Improved Reconstruction Attacks on Encrypted Data Using Range Query Leakage

Marie-Sarah Lacharité, Brice Minaud, Kenneth G. Paterson
Information Security Group
Royal Holloway, University of London
Egham, United Kingdom
Email: {marie-sarah.lacharite.2015, brice.minaud, kenny.paterson}@rhul.ac.uk

***Abstract*—We analyse the security of database encryption schemes supporting range queries against persistent adversaries. The bulk of our work applies to a generic setting, where the adversary's view is limited to the set of records matched by each query (known as *access pattern* leakage). We also consider a more specific setting where *rank* information is also leaked, which is inherent to multiple recent encryption schemes supporting range queries. We provide three attacks.**

First, we consider *full reconstruction*, which aims to recover the value of every record, fully negating encryption. We show that for dense datasets, full reconstruction is possible within an expected number of queries $N \log N + O(N)$, where N is the number of distinct plaintext values. This directly improves on a quadratic bound in the same setting by Kellaris *et al.* (CCS 2016).

Second, we present an *approximate reconstruction* attack recovering all plaintext values in a dense dataset within a constant ratio of error, requiring the access pattern leakage of only $O(N)$ queries.

Third, we devise an attack in the common setting where the adversary has access to an *auxiliary distribution* for the target dataset. This third attack proves highly effective on age data from real-world medical data sets. In our experiments, observing only 25 queries was sufficient to reconstruct a majority of records to within 5 years.

In combination, our attacks show that current approaches to enabling range queries offer little security when the threat model goes beyond snapshot attacks to include a persistent server-side adversary.

***Index Terms*—privacy; cryptanalysis; encrypted database.**

I. Introduction

Various kinds of property-preserving encryption (PPE) schemes have started to see wide deployment, in particular in the area of data storage outsourcing. There, a client encrypts a set of records or documents using a PPE scheme, sends it to the server, and can later query the server and retrieve matching records. By exploiting the special properties of the encryption scheme, the server can index the data just as it would unencrypted data, allowing the server to support efficient search. For example, deterministic encryption allows matching queries to be made, while Order-Preserving/Revealing Encryption (OPE/ORE) allow range queries to be efficiently supported.

At the same time, our understanding of the security that such schemes offer against various kinds of adversary is still developing. This has led to serious attacks being found against some of the early schemes [1], [2], [3], [4], [5], [6], [7] – a good summary of this line of research is available in [8]. A second generation of schemes, which typically use custom indexes rather than legacy indexes, promise to do better, in the sense of provably leaking less information about encrypted data. Perhaps inevitably, the second generation of schemes has been followed by another wave of attacks. Kellaris, Kollios, Nissim, and O'Neill introduced generic reconstruction attacks applicable to any scheme whose range queries leak access pattern or communication volume, for a uniform range query distribution [9]. Grubbs *et al.* presented a snapshot attack on non-deterministic, frequency-hiding OPE schemes when auxiliary information about the plaintext distribution is available [7]. We continue this line of research into generic attacks, which apply even to second-generation encryption schemes, focussing on those schemes that support range queries.

A. Setting and Notation

All three of our attacks share the same general setting, which we introduce here. We let $[a, b]$ denote the set of integers $\{a, \ldots, b\}$ (with $[a, b] = \varnothing$ whenever $a > b$).

a) Records, identifiers, and values: We consider a collection of R records in a database, each with a unique identifier in the set $\mathcal{R}$ and a (not necessarily unique) value $\mathsf{val}(\cdot)$ from some ordered set $\mathcal{X}$, on which range queries are performed. Without loss of generality, we assume that $\mathcal{R} = [1, R]$ and $\mathcal{X} = [1, N]$, with "$<$" being the usual ordering on

the integers. We let $\mathcal{S}_x \subseteq [1, R]$ denote the set of identifiers of all records that contain value x. We assume the attacker knows the set of all record identifiers. (We discuss the implications of this assumption in Appendix C – in any case, with uniform queries, the set of all identifiers is recovered within $O(N)$ queries.)

b) Range queries: A range query $[x, y]$ is defined by its two end points $x \leq y$ in $\mathcal{X}$. In our analyses, except where indicated otherwise, range queries are modelled as uniformly distributed non-empty intervals in $[1, N]$. The uniformity assumption is briefly discussed in the next section. A range query returns, at the very least, a set of matching record identifiers $\mathcal{M} := \{r \in \mathcal{R} : \mathsf{val}(r) \in [x, y]\}$. We also write $\mathcal{M} = \mathcal{S}_{[x,y]} := \cup_{x \leq t \leq y} \mathcal{S}_t$.

c) Adversarial model: The adversarial setting we consider throughout is that of a persistent, passive adversary, able to observe all communication between client and server (as opposed to a snapshot or active adversary). The adversary's goal is to reconstruct some information about the plaintext underlying the database hosted by the server, *i.e.* infer information about the client's data. Such a model captures threats such as an adversary having compromised the server for a sufficient length of time, a man-in-the-middle attacker intercepting communications between client and server, and an honest-but-curious server. From this last perspective, the security model we consider directly expresses the privacy of the user's data with respect to the server – indeed, all of our attacks could be mounted by the server itself.

Specifically, the view of the adversary is limited to some scheme-dependent *leakage* induced by each range query. We consider the following two types of leakage.

d) Access pattern leakage: If, in response to every range query, an adversary is able to observe the set of matching record identifiers $\mathcal{M}$, following [9], we call this *access pattern* leakage. As discussed in [9], access pattern leakage can stem from the actual memory access pattern of the server. This is a rather generic type of leakage: to the best of our knowledge, all known efficient schemes supporting range queries leak the access pattern.

e) Rank information leakage: The *rank* of a value $z \in \mathcal{X}$ is the number of records having a value less than or equal to z, *i.e.* $\mathsf{rank}(z) := |\mathcal{S}_{[1,z]}|$. It can also be interpreted as the highest position a record in $\mathcal{S}_z$ could have in a list of records sorted by value. The rank information leakage for a range query $[x, y]$ is the values $a := \mathsf{rank}(x - 1)$ and $b := \mathsf{rank}(y)$, where $a + 1$ and b can be interpreted as the lowest and highest positions (inclusive) of matching records in the aforementioned sorted list. The number of records returned by a query on range $[x, y]$ is $\sum_{x \leq t \leq y} |\mathcal{S}_t| = b - a$.

It is natural to consider such leakage in schemes supporting range queries, and our attacks directly apply to, among others, Lewi and Wu's ORE scheme, Arx-RANGE, and Kerschbaum's frequency-hiding OPE scheme. More applications are discussed in Section I-C.

Whenever we consider intervals of *values* in $[1, N]$, we shall use the letters $[x, y]$; whenever we consider intervals of *ranks*, we shall use the letters $[a, b]$.

To summarise, if leakage is limited to the access pattern, then a query on range $[x, y]$ leaks $\mathcal{M} = \mathcal{S}_{[x,y]}$. If rank information also leaks, then the leakage is $(a, b, \mathcal{M})$ where $a = \mathsf{rank}(x-1)$ and $b = \mathsf{rank}(y)$. We often assimilate a set of queries $\mathcal{Q}$ with its leakage, hence writing it as $\{(a_k, b_k, \mathcal{M}_k)\}$ if rank information is available, or simply $\{\mathcal{M}_k\}$ if only the access pattern is available.

f) Density of the database: We say that the database is *dense* iff every value $x \in [1, N]$ occurs in at least one record. We assume that the database is dense, except in Section IV.

g) Miscellaneous mathematical notation: The notation $\log$ refers to the natural logarithm. We write H_n for the n-th harmonic number $H_n = \sum_{k=1}^{n} \frac{1}{k}$. Whenever f is a mapping $A \to B$, if $S \subseteq A$, $f(A)$ denotes the image set $\{f(a) : a \in A\}$.

B. Our Contributions

We analyse the security of encryption schemes supporting range queries against persistent, passive adversaries. The bulk of our work applies to any scheme leaking access patterns. We also consider the case where the leakage additionally contains rank information. We present three attacks: full reconstruction, approximate reconstruction, and reconstruction using auxiliary information. An overview of the characteristics of our first two attacks, juxtaposed with previous results, is given in Table I. Note that without rank leakage, full reconstruction is only up to reflection. In Table I, the exact upper bounds on the expected number of queries for the attacks in Sections II-B and II-C require $N \geq 27$ and $N \geq 26$ respectively.

a) Full reconstruction attack: We show that access pattern leakage is sufficient to allow *full reconstruction*, *i.e.* recovering the value of *all* records with complete accuracy (up to reflection), within an expected number of only $N \log N + O(N)$ queries, where N is the number of distinct values. This

TABLE I: Comparison of full and approximate reconstruction attacks exploiting access pattern leakage on encryption schemes allowing range queries, for a set of N distinct plaintext values, with uniformly distributed queries.

	Req'd leakage	Assumptions	Number of queries	
	Rank	Density	Sufficient	Necessary
Full reconstruction attack				
[9]	N	N	$O(N^4 \log N)$	$\Omega(N^4)$
[9]	N	Y	$O(N^2 \log N)$	-
Sec. II-B (Fig. 1)	Y	Y	$N(\log N + 2)$	$N \log(N)/2 - O(N)$
Sec. II-C (Fig. 2)	N	Y	$N(\log N + 3)$	$N \log(N)/2 - O(N)$
Approximate reconstruction attack (within ϵN)				
Sec. III-B (Fig. 6)	N	Y	$\frac{5}{4} N \log(1/\epsilon) + O(N)$	$N \log(1/\epsilon)/2 - O(N)$

directly improves on a $O(N^2 \log N)$ bound by Kellaris *et al.* in the same setting [9]. Furthermore this can be achieved efficiently and *data-optimally* (as soon as the available leakage is sufficient for full reconstruction for *any* algorithm, our algorithm succeeds). Concretely, a bound on the expected number of queries is $N(\log(N) + 3)$ for $N \geq 26$.

In the full version of this paper, we also (i) present an algorithm for full reconstruction when rank leakage is also available, offering a slightly better bound of $N(\log(N) + 2)$ queries for $N \geq 27$, (ii) prove that our algorithms with and without rank leakage are data-optimal, (iii) present faster reconstruction algorithms with and without rank leakage, and (iv) prove a lower bound of $N \log(N)/2 - O(N)$ on the expected number of queries required for full reconstruction with or without rank leakage for any algorithm using access pattern leakage.

Our full reconstruction attacks require only two mild extra assumptions for their success:

1) every $x \in \mathcal{X}$ appears in at least one record (so the sets $\mathcal{S}_x$ are non-empty), also referred to as "density" in [2], [9];
2) the range queries $[x, y]$ are generated uniformly at random.

Assuming that range queries are distributed uniformly at random makes our work directly comparable to Kellaris *et al.*'s [9]. Further, since our full reconstruction algorithms are data-optimal, they consume the minimum possible amount of data for any query distribution. We show that very similar bounds would be obtained for the distribution where left end points are uniformly random; more generally, we conjecture that similar bounds would be achieved for a wide range of "non-pathological" distributions. (This is in contrast to *e.g.* Kellaris *et al.*'s attack in the non-dense case, which relies on a statistical inference approach that directly exploits the expected distribution of range queries.)

As a concrete example, if the values x represent age in years, then with $N = 125$, our attack reduces by a factor of about 100 the expected number of queries that need to be observed for full reconstruction, as compared to [9]. Specifically, the expected number of queries required for their attack is $\frac{N(N+1)}{2} H_{N(N+1)/2} \approx 75,196$, while our attack requires at most $(N+1)(\log(N)+1) + 8\sqrt{N} + 6 \approx 830$ queries, or, if rank leakage is available, $N(\log(N)+1) + 4.4\sqrt{N} + 4 \approx 782$ queries. Thus the gap is significant even for relatively low values of N.

b) Approximate reconstruction attack: In some situations, exactly recovering the value x associated with each record is not necessary, and approximately learning these values may be sufficient. Under the same assumptions as our first attack, we show that $O(N)$ queries suffice for *approximate* reconstruction with only access pattern leakage. More precisely, within an expected number of $\frac{5}{4} N \log(1/\epsilon) + O(N)$ queries, the attacker can reconstruct the values x associated with every record with a maximum error of ϵN, for any arbitrary precision ϵ. Note that the data requirement only grows logarithmically with the desired precision ϵ. An exact upper bound is $(1+\epsilon/4)N \log(1/\epsilon) + (2+4\sqrt{\epsilon})N + 4/\epsilon$ for $N \geq 40$.

Going back to the example where $N = 125$, the approximate reconstruction attack starts to overtake the full reconstruction attack without rank leakage for $\epsilon \approx 0.05$, *i.e.* when reconstructing records up to a precision of 5%. (So an attacker's guesses would never be wrong by more than 2.5% of N.)

In the full version of this paper, we also (i) present faster approximate reconstruction algorithms with and without rank information, and (ii) prove a lower bound $N \log(1/\epsilon)/2 - O(N)$ on the expected number

of queries before *any* algorithm can achieve approximate reconstruction with precision ϵN.

c) Inference attack using auxiliary distribution: Recent research has shown that an attacker equipped with an auxiliary (or reference) distribution for the plaintext can wreak havoc against deterministic encryption schemes in a snapshot attack model by using basic frequency analysis [2]. The assumption that an auxiliary distribution is available is a mild one in practice; it is often the case that the type of data in a given column in an encrypted database is known or can be inferred from the diversity of its values [2], and a rough and ready auxiliary distribution can be obtained from public statistics.

In our third attack, we show how an auxiliary distribution can be combined with rank and access pattern leakage to perform even better reconstruction attacks, with even fewer queries. Our attack essentially converts rank leakage from each query to information about the position of individual records in a sorted list of all records, then maps it back to a small set of possible values using the inverse CDF of the auxiliary distribution.

Because this third attack is not amenable to a rigorous analysis, we resort to an empirical evaluation. In particular, we study the relationship between number of random range queries available and the accuracy of reconstruction of "age in years" data, extracted from a real medical dataset. In our experiments, observing only 50 queries was sufficient to reconstruct 95% of one hospital's records to within 10 years, 55% of its records to within 5 years, and 35% of its records to within 3 years.

C. Applications

To the best of our knowledge, all practical encryption schemes supporting range queries leak access pattern information, making our generic attacks requiring only access pattern leakage widely applicable. However, it should be noted that schemes such as [10], [11] that rely on order-preserving or order-revealing encryption (OPE/ORE) are already vulnerable to powerful statistical inference attacks by snapshot adversaries, such as [2].[1] As a consequence, our attacks are more relevant to second-generation schemes that attempt to reduce or eliminate leakage against snapshot adversaries, and in the remainder we focus on such schemes. These schemes can be divided into two groups: schemes that leak access pattern but not rank information, and those that also leak rank information. The first group includes Blind Seer by Pappas *et al.* [12], as well as a scheme by Faber *et al.* [13]; they also include approaches based on Range Predicate Encryption, such as [14]. These schemes are vulnerable to our full and approximate reconstruction attacks.

The second group includes a variety of recent schemes, such as Lewi and Wu's ORE[2] construction [15] (used in EncKV [16]), Arx [17] (which mentions ongoing collaboration with health data cloud provider Chino and European medical project UNCAP to deploy their scheme), FH-OPE [18], and Cipherbase [19]. While the leakage of these schemes is expressed in different ways, it includes rank information. As a result, these schemes are vulnerable to all three of our attacks. In addition to the aforementioned schemes, the recent POPE scheme by Roche *et al.* [20], and a scheme with similar leakage by Hahn and Kerschbaum [21], should also be mentioned: although they mainly target scenarios where the database is not dense, our third attack still applies.

For all of the aforementioned schemes except FH-OPE, the only attacks we are aware of beside ours is that of Kellaris *et al.* [9], which mainly focuses on the non-dense setting, and requires a significantly larger number of queries. In the non-dense case, the approach of Kellaris *et al.* is to use statistical inference based on the uniformity of queries; in the dense case, they collect all $O(N^2)$ possible queries, resulting in a simple sorting step.

In the particular case of Kerschbaum's FH-OPE scheme [18], an attack was already mounted by Grubbs *e*t al. [7]. Their attack is in the snapshot setting and thus did not use query leakage. Although our attacks require query leakage, they are more generic, as the scheme is only required to leak access patterns (and in some cases rank information). Another difference is that the attack by Grubbs *et al.* requires knowledge of an auxiliary distribution, whereas our first two attacks (full and approximate reconstruction) do not.

D. Full version of the article.

Due to space constraints, some proofs are not included in this version of the paper. In their place, some intuition explaining the query complexity of our algorithms is provided in Sections I-B, II-A, and III-A. Thus the present version is intended to be self-contained. The full version is available at: https://eprint.iacr.org/2017/701.pdf

[1] Any ORE (*a fortiori* OPE) scheme is trivially broken whenever the database is dense, as ordering all ciphertexts allows them to be directly matched with plaintext values.

[2] The Lewi-Wu construction is not strictly an ORE scheme; in particular, it is fully secure against a basic snapshot adversary.

II. Full Reconstruction Attack

The purpose of a full reconstruction attack is to determine the sets $\mathcal{S}_x$ for all $x \in \mathcal{X}$. In other words, a full reconstruction attacks recovers the value associated with every record.

In Section II-A, we start with a simple introductory example. In Section II-B, we focus on the case where range queries leak *rank information*, as defined in Section I-A. In Section II-C, we show how the attack can be extended to work with only access pattern leakage. Throughout this section, we assume that the database is dense, *i.e.* every value appears in at least one record (cf. Section I-A).

In all cases, we prove that $N \log N + O(N)$ queries are enough for full reconstruction (and provide concrete bounds). Throughout this section, we give algorithms that favour simplicity over efficiency. As mentioned in Section I-B, the full version of this paper contains much more efficient algorithms and a lower bound on the number of queries required.

A. Simple Attack with Simplified Query Distribution

We begin with an example that provides some intuition for why $N \log N + O(N)$ queries are enough for full reconstruction with rank information leakage. In this section only, we assume that the left end points x of range queries $[x, y]$ are uniformly random (so left end points are biased toward lower values).

Recall that for a query on range $[x_k, y_k]$, access pattern leakage and rank information leakage mean that an adversary observes $(a_k, b_k, \mathcal{M}_k)$. Our algorithm will succeed iff every value $x \in [1, N]$ appears as the left end point of some query. Because we assumed uniformly random left end points, the expected number of queries until this condition is satisfied is the solution to the standard coupon collector's problem, satisfying:

$$N \cdot H_N \leq N \left(1 + \int_{x=1}^{N} \tfrac{1}{x}\right) = N(1 + \log N)$$

where H_N is the N-th harmonic number.

Suppose every left end point does occur in at least one query. By relabelling, we may assume that the a_k's are in increasing order. The adversary selects any N-subset of indices k_i, where $1 \leq i \leq N$, such that the a_{k_i} are distinct. It then computes:

$$\mathcal{E}_i := \mathcal{M}_{k_i} \setminus \left(\bigcup_{\ell \geq k_{i+1}} \mathcal{M}_\ell\right), \quad 1 \leq i \leq N$$

where we define the union $\cup_{\ell \geq k_{i+1}} \mathcal{M}_\ell$ for $i = N$ to be the empty set. The set $\mathcal{E}_j$ is precisely the set of identifiers for records containing value $j \in \mathcal{X}$; that is, $\mathcal{E}_j = \mathcal{S}_j$. The idea is that $\mathcal{M}_{k_j}$ certainly contains all such records, but possibly also other records. Performing a set subtraction with the sets of record identifiers $\mathcal{M}_\ell$ for $\ell \geq k_{j+1}$ removes the additional records.

This concludes the attack: within an expected number of queries $N \cdot H_N \leq N \log(N) + N$, all left end points are collected, and the values of all records can be recovered. Despite its simplicity, this example captures a large part of the intuition behind our attacks in this section and the next. In particular, the data complexity bounds in the following sections are ultimately based on reductions to (increasingly intricate) variants of the coupon collector's problem.

B. Full Reconstruction with Rank Information

Before moving on to full reconstruction attacks *without* rank information, we first present a more robust algorithm for full reconstruction in the presence of rank information leakage. The general approach of this new algorithm, and in particular the notion of *partition of records*, will form the basis of later algorithms and their analysis. This new algorithm is *data-optimal*: if it fails, then full reconstruction is impossible given the input queries (for any algorithm). As a consequence, this algorithm requires the minimum possible number of queries for *any* distribution of queries.

In particular, it must perform at least as well as the algorithm from the previous section; it must succeed within an expected number of queries $\leq N(\log(N)+1)$ in the case of uniform left end points. We also show that it succeeds with an expected number of queries $\leq N(\log(N) + 2)$ for uniform queries, contrary to the algorithm from the previous section, which requires $\Omega(N^2)$ queries in that setting.

a) The algorithm: The idea of the algorithm is as follows. As before, fix a set of Q queries on ranges $[x_k, y_k]$, $k \leq Q$, with the corresponding leakage observed by the adversary $\{(a_k, b_k, \mathcal{M}_k) : k \leq Q\}$.

Define the following equivalence relation on records. (Recall that records are assimilated with their IDs.) Two records are equivalent iff for each of the Q queries, either both records are matched by the query, or neither is. It is clear – at least intuitively – that two records can be meaningfully distinguished iff they are in distinct equivalence classes. The equivalence class of a record r can be easily computed as:

$$P(r) := \left(\bigcap_{\{k : r \in \mathcal{M}_k\}} \mathcal{M}_k\right) \setminus \left(\bigcup_{\{k : r \notin \mathcal{M}_k\}} \mathcal{M}_k\right),$$

which contains exactly the records whose value is in:

$$V(r) := \left(\bigcap_{\{k : r \in \mathcal{M}_k\}} [x_k, y_k]\right) \setminus \left(\bigcup_{\{k : r \notin \mathcal{M}_k\}} [x_k, y_k]\right).$$

Note that neither $P(r)$ nor $V(r)$ can be empty, as they must contain, at least, r and $\mathsf{val}(r)$ respectively.

Note also that $V(r)$ is unknown to the adversary at this point in the attack.

The set of $P(r)$'s (resp. $V(r)$'s) forms a partition of the set of records (resp. values), and each $P(r)$ contains exactly those records whose value lies in $V(r)$. Due to this bijection, the *partition of records* $\mathcal{P} = \{P(r) : r \in \mathcal{R}\}$ cannot contain more than N elements. On the other hand, since records in the same class cannot be meaningfully distinguished, full reconstruction requires $|\mathcal{P}| = N$.

The crucial point is that, conversely, $|\mathcal{P}| = N$ is enough to enable full reconstruction: if $|\mathcal{P}| = N$, then each $V(r)$ must contain a single value, and $P(r)$ constitutes the set of records having that value. Thus each element of $\mathcal{P}$ corresponds to a single value: $\mathcal{P} = \{\mathcal{S}_x : x \in \mathcal{X}\}$. Hence, in order to find the value of every record, it only remains to sort the elements of $\mathcal{P}$ by value, which can be done using rank as a kind of proxy for value: the value of $P(r)$ has rank:

$$m(r) := \max\Big(\big(\textstyle\bigcap_{\{k:r\in\mathcal{M}_k\}}[a_k, b_k]\big)\setminus \big(\textstyle\bigcup_{\{k:r\notin\mathcal{M}_k\}}[a_k, b_k]\big)\Big).$$

It suffices to sort all values $m(r)$ to correctly sort the corresponding classes $P(r)$ by value. Our algorithm does exactly this. Pseudo-code is provided in Fig. 1. In this algorithm, we immediately compute the map m, and check that $|\{m(r) : r \in \mathcal{R}\}| = N$. Computing P and/or V first is not necessary, but is only used above as an aid to understanding the algorithm.

FULL-RECONSTRUCTION($\mathcal{Q}$):
Input: set of queries $\mathcal{Q} = \{(a_k, b_k, \mathcal{M}_k)\}$.
Output: $\perp$, or map $\mathrm{Val} : \mathcal{R} \to \mathcal{X}$ s.t. $\forall r\ \mathrm{Val}(r) = \mathsf{val}(r)$.

1: $m \leftarrow$ empty map, $\mathrm{Val} \leftarrow$ empty map
2: **for all** $r \in \mathcal{R}$ **do** ▷ *Partitioning* step
3: $\quad m(r) \leftarrow \max\Big(\big(\bigcap_{\{k:r\in\mathcal{M}_k\}}[a_k, b_k]\big)\setminus \big(\bigcup_{\{k:r\notin\mathcal{M}_k\}}[a_k, b_k]\big)\Big)$
4: **end for**
5: $M \leftarrow \{m(r) : r \in \mathcal{R}\}$
6: **if** $|M| < N$ **then**
7: $\quad$ **return** $\perp$
8: **end if**
9: $M \leftarrow \mathrm{sort}(M, \text{increasing})$ ▷ *Sorting* step
10: **for all** $r \in \mathcal{R}$ **do**
11: $\quad \mathrm{Val}(r) \leftarrow$ index of $m(r)$ in M
12: **end for**
13: **return** Val

Fig. 1: Full reconstruction attack with rank leakage.

b) Analysis of the algorithm: We make several claims about the FULL-RECONSTRUCTION algorithm in Fig. 1. First, it is data-optimal: if it fails, then no algorithm can achieve full reconstruction with probability 1 given the same queries as input. The rationale we have provided for the algorithm already gives strong evidence that this is the case, however a more formal treatment is provided in the full version of this paper. Second, we observe that the algorithm still achieves full reconstruction with an expected number of queries upper-bounded by $N(\log(N) + 1)$ in the setting where left end points are uniformly random. This is a direct consequence of data optimality and the observations in Section II-A.

Third, and most importantly, we claim that the algorithm in Fig. 1 achieves a similar data complexity in the case where range queries are uniformly random (unlike the algorithm from Section II-A). More precisely, we show that for $N \geq 27$, the expected number of queries is upper-bounded by $N(\log(N) + 2)$. For simplicity, we assume in the following statement that N is a multiple of 4, but as the success probability of the algorithm and the expected number of queries before it succeeds respectively increase and decrease with N, we can always round N to the next multiple of 4 if necessary.

Proposition 1. *Assume N is a multiple of 4. Assume that the database is dense, and range queries are drawn uniformly at random. Then the probability of success of the algorithm in Fig. 1 after Q queries is lower-bounded by:*

$$1 - 2e^{-Q/(2N+2)} - Ne^{-Q/N}.$$

Moreover, the expected number of queries before the algorithm succeeds is upper-bounded by:

$$N\log(N) + O(N).$$

Concretely, for $N \geq 27$, it is upper-bounded by $N(\log(N) + 1) + 4.4\sqrt{N} + 4 \leq N(\log(N) + 2)$.

A proof of Proposition 1 is given in the full version of this paper. As a concrete example, setting $N = 100$ yields an expected number of queries upper-bounded by 609 to achieve full reconstruction of a dense database, regardless of the number of records.

c) Matching lower bound: In the full version of this paper, we show that the necessary expected number of queries for *any* algorithm to achieve full reconstruction is $\frac{1}{2}N\log(N) - O(N)$.

d) Complexity: The algorithm in Fig. 1 is stated in a way that closely follows the rationale exposed earlier in this section. However it is quite inefficient: line 3 in particular results in a large number

of redundant computations, each involving multiple intersections and unions of potentially large sets. While the number of values N may be small, and we already know that $Q \approx N \log N$ queries are enough for full reconstruction, the number of records R can be quite large. In the full version of this paper, we show a very efficient approach to computing the partition of records, without explicitly computing any set intersections or unions. This results in a time complexity $O(Q(N+R))$, with little overhead over simply reading all queries.

C. Full Reconstruction without Rank Information

We now extend the attack to the case where only the *access pattern* is leaked. Kellaris *et al.* have already shown that $O(N^2 \log N)$ queries are enough to achieve full reconstruction for dense databases [9]. However, we show that $N \log(N) + O(N)$ queries are still enough with the same assumptions. A small caveat is that in the absence of rank information, full reconstruction can be achieved only up to *reflection*, swapping value i with value $N+1-i$. (Of course, this fact also applies to the attacks in [9].)

a) The algorithm: In response to a query on range $[x_k, y_k]$, the adversary now sees only the access pattern leakage $\mathcal{M}_k = \mathcal{S}_{[x_k,y_k]}$. The attack can be divided into a *partitioning* step and a *sorting* step. In the remainder we identify a query with the corresponding set of matching records. Although the algorithm can fail at a few points, it is data-optimal. Pseudo-code is provided in Fig. 2.

First, compute the partition of records $\mathcal{P}$ as in Section II-B:

$$P(r) := \left(\bigcap_{\{k:r\in\mathcal{M}_k\}} \mathcal{M}_k\right) \setminus \left(\bigcup_{\{k:r\notin\mathcal{M}_k\}} \mathcal{M}_k\right)$$
$$\mathcal{P} := \{P(r) : r \in \mathcal{R}\}.$$

Recall that full reconstruction requires $|\mathcal{P}| = N$, and that if $|\mathcal{P}| = N$, every element of $\mathcal{P}$ is the set $\mathcal{S}_x$ for some $x \in \mathcal{X} = [1, N]$. We will call the elements of $\mathcal{P}$ "points". Each such point p corresponds to a distinct value in $\mathcal{X}$ (*viz.* the singleton $\mathsf{val}(p)$).

To achieve full reconstruction, it remains to assign the correct value to each point, which is equivalent to sorting $\mathcal{P}$ according to the value of each point – this is the *sorting* step. First, we set out to find an end point, *i.e.* a point with value 1 or N (which are indistinguishable due to the reflection symmetry mentioned earlier). To this end, we form a maximal union S of queries that does not cover the full set of records, but does cover an interval of values. Build S as follows: start with a query not covering the full set of records, and extend it with overlapping queries until no longer possible without covering the full set of records $\mathcal{R}$. If $\mathcal{R} \setminus S$ is reduced to a single point, then it must be an end point: indeed in that case S must cover an interval of $N-1$ values within $[1, N]$, so the remaining value can only be 1 or N. If $\mathcal{R} \setminus S$ is not a single point, the algorithm fails.

Once an end point is identified, we assume that it corresponds to value 1, which must be true up to reflection. We have thus determined the initial segment of points $I(1)$ containing a single point. We then propagate this information by building each initial segment of points $I(i) = \mathcal{S}_{[1,i]}$ in turn, by induction. Given an initial segment $I(i)$, this amounts to finding the *next* point, which must be $\mathcal{S}_{i+1}$. To do this, we build the intersection T of all queries overlapping $I(i)$ and containing at least one point outside $I(i)$. We then subtract from T all queries overlapping T from the right (*i.e.* overlapping T and not contained in $I(i) \cup T$) so long as T remains non-empty. If the resulting T contains a single point, it must be $\mathcal{S}_{i+1}$; we let $I(i+1) = I(i) \cup T$ and continue. Otherwise the algorithm fails.

Once we have built the last (proper) initial segment $I(N-1)$, we have successfully sorted all points, and hence determined the value of all records. Pseudo-code is given in Fig. 2.

b) Analysis of the algorithm: Once again, the algorithm is data-optimal: if it fails on some input, then full reconstruction with probability 1 is impossible for any algorithm given that input (a proof is provided in the full version of this article). Furthermore, the expected number of queries necessary for the algorithm to succeed in achieving full reconstruction, given uniformly random ranges queries, is $N \log(N) + O(N)$. More precisely, the following holds.

Proposition 2. *Assume N is a multiple of 4. Assume that the database is dense, and range queries are drawn uniformly at random. Then the probability of success of the algorithm in Fig. 2 after Q queries is lower-bounded by:*

$$1 - 4e^{-Q/(2N+2)} - Ne^{-Q/(N+1)}.$$

Moreover, the expected number of queries before the algorithm succeeds is upper-bounded by:

$$N \log(N) + O(N).$$

Concretely, for $N \geq 26$, it is upper-bounded by $(N+1)(\log(N)+1) + 8\sqrt{N} + 6 \leq N(\log(N)+3)$.

A proof of Proposition 2 is provided in the full version of this paper. Since it is one of our main results, we provide here some intuition regarding

FULL-RECONSTRUCTION-AP($\mathcal{Q}$):
Input: set of queries $\mathcal{Q} = \{\mathcal{M}_k\}$.
Output: $\perp$, or map $\mathrm{Val} : \mathcal{R} \to \mathcal{X}$ s.t. $\forall r\, \mathrm{Val}(r) = \mathsf{val}(r)$ or $\forall r\, \mathrm{Val}(r) = N{+}1{-}\mathsf{val}(r)$.

1: $P, I, \mathrm{Val} \leftarrow$ empty maps
2: **for all** $r \in \mathcal{R}$ **do** ▷ *Partitioning* step
3: $\quad P(r) \leftarrow \left(\bigcap_{\{k : r \in \mathcal{M}_k\}} \mathcal{M}_k\right) \setminus \left(\bigcup_{\{k : r \notin \mathcal{M}_k\}} \mathcal{M}_k\right)$
4: **end for**
5: $\mathcal{P} \leftarrow \{P(r) : r \in \mathcal{R}\}$
6: **if** $|\mathcal{P}| < N$ **then**
7: $\quad$ **return** $\perp$
8: **end if**
9: Pick $S \in \mathcal{Q}$ s.t. $|S| < R$ ▷ *Sorting* step
10: **while** $\exists q \in \mathcal{Q}$ s.t. $q \cap S \neq \varnothing, q \setminus S \neq \varnothing, q \cup S \neq \mathcal{R}$ **do** ▷ Searching for end point
11: $\quad S \leftarrow S \cup q$
12: **end while**
13: **if** $\mathcal{R} \setminus S \notin \mathcal{P}$ **then** ▷ Ensuring $|\mathsf{val}(\mathcal{R} \setminus S)| = 1$
14: $\quad$ **return** $\perp$
15: **end if**
16: $I(1) \leftarrow \mathcal{R} \setminus S$ ▷ Found end point
17: **for all** $i \in [1, N-1]$ **do** ▷ Seeking next point
18: $\quad \mathcal{Q}' \leftarrow \{q \in \mathcal{Q} : q \cap I(i) \neq \varnothing, q \setminus I(i) \neq \varnothing\}$
19: $\quad T \leftarrow \left\{\bigcap_{q \in \mathcal{Q}'} q\right\} \setminus I(i)$
20: $\quad$ **while** $\exists q \in \mathcal{Q}$ s.t. $q \cap T \neq \varnothing, q \setminus (T \cup I(i)) \neq \varnothing, T \setminus q \neq \varnothing$ **do**
21: $\quad\quad T \leftarrow T \setminus \{q\}$
22: $\quad$ **end while**
23: $\quad$ **if** $T \notin \mathcal{P}$ **then** ▷ Ensuring $|\mathsf{val}(T)| = 1$
24: $\quad\quad$ **return** $\perp$
25: $\quad$ **end if**
26: $\quad I(i+1) \leftarrow I(i) \cup T$ ▷ Found next point
27: **end for**
28: **for all** $r \in \mathcal{R}$ **do** ▷ Success
29: $\quad \mathrm{Val}(r) \leftarrow \min(\{i : r \in I(i)\})$
30: **end for**
31: **return** Val

Fig. 2: Full reconstruction from access pattern.

why $N \log(N) + O(N)$ queries is still enough, even without rank information. For simplicity, we go back to the setting of the introductory example from Section II-A, and assume that the left end points of range queries are uniformly random. The reasoning from Section II-A already shows that if all N possible left end points appear in some range query, then $|\mathcal{P}| = N$. Thus we have N *points* (elements of $\mathcal{P}$), and the partitioning step succeeds. It remains to sort points.

For the sorting step to succeed, the previous condition is not enough. (Consider the case where all queries are singleton range queries $[x, x]$.) If we strengthen the previous condition very mildly by requiring that (1) all left end points $< N$ appear in some range query on an interval of length at least 2 and (2) there exists a range query of the form $[x, N-1]$ for some $x < N-1$, then with these two conditions the algorithm from Fig. 2 cannot fail at lines 14 or 24, and hence the sorting step succeeds.[3]

In the proof of Proposition 2 (in the full version), the actual requirements are slightly more intricate because range queries are assumed to be uniformly distributed. However, the underlying idea is the same.

c) Complexity: Again, our algorithm was described in such a way to maximise legibility and ease analysis. In the full version, we show that the algorithm in Fig. 2 can be executed efficiently, with time complexity $O(Q(N^2 + R))$.

III. Approximate Reconstruction Attack

In the previous section, we showed that $N \log(N) + O(N)$ uniformly distributed range queries are sufficient to fully reconstruct the exact value in every record. In this section, we present an *approximate* reconstruction attack that seeks to determine the value $\mathsf{val}(r)$ of every record r, up to some precision $k = \epsilon N$. We limit our attention to the general setting where only access pattern leakage is available (a data-optimal variant when rank information is available is given in the full version).

More precisely, for every record r, an approximate reconstruction attack with precision k outputs an interval $[x, x+k]$ such that $\mathsf{val}(r) \in [x, x+k]$. A full reconstruction attack may be regarded as the special case where $k = 0$. In the remainder however, we shall assume $k > 0$ and, without loss of generality, that $k = \epsilon N$ is an integer. Note that our algorithm will either succeed, *i.e.* it correctly recovers all values up to precision ϵ as defined above; or it outputs $\perp$. (In other words, the algorithm can fail with $\perp$, but it cannot output an incorrect answer.)

The approximate reconstruction attack we discuss in this section succeeds within an expected number of queries upper-bounded by $\frac{5}{4} N \log(1/\epsilon) + O(N)$. In particular, it is $O(N)$ for any fixed ϵ, and only grows logarithmically with the precision ϵ.

A. Intuition for Approximate Reconstruction

We first provide some intuition as to why the complexity drops from $O(N \log N)$ to $O(N)$ when a margin of error is allowed in a reconstruction

[3]These conditions are sufficient, but not necessary – which is why the approximate reconstruction algorithm, contrary to the full reconstruction algorithm, is not data-optimal.

attack. Recovering the value of every record required $O(N \log N)$ queries because the problem essentially reduced to a coupon collector's problem on N items, as seen in Section II-A. Each coupon was an integer in the interval $[1, N]$. After $k < N$ distinct coupons have been drawn, drawing a new coupon requires $N/(N-k)$ tries on average, since each new draw has a probability $(N-k)/N$ of yielding a "new" coupon. Recall that the expected number of draws to gather all N coupons is:

$$\sum_{k=0}^{N-1} \frac{N}{N-k} = N \cdot \sum_{k=1}^{N} \frac{1}{k} = N \cdot H_N$$

where H_N is the N-th harmonic number.

The last few coupons are much more expensive to collect than the early ones, in terms of the expected number of draws required. In particular, if we only wish to recover $(1-\epsilon)N$ coupons for some $\epsilon > 0$, then the expected number of draws is:

$$\begin{aligned}\sum_{k=0}^{N-\epsilon N-1} \frac{N}{N-k} &= N \cdot \sum_{k=\epsilon N+1}^{N} \frac{1}{k} \\ &< N \cdot \int_{\epsilon N}^{N} \frac{1}{x} = N \cdot \log(\tfrac{1}{\epsilon})\end{aligned}$$

where ϵN is assumed to be an integer. Thus, the expected number of tries is $O(N)$, with the growth being proportional to $\log(1/\epsilon)$.

B. Approximate Reconstruction Attack

The details of the approximate reconstruction algorithm are provided in Appendix A, see in particular Fig. 6. In this section, we present a brief overview of the algorithm, and discuss its properties.

Compared to the full reconstruction algorithms from Section II, particular care must be taken around a few issues. The main one is that, because the partition of records $\mathcal{P}$ at the core of the previous algorithms no longer satisfies $|\mathcal{P}| = N$, there is no guarantee that its elements correspond to *intervals* of values (as opposed to arbitrary subsets). As a result we eschew this approach, and instead ensure at every step that the subsets of records we build correspond to intervals of values (*i.e.* for every subset S of records built in the course of the algorithm, $\mathsf{val}(S)$ is an interval).

The general idea of the algorithm is to carefully split the set of records into three subsets, called *left*, *middle* and *right* subsets, such that the values corresponding to each subset (*i.e.* the image of each subset by val) form three consecutive intervals of values. Then the algorithm collects queries Q that contain the middle subset and overlap the left one. Such queries are called *left coupons*. After substracting the middle and right subsets from left coupons, it can be observed that the resulting sets form a linear order for set inclusion. This induces an ordering of the value of records appearing in left coupons. Then the process is repeated for right coupons. The algorithm is described in detail in Appendix A.

a) Expected number of queries: Contrary to the full reconstruction algorithms from the previous section, the approximate reconstruction algorithm is not data-optimal. However, we do give a data-optimal variant in the case where rank information is available in the full version of the paper.

The main feature of the approximate reconstruction algorithm is expressed by the following proposition. As before, we assume for simplicity that N is a multiple of 4, and, as before, this has little impact.

Proposition 3. *Assume N is a multiple of 4, $\epsilon N/2$ is an integer, and $\epsilon < 3/4$. Assume that the database is dense, and range queries are drawn uniformly at random. Then the probability of success of the algorithm in Fig. 6 after Q queries is at least:*

$$1 - 4e^{-Q/(2N+2)} - 2e^{-Q\log(1+\epsilon/2)+\epsilon N/2\cdot(\log(1/\epsilon)+1.5)}.$$

Moreover, the expected number of queries before the algorithm succeeds is upper-bounded by:

$$\frac{5}{4} N \log(1/\epsilon) + O(N).$$

Concretely, for $N \geq 40$, it is upper-bounded by $(1+\epsilon/4)N\log(1/\epsilon) + (2+4\sqrt{\epsilon})N + 4/\epsilon$.

A proof of Proposition 3 is given in the full version of this paper (an intuition for the data complexity was also discussed in Section III-B). As a corollary of the expected number of queries given above, if we want to recover the value of all records within a constant *additive* error, *i.e.* $\epsilon = \Theta(1/N)$, then $O(N \log N)$ queries suffice: as one might expect, this matches the full reconstruction attack. Another observation is that for fixed ϵ, $O(N)$ queries suffice; and perhaps surprisingly, the expected number of queries only grows logarithmically with the desired precision.

The formula for the probability of success may appear difficult to parse. It may help to observe that the last term is dominant for small ϵ. Furthermore, if we approximate $\log(1+\epsilon/2)$ by $\epsilon/2$, and disregard the final "$+1.5$", then the last term would dictate a probability of failure upper-bounded by $2/e < 3/4$ for $Q = N\log(1/\epsilon)$ queries, and this upper bound would be divided by $e \approx 2.7$ for every additional $2/\epsilon$ queries. Once again if $\epsilon = \Theta(1/N)$ this matches the approximate behaviour of the full reconstruction attack.

b) Matching lower bound: In the full version of this paper, we show that the expected number of queries before *any* algorithm can achieve approximate reconstruction with precision ϵN is at least $\frac{1}{2}N\log(1/\epsilon) - O(N)$, where the constant in $O(N)$ is independent of ϵ.

c) Complexity: As was the case with the algorithms in Fig. 1 and 2, the algorithm in Fig. 6 was defined with legibility in mind, rather than efficiency. A more efficient variant is discussed in the full version, and achieves a time complexity $O(Q(N^2+R))$ with only access pattern leakage, and $O(Q(N+R))$ with rank information.

IV. Exploiting Auxiliary Information

The previous attacks do not make any assumptions about the distribution of values, but in a real-world scenario, this distribution may be quite predictable. For instance, the values might represent the age of patients in a hospital or salaries in a personnel database. Attacks on database encryption schemes based on statistical inference are commonplace; in this regard our previous attacks are atypical.

In this section, we propose a heuristic algorithm for performing reconstruction attacks against schemes that have access pattern leakage and rank leakage, in the setting where some auxiliary information about the values' distribution is available. Our attack is not amenable to rigorous analysis, unlike our previous two attacks, so we will rely on an empirical evaluation of its performance.

While we introduce a new assumption about the availability of pertinent auxiliary information, we also remove an assumption that applied to previous attacks: we no longer assume the data is dense. So, while query end points are still sampled uniformly at random from $\mathcal{X}$, not all of these values necessarily correspond to a record or records. As in the previous attacks, we assume that the adversary knows the set of record identifiers $\mathcal{R}$ and we treat this set as $[1, R]$ without loss of generality.

A. The Algorithm

As usual, we consider the scenario where a client is issuing uniformly distributed queries on ranges $[x_k, y_k] \subseteq \mathcal{X}$, for $k \leq Q$, and for each query, the adversary observes the leakage $(a_k, b_k, \mathcal{M}_k)$.

a) The algorithm: Let us define *position* as follows: the position $\mathsf{pos}(r)$ of a record r is its index (counting from 1) in the sorted list L of records (sorted by value). From this perspective, the rank of a value x is the position of the last record with value x in L.

The algorithm proceeds in two steps. In Step 1, for each record r, we compute an interval $[a, b]$ such that $\mathsf{pos}(r) \in [a, b]$, or, equivalently, $\mathsf{rank}(\mathsf{val}(r) - 1) + 1 \geq a$ and $\mathsf{rank}(\mathsf{val}(r)) \leq b$. Essentially we are performing an approximate reconstruction attack, except that, instead of outputting an interval of $\mathcal{X}$ containing $\mathsf{val}(r)$, we output an interval of $[1, R]$ containing $\mathsf{pos}(r)$. In Step 2 of the algorithm, we will use the auxiliary information about the a priori distribution of values $\mathcal{X}$ to map each position to the most likely associated value. Compared to the algorithms we have encountered in the previous sections, such an approach can output a guess for the value of a record even when only few range queries are available.

We now explain each step of the algorithm in detail.

Step 1. Recall from Section II-B that the *partition of records* $\mathcal{P} = \{P(r) : r \in \mathcal{R}\}$ is defined by:

$$P(r) := \left(\bigcap_{\{k:r\in\mathcal{M}_k\}} \mathcal{M}_k\right) \setminus \left(\bigcup_{\{k:r\notin\mathcal{M}_k\}} \mathcal{M}_k\right),$$

In the same way, define the *partition of positions* $\{S(r) : r \in \mathcal{R}\}$ by:

$$S(r) := \left(\bigcap_{\{k:r\in\mathcal{M}_k\}} [a_k+1, b_k]\right) \setminus \left(\bigcup_{\{k:r\notin\mathcal{M}_k\}} [a_k+1, b_k]\right).$$

Recall that $[a_k + 1, b_k]$ can be interpreted as the positions of records in $\mathcal{M}_k$: that is, $[a_k + 1, b_k] = \mathsf{pos}(\mathcal{M}_k)$. By looking at the definitions of $P(r)$ and $S(r)$, it is apparent that as a direct result $S(r) = \mathsf{pos}(P(r))$. Thus, $S(r)$ contains precisely the set of positions that record r can occupy in a list of records sorted by value. As result, computing the minimal and maximal possible position of a record r is straightforward: they are precisely $\min(S(r))$ and $\max(S(r))$. This concludes Step 1 of the algorithm. Pseudo-code is provided in Fig. 3.

Step 2. At this point, for each record r, we have computed a possible range $[a + 1, b]$ for its position $\mathsf{pos}(r)$. Now, we would like to output an estimate for its value $\mathsf{val}(r)$. First, from the knowledge of a, b, we compute an (approximation of) the distribution of $\mathsf{val}(r)$; second, using this distribution, we output an estimate for $\mathsf{val}(r)$. Let us call these two steps (2a) and (2b). In the remainder we fix a record r and the corresponding interval of positions $[a + 1, b]$ output by the algorithm in Fig. 3.

Step 2a. By construction, a and b are always the ranks of some values. We can attempt to determine which values using the auxiliary distribution D by evaluating $\mathsf{rank}(x)$ for all values x based on D, and

Input: set of queries $\mathcal{Q} = \{(a_k, b_k, \mathcal{M}_k)\}$.
Output: maps $\mathrm{minPos}, \mathrm{maxPos} : \mathcal{R} \to [1, R]$ s.t. $\forall r, \mathsf{pos}(r) \in [\mathrm{minPos}(r), \mathrm{maxPos}(r)]$.

$S \leftarrow$ empty map
for all $r \in \mathcal{R}$ **do**
  $S(r) \leftarrow \left(\bigcap_{\{k: r \in \mathcal{M}_k\}} [a_k + 1, b_k]\right) \setminus \left(\bigcup_{\{k: r \notin \mathcal{M}_k\}} [a_k + 1, b_k]\right)$
  $\mathrm{minPos}(r) \leftarrow \min(S(r))$
  $\mathrm{maxPos}(r) \leftarrow \max(S(r))$
end for
return $\mathrm{minPos}, \mathrm{maxPos}$

Fig. 3: Computing minimal intervals containing the position of each record.

matching a (resp. b) with the closest result. This yields a simple model for the distribution of $\mathsf{val}(r)$: if $a = \mathsf{rank}(x-1)$ and $b = \mathsf{rank}(y)$, then we can model the distribution of $\mathsf{val}(r)$ as D restricted to $[x, y]$.

The auxiliary distribution D tells us that each value $z \in [1, N]$ occurs within records with some probability p_z, with $\sum_{z=1}^{N} p_z = 1$. Let $q_z = \sum_{i=1}^{z} p_i$ denote the cumulative distribution.

For $z \in [1, N]$, $\mathsf{rank}(z)$ can be seen as a random variable whose distribution is determined by D and follows a binomial distribution:

$$\Pr\left[\mathsf{rank}(z) = a\right] = \binom{R}{a} q_z^a (1 - q_z)^{R-a} \quad (1)$$

with expected value $\mathbb{E}(\mathsf{rank}(z)) = R q_z$.

Given $a \in [1, R]$ and knowing that $a = \mathsf{rank}(z)$ for some z, finding the most likely value of z amounts to choosing z so as to maximise $\Pr\left[\mathsf{rank}(z) = a\right]$. Fixing a, observe that the function $q \mapsto q^a (1-q)^{R-a}$ is concave and reaches its maximum for $q = a/R$. Using (1), it follows that the optimal choice of z is either z or $z + 1$, for z such that a/R lies within $[q_z, q_{z+1}]$. The optimal choice between z and $z + 1$ can be determined by computing (1) above and picking the higher of the two values.

In this way, we can compute the most likely values $x, y \in \mathcal{X}$ such that $a = \mathsf{rank}(x-1)$ and $b = \mathsf{rank}(y)$. Assuming that $\mathsf{rank}^{-1}(a)$ and $\mathsf{rank}^{-1}(b)$ are in fact $x - 1$ and y, then $\mathsf{val}(r) \in [x, y]$, and we can model the distribution of $\mathsf{val}(r)$ as D restricted to $[x, y]$, *i.e.* each value $t \in [x, y]$ occurs with probability $\left(\sum_{i=x}^{y} p_i\right)^{-1} p_t$. This concludes the first step.

Step 2b. We now have (an approximation of) the distribution of $\mathsf{val}(r)$. We wish to output an estimate of $\mathsf{val}(r)$. A simple choice is to output its expected value. For the distribution proposed in the previous step, this means outputting as a guess for $\mathsf{val}(r)$ the expectation of a value drawn according to D conditioned on being within $[x, y]$, namely:

$$\left(\sum_{i=x}^{y} p_i\right)^{-1} \left(\sum_{i=x}^{y} i \cdot p_i\right). \quad (2)$$

This concludes the attack. We discuss several variants of Step 2 in Appendix B. In our experiments, the simple approach presented above already proved to be effective.

b) Complexity: Step 2 requires a negligible amount of computation. As in previous sections, the algorithm in Step 1 can be sped up considerably using techniques we introduce in the full version of the paper, resulting in $O(Q(N + R))$ operations, little more time than it takes to read all queries.

B. Experimental Results

Since this attack is less amenable to rigorous analysis, we carry out an empirical evaluation by simulating queries on age data in years, from 0–124. The data we use are from the 2009 Nationwide Inpatient Sample (NIS), from the Healthcare Cost and Utilization Project (HCUP), run by the Agency for Healthcare Research and Quality in the United States [22]. (We expect results for any other year would be similar.) It includes data at the hospital level – e.g. number of discharges, number of beds – and at the patient discharge level – e.g. demographics, diagnosis, procedure, payer.

Remark. The NIS is processed in a way to protect the privacy of individual patients, physicians, and hospitals. Our experiments were on the AGE attribute only, and we did not attempt to deanonymise any of the data. All authors underwent the HCUP Data Use Agreement training and submitted signed Data Use Agreements to the HCUP Central Distributor.

We extract the age data (in years) from the patient discharge records of the largest 200 hospitals in the 2009 sample. Each of these largest hospitals has between 13,000 and 122,000 records, approximately, for a total of about 4.9 million records. We simulate range queries on individual hospitals' data and attack the query leakage; the auxiliary distribution we use is obtained by averaging over all 200 largest hospitals' records. Query end points are sampled independently and uniformly at random from $[0, 124]$, the range of valid age values according to the NIS Description of Data Elements. The auxiliary attack does not require the data to be dense and, indeed, some of the ages in this range do not appear in the records of any of the largest 200 hospitals. Further, each hospital's data is not necessarily dense with respect to the auxiliary distribution.

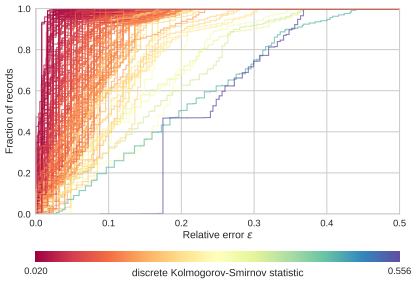
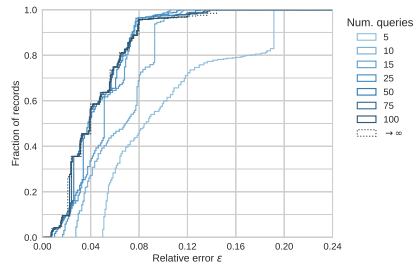


Fig. 4: Fraction of records recovered within ϵ of actual age as number of observed queries tends to infinity, for largest 200 hospitals.

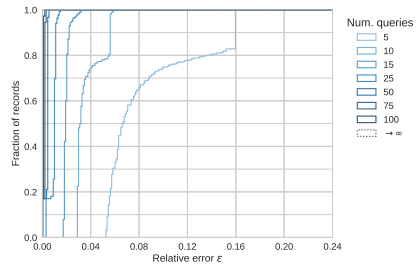
We measure the attack's success by computing the proportion of a hospital's records that was recovered within ϵ, i.e. the fraction of records for which the guessed value is at most ϵN years away from the true value, where N is the number of possible values, 125. It is important to note that ϵ is used only to characterise results and is *not* a parameter of the attack, unlike the approximate reconstruction attack in Section III. Here, there is no guarantee that the guessed values are within ϵ of the true values.

We display results from our experiments in plots with relative error (ϵ) on the x-axis and cumulative fraction of records on the y-axis. Since ages are in the range $[0, 124]$, the margin of error for all records cannot be higher than 125, which corresponds to $\epsilon = 1$. Perfect reconstruction corresponds to a vertical line at $\epsilon = 0$, while successful attacks have steeply rising curves that reach $y = 1$ for small values of ϵ.

Suitability of the auxiliary distribution. First, we demonstrate the extent to which the performance of this attack is limited by the accuracy of the auxiliary distribution. Although each hospital has over 10,000 records, the per-hospital distributions of ages can vary greatly from the auxiliary distribution. (Such differences could be due to regional demographics or specialised departments, e.g. neonatal, pediatric, or geriatric.) Fig. 4 shows the *asymptotic* success of the attack for each of the 200 hospitals: we plot the fraction of records recovered within ϵ as the number of observed queries tends to infinity, meaning that there has been enough leakage to fully determine the partitions of records and the partitions of positions. To measure how closely each hospital's distribution matches the auxiliary distribution, and to investigate the effect of this on the success of our attack, we colour-code each hospital's curve in Fig. 4 with the discrete Kolmogorov-Smirnov (K-S) statistic for the per hospital and aggregate distributions. The K-S



(a) Approximate auxiliary information



(b) Exact auxiliary information

Fig. 5: Fraction of records recovered within ϵ of actual age for one hospital, averaged over 1000 experiments.

statistic is a cumulative goodness-of-fit test; for two discrete distributions, it is the maximum absolute difference between their cumulative distribution functions: $KS(q, q') := \max_{z \in \mathcal{X}} |q_z - q'_z|$. The smaller the K-S statistic, the closer the two distributions. If the attack were carried out with exact knowledge of the frequencies, the relative error for all records would be 0 as the number of observed queries tends to infinity. Fig. 4 illustrates the importance of the closeness of the auxiliary distribution to the actual distribution: for small K-S values (encoded in dark red), the algorithm generally performs better, recovering more records with a smaller relative error.

In the remainder of this section, we focus on one hospital with over 30,000 records whose distribution's closeness to the auxiliary distribution was about average: its K-S statistic (about 0.098) is near the median (about 0.103).

Required number of observed queries. Fig. 5a shows the success of the attack on this particular hospital's data, averaged over 1000 experiments, with values assigned to records after 5, 10, 15, 25, 50, 75, and 100 queries. Fig. 5b shows what its success would be if the auxiliary information were perfect. Even with this simple heuristic attack and only approximate auxiliary information, the number of

queries required to reconstruct most of the database is relatively small. (Recall that for $N = 125$, the expected number of queries for our full reconstruction attack with rank leakage is 782.) After observing only 10 queries, an attacker can already guess the ages of 70% of records within 10 years ($\epsilon = 0.08$). After 25 queries, 95% of records are guessed within 10 years, and 55% within 5 years. After 100 queries, the success of the attack is restricted only by the accuracy of the auxiliary data's distribution.

In Appendix C, we investigate the effect of relaxing two assumptions: that the total number of records is known, and that the set of all record identifiers is known.

V. Conclusions

Building secure databases supporting commonly expected functionality, such as range queries, is a challenging task. Initial solutions based on OPE or ORE offered great usability, but were quickly shown to offer little security in the face of frequency analysis attacks exploiting auxiliary distributions, cf. [2], [7] – and no security at all if the database is dense. Second-generation schemes such as the Lewi-Wu ORE scheme, Arx, and FH-OPE, seek to increase security while preserving all or most of the functionality of OPE. Indeed, significant progress seems to have been achieved against snapshot adversaries: although FH-OPE is still vulnerable to some attacks [7], the Lewi-Wu scheme as well as Arx offer, at least in principle, a high level of security against snapshot attackers[4], in a way that precludes the type of statistical inference attacks that worked so powerfully against earlier candidates.

This left open the question of the security offered by such schemes against persistent adversaries, including an honest-but-curious host server (or any adversary having compromised the server for a sufficient length of time). On that front, the generic attack by Kellaris *et al.* [9] shows that observing the access pattern leakage of $O(N^2 \log N)$ queries is enough to reconstruct the value of all records when the database is dense. Our own attacks reduce the expected number of queries to $N \log(N) + O(N)$ in the same setting. Even for a relatively low number of values $N = 125$, this reduces the required number of queries from around 75,000 to around 800. We also prove a linear upper bound $O(N)$ for approximate reconstruction with a fixed precision (say, 5%). These results come with matching lower bounds, and efficient algorithms (which, in the case of the full reconstruction attack, are also data-optimal).

Furthermore, we investigate the setting where rank information is leaked (as is the case for the Lewi-Wu scheme, Arx, and FH-OPE), and an approximation of the distribution of plaintext values is known to the adversary. In that setting, our experiments on a real-world medical database show that after observing the leakage of as few as 25 queries, the age of a majority of patients could be reconstructed to within 5 years, even with imperfect auxiliary information.

All of our results combine to show that, to the best of our knowledge, no known practical scheme supporting range queries achieves a meaningful level of privacy for the client's data with respect to the server hosting the database (or any other persistent adversary).

It would be interesting to analyse to what extent full or approximate reconstruction can be achieved when the density assumption does *not* hold. Such attacks were explored in [9], where a full reconstruction algorithm requiring the access pattern leakage from $O(N^4 \log N)$ queries was established and shown to be essentially optimal over all data distributions. Still, improvements may well be possible for typical data distributions (as opposed to the pathological ones used to show an $\Omega(N^4)$ lower bound in [9]). Another interesting extension would be to study the effectiveness of such attacks for non-uniform range queries, for instance using real-world range query samples (although by definition our data-optimal algorithms would still behave optimally, their data requirements may change). On the positive side, an interesting open problem is whether it is possible to build database encryption schemes that satisfy some sensible efficiency criteria while offering meaningful security against persistent adversaries.

In the absence of such a scheme, we are not in a position to suggest defences against the attacks presented in this paper, except for employing methods that hide access patterns. This can be done using ORAM or PIR techniques. These approaches are currently rather expensive for use at large scale. They suggest that developing "good enough" oblivious access techniques for stored data, where some security is traded for increased efficiency, would be a fruitful research direction.

Acknowledgements

The authors would like to thank Paul Grubbs and Raluca Ada Popa for their helpful comments on an earlier version of this paper. The first author was supported by the European Union's Hori-

[4]Counterpoints on the realism of basic snapshot adversaries are discussed in [23].

zon 2020 research and innovation programme under grant agreement No. H2020-MSCA-ITN-2014-643161 ECRYPT-NET. The second author was supported by EPSRC Grant EP/L018543/1.

References

[1] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *ACM CCS 15* (I. Ray, N. Li, and C. Kruegel:, eds.), pp. 668–679, ACM Press, Oct. 2015.

[2] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *ACM CCS 15* (I. Ray, N. Li, and C. Kruegel:, eds.), pp. 644–655, ACM Press, Oct. 2015.

[3] P. Grubbs, R. McPherson, M. Naveed, T. Ristenpart, and V. Shmatikov, "Breaking web applications built on top of encrypted data," in *ACM CCS 16* (E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, eds.), pp. 1353–1364, ACM Press, Oct. 2016.

[4] D. Pouliot and C. V. Wright, "The shadow nemesis: Inference attacks on efficiently deployable, efficiently searchable encryption," in *ACM CCS 16* (E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, eds.), pp. 1341–1352, ACM Press, Oct. 2016.

[5] Y. Zhang, J. Katz, and C. 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, Austin, TX, USA, August 10-12, 2016.* (T. Holz and S. Savage, eds.), pp. 707–720, USENIX Association, 2016.

[6] F. B. Durak, T. M. DuBuisson, and D. Cash, "What else is revealed by order-revealing encryption?," in *ACM CCS 16* (E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, eds.), pp. 1155–1166, ACM Press, Oct. 2016.

[7] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart, "Leakage-abuse attacks against order-revealing encryption," in *2017 IEEE Symposium on Security and Privacy*, pp. 655–672, IEEE Computer Society Press, May 2017.

[8] B. Fuller, M. Varia, A. Yerukhimovich, E. Shen, A. Hamlin, V. Gadepally, R. Shay, J. D. Mitchell, and R. K. Cunningham, "Sok: Cryptographically protected database search," in *2017 IEEE Symposium on Security and Privacy*, pp. 172–191, IEEE Computer Society Press, May 2017.

[9] G. Kellaris, G. Kollios, K. Nissim, and A. O'Neill, "Generic attacks on secure outsourced databases," in *ACM CCS 16* (E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, eds.), pp. 1329–1340, ACM Press, Oct. 2016.

[10] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, and J. Zimmerman, "Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation," in *EUROCRYPT 2015, Part II* (E. Oswald and M. Fischlin, eds.), vol. 9057 of *LNCS*, pp. 563–594, Springer, Heidelberg, Apr. 2015.

[11] N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu, "Practical order-revealing encryption with limited leakage," in *FSE 2016* (T. Peyrin, ed.), vol. 9783 of *LNCS*, pp. 474–493, Springer, Heidelberg, Mar. 2016.

[12] V. Pappas, F. Krell, B. Vo, V. Kolesnikov, T. Malkin, S. G. Choi, W. George, A. D. Keromytis, and S. Bellovin, "Blind seer: A scalable private DBMS," in *2014 IEEE Symposium on Security and Privacy*, pp. 359–374, IEEE Computer Society Press, May 2014.

[13] S. Faber, S. Jarecki, H. Krawczyk, Q. Nguyen, M.-C. Rosu, and M. Steiner, "Rich queries on encrypted data: Beyond exact matches," in *ESORICS 2015, Part II* (G. Pernul, P. Y. A. Ryan, and E. R. Weippl, eds.), vol. 9327 of *LNCS*, pp. 123–145, Springer, Heidelberg, Sept. 2015.

[14] Y. Lu, "Privacy-preserving logarithmic-time search on encrypted data in cloud," in *NDSS 2012*, The Internet Society, Feb. 2012.

[15] K. Lewi and D. J. Wu, "Order-revealing encryption: New constructions, applications, and lower bounds," in *ACM CCS 16* (E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, eds.), pp. 1167–1178, ACM Press, Oct. 2016.

[16] X. Yuan, Y. Guo, X. Wang, C. Wang, B. Li, and X. Jia, "EncKV: An encrypted key-value store with rich queries," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, (New York, NY, USA), pp. 423–435, ACM, 2017.

[17] R. Poddar, T. Boelter, and R. A. Popa, "Arx: A strongly encrypted database system." Cryptology ePrint Archive, Report 2016/591, 2016. http://eprint.iacr.org/2016/591.

[18] F. Kerschbaum, "Frequency-hiding order-preserving encryption," in *ACM CCS 15* (I. Ray, N. Li, and C. Kruegel:, eds.), pp. 656–667, ACM Press, Oct. 2015.

[19] A. Arasu, K. Eguro, M. Joglekar, R. Kaushik, D. Kossmann, and R. Ramamurthy, "Transaction processing on confidential data using Cipherbase," in *2015 IEEE 31st International Conference on Data Engineering*, pp. 435–446, April 2015.

[20] D. S. Roche, D. Apon, S. G. Choi, and A. Yerukhimovich, "POPE: Partial order preserving encoding," in *ACM CCS 16* (E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, eds.), pp. 1131–1142, ACM Press, Oct. 2016.

[21] F. Hahn and F. Kerschbaum, "Poly-logarithmic range queries on encrypted data with small leakage," in *Proceedings of the 2016 ACM on Cloud Computing Security Workshop*, CCSW '16, (New York, NY, USA), pp. 23–34, ACM, 2016.

[22] Agency for Healthcare Research and Quality, Rockville, MD, "HCUP Nationwide Inpatient Sample (NIS), Healthcare Cost and Utilization Project (HCUP)," 2009. http://www.hcup-us.ahrq.gov/nisoverview.jsp.

[23] P. Grubbs, T. Ristenpart, and V. Shmatikov, "Why your encrypted database is not secure." Cryptology ePrint Archive, Report 2017/468, 2017. http://eprint.iacr.org/2017/468.

Appendix A
Aproximate Reconstruction Algorithm

In this section, we describe our approximate reconstruction algorithm, introduced in Section III. The input of the algorithm is the access pattern leakage of a set of queries $\mathcal{Q} = \{\mathcal{M}_k : k \leq Q\}$ on ranges $[x_k, y_k]$, with $\mathcal{M}_k = \mathcal{S}_{[x_k,y_k]}$. The algorithm proceeds in two steps.

The first step is to split the set of records into two "halves" (and a middle part) as follows. Let r be an arbitrary record. (All possible choices of this record will be tried until the algorithm succeeds.) Let M denote the intersection of all queries containing r. We wish to find two sets of records half_L and half_R such that $\mathrm{half}_L \cup \mathrm{half}_R$ contains all records, $\mathrm{half}_L \cap \mathrm{half}_R = M$, and finally both $\mathsf{val}(\mathrm{half}_L)$ and $\mathsf{val}(\mathrm{half}_R)$ are intervals. In this way, we partition the set of records into three subsets ($\mathrm{half}_L \setminus M$, M, and $\mathrm{half}_R \setminus M$) such that the corresponding sets of values ($\mathsf{val}(\mathrm{half}_L \setminus M)$, $\mathsf{val}(M)$, and $\mathsf{val}(\mathrm{half}_R \setminus M)$) is a partition of $[1, N]$ into three successive intervals. We will then sort records independently in half_L and half_R.

The exact technique used to build half_L and half_R is given in lines 3-6 in Fig. 6. Each subset is built as a union of two overlapping queries, ensuring that the corresponding value set is an interval, while incurring only a cost $O(N)$ in the expected number of queries for the algorithm to succeed.

The second step is to sort records within $\mathrm{half}_L \setminus M$. To do so, define a *left coupon* as a set of records of the form $q \setminus \mathrm{half}_R$ such that q is a query (*i.e.* $q = \mathcal{M}_k$ for some k) containing M. We claim that the set $\mathcal{C}_L$ of left coupons is linearly ordered for $\subset$. To see this, shift the perspective from records to their values: let $[x_M, y_M] = \mathsf{val}(M)$; then every left coupon is equal to $\mathcal{S}_{[x, x_M - 1]}$ for some $x < x_M$. Indeed, any query q containing M is such that $\mathsf{val}(q) = \mathcal{S}_{[x,y]}$ for some $x \leq x_M \leq y$, and $\mathsf{val}(\mathrm{half}_R) = [x_M, N]$. The linear order $\subset$ on left coupons clearly implies the (reverse) order for the value of new records appearing in each successive coupon. Thus we have identified and sorted $n_L = |\mathcal{C}_L|$ distinct sets of records, all below x_M.

We repeat the process for half_R to partition half_R into $n_R = |\mathcal{C}_R|$ ordered subsets, all above y_M. Finally, we have sorted $n_L + n_R + 1$ distinct subsets of records, whose union covers all records.[5] Since the values appearing in each subset of the global partition must be distinct, the values appearing in the k-th subset must be at least k; likewise they can be at most $k + N - (n_L + n_R + 1)$ (since the ℓ-th set counting from the right can contain values at most $N + 1 - \ell$). Hence if $n_L + n_R + 1 = (1 - \epsilon)N$, we have succeeded in approximate reconstruction with precision ϵN. If that condition is not satisfied, try again at the first step with the next value of r. Pseudo-code is provided in Fig. 6.

Appendix B
Auxiliary Attack Step 2 Variants

Although the approach we took for Step 2 of the auxiliary attack in Section IV is grounded on relevant analysis, and performs well in practice, it remains heuristic. Indeed, define the set of *boundaries* as the image of rank; the name *boundary* comes from the fact that boundaries separate distinct values in the sorted list of records L. From this perspective, each query leaks two boundaries a_k and b_k. At a high level, the problem at hand is to compute the distribution of x, y conditioned on knowing $\mathsf{rank}(x-1) = a$, $\mathsf{rank}(y) = b$, *and* on the knowledge of all other known boundaries a_k and b_k.[6] In the above approach, we disregard boundaries other than a and b, and tackle each of them in isolation, which is a reasonable approximation, but not a perfect one. Although computing the likelihood of a given assignment of boundaries to values is simple enough (it follows a multinomial distribution), solving the previous problem that takes into account all boundaries simultaneously, seems to require, at least naively, searching a space of size exponential in N.

A number of trade-offs between accuracy and processing power are possible however. In the remainder, we mention a few such optimisations for step (2a). Regarding step (2b) of the algorithm, we also briefly discuss other choices for the final estimate of $\mathsf{val}(r)$, depending on what metric we wish to optimise.

Starting with step (2a), one possibility is to compute the assignment of both ends of the interval $[a, b]$ simultaneously. That is, instead of computing z_a and z_b to maximise $\Pr[\mathsf{rank}(z_a) = a]$ and $\Pr[\mathsf{rank}(z_b) = b]$ independently, maximise the joint probability $\Pr[\mathsf{rank}(z_a) = a \wedge \mathsf{rank}(z_b) = b]$, which follows a trinomial distribution:

$$\begin{aligned}&\Pr[\mathsf{rank}(z_a) = a \wedge \mathsf{rank}(z_b) = b] \\ &= \frac{R!}{a!(b-a)!(R-b)!} \\ &\quad \cdot q_{z_a}^{a} (q_{z_b} - q_{z_a})^{b-a} (1 - q_{z_b})^{R-b}. \end{aligned} \tag{3}$$

Another possibility is to observe that in the original approach as well as the one just above, we first compute the most likely values x, y such that $\mathsf{rank}(x-1) = a$ and $\mathsf{rank}(y) = b$, *then* approximate the distribution of $\mathsf{val}(r)$ by D restricted to $[x, y]$. In other words we are forcing $\mathsf{rank}^{-1}(a)$ and $\mathsf{rank}^{-1}(b)$ to take their most likely values and deducing the distribution of $\mathsf{val}(r)$ from there. We could instead compute the entire distribution of x and y conditioned on $\mathsf{rank}(x-1) = a$ and $\mathsf{rank}(y) = b$, using either (1) or (3); then use this entire distribution to infer that of $\mathsf{val}(r)$. More explicitly, the distribution of $\mathsf{val}(r)$ becomes:

$$\begin{aligned}&\Pr[\mathsf{val}(r) = t] \\ &= \sum_{(x,y) \in \mathcal{X}^2} \Pr[\mathsf{rank}(x-1) = a \wedge \mathsf{rank}(y) = b] \\ &\qquad \cdot \Pr[\mathsf{val}(r) = t | \mathsf{rank}(x-1) = a \wedge \mathsf{rank}(y) = b] \end{aligned} \tag{4}$$

where the first term is equal to $\Pr[\mathsf{val}(r) = t | \mathsf{val}(r) \in [x, y]] = \left(\sum_{i=x}^{y} p_i\right)^{-1} p_t$

[5] Recall that we assume $[1, N]$ to be a query; or equivalently, that the set of all records is known. As a result $\mathcal{S}_{[1, x_M - 1]}$, $\mathcal{S}_{[y_M + 1, N]}$ must appear, respectively, as left and right coupons.

[6] Empirical solutions for a similar but distinct problem, akin to finding the most likely simultaneous assignment of all boundaries, are proposed in [2], [7].

APPROXIMATE-RECONSTRUCTION($\mathcal{Q}$):

Input: set of queries $\mathcal{Q} = \{\mathcal{M}_k\}$, real $0 < \epsilon < 1$.

Output: $\perp$, or maps $\mathrm{minVal}, \mathrm{maxVal} : \mathcal{R} \to \mathcal{X}$ s.t. $\forall r, \mathsf{val}(r) \in [\mathrm{minVal}(r), \mathrm{maxVal}(r)]$ or $\forall r, \mathsf{val}(r) \in [N + 1 - \mathrm{maxVal}(r), N + 1 - \mathrm{minVal}(r)]$; and $\forall r, \mathrm{maxVal}(r) - \mathrm{minVal}(r) \leq \epsilon N$.

for all $r \in \mathcal{R}$ **do**
  $M \leftarrow \bigcap_{\{k : r \in \mathcal{M}_k\}} \mathcal{M}_k$ ▷ *Partitioning* step
  Find q_L, q_R s.t. $q_L \cap q_R = M$, maximizing $|q_L \cup q_R|$
  Find q'_L s.t. $q'_L \cap q_L \neq \varnothing$, $q'_L \cap q_R \subseteq M$, maximizing $|q'_L \cup q_L|$
  Find q'_R s.t. $q'_R \cap q_R \neq \varnothing$, $q'_R \cap q_L \subseteq M$, maximizing $|q'_R \cup q_R|$
  if $q'_L \cup q_L \cup q_R \cup q'_R = \mathcal{R}$ **then**
    $\mathrm{half}_L \leftarrow q'_L \cup q_L$
    $\mathrm{half}_R \leftarrow q_R \cup q'_R$
    $\mathcal{C}_L \leftarrow \{q \setminus \mathrm{half}_R : q \in \mathcal{Q}, M \subseteq q\} \setminus \{\varnothing\}$ ▷ Left *sorting* step
    $n_L \leftarrow |\mathcal{C}_L|$
    $(\mathcal{C}_L[1], \ldots, \mathcal{C}_L[n_L|) \leftarrow$ sort $\mathcal{C}_L$ for order $\subset$ ▷ It is a linear order
    $\mathcal{C}_R \leftarrow \{q \setminus \mathrm{half}_L : q \in \mathcal{Q}, M \subseteq q\} \setminus \{\varnothing\}$ ▷ Right *sorting* step
    $n_R \leftarrow |\mathcal{C}_R|$
    $(\mathcal{C}_R[1], \ldots, \mathcal{C}_R[n_R]) \leftarrow$ sort $\mathcal{C}_R$ for order $\subset$ ▷ It is a linear order
    if $N - (n_L + n_R + 1) \leq \epsilon N$ **then**
      for all $r \in \mathcal{R}$ **do** ▷ Success
        if $r \in \mathrm{half}_L$ **then**
          $\mathrm{minVal}(r) \leftarrow n_L + 1 - \min\{i : r \in \mathcal{C}_L[i]\}$
        else if $r \in M$ **then**
          $\mathrm{minVal}(r) \leftarrow n_L + 1$
        else if $r \in \mathrm{half}_R$ **then**
          $\mathrm{minVal}(r) \leftarrow n_L + 1 + \min\{i : r \in \mathcal{C}_R[i]\}$
        end if
        $\mathrm{maxVal}(r) \leftarrow \mathrm{minVal}(r) + N - (n_L + n_R + 1)$
      end for
      return minVal, maxVal
    end if
  end if
end for
return $\perp$

Fig. 6: Approximate reconstruction attack from access pattern leakage.

and the second term can be computed using (3). A merit of this approach is that it fully captures the information leaked by $\mathsf{pos}(r) \in [a, b]$. It does, however, remain heuristic as already discussed – in the sense that we are still ignoring the existence of other known boundaries.

In step (2b), we use the approximation D_v of the distribution of $\mathsf{val}(r)$ output by step (2a) to produce an estimate e for $\mathsf{val}(r)$. We chose to output the expected value of x for $x \leftarrow D_v$. If we define the *error* as the difference $x - e$ between $x \leftarrow D_v$ and the estimate e, then the choice of picking the expectation ensures that the mean error is zero, and also minimises the variance of the error.[7] Other metrics we may wish to minimise include the mean of the absolute error $|x - e|$; in that case we should pick the median of the distribution as our estimate; or we may prefer to minimise the median of the absolute error, which amounts to finding an interval $[s, t] \subseteq \mathcal{X}$ of minimal length and probability at least $1/2$, and outputting $(s + t)/2$ (finding such an interval can be done in time $O(N)$).

Finally, we note that the algorithm could output more than simply an estimate of the value. It could, for example, simply output D_v; or, after estimating

[7] To see this, if D_v assigns probability p_i to $x = i$, then the variance of the error is $\sum_i p_i (i - e)^2$; it has degree two in e and derivative $2(e - \mathbb{E}[D_v])$, so its minimum is reached for $e = \mathbb{E}[D_v]$.

the most likely values for x and y, output the entire range $[x, y]$. Alternatively, the algorithm could compute a confidence interval around the estimated value from (2). This is straightforward using the approximate distribution output by step (2a). Such a confidence interval would be particularly meaningful if the distribution is computed as in (4), since it would then properly capture situations where a boundary falls in the middle of many successive values with low probability, which can introduce a significant amount of uncertainty.

Appendix C
Relaxing Auxiliary Attack Assumptions

Although the auxiliary attack in Section IV does not assume density of the data, it still makes two important assumptions: that the total number of records is known (which is necessary to obtain point guesses from rank values), and that the set of all record identifiers is known. While we have referred to sets of record identifiers as "access pattern leakage", this information need not come from observing disk accesses at the server – the adversary could learn the record identifiers by intercepting the query result, for instance. In such a setting, the adversary would not be able to learn the set of record identifiers or the total number of records by any methods that require (physical) access to the storage media. We briefly discuss the impact of removing these two assumptions.

Number of records. For some distributions – in particular, the age data we used – approximating the total number of records is easy with only a few queries. Consider the following approach: (i) compute the *expected value* of the maximum value in $\mathcal{X}$ that was a query end point in the set of Q queries, then (ii) use the maximum observed rank and the cumulative auxiliary distribution to arrive at an estimate $\tilde{R}$.

Recall that ranges are chosen independently and uniformly at random from $\mathcal{X} = [1, N]$. Let A_Q denote the maximum value of a query end point after Q queries have been made:

$$A_Q = \max_{\text{query } k \in \mathcal{Q}} \max\{x_k, y_k\} = \max_{\text{query } k \in \mathcal{Q}} \{y_k\}.$$

Although this value is unknown, it must correspond to the maximum observed rank and we can compute its expected value. We show in the full version of the paper that the probability that the right end point y of a uniformly random range is equal to $z \in \mathcal{X}$ is $\frac{2z}{N(N+1)}$; and so the probability that is less than or equal to z is $\frac{z(z+1)}{N(N+1)}$. Hence the probability that $A_Q = z$ for any $z \in \mathcal{X}$ is:

$$\begin{aligned}\Pr[A_Q = z] &= \Pr[A_Q \leq z] - \Pr[A_Q \leq z-1] \\ &= \left(\frac{z(z+1)}{N(N+1)}\right)^Q - \left(\frac{(z-1)z}{N(N+1)}\right)^Q.\end{aligned}$$

Simplifying, we find an expected value:

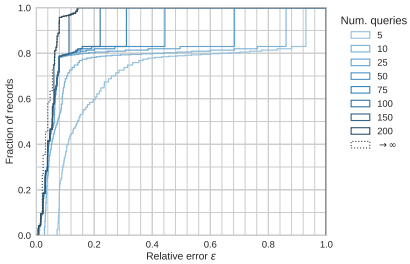
$$\mathbb{E}[A_Q] = N - \frac{1}{(N(N+1))^Q} \sum_{z=1}^{N-1} (z(z+1))^Q.$$

Now, guess that the maximum observed rank, $b_{\max} := \max_{\text{query } k \in \mathcal{Q}} \{b_k\}$, corresponds to the end point $\hat{y}_{max} := \lfloor \mathbb{E}(A_Q) \rceil$. Finally, we can estimate the number of records R using the cumulative distribution function q derived from the auxiliary distribution: $\tilde{R} := \lfloor b_{\max} / q_{\hat{y}_{\max}} \rceil$.

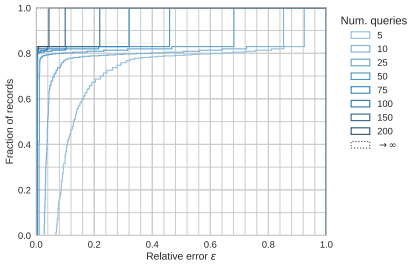
Returning to our experiment, the expected maximum observed end point is 118.6 after just 10 queries, 122.0 after 25 queries, 123.2 after 50 queries, and 123.6 after 75 queries. This heuristic works well for the age dataset because query end points are sampled uniformly at random, while ages above 110 are infrequent and not dense, so it is unlikely to take more than 10 queries for the maximum rank value (i.e. the number of records) to be observed. For other distributions, perhaps the minimum or the mean would be more suitable. Since so few queries are required to estimate the total number of records R given an auxiliary distribution, we are confident that removing this assumption would not have significantly decreased the attack's success in our experiments.

Set of record identifiers. Next, consider the assumption that the adversary knows the set of all record identifiers. If we remove this assumption, then it is clear that the adversary cannot "reconstruct" any records that have not matched at least one query. Fig. 7 shows the results of the experiment when the attacker does not know the set of possible record identifiers before observing any queries, with the aggregate auxiliary distribution (7a) and exact auxiliary information (7b).

In the case of our experiment, the most significant phenomenon arises from the fact that 17% of our chosen hospital's records have value 0 (corresponding to infants). If no query covers the value 0, then the corresponding 17% of records cannot be reconstructed. This results in a sharp jump depending on whether a query covering the value 0 has been issued, visible in the form of vertical lines at the top of the curves in Fig. 7. (Because we average the error over a large number of experiments, the horizontal



(a) Approximate auxiliary information



(b) Exact auxiliary information

Fig. 7: Fraction of records recovered within ϵ of actual age for one hospital, averaged over 1000 experiments, without assumption that set of record identifiers is known.

position of the vertical line at the top of each curve reflects the probability that a query covering the value 0 was issued: indeed if the value 0 has been queried, the corresponding 17% of records are recovered with an error close to 0; while if it was not queried, the records cannot be reconstructed, and are attributed an error of 1.)

When we assume the set of all record identifiers is known, the attacker recovers all records within an error of about $\epsilon = 0.19$ after seeing only 5 queries and using the approximate auxiliary distribution. Without record identifiers, the IDs of 17% of the records cannot be recovered (much less reconstructed) until an expected number of $(N + 1)/2 = 63$ queries have been issued. As a result the performance of the attack, visible on Fig. 7, is significantly worse than in the case where record identifiers are known. (When comparing these graphs to Fig. 5, note that the x-axis now runs from 0 to 1 rather than 0.24 and that the results are for up to 200 queries rather than just 100.) This illustrates the value of knowing the set of record identifiers in our attacks.