings based on a suffix instead of a prefix. Our discussion and attacks easily translate to suffix queries, so we dispense with a separate discussion for them.)

In the the symmetric difference concept space for prefix queries, the concepts $\Delta(C_i, C_j)$ for $i, j \in \Sigma^{\leq \ell}$ are the queries that are prefixes of exactly one of i or j. If i and j themselves have a common prefix, though, some prefix queries will match both i and j. More precisely, if i and j have a length-k common prefix, then $|\Delta(C_i, C_j)| = (|i| - k) + (|j| - k)$. Informally, if the adversary notices that two records are always accessed together or not at all, then it can infer that they share a long common prefix. We will describe how to formalize this intuition with ϵ-nets. Further, if the adversary has a model of the database distribution, it can use frequency analysis to learn the characters of each record, one at a time (reminiscent of the climax of the science-fiction movie *WarGames*).

A *WarGames* attack on prefix search. Most modern text and web search systems support prefix queries on unstructured data [es18], and they are ubiquitous in software-as-a-service (SaaS) products like Salesforce, ServiceNow, and Dropbox [sal18], [sno18], [dro18]. A common [dro18], [sal18] design pattern for these systems is to send a prefix query for every character the user types in the search bar. Since users may find their desired result without finishing their query, the distribution of queries is heavily biased towards shorter prefixes.

Our attack in this setting is simple. First, the adversary runs the approximate equality attack described above, obtaining a partition of the records in the database. Then, for each record, it takes the union of all query results containing that record. Here is where we apply the generalized distance notion discussed earlier: with an ϵ-net, we can ensure that each group in the partition contains records with at least a length-one common prefix, and that the unions we form afterwards are exactly the sets of records with the same first character. The first character of each record is then recovered via frequency analysis, and the attack is iterated to learn the second character, then the third, etc.

Analyzing the attack. We model the queries as being sampled via a two-step process. First, a prefix length ℓ_q is sampled from a Zipf distribution on $[\ell]$. (Recall that the standard Zipf distribution on ℓ elements has $\Pr[\,i\,] = (1/i)/H_\ell$, where $H_\ell = \sum_{m=1}^{\ell} 1/m$ is the ℓth harmonic number.) Then, the query is sampled as a uniformly random element of Σ^{ℓ_q}. Call this distribution over queries π_{ts}.

We first consider, for two words $i, j \in \Sigma^{\leq \ell}$, how the length of i and j's common prefix relates to $\Pr_{\pi_{ts}}[\,\Delta(C_i, C_j)\,]$. If i and j have different lengths and share a length-k prefix, then $\Pr_{\pi_{ts}}[\,\Delta(C_i, C_j)\,] = \frac{1}{H_\ell}\left(\sum_{m=k+1}^{|i|} 1/(m|\Sigma|^m) + \sum_{m=k+1}^{|j|} 1/(m|\Sigma|^m)\right)$. Let $\ell_{\min}$ be the length of the shortest string. If the queries observed by the adversary are an ϵ-net for the symmetric difference concept space and for $\epsilon = \frac{1}{H_\ell}\sum_{m=1}^{\ell_{\min}} 1/(m|\Sigma|^m)$, then, for all i, j having no common prefix, we have the distance $\gamma_{\pi_{ts}}(i, j) > \epsilon$ and a query accessing i and j differently must have occurred.

The VC dimension of this concept space is at most 4, so $\mathcal{O}(\frac{1}{\epsilon}\log\frac{1}{\epsilon\delta})$ queries suffice for this attack to recover the first character of every record with probability at least $1-\delta$. This same analysis can be iterated for the rest of the characters.

Experiments. We implemented the attack using last name data from the Fraternal Order of Police (FOP) database dump, posted online in 2016. It contains the personal information of over 600,000 law enforcement officers in the United States. For auxiliary data, we used public US Census statistics [Bur16] on last name frequencies. We also ran the attack on the FAA ZIP code dataset from the experiments in Section IV-D, but it performed quite poorly, primarily due to the auxiliary data being a poor model of the ZIP code distribution.

In 9 out of 10 trials with only 500 prefix queries sampled according to the distribution described above, we were able to partition the records into groups with at least a one-character prefix in common. The mean number of queries required to do this was 315. Once we obtain this partition, we recovered the first character for over 70% of the last-name records. With the same number of trials for 40,000 queries, we recovered the first and second characters of over 55% of the last-name records. With 3 million queries, we recovered the first three characters for over 40% of last-name records, and we recovered roughly 1,500 three-character last names exactly. The sample complexities given by the ϵ-net theorem above are 1,491, 120,000, and 6 million for recovering 1, 2, and 3 characters—much higher than our experiments indicated. As we saw above, applying these results can give loose bounds but the "true" constants are usually small.

This attack on prefix queries can be improved. Our goal was not simply to construct an accurate reconstruction attack for prefix queries, but to demonstrate the power of the learning-theoretic approach in building and analyzing reconstruction attacks. We can generalize the prefix attack to obtain the three basic steps for this approach: (1) define a concept space and a metric, (2) use an ϵ-net to analyze the number of queries needed to learn approximate equality, then (3) perform an attack on the information about values revealed by approximate equality. We note also that standard results [MU17] on intersections and unions of concept classes can extend this approach to *composite* query classes (e.g. a SQL query which intersects the result of a range query on one column and a prefix query on another).

B. A General Lower Bound on Attacks

The metric γ is defined for *any* query class, and in many cases this leads to privacy implications: for range queries, it is closely related to the distance between record values; for prefix queries, the length of the longest common prefix. A general approximate reconstruction attack should recover values that are close (for γ) to the actual record values, and lower bounds on closeness (for γ) should imply lower bounds on the accuracy of any approximate reconstruction attack. The following theorem gives one such lower bound on the number of queries necessary for any approximate reconstruction attack on any query class, as a function of the desired accuracy ϵ and the VC dimension d of the query class.

Theorem V.1. *Let $\mathcal{Q}$ be a class of queries on $[N]$, π_q a query distribution, and $\mathcal{C} = (\mathcal{Q}, \mathbb{C})$ the associated concept space with VC dimension $\mathsf{d} > 1$. Let $\gamma(i,j) \stackrel{\text{def}}{=} \Pr_{\pi_q}[\,\Delta(C_i, C_j)\,]$ be the distance metric induced on $[N]$ by $\mathcal{Q}$ and π_q. Consider any algorithm that takes as input a database of size R with elements in $[N]$, together with the access pattern leakage of m queries sampled from π_q, and outputs an approximation DB' such that $\gamma(DB[i], DB'[i]) \le \epsilon$ for all $i \in [1,\ldots,R]$, with probability of success at least $1-\delta$ (over the choices of queries from π_q). Then m is in $\Omega(\frac{\mathsf{d}}{\epsilon} + \frac{1}{\epsilon}\log\frac{1}{\delta})$.*

This result is a direct application of PAC learning theory: an algorithm that takes any database as input and outputs a DB' satisfying the stated condition is a PAC learner for the concept space $\mathcal{C}$ defined in the theorem statement. We can thus apply a general lower bound [EHKV89] on the sample complexity of PAC learning to conclude that m must be in $\Omega(\frac{d}{\epsilon} + \frac{1}{\epsilon}\log\frac{1}{\delta})$. With a smaller number of queries m, there will be, with probability at least δ, two values in $[N]$ whose distance γ is strictly greater than ϵ, but which every query given to the algorithm accessed in the same way.

This result is not easy to interpret, so we briefly reflect on its implications. First, it holds even if the adversary knows the exact query distribution and database distribution. Next, note that the same lower bound holds for the existence of an ϵ-net: if the queries fail to form an ϵ-net for the metric γ, then some records that are more than ϵ apart in γ cannot be separated based on access pattern. Since any approximate attack should be able to distinguish such records, in some sense this approximate equality attack is a *minimal* approximate attack. For example, consider both our sacrifical ϵ-ADR and -AOR attacks from Sections III and IV. Recovering approximate values or a partition into buckets with small diameters implies we are able to group together approximately-equal records. From this perspective, the lower bound on the existence of an ϵ-net for γ may be interpreted as a lower bound on the number of queries necessary for *any* form of approximate attack for which γ is a relevant notion of distance—not only an approximate attack attempting to recover values, as in Theorem V.1.

VI. Conclusions

This work initiates the application of learning theory to attacks on encrypted databases which leak access patterns. Our learning-theoretic viewpoint lets us build and analyze approximate reconstruction attacks which are both nearly-optimal in query complexity and effective on real data. We believe this work represents an exciting first step towards building a cohesive theory of security in the presence of access pattern leakage. Towards this, we recommend two main research directions for future work to pursue: first, extend our attacks to other query types of practical importance like edit distance, wildcard, and substring queries. Second, study and apply other results from learning theory, such as active or online learning, to access pattern leakage attacks and defenses.

ACKNOWLEDGMENTS

Portions of this work were written while the first author was visiting Royal Holloway, University of London. This work was supported by NSF Graduate Research Fellowship DGE-1650441, the European Union's Horizon 2020 grant ECRYPT-NET (H2020 643161), ERC Project aSCEND (H2020 639554), and EPSRC Grant EP/M013472/1.

REFERENCES

[BEHW86] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred Warmuth. Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, pages 273–282, New York, NY, USA, 1986. ACM.

[BGC+18] Vincent Bindschaedler, Paul Grubbs, David Cash, Thomas Ristenpart, and Vitaly Shmatikov. The tao of inference in privacy-protected databases. *Proc. VLDB Endow.*, 11(11):1715–1728, July 2018.

[BL76] Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.*, 13(3):335–379, 1976.

[Bur16] US Census Bureau. US Census Bureau name statistics. https://www.ssa.gov/OACT/babynames/, 2016.

[CGPR15] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. In *CCS*, 2015.

[DL] Sanjoy Dasgupta and John Langford. A tutorial on active learning. http://hunch.net/~active_learning/.

[dro18] Dropbox, 2018. https://www.dropbox.com.

[EHKV89] Andrzej Ehrenfeucht, David Haussler, Michael Kearns, and Leslie Valiant. A general lower bound on the number of examples needed for learning. *Inf. Comput.*, 82(3):247–261, September 1989.

[es18] ElasticSearch, 2018. https://www.elastic.co/.

[faa17] Federal Aviation Administration pilot database, 2017. https://www.faa.gov/regulations_policies/pilot_records_database/.

[FVY+17] Benjamin Fuller, Mayank Varia, Arkady Yerukhimovich, Emily Shen, Ariel Hamlin, Vijay Gadepally, Richard Shay, John Darby Mitchell, and Robert K. Cunningham. SoK: Cryptographically protected database search. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 172–191, May 2017.

[GMN+16] Paul Grubbs, Richard McPherson, Muhammad Naveed, Thomas Ristenpart, and Vitaly Shmatikov. Breaking web applications built on top of encrypted data. In *CCS*, 2016.

[Gro11] Greg Grothaus. General implementation of the PQ-tree algorithm, 2011. https://github.com/Gregable/pq-trees.

[GSB+17] Paul Grubbs, Kevin Sekniqi, Vincent Bindschaedler, Muhammad Naveed, and Thomas Ristenpart. Leakage-abuse attacks against order-revealing encryption. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 655–672, May 2017.

[HW86] David Haussler and Emo Welzl. Epsilon-nets and simplex range queries. In *Proceedings of the Second Annual Symposium on Computational Geometry*, SCG '86, pages 61–71, New York, NY, USA, 1986. ACM.

[JR13] Jonathan L. Dautrich Jr. and Chinya V. Ravishankar. Compromising privacy in precise query protocols. In *Joint 2013 EDBT/ICDT Conferences, EDBT '13 Proceedings*, pages 155–166. ACM, 2013.

[KKNO16] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. Generic attacks on secure outsourced databases. In *CCS*, 2016.

[KPT18] Evgenios M. Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. Data recovery on encrypted databases with k-nearest neighbor query leakage. Cryptology ePrint Archive, Report 2018/719, 2018. https://eprint.iacr.org/2018/719.

[KVV94] Michael J. Kearns, Umesh Virkumar Vazirani, and Umesh Vazirani. *An introduction to computational learning theory*. MIT press, 1994.

[LMP18] Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson. Improved reconstruction attacks on encrypted data using range query leakage. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 297–314, May 2018.

[MU17] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, New York, NY, USA, 2nd edition, 2017.

[NKW15] Muhammad Naveed, Seny Kamara, and Charles V Wright. Inference attacks on property-preserving encrypted databases. In *CCS*, 2015.

[sal18] Salesforce.com, 2018. https://www.salesforce.com.

[Sau72] N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145–147, 1972.

[sno18] ServiceNow, 2018. https://www.servicenow.com.

[STW99] John Shawe-Taylor and Robert C. Williamson. Generalization performance of classifiers in terms of observed covering numbers. In *Computational Learning Theory*, pages 274–285, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[swi18] Simplified wrapper and interface generator (SWIG), 2018. http://www.swig.org/.

[Val84] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 1984.

[VC71] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971. Translation by B. Seckler.

[Wik18] Wikipedia contributors. ZIP code — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/ZIP_Code, 2018.

[ZKP16] Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. All your queries are belong to us: The power of file-injection attacks on searchable encryption. In *25th USENIX Security Symposium, USENIX Security 16*, pages 707–720, 2016.

APPENDIX A
STATISTICAL LEARNING THEORY PRIMER

We provide a brief summary of learning theory, using terminology from a recent textbook [MU17].

A. Concept spaces, ϵ-nets, ϵ-samples

Let X be some set of (base) elements. (We consider only finite sets in this work.) A *concept* (also called *event* or *range*) A is a subset of X. Given a probability distribution $\mathcal{D}$ on the set X, let $f_{\mathcal{D}}$ represent the pmf. We can then assign a probability to any concept A in the natural way: $\Pr_{\mathcal{D}}(A)$ is the probability that a single element of X sampled according to $\mathcal{D}$ is in A, i.e., $\Pr_{\mathcal{D}}(A) = \sum_{a \in A} f_{\mathcal{D}}(a)$.

A *concept space* (or *set system*) is a pair $(\mathsf{X}, \mathbb{C})$ where $\mathbb{C}$ is a set of *concepts* (subsets) of X (also called *subset family*). Any concept C can also be viewed as a function from X to $\{0, 1\}$: the function's output on input $x \in \mathsf{X}$ is 1 if $x \in C$ and 0 otherwise. Given a concept space $(\mathsf{X}, \mathbb{C})$ and a sample S of elements drawn from X according to $\mathcal{D}$, we may ask the following questions:

- Does every concept in $\mathbb{C}$ with some not-too-small probability occur in the sample S?
- Is the relative occurrence of every concept of $\mathbb{C}$ in the sample S close to its expectation?

Answering these questions involves analyzing objects called ϵ-nets [HW86] and ϵ-samples.

Definition A.1. *A subset $S \subseteq \mathsf{X}$ is an ϵ-net for the concept space $(\mathsf{X}, \mathbb{C})$ with respect to the distribution $\mathcal{D}$ if for every*

event $C \in \mathbb{C}$ with $\Pr_{\mathcal{D}}(C) \geq \epsilon$, the intersection $S \cap C$ is non-empty.

Definition A.2. *A subset $S \subseteq \mathsf{X}$ is an ϵ-sample (also called* ϵ-approximation*) for the concept space $(\mathsf{X}, \mathbb{C})$ with respect to the distribution $\mathcal{D}$ if for every concept $C \in \mathbb{C}$,*

$$\left| \frac{|S \cap C|}{|S|} - Pr_{\mathcal{D}}(C) \right| \leq \epsilon.$$

Informally, a sample S is an ϵ-sample when every concept's relative frequency in S is within ϵ of its true probability. Thinking of ϵ as being small, this can be seen as saying that S "induces" uniform convergence of relative frequencies to probabilities.

One way to analyze when a set S is an ϵ-net or an ϵ-sample is to characterize the complexity of the concept space. We turn to this next.

B. Shattering, VC dimension, growth functions

The critical measures in determining the complexity of a concept space are the growth function $m^{\mathbb{C}}(n)$ and the Vapnik-Chervonenkis (VC) dimension d, which are related. Given a concept space $(\mathsf{X}, \mathbb{C})$ and a finite sample $S \subseteq \mathsf{X}$, an important object is the *set of subsamples of S induced by* $\mathbb{C}$ (also called *the projection of $\mathbb{C}$ on S*): $\mathbb{C}_S := \{C \cap S\}_{C \in \mathbb{C}}$. The size of this set is the *index of* $\mathbb{C}$ *with respect to* S:

$$\Delta^{\mathbb{C}}(S) := |\mathbb{C}_S| = |\{A \subseteq \mathsf{X} : \exists C \in \mathbb{C} \text{ with } A = C \cap S\}|.$$

Clearly, the index of a concept space relative to a set S is at most $2^{|S|}$ and, when $\mathbb{C}$ is finite, it is at most $|\mathbb{C}|$.

The concept space $(\mathsf{X}, \mathbb{C})$ *shatters* the sample $S \subseteq \mathsf{X}$ if $\mathbb{C}$ induces all possible subsamples of S, i.e., $\Delta^{\mathbb{C}}(S) = 2^{|S|}$.

The *VC dimension* (also called *density* [Sau72] or *capacity* and denoted by d) of a concept space $(\mathsf{X}, \mathbb{C})$ is the largest cardinality (possibly infinite) of a set $S \subseteq \mathsf{X}$ that can be shattered by $\mathbb{C}$. (It is sufficient for only one set of this size to exist; not all sets of this size need to be shattered by $\mathbb{C}$.) VC dimension is an indicator of the complexity of a concept space. Related to VC dimension is the *growth function* of a concept space $(\mathsf{X}, \mathbb{C})$, which is the maximum index of $\mathbb{C}$ over all samples $S \subseteq \mathsf{X}$ of size n: $m^{\mathbb{C}}(n) := \max_{S \subseteq \mathsf{X} : |S| = n} \Delta^{\mathbb{C}}(S)$.

The VC dimension, then, is the largest value of n for which the growth function equals 2^n. Knowing the VC dimension of a concept space is sufficient to determine an upper bound on the growth function: either d is infinite or the growth function is bounded by $\sum_{i=0}^{d} \binom{n}{i}$.

Lemma A.3 (Sauer's Lemma [Sau72])**.** *Let $(\mathsf{X}, \mathbb{C})$ be a concept space having finite VC dimension d. Then, the growth function satisfies $m^{\mathbb{C}}(n) \leq \sum_{i=0}^{d} \binom{n}{i}$.*

This partial sum of binomial coefficients $\sum_{i=0}^{d} \binom{n}{i}$ is denoted by $\mathcal{G}(d, n)$ [MU17]. By induction on d, it is straightforward to show that it is upper-bounded by n^d for $d \geq 2$.

C. Sufficient conditions for ϵ-nets and ϵ-samples

In their groundbreaking paper, Vapnik and Chervonenkis established a lower bound [VC71, Thm. 2] on the probability that a sample S is an ϵ-sample, i.e., that the relative frequencies of events in $\mathbb{C}$ are all within ϵ of their true probabilities.

Theorem A.4 (Sufficient conditions for ϵ-sample [VC71])**.** *Let $(\mathsf{X}, \mathbb{C})$ be a concept space with growth function $m^{\mathbb{C}}(n)$ and VC dimension d. Let $\mathcal{D}$ be a probability distribution on X and let S be a set of size n drawn from X according to $\mathcal{D}$. Then, for any $\epsilon > 0$, the probability that S is an ϵ-sample is at least $1 - 4 \cdot m^{\mathbb{C}}(2n) \cdot e^{-\epsilon^2 n/8}$. In particular, there is an n that is*

$$\mathcal{O}\left(\frac{d}{\epsilon^2} \log \frac{d}{\epsilon} + \frac{1}{\epsilon^2} \log \frac{1}{\delta} \right)$$

such that any sample S of size at least n is an ϵ-sample with probability at least $1 - \delta$.

More precisely, [MU17, Thm. 14.15] establishes that a sample of size at least $\frac{32d}{\epsilon^2} \log \frac{64d}{\epsilon^2} + \frac{16}{\epsilon^2} \log \frac{2}{\delta}$ is an ϵ-sample with probability at least $1 - \delta$.

Inspired by Vapnik and Chervonenkis's work on ϵ-samples, Haussler and Welzel introduced ϵ-nets and derived a lower bound in the case where the distribution over X is uniform [HW86, Thm. 3.7]. Later work extended this bound to arbitrary distributions:

Theorem A.5 (Sufficient conditions for ϵ-net [MU17])**.** *Let $(\mathsf{X}, \mathbb{C})$ be a concept space with growth function $m^{\mathbb{C}}(n)$ and VC dimension d. Let $\mathcal{D}$ be a probability distribution on X and let S be a set of size n drawn from X according to $\mathcal{D}$. Then, for any $\epsilon > 0$, the probability that S is an ϵ-net is at least $1 - 2 \cdot m^{\mathbb{C}}(2n) \cdot e^{-\epsilon n/2}$. In particular, there is an n that is*

$$\mathcal{O}\left(\frac{d}{\epsilon} \log \frac{d}{\epsilon} + \frac{1}{\epsilon} \log \frac{1}{\delta} \right)$$

such that a sample of at least this size is an ϵ-net with probability at least $1 - \delta$. Specifically, a random sample of size at least $\max\{\frac{8d}{\epsilon} \log \frac{16d}{\epsilon}, \frac{4}{\epsilon} \log \frac{2}{\delta}\}$ is an ϵ-net with probability at least $1 - \delta$.

Ehrenfeucht *et al.* prove a lower bound [EHKV89, Cor. 5] on the number of samples needed to obtain an ϵ-net with probability at least $1 - \delta$. Since every ϵ-sample is an ϵ-net, this lower bound also applies to ϵ-samples.

Theorem A.6 (Necessary conditions for ϵ-net or ϵ-sample [EHKV89], [MU17])**.** *Let $(\mathsf{X}, \mathbb{C})$ be a concept space of VC dimension d. Let $\mathcal{D}$ be a probability distribution on X and let S be sample drawn from X according to $\mathcal{D}$. Let $\epsilon > 0$ and $\delta > 0$. Suppose S is an ϵ-net (or ϵ-sample) with probability at least $1 - \delta$. Then $|S| = \Omega\left(\frac{d}{\epsilon} + \frac{1}{\epsilon} \log \frac{1}{\delta}\right)$.*

D. PAC Learning

Introduced by Valiant [Val84], PAC learning is concerned with algorithms which learn from labelled examples. (We restrict our attention to *realizable* and *consistent* PAC learning; for a more general treatment see [KVV94].) Using the terminology above, a learner $\mathcal{L}$ is an algorithm which takes as input a transcript $\{(s_i, C(s_i))\}_{i=1}^{m}$ of elements from X (where each $s_i \leftarrow\!\!\$\ \pi$ for some distribution π on X) along with their labels according to the unknown concept C. A learner

outputs a *hypothesis* $H \in \mathbb{C}$ representing its guess for C. The learner $\mathcal{L}$ is a PAC learner for $\mathbb{C}$ if for any $C \in \mathbb{C}$, distribution π on X, and $0 < \epsilon, \delta < 1$, for a sample (or transcript) of size m in $\mathcal{O}(\text{poly}(\frac{1}{\epsilon}, \frac{1}{\delta}))$ drawn according to π, the *generalization error* $\Pr_\pi[\,\{x \in \mathsf{X} \mid H(x) \neq C(x)\}\,]$ is less than ϵ with probability at least $1 - \delta$. In words, the generalization error is the probability under π that an element is labelled differently by H and C. A central result [BEHW86] in learning theory states that if $\mathcal{C}$ has finite VC dimension, then there exists a (not necessarily efficient) PAC learner for $\mathcal{C}$.

APPENDIX B
PRESENTATION OF PQ-TREES

PQ-trees were introduced in [BL76], and are typically used as tools to solve other algorithmic problems (such as planarity testing for graphs). Given a ground set X with an unknown order, together with a set $\mathcal{I}$ containing intervals of X for that order, a PQ-tree succinctly represents (i.e. in size linear in $|X|$) the set of all orderings of X that are compatible with $\mathcal{I}$. Moreover a PQ-tree can be updated on the fly: given a PQ-tree and a new interval I not previously contained in $\mathcal{I}$, the PQ-tree can be updated in linear time to only contain those orders that are compatible with $\mathcal{I} \cup \{I\}$. (Details of the update procedure are not relevant to our work.)

The structure of a PQ-tree is simple: the leaves of the tree are labelled by the ground set X, each element of X appearing in exactly one leaf. Internal nodes of the tree consist in two types of nodes: P-nodes and Q-nodes. Both types of nodes can have any number of children leaf or non-leaf nodes. P-nodes denote that the children of the node can be ordered in any way. For example, if $X = \{a, b, c\}$, then the tree $P(a, b, c)$ represents the set of all permutations of X. Q-nodes denote that the children of the node can only be ordered either as they appear in the tree, or as the reverse order (a.k.a. its reflection). For example, if $X = \{a, b, c\}$, then the tree $Q(a, b, c)$ represents the orders abc and cba. Nodes combine in the natural way: for example, the tree $Q(a, b, P(c, d))$ represents the possible orders $abcd$, $abdc$, $cdba$, $dcba$.

Initially, when no information whatsoever is known about the order, the PQ-tree consists in a single P-node, with all elements of the ground set as leaves. Conversely, once the order is fully determined, the tree consists in a single Q-node, whose leaves are either in the correct order or its reflection. This corresponds to the usual reflection symmetry, which cannot be broken by learning intervals (as replacing the order on X by its reflection leaves the set of intervals invariant).

In our setting, it is possible that the ground set X, which corresponds to the set of all record IDs, is not known in advance. However PQ-trees can be easily extended to handle that case: to do so, we start with a tree formed of a P-node with a single leaf labelled by a special element $\star$. Whenever the tree is updated with a new set I, if I contains new elements not already among the leaves of the current tree, these new elements are first added as siblings of $\star$. The tree is then updated with I as normal.

APPENDIX C
DESCRIPTION OF ϵ-ADR ATTACK

We will describe our attack and how it resolves the three questions posed in Section IV-D. The first step, **record rank estimation** orients the A_i (question 1), estimates the number of sacrificed records less than A_1 (question 2), and produces an estimate of the range of ranks for each group. The second step, **partition estimation**, uses order statistics to estimate a range of *values* (i.e. a partition of $[N]$) for each range of ranks obtained in the previous step. The third step, **database estimation**, estimates a value for the records in each group given the estimated partition (question 3).

Record rank estimation. To recap, record rank estimation must (1) orient the A_is, (2) guess the number of sacrificed records less than A_1, the first sorted group, and (3) estimate a range of ranks for each A_i. Our attack uses a heuristic for orienting the A_i (this is function OrientSubsets in Algorithm 4): first, measure the proportion of records above and below the middle group. Call these quantities $\hat{p}_a$ and $\hat{p}_b$, respectively. Then, compute the probability of a database value falling above (p_a) and below (p_b) the value $\lceil N/2 \rceil$. If $\hat{p}_a > \hat{p}_b$ and $p_a > p_b$, keep that orientation, else choose the other one. Though quite naive, below we will see this heuristic generally works well for real data distributions.

To guess the number of sacrificed records below the first sorted group (EstimateRank in Algorithm 4), we use a more principled approach. Observe that the sacrificed records are exactly those with values either lower than the value of the smallest left query endpoint (call this value ℓ_{min}) or higher than the value of the largest right endpoint (call this r_{max}). Let $E_{ij} = (\ell_{\text{min}} = i) \cap (r_{\text{max}} = j)$ be the event that ℓ_{min} is i and r_{max} is j. Let the number of sacrificed records be S and r_0 be a random variable denoting the smallest rank for a record in A_1. The RV r_0 takes values in $[0, \ldots, S]$. Conditioned on E_{ij}, the distribution of the number of sacrificed records to the left of i and right of j is binomial with sample size S and probability of success $p_{ij} = \frac{\Pr[\,1,\ldots,i\,]}{\Pr[\,1,\ldots,i\,]+\Pr[\,j,\ldots,N\,]}$ where $\Pr[\,x, \ldots, y\,] = \sum_{k=x}^{y} \pi(k)$ and π is the auxiliary distribution. Thus, for any $r \in [0, \ldots, S]$,

$$\begin{aligned}\Pr[\,r_0 = r\,] &= \sum_{i \le j \in [N]} \Pr[\,r_0 = r \mid E_{ij}\,] \Pr[\,E_{ij}\,] \\ &= \sum_{i \le j \in [N]} \binom{S}{r} p_{ij}^r (1 - p_{ij})^{S-r} \Pr[\,E_{ij}\,] \ .\end{aligned}$$

If the number of queries is Q and the query distribution is uniform, we can compute $\Pr[\,E_{ij}\,]$ via inclusion-exclusion as follows. First, define $f(x, y) = (x - y)(y - x + 1)/N(N + 1)$. Then

$$\begin{aligned}\Pr[\,E_{ij}\,] = f(i, j)^Q &- f(i, j-1)^Q \\ &- f(i+1, j)^Q + f(i+1, j-1)^Q \ .\end{aligned}$$

This is the only part of the attack that uses the uniform distribution on queries. If we let $\pi_q^{[i,j]}$ be the probability that a query is contained in the range $[i, j]$, with a non-

uniform query distribution this expression would be the same except with $f(\cdot,\cdot)$ replaced by $\pi_q^{[\cdot,\cdot]}$. The value $\hat{r_0}$ output by function EstimateRank is then $\mathbb{E}[r_0] = \sum_{r=0}^{S} r \Pr[r_0 = r]$. The expression $\Pr[r_0 = r]$ has $\mathcal{O}(N^2)$ terms, which could make the attack scale poorly. Our implementation uses a heuristic to discard the terms for which $\Pr[E_{ij}]$ is very small, so computing $\mathbb{E}[r_0]$ (a one-time operation) takes only about eighty minutes in the worst case. Once we compute $\hat{r_0}$ we can find the lower and upper ranks for the A_i via addition; see the line assigning r_i in Algorithm 4.

Partition estimation. The output of the previous step is a lower and upper rank (call them r_{lb} and r_{ub}) for each A_i. From this we will recover a lower and upper value (ep_{lb} and ep_{ub}) used by the final step of the attack. To estimate values from ranks, we use order statistics. For a sample $X_1, \ldots, X_s$, the kth order statistic (denoted $X_{(k)}$, $k \in [1, \ldots, s]$) is the kth largest value in the sample. The probability $\Pr[X_{(k)} = u]$ has a simple formula: $\Pr[X_{(k)} = u] = \Pr[X_{(k)} \le u] - \Pr[X_{(k)} \le u - 1]$, where $\Pr[X_{(k)} \le u]$ is

$$\sum_{j=0}^{s-k} \binom{s}{j} (1 - \Pr[1, \ldots, u])^j \Pr[1, \ldots, u]^{s-j} \ .$$

Using this, we estimate $ep_{lb} = \max_x \Pr[X_{(r_{lb})} = x]$ and do the same for ep_{ub}. For a fixed rank and varying x, the distribution of $\Pr[X_{(r_{lb})} = x]$ converges to a Gaussian very quickly, so $\max_x \Pr[X_{(r_{lb})} = x] \approx \mathbb{E}[X_{(r_{lb})}]$.

Database estimation. This is the simplest step—the previous step outputs a partition $[1, ep_1, \ldots, ep_{|\mathcal{B}|}, N]$ of $[N]$ where the records in group b_i are between ep_i and ep_{i+1}, so we need only choose a value in $[ep_i, ep_{i+1}]$ to assign to the records in b_i. Since we are concerned with minimizing the absolute value of the difference between the true value and the guess, the natural choice is the median of the database distribution π, conditioned on the range $[ep_i, ep_{i+1}]$. In Algorithm 4 this is written as RangeMedian(π, ep_i, ep_{i+1}).

APPENDIX D
ATTACK PSEUDOCODE

In this appendix we give detailed pseudocode descriptions of some attacks from the main body of the article.

In Algorithm 3, the following notation related to PQ-trees is used. If $\mathcal{T}$ is a PQ-tree, and T is a node of $\mathcal{T}$, then the leaves of T are defined as the leaves of the subtree rooted at T, and denoted $\mathsf{leaf}(T)$. If $\mathcal{T}$ is a PQ-tree, $\mathsf{root}(\mathcal{T})$ is the root of $\mathcal{T}$.

Algorithm 1 Estimating symval.

Input: Set of queries $\mathcal{Q}$.
Output: Function est-symval approximating symval.

for each record r **do**
  $c(r) \leftarrow |\{q \in \mathcal{Q} : r \in q\}|/|\mathcal{Q}|$
  est-symval$(r) \leftarrow \arg\min_{k \in [N/2]} |p(k) - c(r)|$
end for
return est-symval

Algorithm 2 ADR Algorithm ApproxValue..

ApproxValue($\mathcal{Q}$):
Input: Set of queries $\mathcal{Q}$.
Output: Function est-val approximating val.

for each record r **do**
  $c(r) \leftarrow |\{q \in \mathcal{Q} : r \in q\}|/|\mathcal{Q}|$
  $\widetilde{v}(r) \leftarrow \arg\min_k |c(r) - p(k)|$
end for
$r_A \leftarrow \arg\min_r |\widetilde{v}(r) - N/4|$ ▷ Anchor record
$\widetilde{v_A} \leftarrow \widetilde{v}(r_A)$ ▷ Est. anchor value
for each record r **do**
  $c'(r) \leftarrow |\{q \in \mathcal{Q} : r_A, r \in q\}|/|\mathcal{Q}|$
  $\widetilde{w_L} \leftarrow \arg\min_{k \in [1, \widetilde{v_A}]} |d(\widetilde{v_A}, k) - c'(r)|$
  $\widetilde{w_R} \leftarrow \arg\min_{k \in [\widetilde{v_A}, N]} |d(\widetilde{v_A}, k) - c'(r)|$
  if $c(r) < (p(\widetilde{w_L}) + p(\widetilde{w_R}))/2$ **then**
    est-val$(r) \leftarrow \widetilde{w_L}$
  else
    est-val$(r) \leftarrow \widetilde{w_R}$
  end if
end for

Algorithm 3 AOR Algorithm ApproxOrder.

ApproxOrder($\mathcal{Q}$):
Input: Set of queries $\mathcal{Q}$.
Output: Disjoint subsets of records $A_1, \ldots, A_k$.

$\mathcal{T} \leftarrow$ PQ-tree built from $\mathcal{Q}$.
$T \leftarrow$ FindNodeT$(\mathcal{T}, \mathsf{root}(\mathcal{T}))$
$C_1, \ldots, C_k \leftarrow$ children of T (in order)
return $\mathsf{leaf}(C_1)$, …, $\mathsf{leaf}(C_k)$

FindNodeT($\mathcal{T}, S$):
Input: PQ-tree $\mathcal{T}$ and node S of $\mathcal{T}$.
Output: Deepest node $S' \le S$ with $> R/2$ leaves.

$R \leftarrow |\mathsf{leaf}(\mathsf{root}(T))|$
for each child C of S **do**
  if $|\mathsf{leaf}(\mathcal{T}, C)| > R/2$ **then**
    return FindNodeT$(\mathcal{Q}, \mathcal{T}, C)$
  end if
end for
return S

Algorithm 4 Recovering values from approximate order.

Input: $A'_1, \ldots, A'_k$, e, R,π,π_q,N.
Output: $[x_1, x_2, \ldots, x_R]$ ($\forall i$,$x_i \in [N]$).

$A_1, \ldots, A_k \leftarrow$ OrientSubsets$(A'_1, \ldots, A'_k)$
$\hat{r_0} \leftarrow$ EstimateRank(e, π_q, π)
for all $A_i, i \in [1, \ldots, k]$ **do**
    $r_i \leftarrow r_{i-1} + |A_i|$
    $\mathrm{ep}_i \leftarrow \arg\max_{k \in [N]} \Pr\left[X_{(r_i)} = k \right]$
    $\mathsf{med}_{A_i} \leftarrow$ RangeMedian$(\pi, \mathrm{ep}_{i-1}, \mathrm{ep}_i)$
    for all $\mathsf{ind} \in A_i$ **do**
        $\mathrm{cand}[\mathsf{ind}] = \mathsf{med}_{A_i}$
    end for
end for
return cand