Building and breaking encrypted search schemes for ordered data

Marie-Sarah Lacharité

Thesis submitted to the University of London
for the degree of Doctor of Philosophy

Information Security Group
School of Mathematics and Information Security
Royal Holloway, University of London

2020

Declaration

These doctoral studies were conducted under the supervision of Professor Kenneth G. Paterson.

The work presented in this thesis is the result of original research I conducted, in collaboration with others, whilst enrolled in the School of Mathematics and Information Security as a candidate for the degree of Doctor of Philosophy. This work has not been submitted for any other degree or award in any other university or educational establishment.

Marie-Sarah Lacharité

2020

Acknowledgements

First, thank you to my supervisor Kenny for advising me in a research area that was, in 2015, new to me. His ability to draw on knowledge from across mathematics, cryptography, and computer science while staying grounded in practicality continually impresses me. Thanks as well for motivating me when I needed it the most.

Thank you to my brilliant co-authors, Brice, Kenny, and Paul—I would work with you again anytime. Thank you to my thesis examiners, Keith and Pierre-Alain, for your perceptive comments. Thank you to Adrian Waller for hosting me at Thales in Reading in summer 2017, and thank you to Thomas Ristenpart and Rachit Agarwal for hosting me at Cornell Tech in fall 2018 and winter 2019.

The ECRYPT-NET program provided the warmest welcome to living in Europe that I could ever have imagined. During my three years as an ECRYPT-NET fellow, I had the privilege to attend schools and workshops in nearly a dozen European countries. Thank you to the broad ECRYPT-NET family for providing these incredible technical and cultural opportunities.

Thank you to my family, who have always supported me, wherever in the world I go and whatever I do. Thank you to the colleagues, friends, and lovers who enriched my life during my time as a PhD student. Paul, all my love.

Ta. Merci. Thank you.

July 2020

Abstract

Encrypted search schemes allow a client to query and retrieve its outsourced data without the storage provider having to decrypt it. Usually, these schemes sacrifice some security for better functionality or efficiency. Their security is typically measured using a *leakage function*, which articulates exactly what property of the data or queries is revealed to an adversary. In this thesis, we construct, analyze, and attack encrypted search schemes on ordered data that support equality or range queries.

In the first chapter, we consider leakage functions that reveal repetitions in the data. We first present a modern statistical re-framing of frequency analysis-type attacks that exploit this plaintext repetition leakage in a one-time snapshot of data. Such attacks enable an adversary, equipped with an estimate of the distribution from which the plaintext data has been sampled, to *infer* exactly what the underlying plaintext values are. We introduce and develop the notion of frequency-smoothing encryption (FSE) to thwart these attacks while maintaining the ability to query the data. FSE is a provably secure construction and we also present results of an empirical assessment of its security on real data.

In the second (and all further) chapters, we consider passive, persistent adversaries that observe leakage from multiple queries instead of a snapshot of the encrypted data. In the second chapter, we present reconstruction attacks against range query schemes that leak the *access pattern*—which documents a query matches—and the *ranks* of the query endpoints (but not the endpoints themselves). These attacks require that every value appears at least once in the database, a property of the database we call "density." We present exact and approximate reconstruction attacks. In the latter, the aim is to recover the value of each record within some relative error. We also present a heuristic inference attack that additionally uses an estimate of the data's distribution to reconstruct the values, even when the data is non-dense. The performance of this last attack was evaluated on real data.

In the third chapter, we examine reconstruction attacks that exploit only access pattern leakage, not rank leakage, from range queries. First, we present a new analysis that reduces (from a previous analysis in the literature) the expected number of necessary queries to fully reconstruct dense data. Next, we present and analyze an approximate reconstruction attack for dense data. Reconstructing non-dense ("sparse") data requires a new algorithm with a slightly modified goal: we present such a reconstruction attack, its analysis, and experimental evaluations on both sparse and dense data.

In the last chapter, we present and evaluate attacks exploiting the most restricted kind of leakage yet: *volume* leakage—the number of records matching a query. We present a heuristic reconstruction algorithm for dense data using only volume leakage from range queries, reasoning about its design with a graph-theoretic model. An experimental evaluation of the attack on real data includes an example of strategies that can extend the results from dense datasets to sparse datasets.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Many organizations store our sensitive information, such as medical history, financial records, or personal correspondence. However, they are often breached: billions of these records have leaked in 2019 alone [78]. Data breaches have become so common that there exist services, like have i been pwned [36], that will check if a particular email address or user ID has appeared in a leak.

Security and privacy researchers have investigated ways to make these leaks less harmful while keeping whatever functionality is needed on the stored data. For instance, is there a way to encrypt medical records so that policy makers can still efficiently query the data to look for trends and make decisions, but someone who absconds with the hard drive can recover only encrypted data? Could transaction logs be encrypted in such a way that a curious data center employee does not get access to the data even though they have system administrator privileges on the database server?

It is important to analyze proposed cryptographic techniques thoroughly; believing that data is secure when it is not could be catastrophic. This thesis is mainly about how leakage from encrypted databases can still be exploited by an adversary.

Setting. We consider a simple client-server architecture where the server is storing a database. We sometimes refer to the client as the data owner. Throughout this thesis, "database" refers to a relational database—conceptually, a table where each column is associated with an attribute and contains only values from that attribute's *domain*. Without loss of generality, one of the attributes is a *record identifier* (or a "primary key" in SQL terminology) by which the record can be uniquely identified. This record identifier could simply be a counter; we assume that it is independent from any information associated with that record. We do not consider more advanced

schemes or attacks that involve correlations between columns, which have also been explored in the literature [21].

The rows in this table are referred to as the *records* or *items* in the database. A single record is represented by r and the set of all records by $\mathcal{R}$. We identify a record with its record identifier and use these interchangeably. Throughout this thesis, we consider only one attribute of a database at a time; we assume the database has only two columns—the record identifier and the record's *value*. We write $\mathsf{val}(\mathsf{r})$ to represent the value of a particular record r.

Without loss of generality, the values in an attribute's domain are ordered. For numeric data, this ordering can be the natural one; for any other data, it could be lexical or arbitrary. We extend any relation $(<, \leq, >, \geq, =)$ between the values of records to the records themselves with a superscript indicating that the relation is on their value, e.g., whenever $\mathsf{val}(\mathsf{r}) \leq \mathsf{val}(\mathsf{r}')$, we may write $\mathsf{r} \overset{\mathsf{val}}{\leq} \mathsf{r}'$. We also use these relations on sets of records, e.g., $A \overset{\mathsf{val}}{<} A'$ means that for all records $\mathsf{r} \in A$ and all records $\mathsf{r}' \in A'$, $\mathsf{val}(\mathsf{r}) < \mathsf{val}(\mathsf{r}')$.

employees

id	last_name	first_name	age	zip_code	sex	salary
1	Blum	Manuel	43	11375	M	2100
2	Germain	Sophie	30	10044	F	1900
3	Paterson	Kenneth	68	10022	X	2300

Table 1.1: An example of what is called a database throughout this thesis. We focus on one column at a time and refer to that attribute's value for a particular record r as the *value* of this record, $\mathsf{val}(\mathsf{r})$.

Types of queries. In this thesis, we consider two types of queries on single columns: equality queries and range queries. An equality query is defined by a value in that column's domain. The value need not be numeric. An equality query returns the record identifiers of all rows having the specified value in the specified column. For example, an equality query in SQL could look like

```
> SELECT id FROM employees WHERE last_name = 'Paterson';
```

A range query is defined by its two numeric endpoints. It returns the record identifiers of all rows whose value in the specified column is between or equal to the endpoints. A range query in SQL could look like

```
> SELECT id FROM employees WHERE salary >= 2000 AND salary <= 3000;
```

or

```
> SELECT id FROM employees WHERE salary BETWEEN 2000 AND 3000;
```

All numeric values are assumed to be integers. Throughout this thesis, we consistently use a and b as the endpoints of a range query: $[a, b]$ is a typical range query with $1 \leq a \leq b \leq N$. Since we assume numeric values are integers, we obtain another representation of range queries by simply converting a range to a set:

```
> SELECT id FROM employees WHERE salary IN (2000, ..., 2099, 3000);
```

In Chapter 2, we also consider the "join" operation. A join is a kind of cross product. A join on unencrypted data in SQL may look like

```
> SELECT id FROM t1 JOIN t2 WHERE t1.a=t2.b
```

Density. One way to classify the databases to which our attacks apply is by the *density* of their values for a particular attribute. A dense database is one in which there is at least one record having each possible value in the attribute's domain. A non-dense (i.e., sparse) database is one in which there exists a value that does not appear in any record.

Query distribution. None of our attacks depend on the query distribution; only their analysis does. A common assumption when evaluating database reconstruction attacks is to assume queries are uniformly distributed. For example, consider range queries on the domain $[1, N]$. There are $N(N+1)/2$ such ranges, including N "point" queries of the form $[a, a]$. If queries are uniformly distributed, then the range $[1, 5]$ is as likely as $[1, 1]$, $[N-1, N]$, or $[5, 17]$; they each occur with probability $2/(N(N+1))$.

Assumptions. Throughout, we assume that the support of the distribution is known even if the exact distribution is not. We assume that the attribute values are sampled independently according to a fixed distribution. We also assume that the adversary knows the number of possible data values N in the field targeted by the range queries. Note that this number does not depend on the database under attack, but only on the type of data being targeted. For example, an attribute for month of the year may take values between 1 and 12, even though a particular database might contain no records with month 8. An adversary treat this data as having $N = 12$.

1.1 Background: database security

This section provides some context in the form of an overview of common building blocks and techniques in encrypted databases, focusing on those that support range queries. The material in this section is based on a whitepaper [46] I was invited to publish at Black Hat USA 2019, in conjunction with a briefing I gave.

Database security techniques tend to be tailored to specific threat models. For example, if the database server is trusted, but the network is not, then client-server connections can be encrypted with TLS [38]. If the database server is trusted, but there is a risk of disk theft, then full-disk encryption or page-level encryption of database files and logs can be enabled, e.g., with Transparent Data Encryption [55, 64]. If the database server is not trusted at all, then a system that encrypts all data before uploading it could be employed, e.g., a CipherCloud gateway [18] or a CryptDB [68] proxy server.

All of these solutions, however, leak some information when a query is processed, and some even leak information from the data at rest. The leaked information could be the set of records matching the query, the size of this set, or which records in the database have the same value. This information leaks even to an observer who does not have any cryptographic keys. The source of the leakage can vary; it could be raw bits on disk, network traffic, observed memory accesses, or database logs recovered by forensic analysis. This leakage can be exploited by an attacker to recover values in the database.

Looking ahead, in Chapter 2, we present a construction that mitigates inference attacks on equality leakage from data at rest, while allowing equality queries and allowing range queries, though somewhat inefficiently. In Chapters 3 through 5, we present generic attacks on rank leakage, access pattern leakage, and volume leakage from range queries. These attacks are entirely generic and do not depend on the database implementation. A detailed overview of the thesis follows in Section 1.2. First, the following three subsections provide details about typical methods used to secure databases, based on securing the data while it is in transit, at rest, or in use.

1.1.1 Protecting data in transit

Historically, protecting queries and their results from a network eavesdropper was the first step in securing a database server. The client and server can protect their communications by encrypting them according to the Transport Layer Security (TLS) standard [38]. TLS allows the client and server to authenticate each other and negotiate a session key to encrypt the queries and responses. These measures help prevent an adversary from learning what the query was or which records matched it by intercepting packets.

1.1.2 Protecting data at rest

As individuals and organizations move their data to remote data centers, they may update their threat models to reflect no longer having control over their data's physical environment and to acknowledge the possibility of disk theft. Encryption at the filesystem level or column level addresses this issue.

A typical query process might have the following major steps:

1. The client uses TLS to encrypt its query and sends it to the server.
2. The server decrypts it, consults its search index, and fetches the relevant encrypted pages from disk.
3. It decrypts them in memory, processes the results, re-encrypts them with TLS, and sends them back to the client.
4. The client decrypts them.

The data on disk stays encrypted the entire time, so disk theft is mitigated.

Many major database vendors offer some variant of this type of encryption, usually called "Transparent Data Encryption" (TDE) or "Native Encryption." These solutions usually do not noticeably affect performance; the server can still index the plaintext data, so range queries can be answered efficiently. Some of these solutions also offer more granular field-level encryption, like format-preserving encryption or tokenization, but the data encryption key is usually still managed by the server.

Some examples include Transparent Data Encryption (TDE) offered by Microsoft for SQL Server [55], which encrypts data and transactions logs at the page level; Oracle Database's Transparent Data Encryption [64], which allows column or tablespace encryption of data and encryption of undo and redo logs; IBM's Native Encryption for DB2 [37], which offers tablespace encryption of data and encryption of transaction logs; and MongoDB's Encrypted Storage Engine [59], which encrypts all data files, but not logs.

When the database server manages the data encryption key, it is usually stored in a separate keystore, i.e., not on the same disk as the data. However, it is often accessible to curious database administrators, system administrators, or any user who gains such permissions. To prevent a full database server system compromise from revealing the data, it needs to be encrypted by the client, or via a proxy, before it arrives at the server.

1.1.3 Protecting data in use

How to query encrypted data without letting the database server see raw, unencrypted data is an interesting and challenging problem (and the focus of this thesis). In industry, these solutions are usually called *client-side* field-level encryption: instead of the keys being in a keystore attached to the database server, the client completely controls them.

Such solutions are offered by Microsoft's Always Encrypted [54], which implements field-level encryption in a client-side driver; MongoDB's Field Level Encryption in its version 4.2 release from August 2019 [58]; and companies like CipherCloud [18] or solutions like CryptDB [68] that act as proxies between a client and database server. For more commercial solutions, see [24, Sec. I(A)].

Usually, these client-side encryption solutions offer only two basic types of encryption: deterministic and randomized. Deterministic encryption leaks repetition and makes range queries possible but inefficient, while randomized encryption is secure, but does not support ranges. A third type, called Order-Preserving Encryption (OPE) is also sometimes used. It leaks order and makes range queries as efficient as they are on plaintext.

ID	Value
1	3
2	1
3	15
4	41
5	1

(a) Plaintext

ID	Value
1	`0x18fa83`
2	`0x5449a1`
3	`0x8b7630`
4	`0x10cae8`
5	`0x5449a1`

(b) DE

ID	Value
1	`0x5239fb`
2	`0x8e9d98`
3	`0x5a9f2e`
4	`0x4ff8e1`
5	`0xe89cfb`

(c) RE

ID	Value
1	182
2	84
3	2307
4	8932
5	84

(d) OPE

Figure 1.1: An example illustrating the differences between the same values encrypted with Deterministic Encryption (DE), Randomized Encryption (RE), and Order-Preserving Encryption (OPE).

DE. Deterministic Encryption (DE) is an attractive option for encrypting outsourced data because it is equality-preserving: finding an exact match for a particular value is as easy as finding an exact match for its encryption of that value. This makes it possible for a client to query its data using an encrypted search term, with the server identifying and returning matches to the client without needing to decrypt them. With deterministic encryption, any repeated plaintext values show up as repeated ciphertext values. For example, each record that has a value of "1" will have

the exact same ciphertext. This enables range queries: if the client wants to retrieve all records with values between 1 and 3 (i.e., in [1,3]), it could simply request all records whose values are in the set of encryptions {DE(1), DE(2), DE(3)}. Although range queries are possible with deterministic encryption, the fact remains that any repetitions in the plaintext will show up in the ciphertext. Leaking the equality of values can be exploited when combined with information about the distribution of values. These attacks were evaluated on medical datasets by Naveed, Kamara, and Wright [62], and many others after.

RE. With Randomized Encryption (RE), the server is unable to index the data or group values. All encryptions of "1" would be different, which offers excellent security, but the server would have no way to select all of the records with value "1." Due to this lack of functionality, randomized encryption that leaks nothing about the plaintext (other than perhaps its size) is not considered further in this thesis.

OPE/ORE. Order-Preserving Encryption (OPE) [2, 10] is another method of encrypting numeric data, which does exactly what it sounds like: if $x < y$, then $\mathsf{OPE}(x) < \mathsf{OPE}(y)$. This property enables range searches and sorting encrypted data in a straightforward manner. If the client encrypts values with OPE before sending them to the server, the server can still index the data just as if it was unencrypted, but it does not learn the exact values. When the client wants to perform a range query, all it has to do is encrypt the endpoints of that range.

Unfortunately, even an "ideal" OPE scheme (one that behaves like a random order-preserving function) must leak strictly more than order. In particular, Boldyreva, Chenette and O'Neill proved that about *half* of the plaintext bits must leak [11], which is why any efficient OPE scheme cannot offer much security. Further, some OPE schemes are also deterministic, leaking repeated values, and susceptible to frequency analysis attacks [32]. This property of OPE schemes motivated new types of schemes that sacrifice less security while still allowing range queries and sorting. These include techniques like Order-Revealing Encryption (ORE) [17, 51], which is a generalization of OPE. With OPE, it is possible to look at two ciphertexts to compare them, but with ORE, it is necessary to compute a function over each pair to know which one is smaller. Some schemes use more subtle techniques, such as building a search index that the server can traverse by itself, "destroying" nodes along it as it goes, and requiring the client to "re-generate" them after each query [9]. Although these schemes leak less than OPE, they still have some leakage.

A modern encrypted database might combine all of these types of encryption: TLS for the queries and responses, server-side disk encryption, and client-side field-level encryption. Even with all of these layers, there might still be some exploitable leakage. For instance, the server needs to know which records matched a range query in order to return the correct results, so the access pattern of each query (the identifiers of the records that matched it) could leak. If the server maintains an index that allows range queries, the rank of each query endpoint (the number of records with value less it) could leak. Despite client-server communications being encrypted, their length is not hidden, so the volume of each query (how many records matched it) could leak.

The particular channel through which these properties leak varies: it could be from an adversary man-in-the-middling connections to the database, getting access to undo/redo logs or query profiling logs, or simply observing traffic volume [31]. How to extract this leakage is an important question, but it is orthogonal to this thesis: we focus on what can be learned upon having obtained the leakage.

Other approaches. There exist other solutions for securely querying an encrypted database. For example, Kamara and Moataz [41] developed a structured encryption scheme for relational databases that supports many types of SQL queries and does not leak any frequency information. This scheme is not one that could be added to an existing SQL database in a legacy-friendly manner; it would entirely replace a database and change how queries are treated.

1.2 Thesis structure and main contributions

This thesis is roughly structured in terms of security models and adversarial capabilities, from more limited to less limited adversaries. Chapter 2 introduces a scheme that is designed to thwart a so-called "snapshot attacker" that obtains a one-time copy of the contents of a database. Chapters 3 through 5 present attacks that can be mounted by a stronger adversary that continuously observes information as another party makes range queries on the database. With each chapter, mounting the attack requires less information, culminating in the volume-only attacks in Chapter 5 that can be carried out by an adversary that sees only how many records matched each query.

Overall, the main contributions of this thesis are attacks. Table 1.2 offers an overview of the different attacks considered in this thesis. The following paragraphs then provide overviews of each chapter.

Goal	Output	Success condition(s)
Decryption (§ 2.7)	Map $\theta : \mathcal{C}^+ \to \mathcal{M}$	• $\forall m \in \mathcal{M}$, $\theta\left(\mathcal{H}^{\mathsf{FSE}}(m) \cap \mathsf{DB}\right) = m$
Exact reconstruction (§ 3.2)	Map $\widehat{\mathsf{val}} : \mathcal{R} \to [1, N]$	• $\forall \mathsf{r} \in \mathcal{R}$, $\widehat{\mathsf{val}}(\mathsf{r}) = \mathsf{val}(\mathsf{r})$ ⬍ or $\forall \mathsf{r} \in \mathcal{R}$, $\widehat{\mathsf{val}}(\mathsf{r}) = N + 1 - \mathsf{val}(\mathsf{r})$
Diameter-δ approximate reconstruction (§ 3.3)	Map $\widehat{\mathsf{val}}_{\mathsf{min}}, \widehat{\mathsf{val}}_{\mathsf{max}} : \mathcal{R} \to [1, N]$	• $\forall \mathsf{r} \in \mathcal{R}$, $\mathsf{val}(\mathsf{r}) \in [\widehat{\mathsf{val}}_{\mathsf{min}}(\mathsf{r}), \widehat{\mathsf{val}}_{\mathsf{max}}(\mathsf{r})]$ • $\forall \mathsf{r} \in \mathcal{R}$, $\widehat{\mathsf{val}}_{\mathsf{max}}(\mathsf{r}) - \widehat{\mathsf{val}}_{\mathsf{min}}(\mathsf{r}) \leq \delta N$
ϵ-Approximate reconstruction (§ 4.2)	Map $\widehat{\mathsf{val}} : \mathcal{R} \to [1, N]$	• $\forall \mathsf{r} \in \mathcal{R}$, $\lvert\mathsf{val}(\mathsf{r}) - \widehat{\mathsf{val}}(\mathsf{r})\rvert < \epsilon N$ ⬍ or $\forall \mathsf{r} \in \mathcal{R}$, $\lvert N + 1 - \mathsf{val}(\mathsf{r}) - \widehat{\mathsf{val}}(\mathsf{r})\rvert < \epsilon N$
ϵ-Approximate ordered reconstruction (§ 4.3)	Disjoint subsets $A_1, \ldots, A_k \subset \mathcal{R}$	• $\forall \mathsf{r} \in \mathcal{R}$ with $\mathsf{val}(\mathsf{r}) \in (\epsilon N, N - \epsilon N)$, $\exists i : \mathsf{r} \in A_i$ • $\forall i \in [1, k]$, $\mathsf{diam}(A_i) < \epsilon N$ • $A_1 \overset{\mathsf{val}}{<} \cdots \overset{\mathsf{val}}{<} A_k$ ⬍ or $A_k \overset{\mathsf{val}}{<} \cdots \overset{\mathsf{val}}{<} A_1$
Count reconstruction (§ 5.1)	$\widehat{\mathsf{count}} : [1, N] \to \mathbb{Z}$	• $\forall i \in [1, N]$, $\widehat{\mathsf{count}}(i) = \mathsf{count}(i)$ ⬍ or $\forall i \in [1, N]$, $\widehat{\mathsf{count}}(N + 1 - i) = \mathsf{count}(i)$

Table 1.2: The types of database reconstruction we consider as attack goals in this thesis. For attacks on range query leakage, success conditions for reconstruction up to reflection are marked by ⬍.

Decryption attacks. Chapter 2 is different than the later chapters: its main contributions are definitions of a new type of scheme, Frequency-Smoothing Encryption or *FSE* (Section 2.2), security definitions (Section 2.4) and instantiations of FSE (Sections 2.5 and 2.6). FSE offers the same functionality as deterministic encryption: equality queries and, by using the simple technique of converting an interval of values to a set, range queries. The goal of FSE is to use knowledge of the plaintext's distribution to encrypt in such a way that ciphertext frequencies do not leak information about the underlying plaintext.

Despite being more constructive than Chapters 3 through 5, Chapter 2 also considers attacks on the practical security of a proposed FSE scheme (Section 2.7), where an attacker's goal is not to break the formally defined indistinguishability security notions (q.v. Section 2.4), but to construct a decryption mapping that assigns to each ciphertext its correct plaintext. In FSE, each plaintext message m can have multiple different ciphertexts, represented in its set of homophones $\mathcal{H}^{\mathsf{FSE}}(m)$, so the decryption map may be many-to-one. To succeed, the adversary must successfully decrypt each ciphertext appearing in the snapshot of the database it obtains.

Rank and access pattern leakage attacks. Chapter 3 and all further chapters deal with only range queries of the form $[a, b]$ that are sub-intervals of the integers $[1, N]$. Suppose that for each range query a client makes, the adversary learns two pieces of information about it: *which* records matched the query (the query's access pattern) and *how many* records have value less than the left query endpoint a, and how many have value at most the right query endpoint b. The second piece of information is the *rank* of the query endpoints, $\mathsf{rank}(a-1)$ and $\mathsf{rank}(b)$. Many encrypted database schemes that support range queries leak this information (q.v. Section 3.1), and no prior attacks had been designed to exploit it. With such leakage from sufficiently many different queries, an adversary may attempt to exactly reconstruct the data by assigning a value to each record: for each record $\mathsf{r} \in \mathcal{R}$, it assigns a value $\widehat{\mathsf{val}}(\mathsf{r}) \in [1, N]$. Exact (or full) reconstruction succeeds when the reconstructed values of all records are correct. We present such an attack on dense databases in Section 3.2.

Instead of exactly reconstructing the values of all records, the adversary may find it sufficient to *approximately* reconstruct their values. It can proceed by assigning each record an interval of some small width (relative to the attribute domain's size N) that must contain its correct value. Approximate reconstruction within relative diameter δ succeeds if each record is assigned an interval of width at most δN that contains its actual value. Section 3.3 presents an attack targeting diameter-δ approximate reconstruction on dense databases. Lastly, in Section 3.4, we present a different type of value reconstruction attack that uses an auxiliary distribution and makes no guarantees on the correctness of its output. The success of this attack is measured in terms of the fraction of records whose values were close to their correct values, relative to the domain size N. Our results show that the attack has good success when the auxiliary distribution is close to the data's distribution.

Access pattern leakage attacks. Using only access pattern leakage, and not rank leakage, similar types of exact and approximate reconstruction are possible. First, they require a small adaptation: with only access pattern leakage, it is impossible to distinguish (without additional information) records with value a from records with value $N+1-a$. The reflection symmetry of $[1, N]$ and $[N, 1]$ is explained at the start of Chapter 4.

While our work was the first to consider attacks using rank and access pattern leakage, using only access pattern leakage was not novel. Kellaris, Kollios, Nissim, and O'Neill (KKNO) [42] formalized and derived access pattern attacks for both dense and sparse databases, which respectively require $\Omega\big(N^2 \log N\big)$ and $\Omega\big(N^4 \log N\big)$ uniformly random queries. In Section 4.1, we present an attack that succeeds with fewer

queries on dense databases by using a different technique. We also present an attack in Section 4.2 that targets a form of approximate reconstruction: it assigns a point estimate $\widehat{\mathsf{val}}(\mathsf{r})$ to each record r and succeeds if all of these estimates are within distance ϵN of the correct record values (or their reflections). Both of these attacks apply to dense databases only.

We then introduce a new, more relaxed type of approximate reconstruction for non-dense databases: instead of assigning point values to each record, the goal is to group records with close values together, then sort these groups (up to reflection). This goal and a corresponding attack are described in Section 4.3.

Volume leakage attacks. The last chapter of the main body of this thesis exploits the most restricted kind of leakage yet: we consider an adversary that learns only how many records match each range query. Our attack is designed to work after the adversary has observed each of the $N(N+1)/2$ possible range queries. It does not know which query corresponds to which volume; it sees only the following set of volumes:

$$\big\{\mathsf{vol}([1,1]), \mathsf{vol}([1,2]), \ldots, \mathsf{vol}([1,N]), \mathsf{vol}([2,2]), \ldots, \mathsf{vol}([N-1,N]), \mathsf{vol}([N,N])\big\}.$$

In the absence of access pattern leakage, the adversary can no longer reconstruct a value mapping $\widehat{\mathsf{val}}$ as in the two previous scenarios since it does not learn any record identifiers. Instead, its goal is to determine the database's *counts*: for each value i from 1 to N, it tries to determine the number of records with that value, $\widehat{\mathsf{count}}(i)$.

We were not the first to devise reconstruction attacks on range query volume leakage; again, KKNO [42] also presented some. However, their algorithms strongly rely on the assumption that range queries are drawn independently and uniformly at random, whereas ours requires only the *set* of volumes, not their frequencies. Our algorithm in Section 5.1 uses a graph to reason about a particular set of volumes, called the *elementary volumes*, from which the counts can be reconstructed (and *vice versa*). KKNO proved that, in general, any algorithm successfully achieving exact reconstruction from volume leakage requires $\Omega\big(N^4\big)$ range queries. However, our experimental results in Section 5.2 show that reconstruction is possible for real-world databases with much fewer queries. Although our count reconstruction algorithm is heuristic and its success cannot be analyzed as easily as the algorithms in previous chapters, in Section 5.3 make a series of assumptions that allow reasoning about the structure of the graph and when the algorithm is likely to success.

1.3 Experimental evaluations and datasets

While most of the algorithms in this thesis are amenable to statistical analysis, a few are not. In either case, it can be helpful for the methodology to include experimental evaluations. Such evaluations can allow comparing results for different encoding lengths, attributes, database sizes, or data distributions, or validate assumptions made in the analysis. Each of the other chapters in this thesis contains the results of at least one experiment, sometimes on real datasets (though with simulated queries). This section introduces the datasets we use, how we processed them, and where experimental evaluations are used.

HCUP data. The real datasets were extracted from the yearly Nationwide Inpatient Samples (NIS) from the Healthcare Cost and Utilization Project (HCUP), run by the Agency for Healthcare Research and Quality (AHRQ) in the United States [1]. The AHRQ is a US government agency which collects a vast amount of data on the American healthcare industry. HCUP is one of their core projects meant to track how healthcare is used and paid for by different demographic groups. Within HCUP, there are different types of samples taken every year and made available to researchers, of which the NIS is one such sample. In this thesis, "HCUP data" refers to this NIS data.

This data source was introduced to us in Naveed, Kamara, and Wright's 2015 paper [62], discussed further in Chapter 2. The NIS is the largest publicly available inpatient database in the United States that includes discharge data for all types of payers. It includes data at the hospital level (such as number of discharges, number of beds) and at the patient discharge level (such as demographics, diagnosis, procedure, payer). Acquiring this data involved paying a fee and completing HCUP Data Use Agreement training. We did not attempt to deanonymize any of the data, nor are our attacks designed to deanonymize medical data.

As a means of introduction to the HCUP NIS data, Figure 1.2 contains information about the number of hospitals contained in each year's release and the minimum, maximum, and quartiles for the number of records per hospital for the relevant years. There is not too much year-to-year variation in the number of records per hospital for these years, which is expected since until 2012, the HCUP NIS data was collected as a random sample from all hospitals in the USA. This provides evidence that our experiments would be predictive of our attacks' performance (were they carried out on a real hospital database). In 2012, the sampling methodology for HCUP changed:

		Num. patient records per hospital				
Year	Num. hospitals	Min.	25%	50%	75%	Max.
2004	1004	15	1199	4300	11523	71580
2008	1056	3	889	3439	11170	117372
2009	1050	1	750	3278	10487	121668

Figure 1.2: Number of hospitals and quartiles for number of records per hospital for 2004, 2008 and 2009 HCUP data.

more recent HCUP data is collected using a random sample of *patients* instead of hospitals.

Table 1.3 lists the attributes we chose to extract and use in our experiments, along with their domain sizes, which range from $N = 2$ to $N = 365$. APRDRG refers to the All Patients Refined Diagnosis Related Groups, a patient classification system. For the experiments in Chapters 3–5, we used only those attributes on which range queries are meaningful.

The following is an overview of the experimental evaluations in this thesis and which ones use HCUP data:

- **Section 2.7:** Evaluation of frequency analysis attack on 12 attributes from the largest 200 hospitals in the 2009 HCUP dataset encrypted with a frequency-smoothing encryption (FSE) scheme using interval-based homophonic encoding.
- **Section 3.4:** Evaluation of inferred reconstruction attack using rank and access pattern leakage and estimate of data distribution on `AGE` data from the largest 200 hospitals in the 2009 HCUP dataset.
- **Section 4.3.4:** Evaluation of ϵ-approximate ordered reconstruction attack on synthetic data.
- **Section 5.2:** Evaluation of pre-processing and clique-finding steps on 10 about attributes from about 1000 hospitals in the 2004, 2008, and 2009 HCUP datasets. (Some attributes were not available in all years.)
- **Section 5.3.4:** Evaluation of estimates of graph properties computed in analytical Poisson model on synthetic data.

Every year the AHRQ prescribes a format and size for each attribute collected in the various samples. In extracting per-attribute experimental data from HCUP, we faced three main complications: (1) hospitals do not generally abide by these prescriptions, (2) the prescribed formats change from year to year, and (3) not all attributes exist in

all years of HCUP data. We will describe how we address each of these complications in turn.

Hospitals are strongly encouraged, but not required, to report data in the format dictated by the AHRQ, and some hospitals choose to report their data in incorrect or outdated formats. The AHRQ corrects some of these mistakes before making samples available publicly, but many mistakes still occurred in our data. For example, the attributes `NPR`, `NDX`, and `NCHRONIC` are capped by the AHRQ, but some hospitals still report greater values, which we simply ignored.

In extracting `NPR`, `NDX`, and `NCHRONIC`, we also faced the second complication: the number of values changed (increasing from 16 to 26) in 2009. One other attribute whose format changed is `AGE`. In 2012, the AHRQ mandated that ages be "top-coded" (i.e., all values above a threshold be grouped into one category) at 90 in all samples for privacy reasons. Despite not using any HCUP data from 2012 or later in the experiments in this thesis, we top-coded all `AGE` data in our experiments in Chapter 5. This top-coding was done for two main reasons: first, to make our experiments address practical security risks to real deployments (in which ages may be top-coded), and second, because our paper on which Chapter 5 is based included an additional experiment involving `AGE` data from 2013, and we wanted results to be comparable across years.

The only attribute which did not appear in all three years (namely not in 2004) of HCUP data was `NCHRONIC`. Since we had several other datasets for that attribute and the performance of all attacks on that attribute was similar across experiments, we were not concerned. We were not able to obtain the full 2008 NIS, and as a result we did not have `APRDRG_Severity` or `APRDRG_Risk_Mortality` for that year.

Table 1.3: The HCUP attributes examined in experiments throughout

Attribute (`HCUP label`)	Num. values
Age (`AGE`)	125 or 91
Admission month (`AMONTH`)	12
Admission source (`ASOURCE`)	5
Admission type (`ATYPE`)	6
Patient died (`DIED`)	2
Sex (`FEMALE`)	2
Length of stay (`LOS`)	365
Major diagnostic category (`MDC`)	25
Number of chronic conditions (`NCHRONIC`)	16 or 26
Number of diagnoses (`NDX`)	16 or 26
Number of procedures (`NPR`)	16 or 26
Primary payer (`PAY1`)	6
Ethnicity group (`RACE`)	6
ZIP code income quartile (`ZIPINC`)	4
Disease severity (`APRDRG_Severity`)	4
Mortality risk (`APRDRG_Risk_Mortality`)	4

Chapter 2

Frequency-smoothing encryption

Background. Beginning in 2015, a series of papers using statistical analysis of ciphertexts in databases showed that these techniques can yield devastating inference attacks. They targeted deterministic encryption [62], order-preserving/revealing encryption [32, 62], and searchable encryption [70]. One of the first papers I read upon starting my PhD studies was by Naveed, Kamara, and Wright and dealt with attacks on DE and OPE [62]. The two attacks on DE from this paper (simple frequency analysis and ℓ_p-optimization) motivated the work in this chapter. Here, we seek to answer the question of how to encrypt data in a way that reduces frequency leakage—thus thwarting such attacks—while preserving query functionality.

This chapter is based on the paper "Frequency-smoothing encryption: preventing snapshot attacks on deterministically encrypted data," which I co-authored with Kenny Paterson. It was published in the IACR Transactions on Symmetric Cryptology [50] in 2018 and the full version appears on the IACR's Cryptology ePrint Archive [49].

Introduction. At the heart of the aforementioned inference attacks is classical frequency analysis. In this chapter, we propose and evaluate another classical technique, homophonic encoding, as a means to combat these attacks. We introduce the concept of *frequency-smoothing encryption* (FSE) which generalizes (symmetric) deterministic encryption to the setting of "somewhat randomized" encryption, where each message has a relatively small number of possible ciphertexts (i.e., homophones). FSE is gen-

eral enough to capture schemes that handle initially unknown or changing message distributions. We also show how FSE supports equality queries and database joins.

Classical frequency analysis is a powerful attack against deterministically encrypted data. If the plaintext distribution is not uniform and an adversary has a reference dataset from which it can compute expected plaintext frequencies, then, given access to a snapshot of the encrypted data, the adversary can match frequencies in the encrypted domain with those in the plaintext domain, thus identifying which ciphertext corresponds to which plaintext. This kind of *inference attack*, where statistical techniques are used to infer plaintext information, was used to great destructive effect in the work of Naveed *et al.* [62]: they correctly inferred large amounts of patient information from DE-encrypted hospital records. Their work and related papers investigating inference attacks [70, 39, 32, 7] and leakage-abuse attacks [16, 30, 79, 21, 42, 48] have dented both the industry's and the research community's confidence in its ability to protect outsourced data while preserving query capabilities.

A few proposed OPE/ORE schemes hide frequencies, such as those by Kerschbaum [43] and Boneh *et al.* [13], but they both have limited practicality. Pouliot, Griffy, and Wright [69] developed the notion of weakly randomized encryption (WRE), in which a small amount of randomness is injected into each ciphertext to prevent frequency analysis. We discuss this and other related work in greater detail in Section 2.1.

Homophonic encoding. Our goal is to develop a rigorous means of preventing inference attacks on encrypted databases. Our solution was inspired by an old technique: homophonic encoding (HE). The goal of homophonic encoding (or homophonic substitution) is to flatten the frequency distribution of messages by mapping each plaintext to multiple possible *homophones*, with the number of encodings for each plaintext m ideally being proportional to the frequency of m. Then, although homophonically encoded data may still contain repetitions, the homophones occur roughly equally often; frequency information would not help an adversary who has a copy of the encoded data. Homophonic encoding has not received much formal analysis in this context. Moreover, it is usually applied in contexts where adjacent data items are not independent of one another—for example, letters or words in natural language—which renders it vulnerable to attacks based on analysis of bi-grams rather than single-letter frequencies. In database encryption, this inherent weakness does not arise since each column of the database is encrypted under a separate key and we treat the rows as an unordered set.

Another common feature of HE schemes is variable-length encoding. The usual advantage of variable-length codes—their low average codeword length—is not as much of an advantage in the setting of encrypted databases. In a database table, it is likely that every value in a column is allocated the same amount of storage according to the declared data type of the attribute. Variable-length entries are still possible, however—for instance, by storing a prefix indicating the length of each entry in the column. The MySQL version 5.7 reference manual describes four variable-length data types, all for strings: `VARCHAR`, `VARBINARY`, `BLOB` and `TEXT` [63]. Values in a `VARCHAR` column, for example, are stored with a prefix indicating their length in bytes. While the maximum length of an entry in the column must be specified, the data is not padded. Since we are considering applications where the data items are no longer than a few bytes, it is space-efficient to pad data to a fixed size rather than include a length prefix. For these reasons, we will consider only fixed-length codes.

By combining HE to flatten message distributions and encryption to provide message privacy, we obtain what we call *frequency-smoothing encryption* (FSE). Using HE leads to encryption schemes that are randomized: we ensure that each message has enough homophones to combat frequency analysis, but not so many that they cannot all be computed on the fly and sent to the database for comparison with the relevant column of ciphertexts. We show that making this trade-off between preventing frequency leakage and increasing query complexity is beneficial, at least for certain distributions, and provides schemes that are both secure against snapshot attackers and reasonably efficient.

Limitations. We evaluate our construction against only two types of attacks. The first is security in a somewhat randomized generalization of the standard security notion for DE due to Rogaway and Shrimpton [74]. The second is security against inference attacks made by a snapshot attacker on a per-column basis. Our security proofs and empirical evaluations are respectively focused on these notions. We do not defend against more advanced forms of attack, such as those based on query analysis, as in [16, 30], or attacks based on correlations between columns, as in [21]. Concretely, without some kind of query padding or query batching, it will be possible to carry out frequency analysis on the *queries* made in our schemes, since the number of queries required for a given plaintext m will be roughly proportional to the frequency of m. In addition, Grubbs *et al.* recently pointed out the artificiality of the snapshot attack model in view of real database management systems [31]. These systems often store additional information that an attacker would capture in its snapshot, e.g., prior

queries. Nevertheless, resisting snapshot attacks is necessary for achieving meaningful security in any realistic threat model, and our approach at least achieves this.

Despite some limitations, we believe that our work has significant value since currently, there are few good solutions that address *any* of the recent and severe inference attacks, and we show that at least some forms of attack can be effectively combated at low cost. We consider that our work on frequency-smoothing encryption could form the basis of a more complete solution to the problem of preventing inference attacks on encrypted databases.

Chapter overview. In Section 2.1, we discuss related work in more detail. We then introduce a definition for an FSE scheme (Section 2.2) and explain how FSE could be used in practice (Section 2.3.) We provide two security notions for FSE in Section 2.4. The first, FSE−SMOOTH, prevents frequency analysis attacks by requiring that a collection of FSE ciphertexts be indistinguishable from random data even when the underlying plaintext distribution is known. The second, FSE−PRIV, generalizes the symmetric deterministic encryption security notion [74]. In Section 2.5, we give a generic construction for FSE from any Deterministic Encryption (DE) scheme and any Homophonic Encoding (HE) scheme.

We propose two simple, easy-to-implement HE schemes in Section 2.6. Both schemes can be configured to yield different trade-offs between query efficiency and resistance to frequency analysis attacks. We apply a framework for optimal distinguishers [5] to our setting to show that our HE schemes asymptotically achieve perfect flattening in a statistical sense—even for computationally unbounded adversaries. However, obtaining a typical cryptographic level of security like 2^{-80} might require such long encoding lengths that the numbers of homophones per message make the query complexity impractical. For this reason, we also conduct an empirical analysis of the effectiveness of our FSE schemes with moderate values of the encoding length r (Section 2.7). We attempt to match plaintexts with ciphertexts via frequency analysis, in the same way as Naveed *et al.* [62]. Although this type of attack asks more of the adversary than FSE−SMOOTH or FSE−PRIV, resisting it is nevertheless useful, given a real-world adversary's typical aim of recovering actual plaintext. We use the method of Maximum Likelihood Estimation to derive an efficient algorithm that is statistically optimal in assigning ciphertexts to possible plaintexts (Section 2.7.1), in the same way that frequency analysis is optimal for deterministically encrypted data. We apply this algorithm to FSE-encrypted HCUP medical data and measure the attack's success in terms of the number of hospitals in which a certain fraction of records' data was successfully recovered.

Notation specific to this chapter. Let D be the actual (true) probability distribution from which messages in $\mathcal{M}$ are sampled. The probability of a message m under this distribution is $\mathsf{f}_{\mathsf{D}}(m)$. When a data owner or an adversary must guess or estimate the data's actual distribution D, we use $\tilde{\mathsf{D}}$ for the owner's approximation and $\hat{\mathsf{D}}$ for the adversary's approximation. To refer to the distribution of encoded or encrypted data in the appropriate domain, which may be dependent upon a particular state s, we use D_{s}. Table 2.1 summarizes our notation for these various distributions.

2.1 Related work

One of the works most similar to FSE, and developed in parallel, is Pouliot, Griffy, and Wright's weakly randomized encryption (WRE) scheme [69]. It uses a PRF, a distribution-dependent "salt allocation method," and an IND-CPA secure encryption scheme to create tag-ciphertext pairs. The tags allow equality queries and depend on the "salt" chosen for that encryption. Pouliot *et al.* present various salt allocation methods, such as proportional salt allocation (which resembles our interval-based homophonic encoding scheme in Section 2.6.2), and two Poisson-based variants. The first Poisson salt allocation method is vulnerable to a form of frequency analysis based on solving a knapsack problem. In this method, a Poisson process determines the number of salts per message, and the inter-arrival times of the Poisson events determine the distribution over the salts. However, the last salt for each message has a different distribution, which could be exploited by an adversary to determine which salts correspond to which message by solving a subset-sum problem. In response to our pointing this out by email, the authors updated their paper and introduced a bucketized Poisson variant [69, Sec. 5C1] that is no longer vulnerable to this attack, but introduces false positives in results of equality queries.

A few OPE/ORE schemes have properties similar to frequency-smoothing. The first OPE scheme [2], by Agrawal *et al.*, used a kind of homophonic encoding in its con-

Table 2.1: Overview of our notation for various distributions

Symbol	Domain	Description
D	$\mathcal{M}$	data's actual distribution
$\tilde{\mathsf{D}}$	$\mathcal{M}$	owner's guess of the data's distribution
$\hat{\mathsf{D}}$	$\mathcal{M}$	adversary's guess of the data's distribution
D_{s}	$\mathcal{E}$	encoded data's distribution for an HE or FSE scheme when state is s (introduced in Sec. 2.6.1)

struction. Its goal is not necessarily to hide frequencies, but to hide the input's distribution by transforming it to have some target distribution. The paper used the Kolmogorov–Smirnov test (the maximum difference between the CDFs of the two distributions) to determine whether (i) the input data's distribution was indistinguishable from uniform after flattening, and (ii) the encoded data's distribution was indistinguishable from data with the target distribution (Gaussian, Zipf, or uniform). In their experiments, the data items all had 32 bits and the encodings had 64 bits.

Kerschbaum [43] presented a frequency-hiding OPE scheme that entirely forbids repetition of ciphertexts. However, it has large client-side storage requirements and, because of its order-preserving nature, is vulnerable to partial plaintext recovery attacks in a snapshot attack model [32]. The security notion used is indistinguishability under frequency-analyzing ordered chosen plaintext attack (IND-FA-OCPA). The adversary is tasked with distinguishing between encryptions of two equal-length sequences of plaintexts, not necessarily distinct, which have at least one common randomized order (ranking in which ties may be broken arbitrarily). The IND-FA-OCPA security notion captures the idea that the ciphertext leaks only the randomized order. It does not leak any frequency information, since each message and ciphertext value occurs exactly once. Roche *et al.* [72] introduced a partial order-preserving encoding scheme that uses the same security notion. Boneh *et al.*'s ORE scheme [13] is built from multilinear maps and the authors admit it is too inefficient for practical use. Most of these OPE/ORE schemes that attempt to hide frequencies are generally incomparable to ours since we do not require ciphertexts to be distinct. We purposely allow repetition, which enables us to achieve more flexible trade-offs between security and performance.

A different approach was used in Papadimitriou *et al.*'s splayed additively symmetric homomorphic encryption (SPLASHE) construction [65]. It hides frequencies while supporting aggregate operations such as `COUNT` and `SUM` by expanding each column into as many columns as there are possible values. Their enhanced SPLASHE construction addresses the attendant storage expansion by assigning individual columns to the "most frequent" values and grouping together the "least frequent" values in one column. To distinguish the less frequent values, a column of deterministically encrypted (DE) values is added. The frequencies of the "least frequent" values in this column are smoothed with a rudimentary padding technique. The threshold separating most frequent and least frequent values is chosen to ensure that there are enough records having their own columns so that their entries in the DE column can be used to equalize the counts of the least frequent values' DE values. SPLASHE

was designed for data analytics and in particular it does not support equality queries or joins. It also suffers from significant data expansion, about 10x for a real-world analytics database.

Another construction similar to ours is a secure order-preserving indexing (OPI) that supports efficient point and range queries while hiding frequencies [57]. One advantage of this scheme is that an equality query on the unencrypted data corresponds to one range query on the indices, while our schemes transform each equality query to multiple equality queries. OPI expands the plaintext domain to the ciphertext domain by assigning an interval of indices to each plaintext whose size is proportional to its frequency (much like we do with IBHE in Section 2.6.2). However, there is no formal security analysis of the OPI scheme nor suggestion about how to choose the size of the ciphertext domain.

Summary. Our work addresses a combination of properties that none of these previous schemes has: it allows equality queries, has a formal security analysis, and our concrete constructions have adjustable parameters to attain the desired balance of security and efficiency.

2.2 Definitions

Our goal is to design schemes that output ciphertexts whose frequencies are uniform, so even an adversary who knows the underlying plaintext frequencies cannot infer anything about the data.

In our formal definition of a frequency-smoothing scheme, the Setup algorithm accepts as input the data owner's estimate $\tilde{\mathsf{D}}$ of the messages' distribution and a *distribution adaptation parameter* Δ that indicates how "different" the estimated distribution $\tilde{\mathsf{D}}$ may be from the actual, unknown message distribution D. The choice of "difference" measure will depend on the particular FSE scheme and how it adapts to the message counts it observes and encrypts. For instance, the parameter Δ may be an upper bound on the Kolmogorov–Smirnov statistic or the statistical distance of the two distributions. In a sense, this parameter indicates how much uncertainty is associated with the initial estimated distribution $\tilde{\mathsf{D}}$; it indicates how conservatively a dynamic FSE scheme should allocate homophones. Regardless of what measure of difference is used, we assume that $\Delta = 0$ indicates complete confidence that $\tilde{\mathsf{D}} = \mathsf{D}$, in which case the scheme will be entirely non-adaptive, i.e., static.

Maintaining state allows an FSE scheme to handle initially unknown distributions ($\Delta \neq 0$): by updating the state as messages are encrypted, the scheme can allocate more homophones to the more frequently observed messages. Decryption involves accessing the updated state, and therefore the state must always contain enough information to decrypt any message encrypted with an earlier state. The state also makes explicit that encryption requires knowledge of the plaintext distribution, which the data owner will need to store in practice. Additionally, having a state allows some pre-computation on the message distribution to make encryption or decryption faster. Nevertheless, when the precise message distribution is known from the start or the scheme is static ($\Delta = 0$), the state does not need to be updated after running Setup and the following definitions simplify accordingly.

Recall that one of our assumptions is that the support of the distribution (i.e., the messages with non-zero probability) is always known. We also assume that messages are sampled independently from a fixed distribution. If the distribution changes over time, the estimated distribution $\tilde{\mathsf{D}}$ given as input to Setup would need to be replaced with a set of conditional distributions describing a stochastic process.

Definition 2.1. *A frequency-smoothing encryption (FSE) scheme* FSE *is a quadruple of algorithms* $\mathsf{FSE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *such that:*

- $\mathsf{Setup} : \{0,1\}^* \times \mathcal{D}_{\mathcal{M}} \times \{0,1\}^* \to \mathcal{S}$ *takes a security parameter* $\lambda \in \{0,1\}^*$*, a distribution* $\tilde{\mathsf{D}} \in \mathcal{D}_{\mathcal{M}}$*, and a distribution adaptation parameter* $\Delta \in \{0,1\}^*$ *as input and outputs a state* $\mathsf{s} \in \mathcal{S}$ *that includes a description of the distribution* $\tilde{\mathsf{D}}$ *and maybe other information.*
- $\mathsf{KeyGen} : \{0,1\}^* \to \mathcal{K}$ *takes a security parameter* $\lambda \in \{0,1\}^*$ *as input and outputs a secret key* $\mathsf{sk} \in \mathcal{K}$*.*
- $\mathsf{Encrypt} : \mathcal{K} \times \mathcal{M} \times \mathcal{S} \to \{\mathcal{C} \times \mathcal{S}\} \cup \{\perp\}$ *takes a key* $\mathsf{sk} \in \mathcal{K}$*, a message* $m \in \mathcal{M}$*, and a state* $\mathsf{s} \in \mathcal{S}$ *as input and outputs either a ciphertext* $c \in \mathcal{C}$ *and an updated state* $\mathsf{s}' \in \mathcal{S}$ *or* $\perp$*.*
- $\mathsf{Decrypt} : \mathcal{K} \times \mathcal{C} \times \mathcal{S} \to \mathcal{M} \cup \{\perp\}$ *takes a key* $\mathsf{sk} \in \mathcal{K}$*, a ciphertext* $c \in \mathcal{C}$*, and a state* $\mathsf{s} \in \mathcal{S}$ *as input and outputs either a message* $m \in \mathcal{M}$ *or* $\perp$*.*

Setup, KeyGen, and $\mathsf{Encrypt}$ are randomized algorithms, while $\mathsf{Decrypt}$ is deterministic. For a particular key sk, we call a state s' *attainable* from the state s if $\mathsf{s}' = \mathsf{s}$ or if there exists a finite sequence of messages $m_1, \ldots, m_n \in \mathcal{M}^n$ such that by defining $\mathsf{s}_0 \coloneqq \mathsf{s}$ and $(c_i, s_i) \leftarrow \mathsf{Encrypt}(\mathsf{sk}, m_i, \mathsf{s}_{i-1})$ for $i = 1, \ldots, n$, we get $\mathsf{s}_n = \mathsf{s}'$ with non-zero probability. A frequency-smoothing scheme is *correct* for a distribution $\tilde{\mathsf{D}}$ if for any initial state $\mathsf{s} \leftarrow \mathsf{Setup}(\lambda, \tilde{\mathsf{D}}, \Delta)$, any key $\mathsf{sk} \leftarrow \mathsf{KeyGen}(\lambda)$, any message $m \in \mathsf{supp}(\tilde{\mathsf{D}})$,

and any intermediate state s' attainable from s, if $(c, \mathsf{s}'') \leftarrow \mathsf{Encrypt}(\mathsf{sk}, m, \mathsf{s}')$, then for any later state s''' attainable from s'', $\mathsf{Decrypt}(\mathsf{sk}, c, \mathsf{s}''') = m$ with probability 1. Less formally, any message encrypted after initializing the state can be decrypted later, even if other messages are encrypted in the meantime, potentially updating the state.

For some fixed security parameter λ, distribution $\tilde{\mathsf{D}}$, and distribution adaptation parameter Δ, and any key sk output by $\mathsf{KeyGen}(\lambda)$ with non-zero probability, we let $\mathcal{H}^{\mathsf{FSE}}_{\mathsf{sk},\mathsf{s}}(m)$ be the set of all ciphertexts output by $\mathsf{Encrypt}(\mathsf{sk}, m, \mathsf{s}')$ such that s_0 is any state output by $\mathsf{Setup}(\lambda, \tilde{\mathsf{D}}, \Delta)$ with non-zero probability, s' is a state is attainable from s_0, and s is attainable from s'. Thus, $\mathcal{H}^{\mathsf{FSE}}_{\mathsf{sk},\mathsf{s}}(m)$ is the union of homophone sets of message m for any state that may have come before the state s. We also let $\mathcal{H}^{\mathsf{FSE}}_{\mathsf{sk},\mathsf{s}} \coloneqq \bigcup_{m \in \mathcal{M}} \mathcal{H}^{\mathsf{FSE}}_{\mathsf{sk},\mathsf{s}}(m)$ be the set of all possible encryptions (homophones) of any message with key sk for any state that may have come before s. We assume that the sizes of homophone sets are independent of the choice of $\mathsf{sk} \in \mathcal{K}$, so we may write $\left|\mathcal{H}^{\mathsf{FSE}}_{\mathsf{s}}(m)\right|$ for $\left|\mathcal{H}^{\mathsf{FSE}}_{\mathsf{sk},\mathsf{s}}(m)\right|$ and $\left|\mathcal{H}^{\mathsf{FSE}}_{\mathsf{s}}\right|$ for $\left|\mathcal{H}^{\mathsf{FSE}}_{\mathsf{sk},\mathsf{s}}\right|$. Two immediate corollaries of the correctness property are that $\mathcal{H}^{\mathsf{FSE}}_{\mathsf{sk},\mathsf{s}}(m) \subseteq \mathcal{H}^{\mathsf{FSE}}_{\mathsf{sk},\mathsf{s}'}(m)$ for any state s' attainable from s, and that $\mathcal{H}^{\mathsf{FSE}}_{\mathsf{sk},\mathsf{s}}(m_1)$ and $\mathcal{H}^{\mathsf{FSE}}_{\mathsf{sk},\mathsf{s}}(m_2)$ are disjoint unless $m_1 = m_2$, in which case $\mathcal{H}^{\mathsf{FSE}}_{\mathsf{sk},\mathsf{s}}(m_1) = \mathcal{H}^{\mathsf{FSE}}_{\mathsf{sk},\mathsf{s}}(m_2)$.

2.3 Using FSE

To use frequency-smoothing encryption in the intended setting—on outsourced data that is queryable—the set $\mathcal{H}^{\mathsf{FSE}}_{\mathsf{sk},\mathsf{s}}(m)$ must be easy to compute or describe for any message m given a state s and key sk. For example, this allows a SQL query containing an expression of the form `WHERE attribute = x` to be rewritten as `WHERE attribute IN (x1, x2, ...)`, where the `xi`'s compose the set of `x`'s homophones. This rewriting incurs a query blow-up: a single query for item x is converted into a more complex query for all of x's homophones. We will parameterize our FSE schemes so that this blow-up is manageable while still preventing frequency analysis attacks.

FSE also supports joins, which follows directly from the ability to support equality queries. A join on unencrypted data such as `FROM t1 JOIN t2 WHERE t1.a=t2.b` would instead be written as a `UNION` of join queries having the form `FROM t1 JOIN t2 WHERE t1.a IN (x1, x2, ...) AND t2.b IN (y1, y2, ...)`. There is a join query for each possible plaintext value; the `xi`'s compose its set of homophones in column `a` of table `t1` and the `yi`'s compose its set of homophones in column `b` of table `t2`.

FSE is compatible with range queries, since a range is simply a set of values that can be expanded to a larger set of homophones, but it does not natively maintain the ability to index data for efficient range queries. However, the specific constructions for FSE that follow can be adapted to use OPE, in which case range queries can be efficiently supported. In particular, a deterministic OPE scheme could simply replace DE in the HE-DE construction in Section 2.5.3. Further, the specific HE schemes we present in Section 2.6 do not rely on messages being ordered by frequency—we simply label them according to their frequencies for ease of exposition. Moreover, both schemes preserve numerical ordering.

The state s of an FSE scheme is stored by the data owner or in a proxy that transparently performs the encryption and decryption operations. The state s will typically include an accurate representation of the message distribution, and thus FSE schemes may not be appropriate for very large message spaces. We will evaluate the client-side storage requirements of our FSE schemes as we introduce them, but typically they are in the order of $r \cdot |\mathcal{M}|$ where r is a small parameter. Because each frequency $\mathsf{f}_{\tilde{\mathsf{D}}}(m)$ must be represented with finite precision, the set of possible distributions over the message space, $\mathcal{D}_{\mathcal{M}}$, is finite.

2.4 Security definitions

A frequency-smoothing scheme should do what its name implies: hide the frequency of messages from an attacker with access to a collection of ciphertexts, like a column in a database table. It should also be hard to learn anything about individual plaintexts from ciphertexts without the secret key. We formalize these notions of frequency-smoothing and privacy in two security games.

2.4.1 Frequency smoothing

The frequency-smoothing game $\mathsf{FSE{-}SMOOTH}$ (Figure 2.1) captures the requirement that ciphertexts do not leak any information about message frequencies, by making their distribution indistinguishable from uniform. In the $b = 0$ case of this game, the challenger uses an estimated distribution $\tilde{\mathsf{D}}$ (corresponding to a data owner's guess of its data's distribution) to initialize the state and then encrypts messages sampled according to the true distribution D. In the $b = 1$ case, the challenger samples ciphertexts uniformly at random with replacement from a set having the size of the homophone set if the static scheme were used ($\Delta = 0$) with the data's true distribution D. The adversary receives $\mathsf{n_r}$ ciphertexts, the distribution $\tilde{\mathsf{D}}$ that

the challenger uses to initialize the state when $b = 0$, its own estimate of the data's distribution $\hat{\mathsf{D}}$ (possibly different from $\tilde{\mathsf{D}}$), and the distribution adaptation parameter Δ. The adversary's goal is to distinguish these two cases. Informally, if it is able to distinguish the distribution of the $\mathsf{n_r}$ ciphertexts from uniform, then the message distribution must not have been properly smoothed by the FSE scheme.

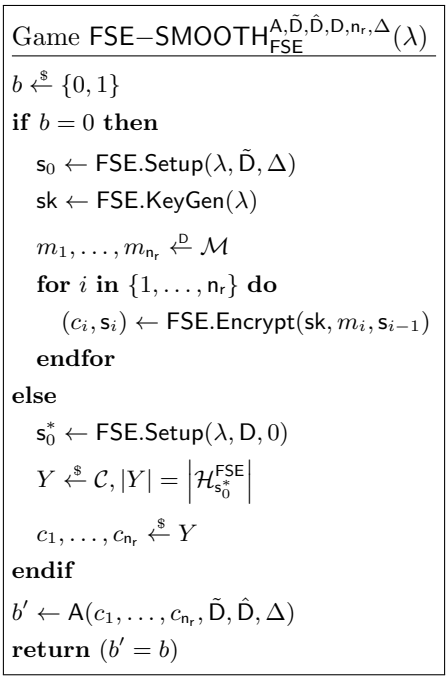


Figure 2.1: The frequency-smoothing game for an FSE scheme.

Definition 2.2. *Consider the game* FSE−SMOOTH *in Figure 2.1. The frequency-smoothing advantage of* A *against the FSE scheme* FSE *is*

$$\mathsf{Adv}^{\mathsf{smooth}}_{\mathsf{FSE}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta) = 2 \cdot \left| \mathrm{Prob}\left[\mathsf{FSE}{-}\mathsf{SMOOTH}^{\mathsf{A},\tilde{\mathsf{D}},\hat{\mathsf{D}},\mathsf{D},\mathsf{n_r},\Delta}_{\mathsf{FSE}}(\lambda) \Rightarrow 1\right] - \frac{1}{2} \right|.$$

Definition 2.3. *An FSE scheme* FSE *is* $(\alpha, t, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta)$-SMOOTH *if for all adversaries* A *running in time at most* t *and receiving at most* $\mathsf{n_r}$ *samples, it holds that*

$$\mathsf{Adv}^{\mathsf{smooth}}_{\mathsf{FSE}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta) \leq \alpha.$$

From the definition of the FSE−SMOOTH game, some necessary conditions are immediately obvious: first, for an FSE scheme to be FSE−SMOOTH for arbitrary $\tilde{\mathsf{D}}$ and D, the distribution adaptation parameter would need to be large and so would the total number of homophones—the latter would need to be about the number of samples, $\mathsf{n_r}$. Therefore, for efficient constructions, it makes sense to consider schemes

that are FSE−SMOOTH for classes of distributions D and $\tilde{\mathsf{D}}$ that are "close enough" according to the distribution adaptation parameter Δ.

Second, for a scheme to be FSE−SMOOTH for arbitrarily large $\mathsf{n_r}$, the size of every message's homophone set must be proportional to the frequency of the corresponding message according to D. This is a consequence of the distribution over the set of all homophones being indistinguishable from uniform and each homophone corresponding to exactly one message.

The FSE−SMOOTH security notion is comprehensive; it captures the possibility that the attacker has different information ($\hat{\mathsf{D}}$) about the messages' actual distribution (D) than the data owner used to initialize the state ($\tilde{\mathsf{D}}$). It also captures the possibility that the adversary has information about the data owner's estimate of the data's distribution ($\tilde{\mathsf{D}}$). In general, the adversary may not know exactly what distribution the data owner used to initialize the state, but we assume that it does—such an adversary is more powerful.

An important case is when the data's distribution is known by both the data owner and the attacker. In Section 2.6, we present FSE schemes that are provably secure when $\mathsf{D} = \tilde{\mathsf{D}} = \hat{\mathsf{D}}$, while in Section 2.7, we present results of an empirical analysis of FSE security when $\mathsf{D} = \tilde{\mathsf{D}} = \hat{\mathsf{D}}$ and compare it to the security of DE when $\hat{\mathsf{D}} = \mathsf{D}$ and $\hat{\mathsf{D}} \approx \mathsf{D}$.

2.4.2 Message privacy

It is not enough for an FSE scheme to hide the frequencies of the messages: even if the ciphertext distribution is uniform, the adversary could still be able to decrypt messages, e.g., if the ciphertexts leak information about message order. Thus, we also need a message privacy notion.

To obtain our message privacy definition, we adapt the deterministic privacy ("detPriv") security notion for DE schemes [74] to our setting. That definition is itself an adaptation of the indistinguishability-from-random-bits ("IND$") notion of security for a nonce-based symmetric encryption scheme [73]. It is also similar to the notion of message privacy we use for DE schemes in Section 2.5.2.

In the detPriv game [74], the adversary is tasked with distinguishing real encryptions of messages m of its choice from random bitstrings selected from the ciphertext space. Our FSE−PRIV game diverges from the detPriv game in two related ways. First, we restrict the adversary to requesting encryptions of messages sampled according to the distribution D, so the challenger can sample the messages on its behalf. This may seem

like a limitation of the adversary's power, but it reflects exactly the scenario we want to model, one in which encryption depends on the plaintext's distribution. Second, we allow the adversary to receive (potentially different) encryptions of the same message. In the deterministic setting, it was assumed without loss of generality that the adversary does not repeat any encryption queries since repeated encryptions would have revealed nothing new. In our setting, the encryption algorithm is probabilistic, so we allow repeated encryptions of m, but ensure they are either real encryptions or sampled from a randomly selected set Y_m of the appropriate size, $\left|\mathcal{H}^{\mathsf{FSE}}_{\mathsf{s}^*_0}(m)\right|$. For different messages m, these sets are disjoint in view of the correctness property of an encryption scheme.

In the $\mathsf{FSE}{-}\mathsf{PRIV}$ game in Figure 2.2, the challenger either initializes the state using the estimated distribution $\tilde{\mathsf{D}}$ and then encrypts messages sampled according to D, or it samples sets Y_m of the "right" size for each message m if the true distribution D had initially been known in the static scheme ($\Delta = 0$). Given $\mathsf{n_r}$ plaintext-ciphertext pairs, the distributions $\hat{\mathsf{D}}$ and $\tilde{\mathsf{D}}$, and the distribution adaptation parameter Δ, the adversary A must determine how the plaintext-ciphertext pairs were generated.

Definition 2.4. *Consider the message privacy game* $\mathsf{FSE}{-}\mathsf{PRIV}$ *in Figure 2.2. The message-privacy advantage of* A *against the FSE scheme* FSE *is*

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{FSE}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta) = 2 \cdot \left| \mathrm{Prob}\left[\mathsf{FSE}{-}\mathsf{PRIV}^{\mathsf{A},\tilde{\mathsf{D}},\hat{\mathsf{D}},\mathsf{D},\mathsf{n_r},\Delta}_{\mathsf{FSE}}(\lambda) \Rightarrow 1\right] - \frac{1}{2} \right|.$$

Definition 2.5. *An FSE scheme* FSE *is* $(\alpha, t, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta)$-$\mathsf{PRIV}$ *if for all adversaries* A *running in time at most* t *and receiving at most* $\mathsf{n_r}$ *plaintext-ciphertext pairs, it holds that* $\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{FSE}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta) \leq \alpha$.

From these definitions, some guidelines arise for creating efficient, secure schemes. First, since homophone set sizes can only increase, the initial homophone set sizes in the $b = 0$ case should be small to leave room to grow the sets corresponding to the most frequent messages. Making a set of homophones too big will only require that some of its members appear with low probability, so the sizes of the final homophone sets in the $b = 0$ case, $\left|\mathcal{H}^{\mathsf{FSE}}_{\mathsf{s}_{\mathsf{n_r}}}(m)\right|$, should be roughly equal to the sizes of the homophone sets in the static $b = 1$ case, $|Y_m| = \left|\mathcal{H}^{\mathsf{FSE}}_{\mathsf{s}^*_0}(m)\right|$.

Recall that in the smoothness game (Figure 2.1), the adversary sees only ciphertexts. Frequency smoothness enforces that the sizes of each message's homophone set must be proportional to that message's frequency or large enough that no ciphertexts are repeated. In the message privacy game (Figure 2.2), the adversary sees plaintext-ciphertext pairs. Message privacy enforces that there is no link between plaintexts and

Game $\mathsf{FSE}\text{-}\mathsf{PRIV}^{\mathsf{A},\tilde{\mathsf{D}},\hat{\mathsf{D}},\mathsf{D},\mathsf{n_r},\Delta}_{\mathsf{FSE}}(\lambda)$

$b \xleftarrow{\$} \{0,1\}$

$m_1,\ldots,m_{\mathsf{n_r}} \xleftarrow{\mathsf{D}} \mathcal{M}$

if $b = 0$ **then**

  $\mathsf{s}_0 \leftarrow \mathsf{FSE.Setup}(\lambda, \tilde{\mathsf{D}}, \Delta)$

  $\mathsf{sk} \leftarrow \mathsf{FSE.KeyGen}(\lambda)$

  for i **in** $\{1,\ldots,\mathsf{n_r}\}$ **do**

    $(c_i, \mathsf{s}_i) \leftarrow \mathsf{FSE.Encrypt}(\mathsf{sk}, m_i, \mathsf{s}_{i-1})$

  endfor

else

  $\mathsf{s}_0^* \leftarrow \mathsf{FSE.Setup}(\lambda, \mathsf{D}, 0)$

  $Y \xleftarrow{\$} \mathcal{C}, |Y| = \left|\mathcal{H}^{\mathsf{FSE}}_{\mathsf{s}_0^*}\right|$

  for i **in** $\{1,\ldots,\mathsf{n_r}\}$ **do**

    if $\exists\, j < i : m_i = m_j$ **do**

      $Y_{m_i} \coloneqq Y_{m_j}$

    else

      $Y_{m_i} \xleftarrow{\$} Y, |Y_{m_i}| = \left|\mathcal{H}^{\mathsf{FSE}}_{\mathsf{s}_0^*}(m_i)\right|$

      $Y \coloneqq Y \setminus Y_{m_i}$

    endif

    $c_i \xleftarrow{\$} Y_{m_i}$

  endfor

endif

$b' \leftarrow \mathsf{A}((m_1,c_1),\ldots,(m_{\mathsf{n_r}},c_{\mathsf{n_r}}), \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \Delta)$

return $(b' = b)$

Figure 2.2: The privacy game for an FSE scheme.

ciphertexts except what is necessary for correctness. Both conditions are necessary for a secure frequency-smoothing scheme. In the next section, we present constructions for FSE that reflect this two-part approach.

2.5 Building FSE from HE and DE

One approach to building an FSE scheme is to first probabilistically encode the messages in a way that smooths the plaintext distribution, then deterministically encrypt the encoded messages. In this section, we present such a two-part, modular construction that composes homophonic encoding (to smooth frequencies) with deterministic symmetric-key encryption (to provide privacy). Sections 2.5.1 and 2.5.2 present definitions for homophonic encoding and deterministic encryption schemes, while Section 2.5.3 describes how to compose them to get an FSE scheme.

2.5.1 Homophonic encoding

We consider stateful encoding schemes that are initially given an estimated distribution of the messages as input.

Definition 2.6. *A (stateful) homophonic encoding scheme* HE *is a triple of algorithms* $(\mathsf{Setup}, \mathsf{Encode}, \mathsf{Decode})$ *such that:*

- $\mathsf{Setup} : \{0,1\}^* \times \mathcal{D}_{\mathcal{M}} \times \{0,1\}^* \to \mathcal{S}$ *is a probabilistic algorithm that takes a configuration parameter* $\lambda \in \{0,1\}^*$*, an estimated distribution* $\tilde{\mathsf{D}}$ *over* $\mathcal{M}$*, and a distribution adaptation parameter* Δ *as input and outputs some state* $\mathsf{s} \in \mathcal{S}$ *that includes a description of the distribution* $\tilde{\mathsf{D}}$ *and any other scheme parameters.*
- $\mathsf{Encode} : \mathcal{M} \times \mathcal{S} \to \{\mathcal{E} \times \mathcal{S}\} \cup \{\perp\}$ *is a probabilistic algorithm that takes a message* $m \in \mathcal{M}$ *and a state* $\mathsf{s} \in \mathcal{S}$ *as input and outputs either an encoded message* $e \in \mathcal{E}$ *and an updated state* $\mathsf{s}' \in \mathcal{S}$*, or* $\perp$*.*
- $\mathsf{Decode} : \mathcal{E} \times \mathcal{S} \to \mathcal{M} \cup \{\perp\}$ *is a deterministic algorithm that takes an encoded message* $e \in \mathcal{E}$ *and a state* $\mathsf{s} \in \mathcal{S}$ *as input and outputs a message* $m \in \mathcal{M}$ *or* $\perp$*.*

We emphasize that all algorithms and parameters in a homophonic encoding scheme are keyless and therefore provide no message privacy.

For some fixed configuration parameter λ, distribution $\tilde{\mathsf{D}}$, and distribution adaptation parameter Δ, let $\mathcal{H}_{\mathsf{s}}^{\mathsf{HE}}(m)$ be the set of all possible encodings (homophones) of the message $m \in \mathcal{M}$ for any state up to the given state s. Also let $\mathcal{H}_{\mathsf{s}}^{\mathsf{HE}} := \bigcup_{m \in \mathcal{M}} \mathcal{H}_{\mathsf{s}}^{\mathsf{HE}}(m)$. In order to use HE for its intended purpose, we require that the set of homophones of a message is easy to compute or describe given a state. Again, call a state s' *attainable* from the state s if $\mathsf{s}' = \mathsf{s}$ or there exists some finite sequence of messages $m_1, \ldots, m_n \in \mathcal{M}^n$ such that setting $\mathsf{s}_0 := \mathsf{s}$ and letting $(e_i, \mathsf{s}_i) \leftarrow \mathsf{Encode}(m_i, \mathsf{s}_{i-1})$ for $i = 1, \ldots, n$, then we have $\mathsf{s}_n = \mathsf{s}'$ with non-zero probability. A homophonic encoding scheme is *correct* for a distribution $\tilde{\mathsf{D}} \in \mathcal{D}_{\mathcal{M}}$ if for all states s output by $\mathsf{Setup}(\lambda, \tilde{\mathsf{D}}, \Delta)$, any message $m \in \mathsf{supp}(\tilde{\mathsf{D}})$, and any state s' attainable from s, if $(e, \mathsf{s}'') \leftarrow \mathsf{Encode}(m, \mathsf{s}')$, then for any s''' attainable from s'', $\mathsf{Decode}(e, \mathsf{s}''') = m$ with probability 1. In particular, the correctness property requires that any two sets of homophones $\mathcal{H}_{\mathsf{s}}^{\mathsf{HE}}(m)$ and $\mathcal{H}_{\mathsf{s}}^{\mathsf{HE}}(m')$ are disjoint unless $m = m'$.

In Figure 2.3, we introduce a game $\mathsf{HE}{-}\mathsf{SMOOTH}$ for HE schemes that is similar to the $\mathsf{FSE}{-}\mathsf{SMOOTH}$ game (Figure 2.1). We also define the advantage of an adversary and the security of an HE scheme in a manner similar to the corresponding $\mathsf{FSE}{-}\mathsf{SMOOTH}$ definitions of the previous section. Note that in the $b = 1$ case of the $\mathsf{FSE}{-}\mathsf{SMOOTH}$ game, the adversary receives ciphertexts sampled uniformly at random from some set

of the right size, while in the $b = 1$ case of the HE−SMOOTH game, the adversary receives ciphertexts sampled uniformly at random from the *actual* set of homophones.

Game $\mathsf{HE{-}SMOOTH}^{\mathsf{A},\tilde{\mathsf{D}},\hat{\mathsf{D}},\mathsf{D},\mathsf{n_r},\Delta}_{\mathsf{HE}}(\lambda)$

$b \xleftarrow{\$} \{0,1\}$
if $b = 0$ **then**
  $\mathsf{s}_0 \leftarrow \mathsf{HE.Setup}(\lambda, \tilde{\mathsf{D}}, \Delta)$
  $m_1, \ldots, m_{\mathsf{n_r}} \xleftarrow{\mathsf{D}} \mathcal{M}$
  for i **in** $\{1, \ldots, \mathsf{n_r}\}$ **do**
    $(e_i, \mathsf{s}_i) \leftarrow \mathsf{HE.Encode}(m_i, \mathsf{s}_{i-1})$
  endfor
else
  $\mathsf{s}^*_0 \leftarrow \mathsf{HE.Setup}(\lambda, \mathsf{D}, 0)$
  $e_1, \ldots, e_{\mathsf{n_r}} \xleftarrow{\$} \mathcal{H}^{\mathsf{HE}}_{\mathsf{s}^*_0}$
endif
$b' \leftarrow \mathsf{A}(e_1, \ldots, e_{\mathsf{n_r}}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \Delta)$
return $(b' = b)$

Figure 2.3: The frequency-smoothing game for an HE scheme.

Definition 2.7. *Consider the game* HE−SMOOTH *in Figure 2.3. The frequency-smoothing advantage of* A *against the homophonic encoding scheme* HE *is*

$$\mathsf{Adv}^{\mathsf{smooth}}_{\mathsf{HE}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta) = 2 \cdot \left| \mathrm{Prob}\left[\mathsf{HE{-}SMOOTH}^{\mathsf{A},\tilde{\mathsf{D}},\hat{\mathsf{D}},\mathsf{D},\mathsf{n_r},\Delta}_{\mathsf{HE}}(\lambda) \Rightarrow 1\right] - \frac{1}{2} \right|.$$

Definition 2.8. *An HE scheme* HE *is* $(\alpha, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta)$-SMOOTH *if for* all *adversaries* A*, it holds that* $\mathsf{Adv}^{\mathsf{smooth}}_{\mathsf{HE}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta) \leq \alpha$.

HE smoothness resembles the Distribution-Transforming Encoder (DTE) security notion from Juels and Ristenpart's work on honey encryption schemes [40]. In that setting, distribution-specific encoders were used to construct encryption schemes that withstand brute-force attacks by yielding plausible plaintexts when decrypting a target ciphertext with incorrect keys. A DTE adversary's goal is to distinguish between single message-encoding pairs where either the message was sampled according to some given distribution, then encoded, or the encoding was first sampled uniformly at random, then the message obtained by decoding. This notion is tailored to their setting and is less suited to the snapshot inference attacks based on frequency analysis that we are considering. In our setting, indistinguishability of a series of samples from one of two distributions is more appropriate than indistinguishability of message-encoding pairs. Nevertheless, the two notions are equivalent in some cases—consider a static HE scheme where the adversary and data owner have perfect distributional knowledge ($\mathsf{D} = \tilde{\mathsf{D}} = \hat{\mathsf{D}}$). Encoded messages in the HE−SMOOTH game can then be

decoded by the adversary, yielding a multi-sample version of DTE security. Thus, in this case, HE−SMOOTH security implies DTE security, and when $\mathsf{n_r} = 1$, they are equivalent.

Our definition of HE smoothness allows the adversary to be computationally unbounded. Our specific HE schemes in Section 2.6 will achieve HE smoothness in this strong sense.

2.5.2 Deterministic encryption

Deterministic encryption is the second ingredient in our modular construction for FSE schemes. We include the standard definition here for completeness.

Definition 2.9. *A deterministic (secret-key) encryption (DE) scheme* DE *is a triple of algorithms* (KeyGen, Encrypt, Decrypt) *with associated sets* $\mathcal{K}$*,* $\mathcal{M}$*, and* $\mathcal{C}$ *such that:*

- $\mathsf{KeyGen} : \{0,1\}^* \to \mathcal{K}$ *is a probabilistic algorithm that takes a security parameter* λ *as input and outputs a secret key* $\mathsf{sk} \in \mathcal{K}$*.*
- $\mathsf{Encrypt} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ *is a deterministic algorithm that takes a secret key* $\mathsf{sk} \in \mathcal{K}$ *and a message* $m \in \mathcal{M}$ *as input, and outputs a ciphertext* $c \in \mathcal{C}$*.*
- $\mathsf{Decrypt} : \mathcal{K} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$ *is a deterministic algorithm that takes a key* $\mathsf{sk} \in \mathcal{K}$ *and a ciphertext* $c \in \mathcal{C}$ *as input and outputs a message* $m \in \mathcal{M}$ *or* $\perp$*.*

A deterministic encryption scheme is correct if $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{Encrypt}(\mathsf{sk}, m)) = m$ for all $m \in \mathcal{M}$ and all $\mathsf{sk} \in \mathcal{K}$. The security notion we choose to use for DE (Figure 2.4) is based on indistinguishability from random bits. Such definitions have already been used in the context of nonce-based symmetric encryption [73] and deterministic authenticated encryption (DAE) for key-wrapping [74]. The adversary adaptively queries an encryption oracle with messages and consistently receives either the corresponding ciphertext or a string of random bits that has the same length as the ciphertext. Without loss of generality, we assume the adversary does not repeat any queries to its encryption oracle. The adversary's goal is to determine whether the oracle is responding with real ciphertexts or random bitstrings. However, to make a definition that is well-suited to the potentially small message spaces we will encounter in our FSE schemes, we deviate from previous definitions in the literature: in the "random bits" case, we sample ciphertexts uniformly at random *without* replacement from a random ciphertext set $Y \subset \mathcal{C}$ of an appropriate size. This makes our definition closer to that of PRI (pseudorandom injection) security for DAE [74, Section 8], though we do not use its decryption oracle.

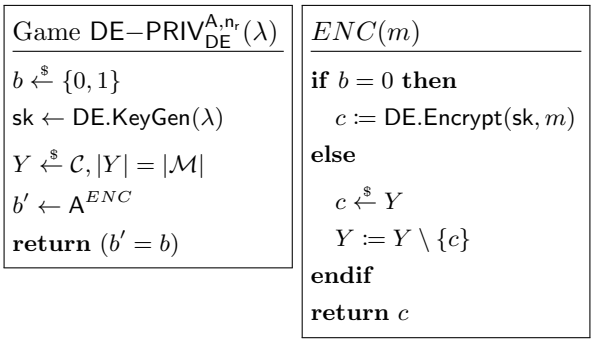


Figure 2.4: The message privacy game for a DE scheme. We assume that A does not repeat queries.

Definition 2.10. *Consider the deterministic privacy game in Figure 2.4. The message privacy advantage of* A *against the deterministic encryption scheme* DE *is*

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{DE}}(\mathsf{A}, \mathsf{n_r}) = 2 \cdot \left| \mathrm{Prob}\left[\mathsf{DE}{-}\mathsf{PRIV}^{\mathsf{A},\mathsf{n_r}}_{\mathsf{DE}}(\lambda) \Rightarrow 1\right] - \frac{1}{2} \right|.$$

Definition 2.11. *A DE scheme* DE *is said to be* $(\alpha, t, \mathsf{n_r})$*-private if for all adversaries* A *running in time at most* t *and making at most* $\mathsf{n_r}$ *encryption queries, it holds that* $\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{DE}}(\mathsf{A}, \mathsf{n_r}) \leq \alpha$.

A block cipher that is a pseudorandom permutation, like AES, meets this definition of privacy. For more flexibility in selecting the message space $\mathcal{M}$, one could pad short strings and use a block cipher, or use a small-domain PRP [61, 71] or a format-preserving encryption scheme [6, 8]. For larger domains, a wide-block PRP or a block cipher mode such as SIV could be used [74].

2.5.3 FSE from HE and DE

Now that we have defined stateful HE schemes, DE schemes, and their security, we are ready to present our modular construction for an FSE scheme.

Definition 2.12. *Let* $\mathsf{HE} = (\mathsf{Setup}, \mathsf{Encode}, \mathsf{Decode})$ *be a stateful homophonic encoding scheme with message space* $\mathcal{M}$ *and encoded message space* $\mathcal{E}$*. Let* $\mathsf{DE} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *be a deterministic encryption scheme with key space* $\mathcal{K}$*, message space* $\mathcal{E}$*, and ciphertext space* $\mathcal{C}$*. The* composed FSE scheme $(\mathsf{HE}, \mathsf{DE})$-$\mathsf{FSE}$ *is defined as follows.*

- Setup *takes a security parameter* $\lambda \in \{0,1\}^*$*, a distribution* $\mathsf{D} \in \mathcal{D}_{\mathcal{M}}$*, and a distribution adaptation parameter* $\Delta \in \{0,1\}^*$ *as input. It runs* $\mathsf{HE.Setup}(\lambda, \mathsf{D}, \Delta)$ *to obtain an initial state* s_0 *and outputs* s_0.

- KeyGen *takes a security parameter* $\lambda \in \{0,1\}^*$ *as input. It runs* $\mathsf{DE.KeyGen}(\lambda)$ *to obtain a key* $\mathsf{sk} \in \mathcal{K}$ *and outputs* sk.
- $\mathsf{Encrypt}$ *takes a key* $\mathsf{sk} \in \mathcal{K}$*, a message* $m \in \mathcal{M}$*, and a state* $\mathsf{s} \in \mathcal{S}$ *as input. It runs* $\mathsf{HE.Encode}(m, \mathsf{s})$ *to obtain* (e, s')*. It then runs* $\mathsf{DE.Encrypt}(\mathsf{sk}, e)$ *to obtain a ciphertext* $c \in \mathcal{C}$*. It outputs* (c, s').
- $\mathsf{Decrypt}$ *takes a key* $\mathsf{sk} \in \mathcal{K}$*, a ciphertext* $c \in \mathcal{C}$*, and a state* $\mathsf{s} \in \mathcal{S}$ *as input. It runs* $\mathsf{DE.Decrypt}(\mathsf{sk}, c)$ *to obtain a message* $e \in \mathcal{E}$ *or* $\perp$*. In the former case, it then runs* $\mathsf{HE.Decode}(e, \mathsf{s})$ *to obtain a message* $m \in \mathcal{M}$ *or* $\perp$*. It outputs* m*, or* $\perp$ *if it occurred in either step.*

When the HE scheme is frequency-smoothing and the DE scheme is message-private, the composed FSE scheme is both frequency-smoothing and private, in the senses of Definitions 2.3 and 2.5.

Theorem 2.13. *Suppose that* HE *is an* $(\alpha_{\mathsf{HE}}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta)$-$\mathsf{SMOOTH}$ *homophonic encoding scheme on* $(\mathcal{M}, \mathcal{E}, \mathcal{S})$ *for some* $\tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D} \in \mathcal{D}_{\mathcal{M}}$ *and that* DE *is an* $(\alpha_{\mathsf{DE}}, t + t_{\mathsf{HE.Setup}} + \mathsf{n_r} \cdot (t_{\mathsf{HE.Encode}} + t_{\mathsf{HE.Decode}}), \mathsf{n_r})$-$\mathsf{PRIV}$ *deterministic encryption scheme on* $(\mathcal{K}, \mathcal{E}, \mathcal{C})$*. Then the FSE scheme* $(\mathsf{HE}, \mathsf{DE})$-$\mathsf{FSE}$ *is*

- $(\alpha_{\mathsf{HE}} + \alpha_{\mathsf{DE}}, t, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta)$-$\mathsf{SMOOTH}$*, and*
- $(\alpha_{\mathsf{HE}} + \alpha_{\mathsf{DE}}, t, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta)$-$\mathsf{PRIV}$*,*

where t_{X} *is the time necessary to run algorithm* X.

Proof. First, consider smoothness of the composed FSE scheme. We prove that $(\mathsf{HE}, \mathsf{DE})$-$\mathsf{FSE}$ is smooth with the given parameters using the sequence of games illustrated in Figure 2.5. The transitions between successive games are based on indistinguishability.

Let A be any SMOOTH adversary for $(\mathsf{HE}, \mathsf{DE})$-$\mathsf{FSE}$ that runs in time at most t, and let Game 0 be the $\mathsf{FSE{-}SMOOTH}$ game, as in Figure 2.1. When $b = 0$, the ciphertexts are obtained by sampling messages m_i from $\mathcal{M}$ according to D, encoding them using $\tilde{\mathsf{D}}$ to initialize the state, and then encrypting them. When $b = 1$, the ciphertexts are chosen uniformly at random from a subset of $\mathcal{C}$ of the correct size (equal to the number of FSE homophones of each message).

Let Game 1 be the same as Game 0 except when $b = 0$: the ciphertexts are obtained by first sampling $\mathsf{n_r}$ encodings e_i uniformly at random from the set of HE homophones, and then encrypting them with DE.

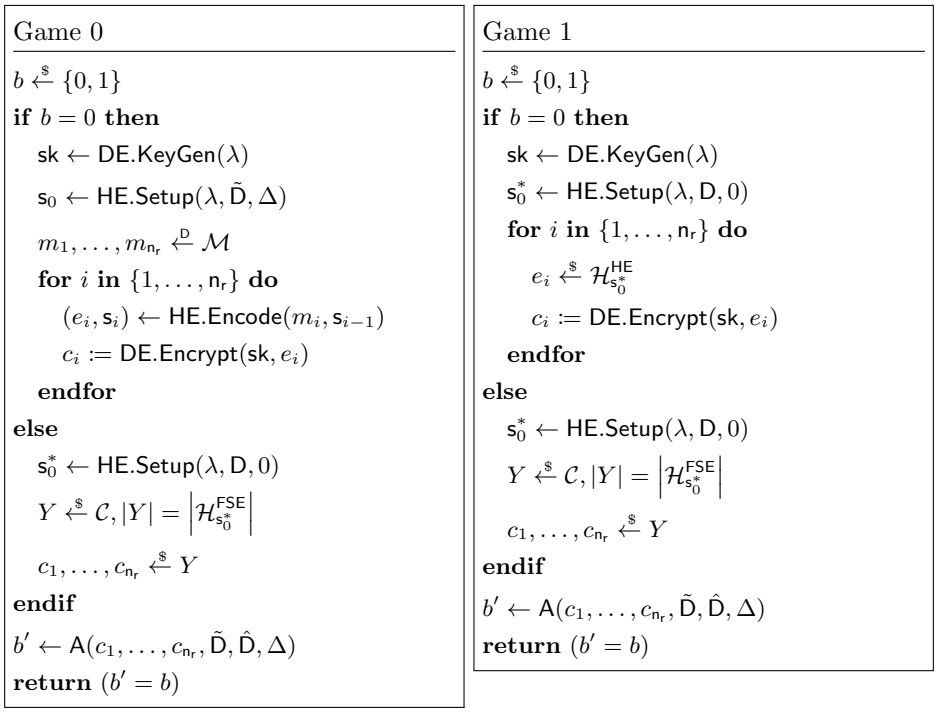


Figure 2.5: Sequence of games in the proof of smoothness of an $(\mathsf{HE}, \mathsf{DE})$-FSE scheme (continued on next page).

Consider the following $(\alpha', \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta)$-SMOOTH adversary A' for HE, which will distinguish games 0 and 1. A' receives $(e_1, \ldots, e_{\mathsf{n_r}}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \Delta)$ and flips a coin $b \in \{0,1\}$. If $b = 0$, it runs $\mathsf{DE.KeyGen}(\lambda)$ to generate a secret key and encrypts the e_i's with it, resulting in c_i's. If $b = 1$, it runs $\mathsf{HE.Setup}(\lambda, \mathsf{D}, 0)$ to generate an initial state s_0^* and samples $\mathsf{n_r}$ c_i's uniformly at random from a subset of $\mathcal{C}$ whose size is $\left|\mathcal{H}_{\mathsf{s}_0^*}^{\mathsf{FSE}}\right|$. It then gives the c_i's, $\tilde{\mathsf{D}}$, $\hat{\mathsf{D}}$, and Δ to A, which returns a bit b'. If $b' = b$, then A' outputs 1. Otherwise, it outputs 0. By definition, the advantage of A' is the absolute difference in the probabilities that A' outputs 1 when its input was real encodings and when its input was uniformly sampled encodings. If A' received real encodings, then A is playing Game 0. If A' received uniformly sampled encodings, then A is playing Game 1. Therefore,

$$\begin{aligned}\mathsf{Adv}_{\mathsf{HE}}^{\mathsf{smooth}}(\mathsf{A}', \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta) = &|\mathsf{Adv}_{\mathsf{FSE}}^{\mathrm{game0}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta) \\ &- \mathsf{Adv}_{\mathsf{FSE}}^{\mathrm{game1}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta)|.\end{aligned}$$

Since HE is $(\alpha_{\mathsf{HE}}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta)$-SMOOTH for adversaries with unbounded runtime, we have

$$|\mathsf{Adv}_{\mathsf{FSE}}^{\mathrm{game0}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta) - \mathsf{Adv}_{\mathsf{FSE}}^{\mathrm{game1}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta)| < \alpha_{\mathsf{HE}}.$$

Game 2

$b \xleftarrow{\$} \{0,1\}$
if $b = 0$ **then**
  $\mathsf{s}_0^* \leftarrow \mathsf{HE.Setup}(\lambda, \mathsf{D}, 0)$
  $Y \xleftarrow{\$} \mathcal{C}, |Y| = \left|\mathcal{H}_{\mathsf{s}_0^*}^{\mathsf{FSE}}\right|$
  for i **in** $\{1, \ldots, \mathsf{n_r}\}$ **do**
    $e_i \xleftarrow{\$} \mathcal{H}_{\mathsf{s}_0^*}^{\mathsf{HE}}$
    if $\exists\ j < i : e_i = e_j$ **do**
      $c_i \coloneqq c_j$
    else
      $c_i \xleftarrow{\$} Y$
      $Y \coloneqq Y \setminus \{c_i\}$
    endif
  endfor
else
  $\mathsf{s}_0^* \leftarrow \mathsf{HE.Setup}(\lambda, \mathsf{D}, 0)$
  $Y \xleftarrow{\$} \mathcal{C}, |Y| = \left|\mathcal{H}_{\mathsf{s}_0^*}^{\mathsf{FSE}}\right|$
  $c_1, \ldots, c_{\mathsf{n_r}} \xleftarrow{\$} Y$
endif
$b' \leftarrow \mathsf{A}(c_1, \ldots, c_{\mathsf{n_r}}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \Delta)$
return $(b' = b)$

Game 3

$b \xleftarrow{\$} \{0,1\}$
if $b = 0$ **then**
  $\mathsf{s}_0^* \leftarrow \mathsf{HE.Setup}(\lambda, \mathsf{D}, 0)$
  $Y \xleftarrow{\$} \mathcal{C}, |Y| = \left|\mathcal{H}_{\mathsf{s}_0^*}^{\mathsf{FSE}}\right|$
  $c_1, \ldots, c_{\mathsf{n_r}} \xleftarrow{\$} Y$
else
  $\mathsf{s}_0^* \leftarrow \mathsf{HE.Setup}(\lambda, \mathsf{D}, 0)$
  $Y \xleftarrow{\$} \mathcal{C}, |Y| = \left|\mathcal{H}_{\mathsf{s}_0^*}^{\mathsf{FSE}}\right|$
  $c_1, \ldots, c_{\mathsf{n_r}} \xleftarrow{\$} Y$
endif
$b' \leftarrow \mathsf{A}(c_1, \ldots, c_{\mathsf{n_r}}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \Delta)$
return $(b' = b)$

Figure 2.5: Sequence of games in the proof of smoothness of an $(\mathsf{HE}, \mathsf{DE})$-$\mathsf{FSE}$ scheme (continued from previous page).

Next, let Game 2 be the same as Game 1 except when $b = 0$, where the $\mathsf{n_r}$ ciphertexts are chosen from a subset of $\mathcal{C}$ of the right size, with repetitions according to the pattern of repetitions in the randomly selected e_i (but otherwise being sampled without replacement, as in the $b = 1$ case of the $\mathsf{DE}{-}\mathsf{PRIV}$ game, cf. Figure 2.4). We can again build an adversary A''—this time for $\mathsf{DE}{-}\mathsf{PRIV}$—that interpolates between Games 1 and 2 and has advantage

$$\mathsf{Adv}_{\mathsf{DE}}^{\mathsf{priv}}(\mathsf{A}'', \mathsf{n_r}) = \left|\mathsf{Adv}_{\mathsf{FSE}}^{\mathrm{game1}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta) - \mathsf{Adv}_{\mathsf{FSE}}^{\mathrm{game2}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta)\right|.$$

A'' flips a coin b and either runs $\mathsf{HE.Setup}(\lambda, \mathsf{D}, 0)$ to get an initial state s_0^*, uniformly samples $\mathsf{n_r}$ encoded messages e_i from $\mathcal{H}_{\mathsf{s}_0^*}^{\mathsf{HE}}$, and queries its ENC oracle with the e_i (avoiding repeated queries to ENC when repeated e_i are encountered), or uniformly samples $\mathsf{n_r}$ ciphertexts from a subset of $\mathcal{C}$ having size $\left|\mathcal{H}_{\mathsf{s}_0^*}^{\mathsf{FSE}}\right|$. It then runs A on these $\mathsf{n_r}$ ciphertexts, $\tilde{\mathsf{D}}$, $\hat{\mathsf{D}}$, and Δ, and outputs 1 if A's output b' equals b. Its running time is therefore the time to run A, $t_{\mathsf{HE.Setup}}$, the time to sample $\mathsf{n_r}$ messages (which we assume is less than $\mathsf{n_r} \cdot t_{\mathsf{HE.Encode}}$), and the time it takes to query its oracle (which we

assume is instantaneous). Since DE is $(\alpha_{\mathsf{DE}}, t + t_{\mathsf{HE.Setup}} + \mathsf{n_r} \cdot t_{\mathsf{HE.Encode}}, \mathsf{n_r})$-$\mathsf{PRIV}$,

$$\left|\mathsf{Adv}^{\text{game1}}_{\mathsf{FSE}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta) - \mathsf{Adv}^{\text{game2}}_{\mathsf{FSE}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta)\right| < \alpha_{\mathsf{DE}}.$$

Finally, we consider Game 3. In the $b = 0$ case of this game, we now sample the c_i's with replacement from a subset of $\mathcal{C}$ of the right size, no longer relying on the e_i's, which were sampled from a set of the same size, to dictate repetitions in the c_i's. It is straightforward to see that the distribution on the c_i's is the same in Game 2 and in Game 3. Hence

$$\left|\mathsf{Adv}^{\text{game2}}_{\mathsf{FSE}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta) - \mathsf{Adv}^{\text{game3}}_{\mathsf{FSE}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta)\right| = 0.$$

Finally, since $\left|\mathcal{H}^{\mathsf{FSE}}_{\mathsf{s}_0^*}\right| = \left|\mathcal{H}^{\mathsf{HE}}_{\mathsf{s}_0^*}\right|$, the $b = 0$ and $b = 1$ cases of Game 3 are identical, so $\mathsf{Adv}^{\text{game3}}_{\mathsf{FSE}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta) = 0$. We therefore have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{smooth}}_{\mathsf{FSE}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta) &= \mathsf{Adv}^{\text{game0}}_{\mathsf{FSE}}(\mathsf{A}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta) \\ &< \alpha_{\mathsf{HE}} + \alpha_{\mathsf{DE}}\end{aligned}$$

for any $\mathsf{FSE}{-}\mathsf{SMOOTH}$ adversary A running in time at most t.

Next, consider message privacy of the composed scheme. We prove via reduction that FSE is $(\alpha_{\mathsf{HE}} + \alpha_{\mathsf{DE}}, t, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta)$-$\mathsf{PRIV}$ by showing that if HE is $(\alpha_{\mathsf{HE}}, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta)$-$\mathsf{HE}{-}\mathsf{SMOOTH}$ and there is an $(\alpha, t, \tilde{\mathsf{D}}, \hat{\mathsf{D}}, \mathsf{D}, \mathsf{n_r}, \Delta)$-$\mathsf{PRIV}$ adversary $\mathsf{A}_{\mathsf{FSE}}$ for FSE, then there is also an $(\alpha - \alpha_{\mathsf{HE}}, t + t_{\mathsf{HE.Setup}} + \mathsf{n_r} \cdot (t_{\mathsf{HE.Decode}} + t_{\mathsf{HE.Encode}}), \mathsf{n_r})$-$\mathsf{PRIV}$ adversary A_{DE} for DE.

A_{DE} can query its provided encryption oracle ENC_{DE} at most $\mathsf{n_r}$ times (without repetition), while it must simulate encrypting $\mathsf{n_r}$ messages sampled according to D (with repetition) for $\mathsf{A}_{\mathsf{FSE}}$. First, A_{DE} initializes the homophonic encoding scheme HE: it runs $\mathsf{HE.Setup}(\lambda, \mathsf{D}, 0)$ to generate a state s_0^*. It samples $\mathsf{n_r}$ encodings e_i uniformly at random with replacement from $\mathcal{H}^{\mathsf{HE}}_{\mathsf{s}_0^*}$. It decodes these e_i's to obtain the messages m_i. That is, for $i = 1$ to $\mathsf{n_r}$, it sets $m_i \coloneqq \mathsf{HE.Decode}(e_i, \mathsf{s}_0^*)$. Next, it queries ENC_{DE} with each of the distinct encodings e_i to obtain $c_1, \ldots, c_{\mathsf{n_r}}$. It provides $\mathsf{A}_{\mathsf{FSE}}$ with the distributions $\tilde{\mathsf{D}}$ and $\hat{\mathsf{D}}$, the distribution adaptation parameter Δ, and the $\mathsf{n_r}$ plaintext-ciphertext pairs $((m_1, c_1), \ldots, (m_{\mathsf{n_r}}, c_{\mathsf{n_r}}))$. Eventually, $\mathsf{A}_{\mathsf{FSE}}$ outputs a bit b'. A_{DE} then outputs the same bit.

Note that $\mathsf{A}_{\mathsf{FSE}}$'s view is exactly the same as in the $\mathsf{FSE}{-}\mathsf{PRIV}$ game in Figure 2.2. If ENC_{DE} is operating with $b_{\mathsf{DE}} = 0$ (real ciphertexts), then A_{DE} is perfectly simulating

the $b = 0$ case for $\mathsf{A}_{\mathsf{FSE}}$ since, by the HE−SMOOTH property, encodings sampled uniformly at random from $\mathcal{H}^{\mathsf{HE}}_{\mathsf{s}_0^*}$ have the same distribution as if they were encodings of messages sampled according to D, with an initial state determined by $\tilde{\mathsf{D}}$.

If ENC_{DE} is operating with $b_{\mathsf{DE}} = 1$ (random bitstrings without replacement), then A_{DE} is perfectly simulating the $b = 1$ case for $\mathsf{A}_{\mathsf{FSE}}$. By the HE−SMOOTH property, the distribution of encodings of messages sampled according to D is uniform on the set of all homophones $\mathcal{H}^{\mathsf{HE}}_{\mathsf{s}_0^*}$. Since this set of homophones is partitioned into the sets of individual messages' homophones, the distribution on the latter is thus uniform as well. Hence, as required, each message's encoding (and thus its ciphertext) is chosen uniformly at random from a set of the correct size with replacement. Therefore, A_{DE}'s advantage is at least $\mathsf{A}_{\mathsf{FSE}}$'s advantage less the probability that the HE encodings were distinguishable: $\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{DE}}(\mathsf{A}_{\mathsf{DE}}, \mathsf{n}_\mathsf{r}) > \alpha - \alpha_{\mathsf{HE}}$. The running time of A_{DE} is at most the time to run $\mathsf{A}_{\mathsf{FSE}}$, $t_{\mathsf{HE.Setup}}$, sample n_r values from $\mathcal{H}^{\mathsf{HE}}_{\mathsf{s}_0^*}$ (which we again assume is less than $\mathsf{n}_\mathsf{r} \cdot t_{\mathsf{HE.Encode}}$), decode n_r items, and make at most n_r queries to its encryption oracle (which we assume is instantaneous), achieving the required bounds. □

The modularity of the composed approach provides flexibility in choosing component schemes. However, the separate decoding and decryption steps are not particularly conceptually elegant.

2.6 Some static HE schemes

In this section, we narrow our focus to the case in which the data's actual distribution is known to both the data owner and the adversary ($\tilde{\mathsf{D}} = \hat{\mathsf{D}} = \mathsf{D}$) and the homophonic encoding scheme is *static* ($\Delta = 0$). We will write $\mathsf{Adv}^{\mathsf{smooth}}_{\mathsf{HE}}(\mathsf{A}, \mathsf{D}, \mathsf{n}_\mathsf{r})$ for the adversary's advantage in this case.

We begin with a general result about an adversary's smoothness advantage against an HE scheme. Then, we present two concrete homophonic encoding schemes. We will prove the smoothness of both schemes using the general bound we now develop.

2.6.1 Bounding an HE−SMOOTH adversary's advantage

When the distribution is public and the HE scheme is static, we can re-interpret the HE−SMOOTH game from Figure 2.3 in terms of distribution over the encoded message space $\mathcal{E}$ that results from sampling messages according to D and encoding them. Let D_s be this distribution—for a static HE scheme, it depends solely on the initial state s output by $\mathsf{Setup}(\lambda, \mathsf{D})$. (For an arbitrary homophonic encoding scheme,

the distribution over the encoding space will involve a stochastic process.) Since a message m's homophone is chosen uniformly at random, each of its homophones e will have frequency $\mathsf{f}_{\mathsf{D}_s}(e) = \mathsf{f}_{\mathsf{D}}(m)/\left|\mathcal{H}^{\mathsf{HE}}(m)\right|$.

The adversary must distinguish receiving $\mathsf{n_r}$ samples drawn according to $\mathsf{D_s}$ and $\mathsf{n_r}$ samples drawn according to the uniform distribution over the set of homophones. We bound an $\mathsf{HE{-}SMOOTH}$ adversary's advantage using Baignères, Junod, and Vaudenay's statistical framework for analyzing distinguishers [5]. The result we use, and will restate, shows that the error probability of an optimal distinguisher given a number of samples from two close distributions D_0 and D_1 can be bounded in terms of the *Kullback–Leibler (KL) divergence* of D_0 with respect to D_1, which is defined as

$$\mathrm{KL}\left(\mathsf{D}_0, \mathsf{D}_1\right) \coloneqq \sum_{m \in \mathcal{M}} \mathsf{f}_{\mathsf{D}_0}(m) \cdot \log \frac{\mathsf{f}_{\mathsf{D}_0}(m)}{\mathsf{f}_{\mathsf{D}_1}(m)}$$

for two distributions D_0 and D_1 having support $\mathcal{M}$. In particular, when D_1 is the uniform distribution $\mathsf{U}_{|\mathcal{M}|}$ over $\mathcal{M}$, we can write the KL divergence in terms of D_0's Shannon entropy, $H(\mathsf{D}_0)$:

$$\begin{aligned}
\mathrm{KL}\left(\mathsf{D}_0, \mathsf{D}_1\right) &= \sum_{m \in \mathcal{M}} \mathsf{f}_{\mathsf{D}_0}(m) \cdot \log \frac{\mathsf{f}_{\mathsf{D}_0}(m)}{\mathsf{f}_{\mathsf{D}_1}(m)} \\
&= \sum_{m \in \mathcal{M}} \mathsf{f}_{\mathsf{D}_0}(m) \cdot \left(\log(\mathsf{f}_{\mathsf{D}_0}(m)) + \log|\mathcal{M}|\right) \\
&= \log|\mathcal{M}| + \sum_{m \in \mathcal{M}} \mathsf{f}_{\mathsf{D}_0}(m) \cdot \log(\mathsf{f}_{\mathsf{D}_0}(m)) \\
&= \log|\mathcal{M}| - H(\mathsf{D}_0).
\end{aligned}$$

Therefore, the Kullback–Leibler divergence of D_0 from the uniform distribution is the natural log of the support's size less the Shannon entropy of D_0 in nats (the "natural unit of information").

We now restate Baignères, Junod, and Vaudenay's result in the following lemma.

Lemma [5, Theorem 6]. *Let D_0 and D_1 be distributions over a set $\mathcal{M}$ having the same support, and suppose that $\mathsf{f}_{\mathsf{D}_0}(m)$ and $\mathsf{f}_{\mathsf{D}_1}(m)$ are close for all $m \in \mathcal{M}$. Then, the total error of an optimal distinguisher limited to $\mathsf{n_r}$ samples, for large $\mathsf{n_r}$, is approximately*

$$\Phi\left(-\sqrt{\frac{\mathsf{n_r} \cdot \mathrm{KL}\left(\mathsf{D}_0, \mathsf{D}_1\right)}{2}}\right)$$

where $\Phi(\cdot)$ is the cdf of the standard normal distribution.

The requirement that $\mathsf{f}_{\mathsf{D}_0}(m)$ and $\mathsf{f}_{\mathsf{D}_1}(m)$ be close for all $m \in \mathcal{M}$ allows a Taylor series expansion to be truncated at the second order with only small error [5, Prop. 5]. In our setting, this requirement will not be a restriction since it is necessary for smoothness anyway.

Applying the Lemma of Baignères, Junod, and Vaudenay yields the following theorem.

Theorem 2.14. *Let* HE *be a static homophonic encoding scheme with message space* $\mathcal{M}$ *and encoded message space* $\mathcal{E}$*. Let* $\mathsf{D} \in \mathcal{D}_{\mathcal{M}}$ *be a public distribution over* $\mathcal{M}$*, and let* D_s *be the distribution over* $\mathcal{E}$ *resulting from sampling messages according to* D *and encoding them with a state* s *output by* $\mathsf{HE.Setup}(\lambda, \mathsf{D})$*. If* $\mathsf{f}_{\mathsf{D}_\mathsf{s}}(e)$ *is close to* $1/\left|\mathcal{H}_\mathsf{s}^{\mathsf{HE}}\right|$ *for all encodings* $e \in \mathcal{H}_\mathsf{s}^{\mathsf{HE}}$*, then, for any* $\mathsf{HE}{-}\mathsf{SMOOTH}$ *adversary* A*, and for sufficiently large* $\mathsf{n_r}$*,*

$$\mathsf{Adv}^{\mathsf{smooth}}_{\mathsf{HE}}(\mathsf{A}, \mathsf{D}, \mathsf{n_r}) \leq \left| \frac{1}{2} - \Phi\left(-\sqrt{\frac{\mathsf{n_r} \cdot \mathrm{KL}\left(\mathsf{D}_\mathsf{s}, \mathsf{U}_{|\mathcal{H}_\mathsf{s}^{\mathsf{HE}}|}\right)}{2}} \right) \right|$$

where $\Phi(\cdot)$ *is the cdf of the standard normal distribution.*

Proof. The proof follows directly from Baignères, Junod, and Vaudenay's result applied to the distribution over encodings and the definition of advantage in terms of total error. □

Theorem 2.14 applies even to computationally unbounded adversaries. Recall that the cdf of the standard normal distribution, Φ, equals $1/2$ at 0, so the closer $\mathsf{n_r} \cdot \mathrm{KL}\left(\mathsf{D}_\mathsf{s}, \mathsf{U}_{|\mathcal{H}_\mathsf{s}^{\mathsf{HE}}|}\right)$ is to 0, the smaller is any $\mathsf{HE}{-}\mathsf{SMOOTH}$ adversary's advantage. Hence, in order to establish a smoothness bound on any particular static scheme HE, it is sufficient to prove bounds on $\mathrm{KL}\left(\mathsf{D}_\mathsf{s}, \mathsf{U}_{|\mathcal{H}_\mathsf{s}^{\mathsf{HE}}|}\right)$. Finally, using the fact that the pdf of a standard normal distribution peaks at 0 with value $1/\sqrt{2\pi}$, we see that a good upper bound on $\mathsf{Adv}^{\mathsf{smooth}}_{\mathsf{HE}}(\mathsf{A}, \mathsf{D}, \mathsf{n_r})$ is given by

$$\mathsf{Adv}^{\mathsf{smooth}}_{\mathsf{HE}}(\mathsf{A}, \mathsf{D}, \mathsf{n_r}) \leq \frac{1}{2\sqrt{\pi}} \cdot \sqrt{\mathsf{n_r} \cdot \mathrm{KL}\left(\mathsf{D}_\mathsf{s}, \mathsf{U}_{|\mathcal{H}_\mathsf{s}^{\mathsf{HE}}|}\right)} \,. \tag{2.1}$$

This suggests that to make the adversary's advantage very small, we need $\mathrm{KL}\left(\mathsf{D}_\mathsf{s}, \mathsf{U}_{|\mathcal{H}_\mathsf{s}^{\mathsf{HE}}|}\right) \ll 1/\mathsf{n_r}$.

We now turn to the analysis of two specific static encoding schemes. For convenience in what follows, we assume that $\mathcal{M} \subseteq \{0,1\}^n$.

2.6.2 Interval-based homophonic encoding

Informally, r-bit interval-based homophonic encoding (IBHE) encodes messages as r-bit strings by partitioning $\{0,1\}^r$ according to the message distribution D: message m will be allocated an interval of about $\mathsf{f}_\mathsf{D}(m) \cdot 2^r$ bitstrings. The idea is that each r-bit homophone will be used roughly equally often when encoding messages sampled from D, since the number of homophones per message is proportional to its frequency.

One way (others are possible) of partitioning the set of r-bit strings according to D is as follows. Suppose, without loss of generality, that the messages in $\mathsf{supp}(\mathsf{D}) = \{m_1, m_2, \ldots\}$ are numbered by increasing frequency according to D. Now, consider the cumulative distribution F_D. To simplify notation, let there be a 0th message m_0 with $\mathsf{F}_\mathsf{D}(m_0) \coloneqq 0$. Then, the homophone set of any message $m_i \in \mathsf{supp}(\mathsf{D})$ is defined to be

$$\left\{ \lfloor 2^r \cdot \mathsf{F}_\mathsf{D}(m_{i-1}) \rceil, \ldots, \lfloor 2^r \cdot \mathsf{F}_\mathsf{D}(m_i) \rceil - 1 \right\},$$

where integers in this set are represented with r bits, and $\lfloor x \rceil$ represents the integer nearest to x. This interval has size approximately $2^r \cdot \mathsf{f}_\mathsf{D}(m_i)$, as desired. The encoding algorithm for IBHE simply selects an encoding e of m_i uniformly at random from the relevant interval.

It is clear that the encoding bitlength r must be at least $\log_2 |\mathsf{supp}(\mathsf{D})|$ so each message can have at least one possible encoding. In addition, r must be big enough so that each message is assigned a non-empty interval using this partitioning technique. The following straightforward proposition relates a message distribution D, an IBHE encoding length r, and a lower bound on the number of homophones h each message has.

Proposition 2.15. *Let D be a distribution over the message space $\mathcal{M}$, with messages in $\mathsf{supp}(\mathsf{D}) = \{m_1, m_2 \ldots\}$ numbered by increasing frequency, and let $h \geq 1$ be a positive integer. Then, when encoded with r-bit IBHE, every message $m_i \in \mathsf{supp}(\mathsf{D})$ has at least h homophones if and only if $r \geq r_{\textit{min-}h}$, where*

$$r_{\textit{min-}h} \coloneqq \left\lceil \max_{1 \leq i \leq |\mathsf{supp}(\mathsf{D})|} \log_2 \frac{i \cdot h - 0.5}{\mathsf{F}_\mathsf{D}(m_i)} \right\rceil.$$

Proof. Let ℓ_i and r_i represent the left and right endpoints (inclusive) of message m_i's homophone set: $\ell_i \coloneqq \lfloor 2^r \cdot \mathsf{F}_\mathsf{D}(m_{i-1}) \rceil$ and $r_i \coloneqq \lfloor 2^r \cdot \mathsf{F}_\mathsf{D}(m_i) \rceil - 1$, so the size of this set is $\left|\mathcal{H}^{\mathsf{HE}}(m_i)\right| = r_i - \ell_i + 1$. By definition, $\ell_1 = 0$ and $\ell_i = r_{i-1} + 1$ for $i = 2, \ldots, |\mathsf{supp}(\mathsf{D})|$.

Suppose every message in $\mathsf{supp}(\mathsf{D})$ has at least h homophones. This happens if and only if, for each i from 1 to $|\mathsf{supp}(\mathsf{D})|$, we have

$$\begin{aligned} & r_i \geq i \cdot h - 1 \\ \Leftrightarrow \quad & \lfloor 2^r \cdot \mathsf{F}_\mathsf{D}(m_i) \rceil - 1 \geq i \cdot h - 1 \\ \Leftrightarrow \quad & 2^r \cdot \mathsf{F}_\mathsf{D}(m_i) \geq i \cdot h - 0.5 \\ \Leftrightarrow \quad & r \geq \log_2 \frac{i \cdot h - 0.5}{\mathsf{F}_\mathsf{D}(m_i)}. \end{aligned}$$

Since this inequality must hold for all i and r is an integer, we obtain the desired expression for $r_{\text{min-}h}$. □

For correctness—to ensure that no message in the support of D is assigned an empty homophone set—setting $r \geq r_{min-1}$ is necessary and sufficient. It is possible to obtain a simpler sufficient (though not necessary) condition for every message in the support of D to have at least h homophones by noting that messages are ordered according to increasing frequency, so $\mathsf{F}_\mathsf{D}(m_i) \geq i \cdot f_\mathsf{D}(m_1)$.

Corollary 2.16. *If messages are encoded with r-bit IBHE for some $r \geq \log_2 \frac{h}{\mathsf{f}_\mathsf{D}(m_1)}$, then every message $m \in \mathcal{M}$ has at least h homophones.*

Proof. For any i from 1 to $|\mathcal{M}|$, we have

$$\log_2 \frac{h}{\mathsf{f}_\mathsf{D}(m_1)} \geq \log_2 \frac{i \cdot h - 0.5}{i \cdot \mathsf{f}_\mathsf{D}(m_1)} \geq \log_2 \frac{i \cdot h - 0.5}{\mathsf{F}_\mathsf{D}(m_i)}. \qquad \square$$

Therefore, the condition $r \geq \log_2 \frac{h}{\mathsf{f}_\mathsf{D}(m_1)}$ is enough to guarantee that all messages have at least h homophones.

Definition 2.17. *The interval-based homophonic encoding (IBHE) scheme with message space $\mathcal{M} \subseteq \{0,1\}^n$ is defined as follows:*

- $\mathsf{Setup} : (\lambda, \mathsf{D}) \mapsto \mathsf{s}$, *computes the maximum r of the minimum encoding length r_{min-1} and the encoding length $r_{\mathsf{D},\lambda}$ determined by D and λ, and outputs the state $\mathsf{s} \coloneqq (r, \mathsf{D})$.*
- $\mathsf{Encode} : (m, \mathsf{s}) \mapsto e \cup \perp$, *chooses an integer e uniformly at random from the set of m's homophones $\mathcal{H}_\mathsf{s}^\mathsf{HE}(m) \coloneqq \left\{ \lfloor 2^r \cdot \mathsf{F}_\mathsf{D}(m_{i-1}) \rceil, \ldots, \lfloor 2^r \cdot \mathsf{F}_\mathsf{D}(m_i) \rceil - 1 \right\}$, and outputs either the r-bit representation of e, or $\perp$ if $m \notin \mathsf{supp}(\mathsf{D})$.*
- $\mathsf{Decode} : (e, \mathsf{s}) \mapsto m \cup \perp$, *determines the message $m_i \in \mathsf{supp}(D)$ such that $e \in \{\mathsf{F}_\mathsf{D}(m_{i-1}), \ldots, \mathsf{F}_\mathsf{D}(m_i) - 1\}$, and outputs either $m \coloneqq m_i$, or $\perp$ if no such m_i exists.*

Note that it is possible for the encoded bitlength r to be smaller than the data's bitlength n, in which case IBHE compresses data. Also note that IBHE's Encode and Decode algorithms need access to tables mapping the messages m_i to their intervals

$$\left\{ \lfloor 2^r \cdot \mathsf{F}_\mathsf{D}(m_{i-1}) \rceil, \ldots, \lfloor 2^r \cdot \mathsf{F}_\mathsf{D}(m_i) \rceil - 1 \right\}$$

via the cdf F_D of D, and *vice versa*. Since each interval can be represented by $2r$ bits, the total client-side storage for these tables is $4r \cdot |\mathsf{supp}(\mathsf{D})|$ bits.

In order to apply Theorem 2.14 to bound the HE-smoothness of IBHE, and thereby Theorem 2.13 to construct an FSE scheme, we need an upper bound on the Kullback–Leibler divergence of the encoded data's distribution D_s relative to the uniform distribution $\mathsf{U}_{|\mathcal{H}^{\mathsf{HE}}_\mathsf{s}|}$. We derive this bound using the following result:

Lemma [5, Proposition 5]. *Let D_0 and D_1 be distributions over a set $\mathcal{M}$ having the same support, and suppose that $\mathsf{f}_{\mathsf{D}_0}(m)$ and $\mathsf{f}_{\mathsf{D}_1}(m)$ are close for all $m \in \mathcal{M}$. Then, the Kullback–Leibler divergence of D_0 with respect to D_1 is*

$$\mathrm{KL}\left(\mathsf{D}_0, \mathsf{D}_1\right) \approx \frac{1}{2} \sum_{m \in \mathcal{M}} \frac{(\mathsf{f}_{\mathsf{D}_0}(m) - \mathsf{f}_{\mathsf{D}_1}(m))^2}{\mathsf{f}_{\mathsf{D}_1}(m)}.$$

We use this result in the proof of the following lemma: it says that for IBHE, if the encoding length r is at least $r_{\text{min-}h}$, as defined in the statement of Proposition 2.15, then this bound is approximately $1/2h^2$.

Lemma 2.18. *Let D be a distribution over $\mathcal{M}$ and suppose that m_1 is the least frequent message in the support of D. Suppose that the encoding length r in the IBHE scheme is such that $r \geq r_{min\text{-}h}$ for some positive integer h and let $\mathsf{s} \coloneqq (r, \mathsf{D})$. Then,*

$$\mathrm{KL}\left(\mathsf{D}_\mathsf{s}, \mathsf{U}_{2^r}\right) \leq \frac{1}{2h^2}.$$

Proof. For ease of notation, suppose $\mathcal{E} = \mathcal{H}^{\mathsf{HE}}_\mathsf{s} = \{0,1\}^r$ and write $\mathcal{H}^{\mathsf{HE}}$ for $\mathcal{H}^{\mathsf{HE}}_\mathsf{s}$. Recall that messages in the support of D are numbered by increasing frequency, and since $r \geq r_{\text{min-}h}$, each of these messages has at least h homophones in $\mathcal{E}$. In order to apply Baignères *et al.*'s proposition to estimate the KL divergence, we need $\mathsf{f}_{\mathsf{D}_\mathsf{s}}(e)$ to be close to 2^{-r} for all encodings $e \in \mathcal{E}$. Let $\delta_i \coloneqq \lfloor \mathsf{F}_\mathsf{D}(m_i) \cdot 2^r \rceil - \mathsf{F}_\mathsf{D}(m_i) \cdot 2^r$ be a rounding error associated with each message in $\mathsf{supp}(\mathsf{D})$, so $\delta_i \in (-0.5, 0.5]$. For convenience, set $\delta_0 \coloneqq 0$. Then, we can express the size of a message's homophone set as

$$\left|\mathcal{H}^{\mathsf{HE}}(m_i)\right| = \mathsf{f}_\mathsf{D}(m_i) \cdot 2^r + \delta_i - \delta_{i-1}. \tag{2.2}$$

Let $e \in \mathcal{H}^{\mathsf{HE}}(m_i)$ be one of m_i's homophones. Using the previous equation and recalling how $\mathsf{D_s}$ is defined, we get

$$\frac{\mathsf{f}_{\mathsf{D_s}}(e)}{2^{-r}} = \frac{\mathsf{f}_{\mathsf{D}}(m_i) \cdot 2^r}{|\mathcal{H}^{\mathsf{HE}}(m_i)|} = 1 + \frac{\delta_{i-1} - \delta_i}{|\mathcal{H}^{\mathsf{HE}}(m_i)|}.$$

Since the difference of the rounding errors, $\delta_{i-1} - \delta_i$, could take on any value in the interval $(-1, 1)$, we use the fact that $r \geq r_{\text{min-}h}$ to bound $\left|\mathcal{H}^{\mathsf{HE}}(m_i)\right|$, ensuring that $\mathsf{f}_{\mathsf{D_s}}(e)$ and 2^{-r} are close. We are now able to use the approximation in the aforementioned proposition:

$$\begin{aligned}
\mathrm{KL}\left(\mathsf{D_s}, \mathsf{U}_{2^r}\right) &\approx \frac{1}{2} \sum_{e \in \mathcal{E}} \frac{(\mathsf{f}_{\mathsf{D_s}}(e) - 2^{-r})^2}{2^{-r}} \\
&\approx 2^{r-1} \sum_{e \in \mathcal{E}} (\mathsf{f}_{\mathsf{D_s}}(e) - 1/2^r)^2 \\
&\approx 2^{r-1} \sum_{i=1}^{|\mathsf{supp}(\mathsf{D})|} \left|\mathcal{H}^{\mathsf{HE}}(m_i)\right| \cdot \left(\frac{\mathsf{f}_{\mathsf{D}}(m_i)}{|\mathcal{H}^{\mathsf{HE}}(m_i)|} - 1/2^r\right)^2 \\
&\approx 2^{r-1} \sum_{i=1}^{|\mathsf{supp}(\mathsf{D})|} \left(\frac{\mathsf{f}_{\mathsf{D}}(m_i)^2}{|\mathcal{H}^{\mathsf{HE}}(m_i)|} - \frac{2 \cdot \mathsf{f}_{\mathsf{D}}(m_i)}{2^r} + \frac{\left|\mathcal{H}^{\mathsf{HE}}(m_i)\right|}{2^{2r}}\right) \\
&\approx 2^{r-1} \sum_{i=1}^{|\mathsf{supp}(\mathsf{D})|} \left(\frac{\mathsf{f}_{\mathsf{D}}(m_i)^2}{|\mathcal{H}^{\mathsf{HE}}(m_i)|}\right) - 1 + \frac{1}{2}.
\end{aligned}$$

Next, we simplify the sum using Equation 2.2:

$$\begin{aligned}
\sum_{i=1}^{|\mathsf{supp}(\mathsf{D})|} \frac{\mathsf{f}_{\mathsf{D}}(m_i)^2}{|\mathcal{H}^{\mathsf{HE}}(m_i)|} &= \sum_{i=1}^{|\mathsf{supp}(\mathsf{D})|} \frac{\left(\left|\mathcal{H}^{\mathsf{HE}}(m_i)\right| - (\delta_i - \delta_{i-1})\right)^2}{2^{2r} \cdot |\mathcal{H}^{\mathsf{HE}}(m_i)|} \\
&= \frac{1}{2^{2r}} \sum_{i=1}^{|\mathsf{supp}(\mathsf{D})|} \left(\left|\mathcal{H}^{\mathsf{HE}}(m_i)\right| - 2(\delta_i - \delta_{i-1}) + \frac{(\delta_i - \delta_{i-1})^2}{|\mathcal{H}^{\mathsf{HE}}(m_i)|}\right) \\
&= \frac{1}{2^r} + \frac{1}{2^{2r}} \sum_{i=1}^{|\mathsf{supp}(\mathsf{D})|} \frac{(\delta_i - \delta_{i-1})^2}{|\mathcal{H}^{\mathsf{HE}}(m_i)|},
\end{aligned}$$

where the middle term collapsed to zero since $\delta_0 = \delta_{|\mathsf{supp}(\mathsf{D})|} = 0$. Finally, by noting that $\delta_i \in (-0.5, 0.5]$ guarantees that $(\delta_i - \delta_{i-1})^2 \leq 1$, using the assumption that each message has at least h homophones, and hence that $|\mathsf{supp}(\mathsf{D})|$ can be at most $2^r/h$, we get the bound

$$\sum_{i=1}^{|\mathsf{supp}(\mathsf{D})|} \frac{(\delta_i - \delta_{i-1})^2}{|\mathcal{H}^{\mathsf{HE}}(m_i)|} \leq |\mathsf{supp}(\mathsf{D})| \, \frac{1}{h} \leq \frac{2^r}{h^2}.$$

Combining the previous equations and inequalities yields the desired bound:

$$\mathrm{KL}\left(\mathsf{D}_{\mathsf{s}}, \mathsf{U}_{2^r}\right) \leq \frac{1}{2h^2}. \qquad \square$$

Suppose one has a distribution D, $\mathsf{n_r}$ samples, and a given target ϵ for the frequency-smoothing advantage $\mathsf{Adv}^{\mathsf{smooth}}_{\mathsf{HE}}(\mathsf{A}, \mathsf{D}, \mathsf{n_r})$ for the IBHE scheme. Using the approximation in Equation 2.1 at the end of Section 2.6.1 and the bound from Lemma 2.18, we obtain after some manipulation the requirement $h \geq \frac{\sqrt{\mathsf{n_r}}}{2\sqrt{2\pi}\epsilon}$. Combining this value with the sufficient condition from Corollary 2.16 enables us to derive a minimum bitlength for r to use in the IBHE scheme:

$$r \geq \log_2 \frac{\sqrt{\mathsf{n_r}}}{2\sqrt{2\pi}\epsilon \cdot \mathsf{f_D}(m_1)}.$$

A consequence of this inequality is that to halve the upper bound on an adversary's advantage ϵ, the minimum encoding length must increase by 1 bit.

A numerical example. Suppose D is such that the least frequent message occurs with probability $\mathsf{f_D}(m_1) = 2^{-5}$. Suppose $\mathsf{n_r} = 2^{10}$ and $\epsilon = 2^{-10}$. Then we get $h \geq 2^{15}/2\sqrt{2\pi} \approx 2^{12.7}$. Applying the bound from Corollary 2.16 to guarantee $r \geq r_{\text{min-}h}$, we find that we need $r \geq 18$ to limit the frequency-smoothing advantage of any adversary to at most 2^{-10} against IBHE for these parameters.

Variants. We now describe, with practicality in mind, two variants of IBHE, one of which we will use in our evaluation in Section 2.7.

- Variant 1: Append encodings to messages rather than entirely replacing them. This enables, for instance, faster decoding when processing query results.

- Variant 2: Modify how intervals (homophone sets) are allocated in such a way that smaller encoding bitlengths are possible (as long as they are still at least $\log_2 |\mathsf{supp}(\mathsf{D})|$). Some distributions can yield prohibitively large values of r_{min-1} if $\mathsf{f_D}(m_1)$ is relatively tiny.

The change to how intervals of $\{0, \ldots, 2^r - 1\}$ are assigned can be interpreted simply as building intervals (in the same way as before) for a modified distribution D'. The procedure shown in Algorithm 2.1 takes as input a distribution D and a desired encoding length. It outputs a second distribution, D', with the same support as D that can be used to construct intervals, encode, and decode with the desired encoding length. Starting with the least frequent message, this algorithm changes the distribution just enough that one homophone is assigned to each "too small" message. It does this

until each of the remaining messages can be assigned at least one homophone after being scaled to share the error introduced by assigning "too many" homophones to the least frequent messages. When $r \geq r_{min-1}$, this algorithm does not change the distribution.

The resulting modified IBHE scheme would run this algorithm as part of Setup and use the adjusted distribution D' in the state, $\mathsf{s} \coloneqq (r, \mathsf{D}')$, for all encoding and decoding. The original distribution D does not need to be stored.

Algorithm 2.1 Distribution adjustment algorithm for IBHE

Input: distribution D over $\mathcal{M}$, desired encoding length r with $r \geq \log_2 |\mathsf{supp}(\mathsf{D})|$.
Output: distribution D'.

$isBigEnough \leftarrow False$
$scaleFactor \leftarrow 1$
$(m_1, \ldots, m_{|\mathsf{supp}(\mathsf{D})|}) \leftarrow \mathcal{M} \cap \mathsf{supp}(\mathsf{D})$ where $\mathsf{f}_\mathsf{D}(m_1) \leq \ldots \leq \mathsf{f}_\mathsf{D}(m_{|\mathcal{M}|})$
for all $i \in \{1, \ldots, |\mathsf{supp}(\mathsf{D})|\}$ **do**
    if $i = 1$ **then**
        if $\mathsf{f}_\mathsf{D}(m_i) < 1/2^{r+1}$ **then**
            $\mathsf{f}_{\mathsf{D}'}(m_i) \coloneqq 1/2^{r+1}$
            $scaleFactor \leftarrow (1 - \mathsf{f}_\mathsf{D}(m_i))/(1 - \mathsf{f}_{\mathsf{D}'}(m_i))$
        else
            $\mathsf{f}_{\mathsf{D}'}(m_i) \coloneqq \mathsf{f}_\mathsf{D}(m_i)$
    else
        if $isBigEnough$ **then**
            $pmf_{\mathsf{D}'}(m_i) \coloneqq \mathsf{f}_\mathsf{D}(m_i)/scaleFactor$
        else
            if $\mathsf{f}_\mathsf{D}(m_i) \geq 1/2^r \cdot scaleFactor$ **then**
                $isBigEnough \leftarrow True$
                $\mathsf{f}_{\mathsf{D}'}(m_i) \coloneqq \mathsf{f}_\mathsf{D}(m_i)/scaleFactor$
            else
                $\mathsf{f}_{\mathsf{D}'}(m_i) \coloneqq 1/2^r$
                $scaleFactor \coloneqq (1 - \mathsf{F}_\mathsf{D}(m_i))/(1 - \mathsf{F}_{\mathsf{D}'}(m_i))$
return D' defined by $\mathsf{f}_{\mathsf{D}'}$

2.6.3 Banded homophonic encoding

We next present a simple homophonic encoding scheme that appends tags to messages rather than replacing them entirely. The tags can have any length $l \geq 1$ and each message has at most 2^l homophones. Let D be some distribution over $\mathcal{M}$ and again suppose that the messages in $\mathsf{supp}(\mathsf{D})$ are numbered according to their frequencies:

$$\mathsf{f}_\mathsf{D}(m_1) \leq \mathsf{f}_\mathsf{D}(m_2) \leq \ldots \leq \mathsf{f}_\mathsf{D}(m_{|\mathsf{supp}(\mathsf{D})|}).$$

Based on these frequencies, each message has a "band" that determines the number of possible tags that can be appended to it and therefore the number of homophones it has. Divide the interval $(0, \mathsf{f}_\mathsf{D}(m_{|\mathsf{supp}(\mathsf{D})|})]$ into 2^l bands each of width $\mathsf{w} \coloneqq \mathsf{f}_\mathsf{D}(m_{|\mathsf{supp}(\mathsf{D})|})/2^l$, numbered 1 to 2^l. The messages whose frequencies are in band

i, in the interval $((i-1)\cdot \mathsf{w}, i\cdot \mathsf{w}]$, will each have i homophones. In particular, the most frequent message, $m_{|\mathcal{M}|}$, will have 2^l homophones—all possible l-bit strings can be appended to it.

Definition 2.19. *The banded homophonic encoding (BHE) scheme with message space $\mathcal{M} \subseteq \{0,1\}^n$ is defined as follows:*

- $\mathsf{Setup} : (\lambda, \mathsf{D}) \mapsto \mathsf{s}$ *computes the tag length l determined by λ and D, the band width $\mathsf{w} := \mathsf{f_D}(m_{|\mathsf{supp(D)}|})/2^l$, and outputs $\mathsf{s} := (l, \mathsf{w}, \mathsf{D})$.*
- $\mathsf{Encode} : (m, \mathsf{s}) \mapsto m \parallel t \ \cup \{\perp\}$ *computes message m's frequency band, $\mathsf{b} := \lceil \mathsf{f_D}(m)/\mathsf{w} \rceil$, picks an integer t uniformly at random in $\{0, 1, ..., \mathsf{b}-1\}$, and outputs either the $(n+l)$-bit string $m \parallel t$, where t is represented using l bits, or $\perp$ if $m \notin \mathsf{supp(D)}$.*
- $\mathsf{Decode} : (e, \mathsf{s}) \mapsto \mathsf{Trunc}\,(e, n)$ *removes the last l bits of e to recover m.*

The main advantages of this banded HE scheme are that there is no minimum tag length and decoding is fast—in particular, it does not need any table of frequency information to decode. Encoding requires storing a table of $l \cdot |\mathcal{M}|$ bits.

Another feature is that if the distribution changes, the scheme can adapt to the new frequencies without re-encoding every data item. This can be done by using so-far-unused l-bit tags if an item's frequency increases (effectively increasing its band number), or by initially over-sizing l and using a deliberately under-sized set of homophones and, if an item's frequency decreases, re-scaling the bands used for all the other items. For queries to continue to return complete results, either the all-time maximum band number of each message will need to be stored, or all 2^l homophones of each message will need to be checked. By contrast, the interval-based encoding scheme cannot adapt to changes in the distribution without re-encoding all of the messages.

A negative aspect of the banded homophonic encoding scheme is that the total number of encodings, $\left|\mathcal{H}_{\mathsf{s}}^{\mathsf{HE}}\right|$, is not fixed. For Theorem 2.14 to apply, the distribution of the encoded data must already be close enough to the uniform distribution on its homophones. Consider the rounding error for each message: let it be

$$\delta_i := \left\lceil 2^l \cdot \mathsf{f_D}(m)/\mathsf{f_D}(m_{|\mathsf{supp(D)}|}) \right\rceil - 2^l \cdot \mathsf{f_D}(m)/\mathsf{f_D}(m_{|\mathsf{supp(D)}|}),$$

where $\delta_i \in [0,1)$ for each m_i, from $1 \le i \le |\mathsf{supp(D)}|$. The total number of homophones is then $\left|\mathcal{H}_{\mathsf{s}}^{\mathsf{HE}}\right| = \frac{2^l}{\mathsf{f_D}(m_{|\mathsf{supp(D)}|})} + \sum_{i=1}^{|\mathsf{supp(D)}|} \delta_i$. Whereas the total number of homophones was predictable (indeed fixed) for IBHE, here it may vary by as much as

$|\mathsf{supp}(\mathsf{D})| - 1$ depending on the distribution and the rounding errors δ_i it produces. For the encoded data's distribution to be close enough to uniform so we can apply Theorem 2.14, we require $|\mathsf{supp}(\mathsf{D})| \ll \frac{2^l}{\mathsf{f}_\mathsf{D}(m_{|\mathsf{supp}(\mathsf{D})|})}$. This unpredictability indicates that values of l for BHE will need to be much higher than values of r for IBHE to guarantee smoothness. This is quantified in the following lemma.

Lemma 2.20. *Let* D *be a distribution over* $\mathcal{M}$ *and suppose that* $m_{|\mathsf{supp}(\mathsf{D})|}$ *is the most frequent message according to* D*. Suppose that* l *in the BHE scheme is such that* $|\mathsf{supp}(\mathsf{D})| \ll \frac{2^l}{\mathsf{f}_\mathsf{D}(m_{|\mathsf{supp}(\mathsf{D})|})}$*, and let* $\left|\mathcal{H}_\mathsf{s}^\mathsf{HE}\right|$ *be the size of the resulting set of homophones. Then*

$$\mathrm{KL}\left(\mathsf{D}_\mathsf{s}, \mathsf{U}_{|\mathcal{H}_\mathsf{s}^\mathsf{HE}|}\right) \leq \frac{|\mathsf{supp}(\mathsf{D})| \cdot \mathsf{f}_\mathsf{D}(m_{|\mathsf{supp}(\mathsf{D})|})}{2^{l+1}}.$$

Proof. For ease of notation, suppose $\mathcal{E} = \bigcup_{m \in \mathsf{supp}(\mathsf{D})} \mathcal{H}_\mathsf{s}^\mathsf{HE}(m)$, and write $\mathcal{H}^\mathsf{HE}$ for $\mathcal{H}_\mathsf{s}^\mathsf{HE}$. Recall that the number of homophones of a message $m \in \mathsf{supp}(\mathsf{D})$ is its band number, $\left\lceil 2^l \cdot \mathsf{f}_\mathsf{D}(m)/\mathsf{f}_\mathsf{D}(m_{|\mathsf{supp}(\mathsf{D})|})\right\rceil$, where $m_{|\mathsf{supp}(\mathsf{D})|}$ is the most frequent message according to D. Letting $\delta_i \coloneqq \left|\mathcal{H}^\mathsf{HE}(m_i)\right| - 2^l \cdot \mathsf{f}_\mathsf{D}(m_i)/\mathsf{f}_\mathsf{D}(m_{|\mathsf{supp}(\mathsf{D})|})$, we can write

$$\left|\mathcal{H}^\mathsf{HE}\right| = \frac{2^l}{\mathsf{f}_\mathsf{D}(m_{|\mathsf{supp}(\mathsf{D})|})} + \sum_{i=1}^{|\mathsf{supp}(\mathsf{D})|} \delta_i. \tag{2.3}$$

By assumption, $|\mathsf{supp}(\mathsf{D})| \ll \frac{2^l}{\mathsf{f}_\mathsf{D}(m_{|\mathsf{supp}(\mathsf{D})|})}$, so Theorem 2.14 applies and we can use the following approximation for the Kullback–Leibler divergence:

$$\begin{aligned}
\mathrm{KL}\left(\mathsf{D}_\mathsf{s}, \mathsf{U}_{|\mathcal{H}^\mathsf{HE}|}\right) &\approx \frac{1}{2}\sum_{e \in \mathcal{E}} \frac{\left(\mathsf{f}_{\mathsf{D}_\mathsf{s}}(e) - 1/\left|\mathcal{H}^\mathsf{HE}\right|\right)^2}{1/|\mathcal{H}^\mathsf{HE}|} \\
&\approx \frac{|\mathcal{H}^\mathsf{HE}|}{2} \sum_{i=1}^{|\mathsf{supp}(\mathsf{D})|} \left|\mathcal{H}^\mathsf{HE}(m_i)\right| \cdot \left(\frac{\mathsf{f}_\mathsf{D}(m_i)}{|\mathcal{H}^\mathsf{HE}(m_i)|} - \frac{1}{|\mathcal{H}^\mathsf{HE}|}\right)^2 \\
&\approx \frac{|\mathcal{H}^\mathsf{HE}|}{2} \sum_{i=1}^{|\mathsf{supp}(\mathsf{D})|} \left(\frac{\mathsf{f}_\mathsf{D}(m_i)^2}{|\mathcal{H}^\mathsf{HE}(m_i)|} - \frac{2 \cdot \mathsf{f}_\mathsf{D}(m_i)}{|\mathcal{H}^\mathsf{HE}|} + \frac{\left|\mathcal{H}^\mathsf{HE}(m_i)\right|}{|\mathcal{H}^\mathsf{HE}|^2}\right) \\
&\approx \frac{|\mathcal{H}^\mathsf{HE}|}{2} \left(\sum_{i=1}^{|\mathsf{supp}(\mathsf{D})|} \frac{\mathsf{f}_\mathsf{D}(m_i)^2}{|\mathcal{H}^\mathsf{HE}(m_i)|}\right) - 1 + \frac{1}{2}.
\end{aligned}$$

Next, we estimate the sum using the fact that $\delta_i \in [0, 1)$ for $i = 1, \ldots, |\mathsf{supp}(\mathsf{D})|$:

$$\begin{aligned}
\sum_{i=1}^{|\mathsf{supp}(\mathsf{D})|} \frac{\mathsf{f}_\mathsf{D}(m_i)^2}{|\mathcal{H}^\mathsf{HE}(m_i)|} &= \sum_{i=1}^{|\mathsf{supp}(\mathsf{D})|} \frac{\mathsf{f}_\mathsf{D}(m_i)^2}{2^l \cdot \mathsf{f}_\mathsf{D}(m_i)/\mathsf{f}_\mathsf{D}(m_{|\mathsf{supp}(\mathsf{D})|}) + \delta_i} \\
&\leq \sum_{i=1}^{|\mathsf{supp}(\mathsf{D})|} \frac{\mathsf{f}_\mathsf{D}(m_i)^2}{2^l \cdot \mathsf{f}_\mathsf{D}(m_i)/\mathsf{f}_\mathsf{D}(m_{|\mathsf{supp}(\mathsf{D})|})} \\
&\leq \frac{\mathsf{f}_\mathsf{D}(m_{|\mathsf{supp}(\mathsf{D})|})}{2^l}.
\end{aligned}$$

Finally, combining this upper bound on the sum with an upper bound on the total number of homophones from Equation 2.3 yields the desired bound:

$$\begin{aligned}\mathrm{KL}\left(\mathsf{D}_{\mathsf{s}}, \mathsf{U}_{|\mathcal{H}^{\mathsf{HE}}|}\right) &\leq \frac{\frac{2^l}{\mathsf{f}_{\mathsf{D}}(m_{|\mathsf{supp}(\mathsf{D})|})} + |\mathsf{supp}(\mathsf{D})|}{2}\left(\frac{\mathsf{f}_{\mathsf{D}}(m_{|\mathsf{supp}(\mathsf{D})|})}{2^l}\right) - \frac{1}{2} \\ &\leq \frac{|\mathsf{supp}(\mathsf{D})| \cdot \mathsf{f}_{\mathsf{D}}(m_{|\mathsf{supp}(\mathsf{D})|})}{2^{l+1}}.\end{aligned}$$

□

Suppose one has a distribution D, $\mathsf{n_r}$ samples, and a given target ϵ for the frequency-smoothing advantage $\mathsf{Adv}^{\mathsf{smooth}}_{\mathsf{HE}}(\mathsf{A}, \mathsf{D}, \mathsf{n_r})$ for the BHE scheme. Using the approximation in Equation 2.1 at the end of Section 2.6.1 and the bound from Lemma 2.20, we obtain the requirement

$$l \geq \log_2\left(\frac{\mathsf{n_r} \cdot |\mathsf{supp}(\mathsf{D})| \cdot \mathsf{f}_{\mathsf{D}}(m_{|\mathsf{supp}(\mathsf{D})|})}{(2\epsilon)^2 \cdot \pi}\right) - 1.$$

Note that since $\mathsf{f}_{\mathsf{D}}(m_{|\mathsf{supp}(\mathsf{D})|})$ is the maximum frequency, $\mathsf{f}_{\mathsf{D}}(m_{|\mathsf{supp}(\mathsf{D})|}) \geq \frac{1}{|\mathsf{supp}(\mathsf{D})|}$, so regardless of the distribution, the added bitlength l must be at least $\log_2\left(\frac{\mathsf{n_r}}{(2\epsilon)^2 \cdot \pi}\right) - 1$.

A numerical example. Suppose $\mathsf{n_r} = 2^{10}$ and $\epsilon = 2^{-10}$, and let D be the given distribution on the message space $\mathcal{M}$. A lower bound on the required tag length l in the BHE scheme is $\log_2\left(\frac{2^{10}}{(2 \cdot 2^{-10})^2 \cdot \pi}\right) - 1 \approx 25$. The minimum value of l needed for a specific distribution may be greater still. Recall the similar example at the end of Section 2.6.2: for the same values of $\mathsf{n_r}$ and ϵ, the minimum required encoding bitlength for interval-based HE was $r \geq 12.7 + \log_2 \frac{1}{\mathsf{f}_{\mathsf{D}}(m_1)}$. With banded HE, the minimum *additional* bitlength is $l = 25$.

2.7 Practical security

We have introduced definitions and general constructions that we proved secure with respect to our expressly defined security notions. However, as we have seen in some numerical examples for our encoding schemes, achieving typical cryptographic security levels for our notion of FSE−SMOOTH security could require large encoding lengths for some distributions, leading to a serious blow-up in query complexity (cf. Section 2.3). Given this limitation, we choose to perform an empirical evaluation of the security of FSE against frequency analysis attacks. Of course, we are also interested in achieving FSE−PRIV, but this is easily done using our HE-DE construction with an appropriate DE component, e.g., a block cipher such as AES.

In this section, therefore, we adopt a more pragmatic approach, working with moderate encoding lengths and switching to a more practical metric of evaluation, since we already know that we will not attain cryptographic levels of security for arbitrary distributions. The security metric we work with in this section is the number of data items that an attacker can correctly decrypt, which has been used for assessing the effectiveness of inference attacks in the literature [32, 62] and closely reflects a real-world adversary's aim of plaintext recovery. This approach is similar to the paradigm of *accelerated provable security*, also called *prove-then-prune* [35]: we designed a scheme and proved its security based on the security of its primitives, but we relax the primitives for practical use and rely on cryptanalysis to assess security.

We evaluate an FSE scheme built from static HE and DE using our modular construction. For the HE component, we use IBHE (Section 2.6.2) with the distribution adjustment algorithm (Variant 2 at the end of that section). Our attacks on FSE are in the public distribution setting, where $\tilde{\mathsf{D}} = \hat{\mathsf{D}} = \mathsf{D}$. This grants the adversary greater power than in the scenario considered by Naveed *et al.* [62], where the distribution $\hat{\mathsf{D}}$ is only approximately D.

Our aim is to reduce the attacker's success rate in recovering plaintext to that of a naïve guessing attack, which is, in any case, not preventable. We develop a maximum likelihood attack for this setting, and then assess its performance using the same HCUP datasets to which Naveed *et al.* applied DE and carried out inference attacks [62]. This allows us to compare the security of FSE and of DE, and of FSE to naïve guessing attacks.

2.7.1 A maximum likelihood attack on static FSE

Given the selected metric of success—the number of records an attacker can correctly decrypt—we must determine how an attacker would maximize this number. We apply the technique of maximum likelihood estimation (MLE) to derive an efficient attack on a static FSE scheme under the assumption that only frequency information is meaningful. MLE is an asymptotically optimal technique; as the number of samples tends toward infinity, the maximum likelihood estimator is an unbiased estimator with the smallest variance (spread). This means that the expected value of the estimator is the true value.

Our analysis relies on the following two assumptions. The first is that a static FSE scheme's $\mathsf{Encrypt}$ algorithm outputs each of a message's homophones with equal probability. This property holds for composed FSE schemes arising from both of our static HE constructions. It is reasonable to assume that it would hold for any static FSE

scheme since the state is not updated in such schemes and, after all, the goal of a frequency-smoothing scheme is to smooth the distribution such that it becomes indistinguishable from uniform. Our second assumption is that the adversary considers only "proper" deterministic decryption functions—its solution cannot map one ciphertext to multiple plaintexts, nor can it assign one plaintext more homophones than it has. This rules out attacks that may otherwise appear to perform well, such as simply guessing that *every* item is the plaintext having the highest frequency in the reference distribution. Such a naïve attack could actually perform better than the MLE attack with respect to our chosen success metric.

We let $n(c)$ denote the number of times that ciphertext $c \in \mathcal{C}$ occurs in the database DB. According to the MLE approach, a most likely decryption θ maximizes the likelihood $L(\theta|\mathsf{DB}) \coloneqq \text{Prob}[\mathsf{DB}|\theta]$. Thus we wish to compute

$$\begin{aligned}
\arg\max_{\theta} \ \text{Prob}[\mathsf{DB}|\theta] &= \arg\max_{\theta} \prod_{c\in\mathcal{C}} \left(\frac{\mathsf{f}_{\mathsf{D}}(\theta(c))}{|\mathcal{H}^{\mathsf{FSE}}(\theta(c))|} \right)^{n(c)} \\
&= \arg\max_{\theta} \prod_{m\in\mathsf{supp}(\mathsf{D})} \left(\frac{\mathsf{f}_{\mathsf{D}}(m)}{|\mathcal{H}^{\mathsf{FSE}}(m)|} \right)^{\sum_{c\in\theta^{-1}(m)} n(c)} \\
&= \arg\max_{\theta} \sum_{m\in\mathsf{supp}(\mathsf{D})} \left(\sum_{c\in\theta^{-1}(m)} n(c) \right) \cdot \log \frac{\mathsf{f}_{\mathsf{D}}(m)}{|\mathcal{H}^{\mathsf{FSE}}(m)|}
\end{aligned}$$

where at the first step, we use the fact that messages are sampled independently, and at the last step, we use the fact that maximizing a product of terms can be achieved by maximizing the sum of the logs of those terms. To maximize this expression, θ should map the most frequently occurring ciphertexts (with largest $n(c)$ values) to the messages with the largest "scaled frequencies" $\mathsf{f}_{\mathsf{D}}(m)/\left|\mathcal{H}^{\mathsf{FSE}}(m)\right|$. This observation leads directly to the following attack.

When not all possible ciphertexts appear in DB, the sizes of the sets $\theta^{-1}(m)$ can be strictly less than $\left|\mathcal{H}^{\mathsf{FSE}}(m)\right|$. In this case, we scale the number of homophones of each message by the fraction of unique ciphertexts in $\mathcal{C}$ that occur in DB.

So, suppose the adversary has $\mathsf{n_r}$ FSE-encrypted items, each of whose underlying plaintext was sampled independently from $\mathcal{M}$ according to the known distribution D. The adversary can compute the number of homophones $\left|\mathcal{H}^{\mathsf{FSE}}_{\mathsf{s}}(m)\right|$ for each m in $\mathsf{supp}(\mathsf{D})$, since this set's size depends on the state s, which in turn depends only on the distribution and not the particular choice of key. Further, suppose $\left|\mathcal{H}^{\mathsf{FSE}}\right| = |\mathcal{C}|$, so that every possible ciphertext appears at least once.

The adversary's goal is to find the correct many-to-one decryption mapping $\theta : \mathcal{C} \to \mathcal{M}$. The attack proceeds as follows. First, label the distinct observed ciphertexts so their counts are in decreasing order: $n(c_1) \geq n(c_2) \geq \cdots \geq n(c_{|\mathcal{C}|})$. Also label the plaintext items in the support of D so their scaled frequencies are in decreasing order:

$$\frac{\mathsf{f}_{\mathsf{D}}(m_1)}{|\mathcal{H}_{\mathsf{s}}^{\mathsf{FSE}}(m_1)|} \geq \frac{\mathsf{f}_{\mathsf{D}}(m_2)}{|\mathcal{H}_{\mathsf{s}}^{\mathsf{FSE}}(m_2)|} \geq \cdots \geq \frac{\mathsf{f}_{\mathsf{D}}(m_{|\mathsf{supp}(\mathsf{D})|})}{\left|\mathcal{H}_{\mathsf{s}}^{\mathsf{FSE}}(m_{|\mathsf{supp}(\mathsf{D})|})\right|}.$$

Then, the attack sets θ so that

$$\begin{aligned}\theta :&\{c_1, \ldots, c_{|\mathcal{H}_{\mathsf{s}}^{\mathsf{FSE}}(m_1)|}\} \mapsto m_1,\\ \theta :&\{c_{|\mathcal{H}_{\mathsf{s}}^{\mathsf{FSE}}(m_1)|+1}, \ldots, c_{|\mathcal{H}_{\mathsf{s}}^{\mathsf{FSE}}(m_1)|+|\mathcal{H}_{\mathsf{s}}^{\mathsf{FSE}}(m_2)|}\} \mapsto m_2,\end{aligned}$$

and so on, until the $\left|\mathcal{H}_{\mathsf{s}}^{\mathsf{FSE}}(m_{|\mathsf{supp}(\mathsf{D})|})\right|$ least frequent ciphertexts are mapped to $m_{|\mathsf{supp}(\mathsf{D})|}$.

This efficient procedure creates a decryption mapping θ that is not necessarily unique: if two or more encrypted data item counts are the same, then permuting them will result in decryption mappings that are equally likely. Similarly, if two or more scaled plaintext frequencies are the same, then permuting them will result in equally likely decryption mappings. In our experiments, such ties were broken randomly.

Notice that if deterministic encryption were used in place of FSE, so that $\left|\mathcal{H}_{\mathsf{s}}^{\mathsf{FSE}}(m)\right| = 1$ for each message m, then this attack reduces to a basic frequency analysis attack of the type used by Naveed, Kamara, and Wright [62]. Thus, our attack generalizes frequency analysis on deterministic encryption.

This attack is easily modified for the case where the attacker and data owner have different information about the data's distribution ($\hat{\mathsf{D}} \neq \tilde{\mathsf{D}}$). In this case, the attacker would number the plaintext items according to $\mathsf{f}_{\hat{\mathsf{D}}}(m)/\left|\mathcal{H}_{\tilde{\mathsf{s}}}^{\mathsf{FSE}}(m)\right|$, where $\tilde{\mathsf{s}}$ depends only on $\tilde{\mathsf{D}}$.

2.7.2 Experimental results

We use the aforementioned MLE attack to simulate an attacker attempting to decrypt FSE-encrypted records in a database. We individually attack the 12 columns of 200 medical databases (one database per hospital). To obtain the distribution D, we use HCUP data from 2009 (q.v. Section 1.3). In Table 2.2, we list the 12 target attributes, ordered by the number of values they can have, and a typical minimum encoding length for the IBHE scheme. (Different per-hospital distributions could result in slightly different r_{min} values.)

We simulate FSE-encrypting and then attacking the HCUP data of the *individual* largest hospitals using each of the hospitals' data to define a *per-hospital* reference distribution for each of the 12 target attributes. We assume this per-hospital distribution is always known to the attacker. This experimental setup is good for the attacker—in reality, it is likely that an attacker attempting to steal a particular hospital's data would only have access to, say, national statistics from previous years (as in [62]). To simplify our analysis, we ignore all values that were identified as missing, invalid, unavailable, or inconsistent.

Table 2.2: Typical r_{min} values for attributes in FSE MLE attack

Attribute	Num. values	Typical r_{min} (IBHE)
Length of stay (`LOS`)	365	23
Age (`AGE`)	125	20
Major diagnostic category (`MDC`)	25	10
Admission month (`AMONTH`)	12	4
Admission type (`ATYPE`)	6	12
Primary payer (`PAY1`)	6	7
Ethnicity group (`RACE`)	6	7
Admission source (`ASOURCE`)	5	10
Disease severity (`APRDRG_Severity`)	4	10
Mortality risk (`APRDRG_Risk_Mortality`)	4	10
Patient died (`DIED`)	2	5
Sex (`FEMALE`)	2	1

Our results are presented in a series of graphs in Figure 2.6, one for each attribute, and with various encoding lengths r for each attribute. These graphs show complementary cumulative distributions, since we are interested in the number of databases for which *at least* some fraction of the records were recovered. We consider each attribute separately, so "percentages of records recovered" refers not to entire records in a database, but to the values of a particular attribute in those records.

Our goal, informally, is that attacking FSE is hard—in particular, at least as hard as attacking DE. If our attacks are less successful against FSE than DE, then the lines corresponding to FSE will be to the left of and below those for DE, and the area under them will be smaller.

The trivial guessing attack. An adversary can always simply guess that every ciphertext it sees corresponds to the most likely plaintext. It would succeed well

with this metric for certain attributes, regardless of the encryption method. This is the case, for example, with the binary attribute `DIED` where there is one very likely plaintext since most patients survive their hospital visits. Each attribute's graph in Figure 2.6 includes a solid gray line, labeled "max f_{D}", that represents the success rate of this trivial attack. No encryption method can force the trivial attacker below this line, so—according to the metric chosen for our evaluation—little security is achievable for certain attributes like `DIED` using any form of encryption.

Our MLE approach does not capture this trivial attack since it looks for a *correct* decryption mapping that respects the numbers of homophones each plaintext has. Thus, it is possible for the trivial attack to actually perform better than a statistically optimal attack. As can be seen from the graphs, by setting r appropriately, we can ensure that this is the case, making the MLE attack worse than simple guessing. Since it is not possible for any encryption scheme to protect against simple guessing attacks, the fact that the MLE attack is made worse than the trivial attack by homophonic encoding is a positive feature of our approach. Indeed, once this is achieved for a particular value of r, there is no benefit in increasing r further (except perhaps to disguise which database column is which).

Comparison with DE. Naveed *et al.* individually attacked 200 databases of DE-encrypted medical data from 2009 using aggregated 2004 data for the auxiliary distribution [62]. The power of frequency analysis attacks on DE can be further strengthened by assuming the attacker knows the exact per-database distributions rather than an aggregated distribution. In evaluating DE, we consider both situations, yielding two curves for DE in each graph: one that uses an aggregated distribution ($\hat{\mathsf{D}} \approx \mathsf{D}$, similar to [62], but from the same year) and the other, a per-database distribution ($\hat{\mathsf{D}} = \mathsf{D}$). Our experiments attacking FSE always assume that the adversary has exact knowledge of the data's distribution D, giving it the most power.

For some attributes, frequency analysis on DE even with aggregated auxiliary data recovers nearly *all* records in *all* 200 databases (e.g., `APRDRG_Risk_Mortality`, `DIED`, `FEMALE`), and per-hospital distributions perform even better, recovering nearly 100% of records correctly in every case. And, as can be seen from our graphs in Figure 2.6, FSE withstands attacks much better than DE in the majority of cases, even when the adversary is given the per-hospital distributions. The results for `AGE`, `LOS`, and `MDC` are particularly encouraging. One exception is `DIED`; using FSE barely reduces the number of records an attacker can recover, even with large encoding lengths. The reason is that `DIED` is binary and one value accounts for over 98% of records in a data set, on average. Thus the MLE attack will still succeed with high probability, as it

will assign the majority of ciphertexts to the high probability value and be correct most of the time. As noted earlier, in such a situation, the trivial plaintext recovery attack that just assigns every ciphertext to the most likely plaintext value performs even better and is also unavoidable for *any* encryption scheme.

Limit case. As the encoding length r increases, there are fewer repeated ciphertexts, and eventually, no ciphertext occurs more than once. Given $\mathsf{n_r}$ ciphertext items, our MLE attack assigns approximately $\mathsf{n_r} \cdot \mathsf{f_D}(m)$ of them to message m. For large enough $\mathsf{n_r}$, we can approximate this assignment of plaintexts to ciphertexts in the following manner: for each ciphertext, the attacker independently samples $\mathcal{M}$ according to D to determine its guess. The probability that any single ciphertext is assigned the correct plaintext is then $f \coloneqq \sum_{m \in \mathcal{M}} \mathsf{f_D}(m)^2$, and the number of correct guesses then follows a binomial distribution with $\mathsf{n_r}$ trials and success probability f. We have simulated such an attack strategy using each individual hospital's distribution and indicated the resulting curves with $r \to \infty$ in the graphs. The fraction of records recovered quickly converges to this random guessing strategy, even using encoding lengths much less than r_{min}.

Distribution adjustment algorithm. We use the distribution adjustment algorithm from Algorithm 2.1: when the desired encoding length is less than r_{min}, intervals are constructed in a different way that guarantees even the least frequent items have at least one homophone. The values of r_{min} were typically highest for `AGE` (20) and `LOS` (23). Using an encoding length of 8 for `AGE` still resulted in fewer records decrypted than with DE. For `LOS`, whose minimum unencoded bitlength is 9, there was a drastic drop in the percentage of records recovered even with an encoding length of only 10. Using only DE, 50% of hospitals had at least 80% of their records recovered, while with 10-bit IBH encoding, no hospital had more than 22% of its records recovered.

Query complexity. The parameter r affects query complexity in addition to affecting storage cost: an equality query for one item becomes an equality query for each of its homophones. For large enough encoding lengths r, our results indicate that the statistically optimal MLE attack offers no advantage over guessing—even when the attacker has precise knowledge of the underlying data's distribution. However, the results quickly converge to random guessing for all attributes, and the effect on query complexity is manageable. For example, encoding `AGE` with $r = 10$ bits results in a query expansion of $2^r \cdot \mathsf{f_D}(0) \approx 2^7$ in the worst case (for the most frequent age, 0). Encoding `MDC` with $r = 10$ bits results in a query expansion of about 2^8 for the most frequent item.

Limitations. For a few attributes, such as `ASOURCE` and `RACE`, even an attacker using the random guessing strategy succeeds more often than may be acceptable. In these cases, higher values of r cannot help limit the adversary's success. These attributes had few possible plaintext values (5 and 6 respectively) and their unencoded distributions were skewed: for example, the most common `ASOURCE` value was about 2^9 times more frequent than the least common value. As we noted previously, such guessing attacks are unavoidable in this situation.

2.8 Conclusions

Frequency-smoothing encryption provides a means to prevent inference attacks on snapshots of encrypted databases. While deterministic encryption allows equality queries on encrypted data, it must leak equality of plaintexts to do so. Our FSE constructions allow randomizing ciphertexts in a controlled way to maintain support for equality queries, while making the ciphertexts' distribution uniform. Our schemes are static and we did not analyze them in the case where the data owner's guess of their distribution, $\tilde{\mathsf{D}}$, differed from their actual distribution, D, or when the adversary's approximation of the distribution, $\hat{\mathsf{D}}$, was different. We presented general definitions that would allow considering such cases in future work.

Low-entropy distributions are those that would benefit the most from frequency smoothing. However, for the scheme to allow efficient equality queries (both formulation by the client and processing by the server), the number of homophones of each item must not be too large, because an equality query on unencrypted data is replaced by a set membership query for each of its homophones. Also, if values have different sizes, then they still need to be encoded to a fixed length, which might result in high ciphertext expansion and thus increased storage requirements at the server.

It would be interesting to explore the effect of HE on the success of pairwise column attacks for OPE [21] and on the success of other inference attacks that exploit cross-column correlations [7]. Addressing the same issue would be of great interest for indices in searchable encryption, in particular for inference attacks exploiting word co-occurrence [70] or attacks that use subsets of known documents [16]. Also, our general definition of FSE is conducive to the development of schemes that can adapt to changing distributions in the underlying data. Future work could assess how the attack prevention capability of our static HE techniques degrades as the distribution changes gradually, to understand how much change can be tolerated.

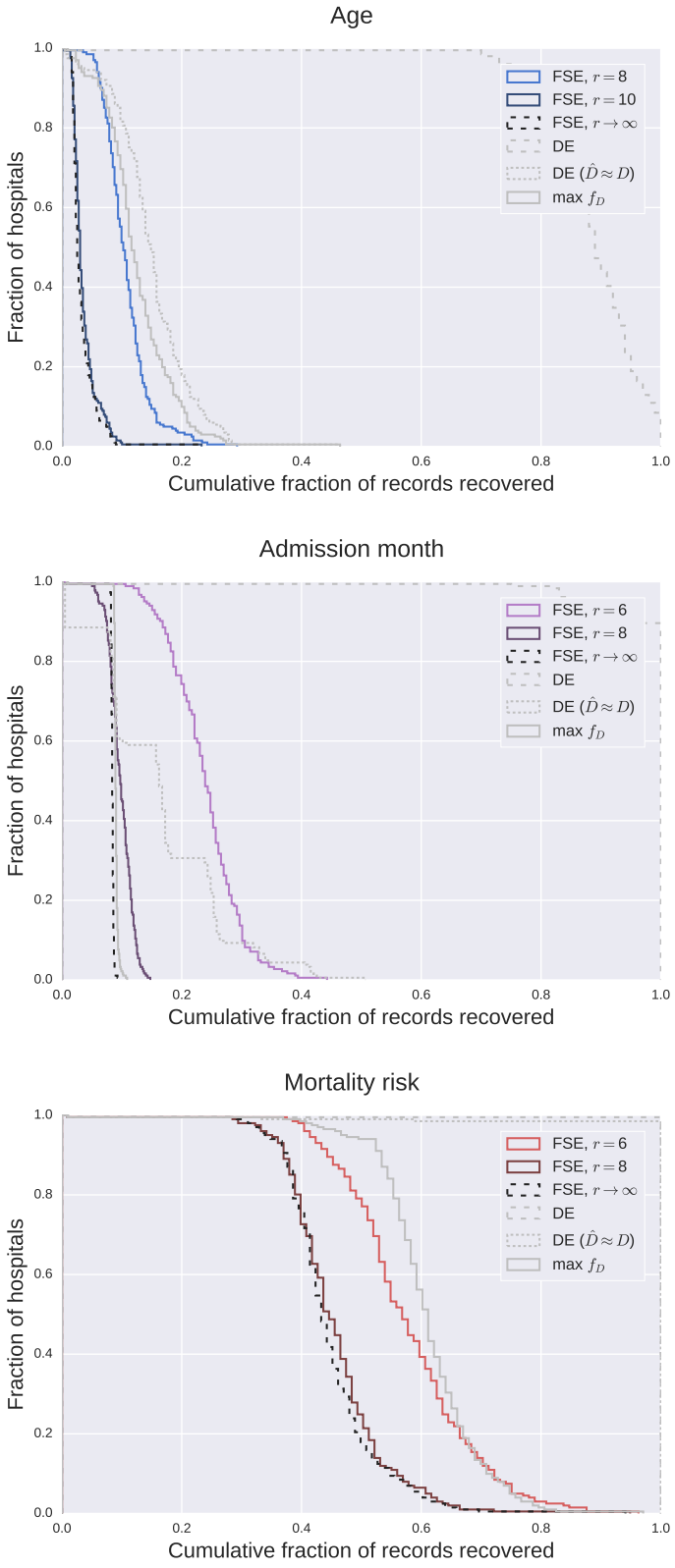


Figure 2.6: Our experimental results by attribute: complementary cumulative distributions (continued on next pages).

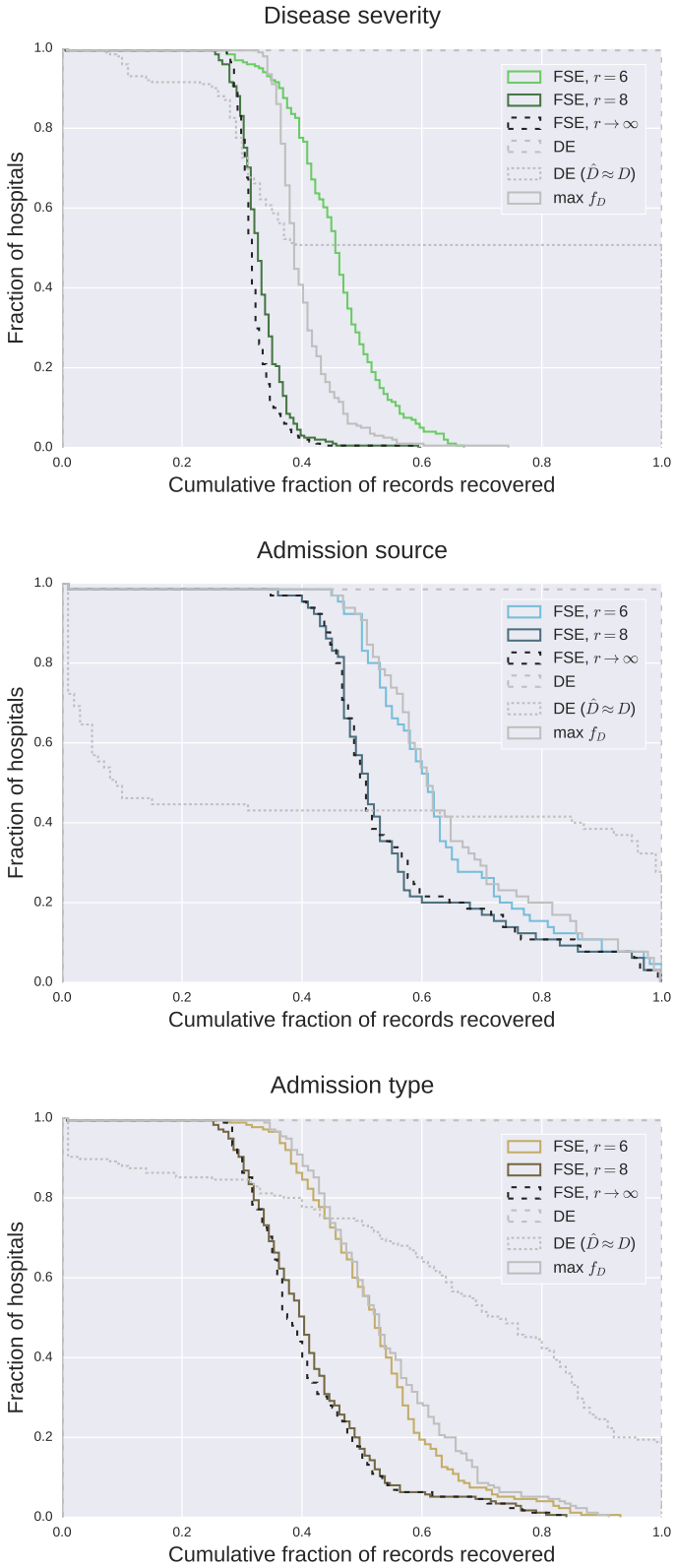


Figure 2.6: Our experimental results by attribute: complementary cumulative distributions (continued from previous page).

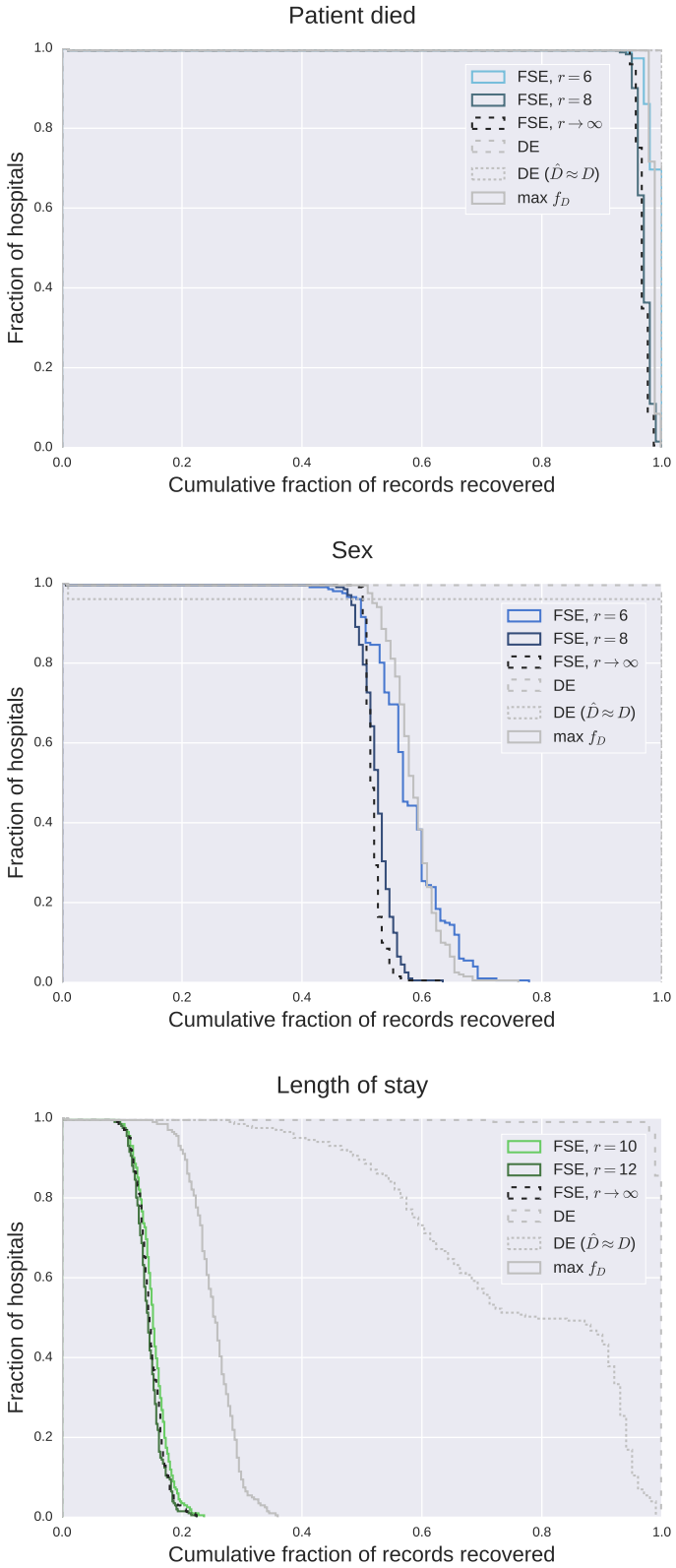


Figure 2.6: Our experimental results by attribute: complementary cumulative distributions (continued from previous page).

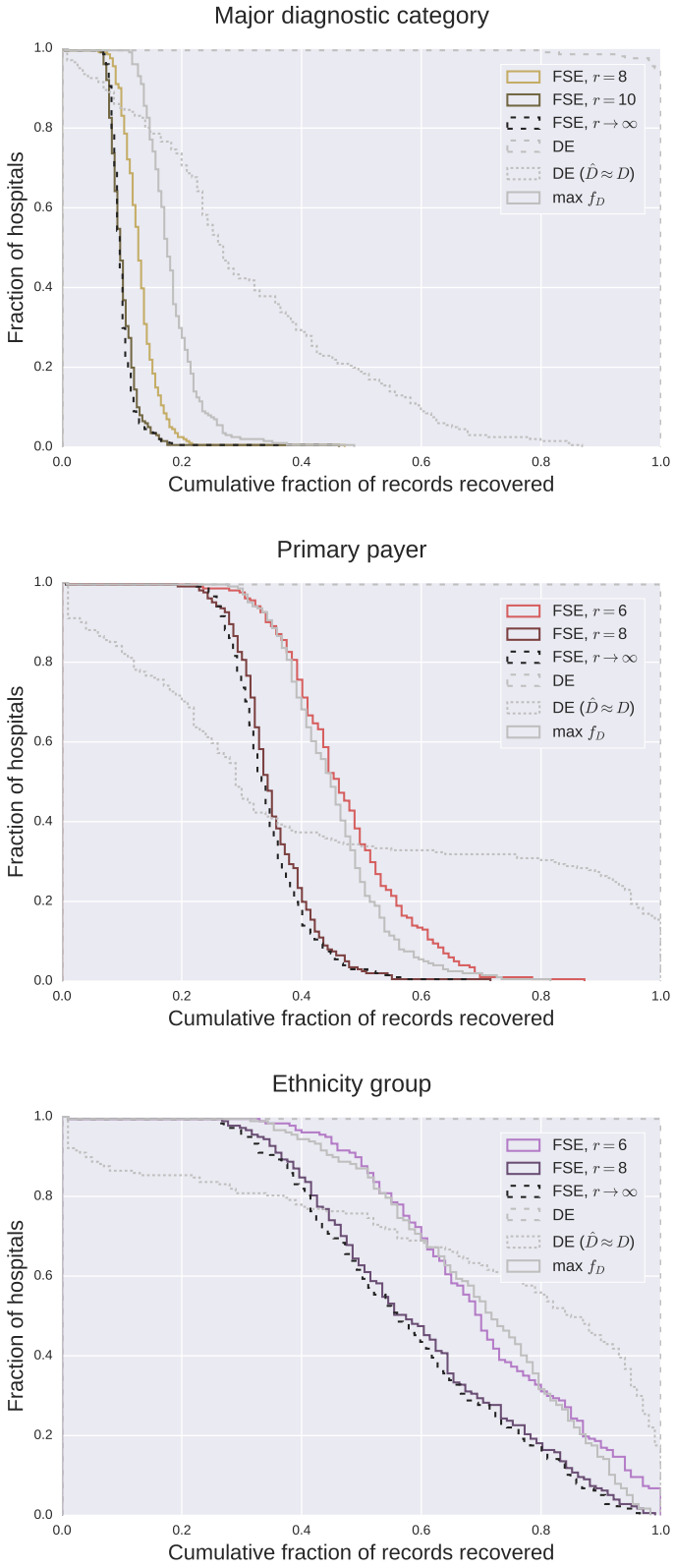


Figure 2.6: Our experimental results by attribute: complementary cumulative distributions (continued from previous page).

Chapter 3

Rank and access pattern attacks

Background. In 2016, Kellaris, Kollios, Nissim, and O'Neill (KKNO) published a paper about generic attacks on range query leakage [42]. KKNO were the first to derive such attacks that are *generic* (i.e., applied to classes of schemes, regardless of their implementation details) and *independent of data distribution*. One of the types of leakage targeted in KKNO's paper was access pattern leakage.

After reading KKNO's paper with Kenny Paterson and Brice Minaud, we observed that many schemes (q.v. Section 3.1) supporting range queries have an additional form of leakage, *rank leakage*. The rank leakage of a range query $[a, b]$ is the number of records with value strictly less than the left query endpoint, $\mathsf{rank}(a-1)$, and the number of records with value at most the right query endpoint, $\mathsf{rank}(b)$. We devised generic attacks exploiting rank and access pattern leakage. As expected, since our attacks use an additional kind of leakage, they require leakage from fewer queries to accomplish the same kind of exact reconstruction targeted by KKNO.

The resulting paper is "Improved Reconstruction Attacks on Encrypted Data Using Range Query Leakage," co-authored with Brice Minaud and Kenny Paterson. It was published at IEEE S&P 2018 [48] and the full version appears on the IACR's Cryptology ePrint Archive [47]. My main contribution to the paper was designing and implementing an evaluation of the inferred reconstruction attack (here in Section 3.4). This chapter also contains the following two contributions that were not in our paper. First, I provide an upper bound on the expected number of queries till diameter-δ approximate reconstruction succeeds (by adapting Proposition 3 [47, Appendix B.4]

to use rank leakage). Second, I demonstrate why the approximate reconstruction algorithm is not leakage-optimal by way of examples where our algorithm does not achieve diameter-δ approximate reconstruction, but an extension of it does. (This arises when the number of records in the database is relatively small.)

Persistent adversaries. In Chapter 2, we built a scheme that achieves security in a specific threat model: FSE is built for security against a snapshot adversary that obtains a one-time copy of the database, meant to model disk theft. Against a different type of threat, such as a persistent adversary that observes queries, FSE may provide much less security. Recall that with FSE, an equality query for one item is translated into as many equality queries as that item has homophones. If the adversary can count individual homophone queries and identify when the homophone queries for one item end and those for another item begin, then it knows the number of homophones for the queried value. Combined with knowledge of the frequencies of values in the database, it could then infer exactly what the item is by determining how many homophones each value should have. In this way, it could uniquely identify the queried value and which records have that value. In other words, if the adversary knows which set of encrypted values are homophones of a particular value, then frequency-smoothing encryption is effectively reduced to deterministic encryption.

In this (and further) chapters, the focus shifts to attacks and we no longer consider solutions for specific threat models. Instead, we consider *generic* attacks on abstract forms of leakage that can arise due to the scheme specification or implementation in various threat models.

For concrete examples of leakage, it may be helpful to consider what information may leak at two particular points: during setup and during query execution. Setup leakage could include properties such as the total number of rows (items) in the database, the number of columns (attributes), which items have the same values, or even the order of values in a column. Query leakage can include the query itself, the width of the range, or the number of results. How exactly these properties leak varies depending on the particular scheme or implementation. For instance, the size of the database could leak to a passive network adversary that observes traffic during the initial upload, or to a curious database server administrator that can monitor disk usage. The number of results to a particular query may also leak to a network adversary, or they may be encoded in a search index computed at the time of setup and available to a database administrator.

Rank and access pattern leakage. In this chapter, we consider attacks on two types of leakage: *rank* of values or records and *access pattern* of range queries. For a fixed attribute, the rank of a value is the number of items in the database whose value is less than or equal to that value. It is sometimes convenient to also define rank as a property of records: the rank of a record is the number of items in the database (including itself) whose value is less than or equal to this value. We define a rank function, $\mathsf{rank}(\cdot)$, that maps either a value or record to the number of records whose value is less than or equal to it:

$$\mathsf{rank}(\mathsf{r}^*) \coloneqq |\{\mathsf{r} \in \mathcal{R} : \mathsf{val}(\mathsf{r}) \leq \mathsf{val}(\mathsf{r}^*)\}|\,,$$
$$\mathsf{rank}(a) \coloneqq |\{\mathsf{r} \in \mathcal{R} : \mathsf{val}(\mathsf{r}) \leq a)\}|\,.$$

The access pattern of a range query is the set of record identifiers of the matching items, i.e., the identifiers for records whose value falls in the given range. For notational simplicity, define a null value 0 and set $\mathsf{rank}(0) \coloneqq 0$.

Chapter overview. Before presenting our attacks, we first discuss existing schemes that leak rank and access pattern in Section 3.1—these are all schemes to which our attacks apply. The three attacks presented in this chapter assume that the number of possible values, N, the total number of records, n_r, and the set of record identifiers, $\mathcal{R}$, are public. The first two attacks require data to be dense. Section 3.2 presents an *exact reconstruction* attack whose goal is to reconstruct the value of every record. If records can take values within $\{1, \ldots, N\}$, then the expected number of uniformly random range queries till our algorithm succeeds is at most $N \log N + N + 6.6\sqrt{N} + 7$.

In the following section, we consider a different goal: *approximately reconstructing* the value of every record by identifying an interval of diameter at most δN that contains the record's value. The expected number of queries until our diameter-δ approximate reconstruction algorithm succeeds is $(N+1)(\log(2/\delta)+4)$. Diameter-δ approximate reconstruction outputs two maps, $\widehat{\mathsf{val}}_{\mathsf{min}}$ and $\widehat{\mathsf{val}}_{\mathsf{max}}$, used to assign intervals of values $[\widehat{\mathsf{val}}_{\mathsf{min}}(\mathsf{r}), \widehat{\mathsf{val}}_{\mathsf{max}}(\mathsf{r})]$ to each record r with width $\widehat{\mathsf{val}}_{\mathsf{max}}(\mathsf{r}) - \widehat{\mathsf{val}}_{\mathsf{min}}(\mathsf{r}) \leq \delta N$. It is possible that some records are assigned intervals of width δN and some are assigned intervals of strictly smaller width. Recall the similar goal of ϵ-approximate reconstruction (q.v. Table 1.2). Although diameter-δ approximate reconstruction does imply to 2ϵ-approximate reconstruction, we use the different parameter δ to indicate that the output of a diameter-δ approximate reconstruction algorithm may provide strictly more information about the records' values.

Lastly, in Section 3.4, we craft a heuristic *inferred reconstruction* attack that uses an estimate of the data's distribution (the *auxiliary* distribution) to guess each record's value after even fewer queries. We design experiments to evaluate various aspects of this attack and simulate queries on age data from the HCUP medical records (q.v. Section 1.3).

3.1 Schemes that leak rank

We say that a scheme "leaks rank" if it is designed such that a party other than the client making the range query $[a, b]$ can learn the ranks of the endpoints, $\mathsf{rank}(a-1)$ and $\mathsf{rank}(b)$. For example, it could be an observer of the encrypted query tokens, or the server processing the query. In this section, we describe some schemes that leak rank and explain how they do so.

Consider a database DB whose records are retrieved via range queries on some fixed attribute. Without any encryption, the database clearly reveals rank to any observer: it can simply count the number of records with each value. However, some forms of OPE reveal rank in this setting as well. Suppose values are encrypted with equality-preserving OPE: for any two records r and r', $\mathsf{Enc}(\mathsf{val}(\mathsf{r})) < \mathsf{Enc}(\mathsf{val}(\mathsf{r}'))$ if and only if $\mathsf{val}(\mathsf{r}) < \mathsf{val}(\mathsf{r}')$. It is then still easy for an observer to count how many records have value less than or equal to a particular record r: it simply counts the number of records with encrypted value less than or equal to that record's encrypted value, $\mathsf{Enc}(\mathsf{val}(\mathsf{r}))$. This allows an adversary to learn ranks, but not exactly what values the ranks correspond to, unless the database is dense. In general, OPE and ORE are already vulnerable to other attacks based on the revealed orders of all records' values, e.g., [62]. With more general forms of OPE that do not preserve equality, rank does not necessarily leak from the static database since $\mathsf{val}(\mathsf{r}) = \mathsf{val}(\mathsf{r}')$ does not disallow any ordering of $\mathsf{Enc}(\mathsf{val}(\mathsf{r}))$ and $\mathsf{Enc}(\mathsf{val}(\mathsf{r}'))$. Some other schemes that are not vulnerable to a snapshot attacker but leak rank include the following.

First, the Lewi-Wu ORE scheme [51] leaks rank. This scheme can encrypt values with a "left" encryption function $\mathsf{Enc}_{\mathsf{L}}$ or a "right" encryption function $\mathsf{Enc}_{\mathsf{R}}$, and is able to compare only left and right ciphertexts. The encrypted values in the database would be "right" ciphertexts and, by themselves, semantically secure, but when compared with a query endpoint—a "left" ciphertext—reveal rank. The length of a "right" ciphertext is linear in the size of the plaintext space. A "left" ciphertext is simply a key and an index (identifying a particular component of a "right" ciphertext). Given a "left" and a "right" ciphertext, one may determine whether one is less than, equal

to, or greater than the other. Therefore, a query for items in the range $[a, b]$ would be encrypted as $(\mathsf{Enc}_{\mathsf{L}}(a), \mathsf{Enc}_{\mathsf{L}}(b))$ and would leak $\mathsf{rank}(a)$, $\mathsf{rank}(a-1)$, $\mathsf{rank}(b)$, and $\mathsf{rank}(b-1)$.

Second, the Arx-Range index [9], part of the Arx database system [67], also leaks rank. In this system, the server traverses a tree to identify which items (leaf nodes) fall in a particular range. The non-leaf nodes are chained garbled comparison circuits: the output of one garbled circuit determines which of its children to visit next, and provides the information required to encode the input to the next child. Specifically, each node is a boolean circuit with n inputs (corresponding to the n bits of the input value) and 1 output, representing the result of the comparison of the input with a fixed, secret value encoded in the circuit. The server traverses the index twice—once for each endpoint of the range—and returns all leaf nodes between the two endpoint leaves. A query for items in the range $[a, b]$ leaks to the server $\mathsf{rank}(a-1)$ and $\mathsf{rank}(b)$.

Third, Kerschbaum's frequency-hiding OPE [43] (and the corrected version of this scheme [53] that explicitly fixes a randomized order) leaks rank of the query endpoints. In this scheme, the client stores a tree index that has an entry for each value stored in the database (including repetitions). The need for storing this tree arises from the fact that repeated values are encrypted to different ciphertexts—the client needs to record whether a repeated value was inserted to the left or right of an existing value. When the client wants to query the database with a range $[a, b]$, it traverses its search tree twice: it finds the smallest encryption of a, $\min(a)$, and the largest encryption of b, $\max(b)$. Then, it sends the range $[\min(a), \max(b)]$ to the server. Since encryption is order-preserving, the server learns $\mathsf{rank}(a-1)$ and $\mathsf{rank}(b)$ simply by counting records.

Next, the Cipherbase [4] database system also leaks rank. Its custom range indexes are B-trees (shallow trees with high fan-out) created and updated by a trusted hardware module, but visible to the (untrusted) database server. Each entry is an IND-CPA ciphertext of the record identifier, but they are ordered by plaintext value. In other words, the range index reveals (randomized) order of records' values, e.g., if the items in a node are $\mathsf{Enc}(\mathsf{r}_1)$, $\mathsf{Enc}(\mathsf{r}_2)$, and $\mathsf{Enc}(\mathsf{r}_3)$, then $\mathsf{val}(\mathsf{r}_1) \leq \mathsf{val}(\mathsf{r}_2) \leq \mathsf{val}(\mathsf{r}_3)$. This passive leakage of order—but not equality—is similar to Kerschbaum's frequency-hiding OPE scheme. As a range is queried, the trusted hardware traverses the index to find the endpoints of the query in the B-tree. Therefore, the untrusted server learns $\mathsf{rank}(a-1)$ and $\mathsf{rank}(b)$.

Another scheme, Roche *et al.*'s Partial Order Preserving Encoding (POPE) [72], offers frequency-hiding while aiming to prevent direct comparisons between many cipher-

texts. Their scheme is suited to settings where many new records are continuously added and there are relatively few range queries. POPE uses a structure similar to a B-tree where nodes are augmented with buffers of unsorted items. Ciphertexts are semantically secure; sorting is done by sending them to the client. Specifically, when a range query is issued, the server traverses the tree to find the encrypted endpoints. For every node encountered along the two paths, it sends the client the pivot values and unsorted buffer. The client helps empty the buffer by decrypting the items and telling the server which child buffer to move them to. In the end, there will be paths from the root to two particular leaves, and these leaves will be the only nodes in the path with non-empty buffers. Finally, the server answers the range query by sending the client all items in the unsorted buffers of every node between the two query endpoints. Therefore, after a range query for $[a, b]$, the server learns $\mathsf{rank}(a-1)$ by counting items to the "left" of the left endpoint and $\mathsf{rank}(b)$ by counting items to the "left" of the right endpoint.

All of the aforementioned schemes also leak access pattern to the server. Specifically, after the client queries a range $[a, b]$, the server learns the subset of records $\mathsf{R}_{[a,b]} \coloneqq \{\mathsf{r} \in \mathcal{R} : a \leq \mathsf{val}(\mathsf{r}) \leq b\}$.

3.2 Exact reconstruction for dense data

Our first algorithm reconstructs the value of each record given rank leakage and access pattern leakage from some queries. Specifically, suppose that for every queried range $[a, b]$, the ranks $\mathsf{rank}(a-1)$ and $\mathsf{rank}(b)$, and the set of matching records $\mathsf{R}_{[a,b]}$ leak. Since the exact query endpoints a and b are not known, represent this leakage by $x_i \coloneqq \mathsf{rank}(a-1)$, $y_i \coloneqq \mathsf{rank}(b)$, and $\mathsf{R}_i \coloneqq \mathsf{R}_{[a,b]}$. The goal is to output a mapping $\widehat{\mathsf{val}}$ such that for all records $\mathsf{r} \in \mathcal{R}$, $\widehat{\mathsf{val}}(\mathsf{r}) = \mathsf{val}(\mathsf{r})$.

Reasoning behind the algorithm. The idea behind this algorithm is to partition the set of records based on which subset of queries matched them, and assign values to each group using rank information. If two records have the same value, then every possible range query must match both or neither of those records. If two records have different values, then there exists a range query that matches one record but not the other. If all queries are issued with non-zero probability, then the set of records will eventually be divided into equivalence classes that correspond exactly to groups of records having the same value. We refer to this set of equivalence classes as the partition of records, $\mathcal{P}_R$.

Once records are grouped in this manner, values must be assigned to them. Suppose the records $\mathcal{R}$ were in a list ordered by value with ties broken arbitrarily. In such a list, records with the smallest value, 1, would be in positions 1 through $\mathsf{rank}(1)$, records with value 2 would be in positions $\mathsf{rank}(1)+1$ through $\mathsf{rank}(2)$, and so on. The leakage (x_i, y_i, R_i) can be interpreted as a restriction on the positions of the records R_i in this list: they must occupy positions $x_i + 1$ through y_i. Since there is a bijection between records and their position in this list (or many such bijections when there are multiple records with the same value), the positions 1 through $|\mathcal{R}|$ can be divided into some number of intervals based on which queries "matched" those positions. We refer to this set of intervals as the partition of positions, $\mathcal{P}_P$.

When not all queries have been issued, it is possible for two non-consecutive intervals to correspond to records matched by the same set of queries. For example, suppose the number of possible values is $N = 3$ and the issued queries are $[1, 3]$ and $[2, 2]$. Rank leakage reveals the *number* of records with each value, but since no query so far has matched only records with value 1 and not 3 (or *vice versa*), the partition of records has 2 classes, while the partition of positions has 3. Forcing the partition of positions to consist of *intervals* means that two non-neighboring intervals could match exactly the same set of queries, so the number of intervals $|\mathcal{P}_P|$ is at least the number of elements in the partition of records $|\mathcal{P}_R|$.

Once the number of equivalence classes of records $|\mathcal{P}_R|$ equals N, each equivalence class of positions must correspond to a single interval—the positions occupied by records with that particular value. Thus, values are assigned to partitions of records based on the index of the class of positions that matched exactly the same set of queries.

Algorithm details. Pseudocode is in Algorithm 3.1. First, the partitions of records $\mathcal{R}$ and positions $\{1, \ldots, |\mathcal{R}|\}$ are created by building the two maps, Q_R and Q_P, which record the sets of queries that match each record and the sets of queries that match each position respectively. Two records r and r' are in the same equivalence class iff the set of queries for which $\mathsf{r} \in \mathsf{R}_i$ is exactly the set of queries for which $\mathsf{r}' \in \mathsf{R}_i$. The set $\{1, \ldots, |\mathcal{R}|\}$ represents the positions of records in a hypothetical list where they are sorted by value, so two positions j and j' are in the same equivalence class iff the set of queries for which $j \in [x_i + 1, y_i]$ is exactly the set of queries for which $j' \in [x_i + 1, y_i]$.

Let (x_i, y_i, R_i) be the leakage of the ith query, where x_i is the rank of 1 less than the left query endpoint, y_i is the rank of the right query endpoint, and R_i is the set of

Algorithm 3.1 Full reconstruction attack with rank and access pattern leakage

Input: query leakage $\{(x_i, y_i, \mathsf{R}_i)\}_{i=1}^{\mathsf{n_q}}$.

Output: $\perp$ or map $\widehat{\mathsf{val}} : \mathcal{R} \to \{1, \ldots, N\}$ such that $\widehat{\mathsf{val}}(\mathsf{r}) = \mathsf{val}(\mathsf{r})$ for all $\mathsf{r} \in \mathcal{R}$.

1: $\widehat{\mathsf{val}}$, Q_R, $Q_P \leftarrow$ empty maps
2: **for all** $i \in \{1, \ldots, \mathsf{n_q}\}$ **do**
3: **for all** $\mathsf{r} \in \mathsf{R}_i$ **do**
4: $Q_R[\mathsf{r}] \leftarrow Q_R[\mathsf{r}] \cup \{i\}$
5: **for all** $j \in \{x_i + 1, \ldots, y_i\}$ **do**
6: $Q_P[j] \leftarrow Q_P[j] \cup \{i\}$
7: Form partition $\mathcal{P}_R$ of $\mathcal{R}$ with equivalence relation defined by $\mathsf{r} \equiv \mathsf{r}' \Leftrightarrow Q_R[\mathsf{r}] = Q_R[\mathsf{r}']$
8: **if** $|\mathcal{P}_R| \neq N$ **then**
9: **return** $\perp$
10: Form partition $\mathcal{P}_P$ of $\{1, \ldots, |\mathcal{R}|\}$ with equivalence relation defined by $j \equiv j' \Leftrightarrow Q_P[j] = Q_P[j']$
11: Order $\mathcal{P}_P = ([J_1], \ldots, [J_N])$ by $[J] < [J']$ iff $j < j'$ for all $j \in [J]$ and all $j' \in [J']$
12: **for all** $[R] \in \mathcal{P}_R$ **do**
13: $\hat{k} \leftarrow k \in \{1, \ldots, N\}$ such that $Q_R[R] = Q_P[J_k]$
14: **for all** $\mathsf{r} \in [R]$ **do**
15: $\widehat{\mathsf{val}}(\mathsf{r}) \leftarrow \hat{k}$
16: **return** $\widehat{\mathsf{val}}$

records in $\mathcal{R}$ that match this range. First, $Q_R[\mathsf{r}]$ is updated with the query index i for all records r in R_i (line 4), and $Q_P[j]$ is similarly updated for all positions j in $[x_i+1, y]$ (line 6). After processing all queries, the records are partitioned by which set of queries they matched ($\mathcal{P}_R$, line 7). If the number of equivalence classes of records, $|\mathcal{P}_R|$, is not the number of possible values a record can have, N, then it is not possible to assign a value to each record—either not enough queries have been observed yet, or not every value appears in at least one record—so the algorithm aborts. If $|\mathcal{P}_R| = N$, however, then the partition of positions $\mathcal{P}_P$ is also formed (line 10) and its classes ordered in the natural way (line 11). Recall that since $|\mathcal{P}_P| \geq |\mathcal{P}_R| = N$, the partition of positions will consist of N intervals of positions, so this ordering is well defined.

Finally, each class of records is assigned a class of positions, specifically, the one that matched exactly the same set of queries. The records in each class are assigned the value equal to the index of its assigned class of positions (lines 12–15). Records whose position is in the first equivalence class must have value 1, records whose position is in the second equivalence class must have value 2, and so on.

3.2.1 Leakage optimality

Algorithm 3.1 is optimal in the sense that it succeeds whenever any other *correct* algorithm succeeds on the same leakage. A correct algorithm is one that, for each possible input, either (a) with probability 1, returns $\perp$ (failure), or (b) with probability

Table 3.1: Examples of non-dense databases that are indistinguishable with only rank and access pattern leakage with $N = 5$.

	DB_1	DB_2	DB_3	DB_4	DB_5
$\mathsf{val}(\mathsf{r}_1)$	1	1	1	1	2
$\mathsf{val}(\mathsf{r}_2)$	1	1	1	1	2
$\mathsf{val}(\mathsf{r}_3)$	2	2	2	3	3
$\mathsf{val}(\mathsf{r}_4)$	3	3	4	4	4
$\mathsf{val}(\mathsf{r}_5)$	4	5	5	5	5

1, returns a map $\widehat{\mathsf{val}}$ that agrees with the map val defining any database consistent with the given leakage (success). In particular, a correct algorithm must always return $\perp$ unless its input leakage specifies a *unique* database.

For some fixed set of record identifiers $\mathcal{R}$, if there exist two value mappings val and val' that generate the same leakage (possibly from different range queries), then it is impossible for an algorithm to choose the correct one with probability 1. One way in which the same query leakage could arise from two sets of queries on two distinct databases (defined by their value mappings) is when the database is not dense. Suppose the number of possible values a record can have is $N = 5$. Then, for any range query and any two of the non-dense databases in Table 3.1, there exists another range query such that the leakage of the first query on the first database is identical to the leakage from the second query on the second database.

For instance, leakage $(0, 3, \{\mathsf{r}_1, \mathsf{r}_2, \mathsf{r}_3\})$ could arise from query $[1, 2]$ on DB_1, DB_2, or DB_3, from query $[1, 3]$ on DB_4, or from query $[1, 3]$ or $[2, 3]$ on DB_5. When not all values appear in at least one record, there is no way to distinguish which specific value does not appear using only rank and access pattern leakage.

Another way in which the same query leakage could arise from two distinct databases is when not every pair of adjacent values has been "split" by at least one query. The following proposition proves that Algorithm 3.1 is leakage-optimal by showing how to construct different databases, with different value mappings, that yield the same leakage when the algorithm fails.

Proposition 3.1. *Let A be any correct exact reconstruction algorithm that takes as input the rank and access pattern leakage from some range queries on a set of records $\mathcal{R}$ that have integer values between 1 and N. Then, whenever A succeeds on a particular input, so does Algorithm 3.1.*

Proof. Suppose that Algorithm 3.1 did not succeed, so the number of groups in the partition of records $\mathcal{P}_R$ is strictly less than N (line 9). Either all of the records in each class have the same value, or there is at least one class containing records with two or more different values.

In the former case, the number of different values appearing in the database must be less than N, i.e., the database is not dense. Then, there exist two adjacent values v^* and $v^* + 1$ in $\{1, \ldots, N\}$ such that one of them is in the range/image of val and the other is not. Suppose v^* is the one in val's range/image—the other case is similar—and let R^* be the set of records whose value is v^*. Consider another mapping val' that agrees with val on every record except the ones in R^*, which instead have $\mathsf{val}'(\mathsf{r}) = v^* + 1$. Then, the same query leakage could arise from a (possibly different) set of queries, as the following observations indicate:

- Any range of the form $[a, v^*]$ has the same leakage with val as $[a, v^* + 1]$ with val'.
- The range $[v^* + 1, v^* + 1]$ has the same leakage with val as $[v^*, v^*]$ with val'.
- Any range of the form $[v^* + 1, b]$ with $b > v^* + 2$ has the same leakage with val as $[v^* + 2, b]$ with val'.
- Any range $[a, b]$ with $a \neq v^* + 1$ and $b \neq v^*$ already matches both or neither of v^* and $v^* + 1$, and therefore will have the same leakage with val'.

Thus, it is possible to form another value mapping that gives rise to the same leakage, and therefore no correct algorithm A can succeed on this input.

In the latter case, by the pigeonhole principle, there must exist a class of records $[R]$ of $\mathcal{P}_R$ containing two sets of records R_1^* and R_2^* (of size at least 1 each) with values v_1^* and $v_2^* \neq v_1^*$. Since these two sets of records are in the same partition, every query either returned both sets of records or neither. Hence, swapping their values would not change the leakage, so no correct algorithm A would succeed on this same input either. □

3.2.2 Query analysis

The leakage optimality of Algorithm 3.1 means that if it fails, full reconstruction is impossible for any correct algorithm given the same query leakage: this algorithm requires at most as many queries as any other correct algorithm. The actual number of queries required, however, depends on their distribution. A necessary (but definitely not sufficient) condition of the query distribution is that every value is in at least one

range. Suppose all $N(N+1)/2$ ranges occur with uniform probability. Then, as the next theorem shows, the expected number of required queries for the algorithm to succeed is $\mathcal{O}(N \log N)$.

Theorem 3.2. *Let N be the number of possible values a record can have, and let* DB *be a set of records where each value appears in at least one record. Then, the probability that Algorithm 3.1 fails given the leakage from $\mathsf{n_q}$ queries drawn uniformly at random is at most*

$$2 \cdot e^{-\mathsf{n_q}/(2N+2)} + N \cdot e^{-\mathsf{n_q}/N}.$$

Proof. Let $\{[a_i, b_i]\}_{i=1}^{\mathsf{n_q}}$ be the set of sampled queries and $\{(x_i, y_i, \mathsf{R}_i)\}_{i=1}^{\mathsf{n_q}}$ be the corresponding leakage. Assume N is even. Algorithm 3.1 succeeds iff the partition $\mathcal{P}_R$ contains N elements, which is equivalent to the following: for each value $v \in \{1, \ldots, N\}$,

$$\{v\} = \left(\bigcap \{[a_i, b_i] : v \in [a_i, b_i]\}\right) \setminus \left(\bigcup \{[a_i, b_i] : v \notin [a_i, b_i]\}\right). \tag{3.1}$$

That is, the smallest intersection of all ranges containing the value v, minus any range not containing v, must be only the singleton set v.

Claim 3.3. *This event is implied by the following three events:*

1. *There is at least one query with $a = v$ for each v from 1 to $N/2$.*
2. *There is at least one query with $b = v$ for each v from $N/2 + 1$ to N.*
3. *There are two queries whose union is $[N/2 + 1, N]$.*

Proof of Claim 3.3. First, let v be any value in $\{1, \ldots, N/2\}$, and suppose there is some other $w \neq v \in \{1, \ldots, N\}$ in the set on the right-hand side of Equation 3.1. Event 1 implies that there is a query $[a, b]$ with $a = v$, so v is the smallest item in the intersection of sets on the right-hand side of Equation 3.1. Therefore, w must be strictly greater than v. Event 1 also implies that there are queries $[a, b]$ with $a = v+1$, $a = v + 2$, and so on up to $a = N/2$, so the set $\{v + 1, \ldots, N/2\}$ will be in the union of sets on the right-hand side of Equation 3.1. Thus, w, if it exists, must be strictly greater than $N/2$. Event 3 implies that the union of sets on the right-hand side of Equation 3.1 contains all values between $N/2 + 1$ and N, so there can exist no other $w \neq v$ in the set.

If v is in $\{N/2 + 1, \ldots, N\}$, then Event 3 implies that the intersection of sets on the right-hand side of Equation 3.1 is contained in $\{N/2 + 1, \ldots, N\}$, so the entire set must be contained in it as well. Therefore, w must be at least $N/2 + 1$. Event 2 implies that there is a query $[a, b]$ with $b = v$, so v must be the largest value in the

intersection of sets on the right-hand side of Equation 3.1. Therefore, w must be in $\{N/2+1, \ldots, v-1\}$. However, Event 2 also implies that there are queries $[a, b]$ with $b = N/2+1$, $b = N/2+2$, and so on up to $b = v-1$, which will be in the union on the right-hand side of Equation 3.1. Thus, no other $w \neq v$ can be in the final set. This concludes the proof of Claim 3.3. □

Next, returning to the proof of Theorem 3.2, observe that Event 1 and Event 2 occur with the same probability due to reflection, so consider only Event 1 for now. The probability that Event 1 does not occur, i.e., that at least one value $v \in \{1, \ldots, N/2\}$ does *not* arise as the left endpoint of any of the $\mathsf{n_q}$ queries, is

$$\begin{aligned}
\text{Prob}[\neg\text{Event 1}] &\leq \sum_{v=1}^{N/2} \text{Prob}[a_i \neq v \text{ for all } i] && \text{(union bound)} \\
&= \sum_{v=1}^{N/2} \left(1 - \frac{N-(v-1)}{N(N+1)/2}\right)^{\mathsf{n_q}} && \text{(uniformity of queries)} \\
&\leq \left\lfloor \frac{N}{2} \right\rfloor \left(1 - \frac{N-(N/2-1)}{N(N+1)/2}\right)^{\mathsf{n_q}} \\
&\leq \frac{N}{2} \left(1 - \frac{N-(N/2-1)}{N(N+1)/2}\right)^{\mathsf{n_q}} \\
&= \frac{N}{2} \left(1 - \frac{N+2}{N(N+1)}\right)^{\mathsf{n_q}} \\
&\leq \frac{N}{2} \left(1 - \frac{N+1}{N(N+1)}\right)^{\mathsf{n_q}} \\
&= \frac{N}{2} \left(1 - \frac{1}{N}\right)^{\mathsf{n_q}} \\
&\leq \frac{N}{2} \left(e^{-\frac{1}{N}}\right)^{\mathsf{n_q}} && \text{(Bound A.7).}
\end{aligned}$$

Event 3 can happen in a number of ways. We lower-bound the probability of it happening by considering a special case that can be interpreted as the intersection of two events, and therefore can apply a union bound to the complement of these events. Consider only pairs of queries of the form $[N/2+1, b]$ and $[a, N]$, where in the "left" queries, $b \in \{\lfloor 3N/4 \rfloor, \ldots, N\}$ and in the "right" queries, $a \in \{N/2+1, \ldots, \lfloor 3N/4 \rfloor + 1\}$. Then, the union of any "left" query and any "right" query is $\{N/2+1, \ldots, N\}$, the desired set. The number of possible values of b in a "left" query is

$$N - \lfloor 3N/4 \rfloor + 1 \geq N - (3N/4) + 1 = N/4 + 1 \geq N/4,$$

while there are

$$\lfloor 3N/4 \rfloor + 1 - N/2 \geq (3N/4 - 1) + 1 - (N/2) \geq N/4$$

options for a in a "right" query. Therefore, the probability that any single query is *not* a "left" query (or not a "right" query) is at most $1 - \frac{N/4}{N(N+1)/2} = 1 - \frac{1}{2(N+1)}$ and the probability that Event 3 does not occur is

$$\begin{aligned}
\text{Prob}[\neg\text{Event 3}] &\leq \text{Prob}[\text{no "left"} \cup \text{ no "right"}] \\
&\leq \text{Prob}[\text{no "left"}] + \text{Prob}[\text{no "right"}] \qquad \text{(union bound)} \\
&\leq 2\left(1 - \frac{1}{2(N+1)}\right)^{\mathsf{n_q}} \\
&\leq 2\left(e^{-\frac{1}{2(N+1)}}\right)^{\mathsf{n_q}} \qquad \text{(Bound A.7)}.
\end{aligned}$$

Finally, the probability that Algorithm 3.1 fails is at most

$$\begin{aligned}
&\text{Prob}[\neg\text{Event 1}, \neg\text{Event 2, or } \neg\text{Event 3}] \\
&\leq \text{Prob}[\neg\text{Event 1}] + \text{Prob}[\neg\text{Event 2}] + \text{Prob}[\neg\text{Event 3}] \qquad \text{(union bound)} \\
&\leq N \cdot e^{-\mathsf{n_q}/N} + 2 \cdot e^{-\mathsf{n_q}/(2N+2)},
\end{aligned}$$

thus concluding the proof of Theorem 3.2. □

Therefore, given the leakage from $\mathsf{n_q}$ uniformly sampled queries, Algorithm 3.1 succeeds with probability at least $1 - N \cdot e^{-\mathsf{n_q}/N} - 2 \cdot e^{-\mathsf{n_q}/(2N+2)}$. Next, we determine an upper bound on the expected number of uniformly random queries until Algorithm 3.1 succeeds. This bound has slightly worse constants than the one in the full version of our paper [47, Proposition 1, Appendix B.2], but applies to any $N \geq 4$ instead of $N \geq 27$.

Corollary 3.4. *The expected number of queries until Algorithm 3.1 succeeds is at most* $N \log N + \mathcal{O}(N)$. *Specifically, for* $N \geq 4$, *it is at most* $N \log N + N + 6.6\sqrt{N} + 7$.

Proof. Let X be a random variable representing the number of queries (sampled uniformly at random) until the algorithm succeeds. By Theorem 3.2, we have

$$\text{Prob}[\mathsf{X} \leq x] \geq 1 - N \cdot e^{-x/N} - 2 \cdot e^{-x/(2N+2)},$$

or, written as an inverse cumulative distribution function,

$$\text{Prob}[\mathsf{X} \geq x] \leq N \cdot e^{-(x-1)/N} + 2 \cdot e^{-(x-1)/(2N+2)}.$$

By Formula A.4, the expected value is

$$\begin{aligned}
\mathbb{E}[\mathsf{X}] &= \sum_{x=1}^{\infty} \text{Prob}[\mathsf{X} \geq x] \\
&= \sum_{x=1}^{\lceil N \log N \rceil} \text{Prob}[\mathsf{X} \geq x] + \sum_{x=\lceil N \log N \rceil + 1}^{\infty} \text{Prob}[\mathsf{X} \geq x] \\
&\leq N \log N + 1 + \left(\sum_{x=\lceil N \log N \rceil}^{\infty} N \cdot e^{-x/N} \right) + \left(\sum_{x=\lceil N \log N \rceil}^{\infty} 2 \cdot e^{-x/(2N+2)} \right).
\end{aligned}$$

The first sum satisfies

$$\begin{aligned}
\sum_{x=\lceil N \log N \rceil}^{\infty} N \cdot e^{-x/N} &= N \cdot \left(\frac{e^{-\lceil N \log N \rceil / N}}{1 - e^{-1/N}} \right) && \text{(Formula A.6)} \\
&\leq N \cdot \left(\frac{e^{-\log N}}{1 - e^{-1/N}} \right) \\
&= \frac{1}{1 - e^{-1/N}} \\
&\leq N + 1 && \text{(Bound A.10).}
\end{aligned}$$

Similarly, the second sum satisfies

$$\begin{aligned}
&\sum_{x=\lceil N \log N \rceil}^{\infty} 2 \cdot e^{-x/(2N+2)} \\
&= 2 \cdot \sum_{x=\lceil N \log N \rceil}^{\infty} e^{-x/(2N+2)} \\
&= 2 \cdot \left(\frac{e^{-\lceil N \log N \rceil / (2N+2)}}{1 - e^{-1/(2N+2)}} \right) && \text{(Formula A.6)} \\
&\leq 2 \cdot \left(\frac{e^{-(N \log N)/(2N+2)}}{1 - e^{-1/(2N+2)}} \right) \\
&= 2 \cdot \left(\frac{e^{-(\log N)/2} \cdot e^{\log N/(2N+2)}}{1 - e^{-1/(2N+2)}} \right) && \left(\frac{N \log N}{2(N+1)} = \frac{\log N}{2} \left(1 - \frac{1}{N+1} \right) \right) \\
&= \frac{2}{\sqrt{N}} \cdot \left(\frac{e^{\log N/(2N+2)}}{1 - e^{-1/(2N+2)}} \right) \\
&\leq \frac{2}{\sqrt{N}} \cdot \left(\frac{e^{(N-1)/(2N+2)}}{1 - e^{-1/(2N+2)}} \right) && \text{(Bound A.8)} \\
&\leq \frac{2}{\sqrt{N}} \cdot \left(\frac{\sqrt{e}}{1 - e^{-1/(2N+2)}} \right) && \left(\frac{N-1}{2N+2} = \frac{1 - 1/N}{2 + 2/N} < \frac{1}{2} \right) \\
&\leq \frac{2\sqrt{e}}{\sqrt{N}} \cdot (2N+3) && \text{(Bound A.10)} \\
&= 4\sqrt{e} \cdot \sqrt{N} + \frac{6\sqrt{e}}{\sqrt{N}} \\
&\approx 6.6\sqrt{N} + \frac{9.9}{\sqrt{N}}.
\end{aligned}$$

Combining these inequalities, we obtain

$$\begin{aligned}\mathbb{E}[\mathsf{X}] &\leq N \log N + 1 + N + 1 + 6.6\sqrt{N} + \frac{9.9}{\sqrt{N}} \\ &\leq N \log N + N + 6.6\sqrt{N} + 7 \qquad \text{for } N \geq 4. \qquad \square\end{aligned}$$

3.2.3 Efficiency analysis

The maps Q_R and Q_P in Algorithm 3.1 each contain one entry for every record in the database, and each entry is mapped to a set of up to $\mathsf{n_q}$ integers, representing queries. These two maps represent the largest storage requirements of the algorithm. There are two primary ways to reduce their size. They are described in detail in this section and also apply to the approximate and inferred reconstruction algorithms in the following sections.

First, consider the sets of query identifiers in each of the maps: their only purpose is to group records and positions by the queries they matched (lines 7 and 10) and then match each group of records with a group of positions (line 13). Since there is no need to recover the queries that matched a record, these query identifier sets can be represented more efficiently using one-way, collision-resistant maps. Let $\mathcal{H}(\cdot)$ be a collision-resistant hash function mapping arbitrary-length binary strings to $\ell_{\mathcal{H}}$-bit strings. Instead of simply appending a query identifier to an entry in Q_R or Q_P, all entries in these two maps could be initialized to $0^{\ell_{\mathcal{H}}}$ and lines 4 and 6 in Algorithm 3.1 could be replaced with the following:

4: $\quad Q_R[\mathsf{r}] \leftarrow \mathcal{H}(Q_R[\mathsf{r}] \parallel i)$

6: $\quad Q_P[j] \leftarrow \mathcal{H}(Q_P[j] \parallel i)$

As long as each query is processed before the next (in the loop starting on line 2), such a simple incremental sequence hashing technique suffices for correctness of the algorithm and reduces the size of Q_R and Q_P to be independent of the number of queries, $\mathsf{n_q}$.

Second, the representation of the map Q_P can be scaled down from containing $|\mathcal{R}|$ elements to containing about N elements in a sorted list. Instead of recording the set of queries that match every position from 1 to $|\mathcal{R}|$, it is sufficient to record the set of queries at only the positions where this set changes. Each item in the list is a pair (j, h), where j is a position and h is a sequence hash, representing a set of query identifiers. The list is sorted by the value of j and initially contains only the items $(0, 0^{\ell_{\mathcal{H}}})$ and $(|\mathcal{R}| + 1, 0^{\ell_{\mathcal{H}}})$. These two items will never be modified, but serve

as artificial boundaries that ensure endpoints are properly handled when $y = |\mathsf{DB}|$ (lines 6c and 6e).

Let (j_k, h_k) be the kth entry in this sorted list. Its meaning can be interpreted as "the items in positions $\{j_k, j_k+1, \ldots, j_{k+1}-1\}$ match the set of queries represented by the sequence hash h_k." To handle leakage from the ith query, (x_i, y_i, R_i), the list must be updated to reflect the following properties: (i) elements in positions $x_i+1, \ldots, y_i$ matched this range, (ii) elements in positions less than or equal to x_i did not, and (iii) elements in positions greater than or equal to y_i+1 did not. For simplicity of the algorithm, these properties will be addressed in the order (iii), (i), and (ii). First, the list is scanned for an entry with position equal to y_i+1, and if it does not yet exist, (y_i+1, h) is inserted into it, where h is a copy of the sequence hash of the *previous* element in the list. Next, the list is scanned for an entry with position equal to x_i+1. If it does not yet exist, (x_i+1, h) is inserted into the list, where h is a copy of the sequence hash of the *previous* element in the list. Finally, for every item in the list with position greater than or equal to x_i+1 and at most y_i, its sequence hash is updated to include the current query index i.

The list Q_P contains at most $N+2$ entries of the form (j, h); the possible positions j are $\mathsf{rank}(0) \coloneqq 0$, $\mathsf{rank}(1), \ldots, \mathsf{rank}(N)$, and $\mathsf{n_r}+1$. Recall that when forming the partitions of positions and records as on lines 7 and 10, they have exactly the same size. With the list representation of Q_P, however, this is harder to see, since there can be two non-adjacent positions j and j' that matched exactly the same set of queries. For example, with query leakage $\{(1, 5, \mathsf{R}_1), (2, 4, \mathsf{R}_2)\}$, the partition of positions Q_P would be represented as $\{(0, 0^{\ell_{\mathcal{H}}}), (1, \mathcal{H}(0^{\ell_{\mathcal{H}}} \parallel 1)), (2, \mathcal{H}(\mathcal{H}(0^{\ell_{\mathcal{H}}} \parallel 1) \parallel 2))$, $(5, \mathcal{H}(0^{\ell_{\mathcal{H}}} \parallel 1)), \ldots, (\mathsf{n_r}+1, 0^{\ell_{\mathcal{H}}})\}$.

Specifically, using these two efficiency improvements results in the changes to Algorithm 3.1 specified in Figure 3.1. When one line in the original algorithm is replaced by multiple lines, a letter is appended to the original number. Using the sequence hashing technique and the compressed representation of Q_P, the main loop (lines 2–6) involves computing at most $2\sum_{i=1}^{\mathsf{n_q}} |\mathsf{R}_i| \leq 2\mathsf{n_q}\,|\mathcal{R}|$ hashes. Partitioning the set of records based on Q_R (line 7) requires scanning all $|\mathcal{R}|$ of its entries, while matching the N equivalence classes (line 13) requires $\mathcal{O}(N \log N)$ operations.

This concludes the analysis of Algorithm 3.1, which fully reconstructs the values of dense data given rank and access pattern leakage. If this algorithm fails, then full reconstruction is impossible by any other correct algorithm given the same query leakage. The expected number of queries required for it to succeed is $N \log N + \mathcal{O}(N)$.

1: $\widehat{\mathsf{val}}$, $Q_R \leftarrow$ empty maps; $Q_P \leftarrow \big((0, 0^{\ell_{\mathcal{H}}}), (|\mathcal{R}|+1, 0^{\ell_{\mathcal{H}}})\big)$
5: $\big((j_1, h_1), \ldots, (j_{|Q_P|}, h_{|Q_P|})\big) \leftarrow Q_P$ where $j_1 < \cdots < j_{|Q_P|}$
6a: **if** $\nexists k$ such that $j_k = y_i + 1$ **then**
6b: Find max k' such that $j_{k'} < y_i + 1$ and $(j_{k'}, h_{k'}) \in Q_P$
6c: Insert $(y_i + 1, h_{k'})$ into Q_P after $(j_{k'}, h_{k'})$
6d: **if** $\nexists k$ such that $j_k = x_i + 1$ **then**
6e: Find max k' such that $j_{k'} < x_i + 1$ and $(j_{k'}, h_{k'}) \in Q_P$
6f: Insert $(x_i + 1, h_{k'})$ into Q_P after $(j_{k'}, h_{k'})$
6g: **for** $(j, h) \in Q_P : j \in \{x_i + 1, \ldots, y_i\}$ **do**
6h: Replace h with $\mathcal{H}(h \parallel i)$ in Q_P
10: $\big((j_0, 0^{\ell_{\mathcal{H}}}), (j_1, h_1), \ldots, (j_N, h_N), (j_{N+1}, 0^{\ell_{\mathcal{H}}})\big) \leftarrow Q_P$ where $0 = j_0 < j_1 < \cdots < j_N < j_{N+1} = |\mathcal{R}| + 1$
11: Define $\mathcal{P}_P = ([J_1], \ldots, [J_N])$ where $[J_i] = \{j_i, \ldots, j_{i+1} - 1\}$
13: $\hat{k} \leftarrow k \in \{1, \ldots, N\}$ such that $(k, Q_R[R]) \in Q_P$

Figure 3.1: Changes to Algorithm 3.1 to improve efficiency

3.3 Approximate reconstruction for dense data

For full reconstruction to succeed, the query leakage must induce a partition of records $\mathcal{P}_R$ of size N (line 8 in Algorithm 3.1). Corollary 3.4 proved that the leakage from $N \log N + \mathcal{O}(N)$ uniformly random queries are sufficient for this condition to hold. Observing the leakage from so many queries may not always be realistic (assuming the bound is somewhat tight), so we now adapt the algorithm to reconstruct values *approximately* with fewer queries.

When the size of the partition of records $\mathcal{P}_R$ is strictly less than N, all is not lost. Recall that the number of classes in the partition of positions $\mathcal{P}_P$ is at least the number of classes in the partition of records: since the classes of positions must be intervals, there can be more than the number of classes of records. In particular, one class of records can correspond to multiple classes of position intervals. Therefore, when $|\mathcal{P}_R| < N$, the algorithm can no longer assign a single value to each equivalence class of records: it must assign an interval of values. The size of this interval depends on both the number of classes of positions, $|\mathcal{P}_P|$, and how many classes of positions matched the same set of queries. First, each class of *positions* is assigned a range of values: given $|\mathcal{P}_P|$ such classes, the kth class is assigned the range k through $k + N - |\mathcal{P}_P|$, since the "extra" $N - |\mathcal{P}_P|$ values that do not have their own class could be part of any of the $|\mathcal{P}_P|$ classes of positions. Next, each class of records is matched to its corresponding class(es) of positions by identifying those that matched the same set of queries. If there is only one such class of positions, records in that class are assigned its corresponding interval of values (of width $N - |\mathcal{P}_P|$). If there are multiple classes of positions that matched the same set of queries as a particular class

of records, then the interval of values assigned to it is simply the smallest interval containing all of those intervals of values, i.e., it ranges from the smallest value of any matching class of positions to the largest value of any matching class of positions.

If the width of the interval (i.e., the difference between the greatest and least values in it) assigned to each class of records is at most δN, then the algorithm has achieved *diameter-δ approximate reconstruction*. The case $\delta = 0$ corresponds to full reconstruction as in the previous section. Since the value of any record is in the interval 1 to N (whose width corresponds to the parameter $\delta = 1 - 1/N$), we assume in the rest of the section that the target parameter δ is in $(0, 1 - 1/N)$. Since each class of records is assigned an interval of width at least $N - |\mathcal{P}_P|$, the precision achieved by a partition of positions of size $|\mathcal{P}_P|$ is, at best, $1 - |\mathcal{P}_P|/N$. In other words, a necessary (but not sufficient) condition for achieving diameter-δ approximate reconstruction is that the partition of positions has size at least $N - \delta N$. The size of the partition of records, however, does not provide any guarantees on the algorithm's success. When $|\mathcal{P}_R| = N$, the algorithm succeeds with $\delta = 0$, full reconstruction. However, even when there are $N - 1$ classes of records, it is possible that one of them is assigned the range of values 1 to N. Consider leakage from the singleton queries $[2, 2]$ through $[N - 1, N - 1]$ and the query $[1, N]$—records with values 1 and N will be in the same class, and match the same queries as the first and last intervals of positions, which means they will be assigned the range of values $[1, N]$.

The pseudocode in Algorithm 3.2 describes the details of this approximate reconstruction attack using rank and access pattern leakage. The algorithm begins by processing leakage in the same way as in the full reconstruction attack (lines 2–6). It forms the partition of records as before (line 7), then forms an ordered "interval" partition of positions (lines 8–9). Here, two positions that matched exactly the same queries are not necessarily in the same equivalence class—all positions between them must also have matched the same set of queries. Next, each class of records is assigned a range of values. The algorithm first finds the smallest and largest indices of classes of positions that matched the same set of queries as this class of records (lines 11–12). Then, it takes into account the number of classes of positions ($|\mathcal{P}_P|$) to determine the width of the assigned range of values. If there is a single class of records assigned an interval of width larger than δN, then diameter-δ approximate reconstruction has failed and the algorithm returns $\perp$ (lines 13–14).

Algorithm 3.2 Diameter-δ approximate reconstruction attack with rank and access pattern leakage

Input: real $\delta \in (0, 1 - 1/N)$, query leakage $\{(x_i, y_i, \mathsf{R}_i)\}_{i=1}^{\mathsf{n_q}}$.

Output: $\perp$ or maps $\widehat{\mathsf{val}}_{\mathsf{min}}, \widehat{\mathsf{val}}_{\mathsf{max}} : \mathcal{R} \to \{1, \ldots, N\}$ such that for all records $\mathsf{r} \in \mathcal{R}$, $\mathsf{val}(\mathsf{r}) \in \{\widehat{\mathsf{val}}_{\mathsf{min}}(\mathsf{r}), \ldots, \widehat{\mathsf{val}}_{\mathsf{max}}(\mathsf{r})\}$ and $\widehat{\mathsf{val}}_{\mathsf{max}}(\mathsf{r}) - \widehat{\mathsf{val}}_{\mathsf{min}}(\mathsf{r}) \leq \delta N$.

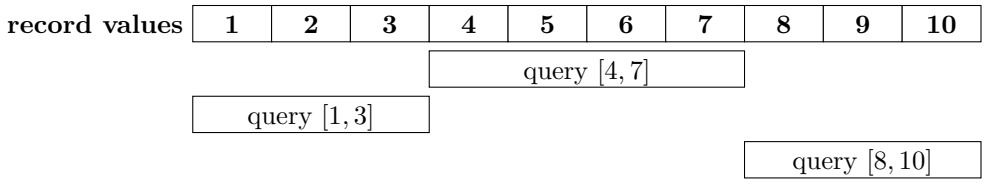
$\widehat{\mathsf{val}}_{\mathsf{min}}$, $\widehat{\mathsf{val}}_{\mathsf{max}}$, Q_R, $Q_P \leftarrow$ empty maps
for all $i \in \{1, \ldots, \mathsf{n_q}\}$ **do**
  for all $\mathsf{r} \in \mathsf{R}_i$ **do**
    $Q_R[\mathsf{r}] \leftarrow Q_R[\mathsf{r}] \cup \{i\}$
  for all $j \in \{x_i + 1, \ldots, y_i\}$ **do**
    $Q_P[j] \leftarrow Q_P[j] \cup \{i\}$
Form partition $\mathcal{P}_R$ of $\mathcal{R}$ with equivalence relation defined by $\mathsf{r} \equiv \mathsf{r}' \Leftrightarrow Q_R[\mathsf{r}] = Q_R[\mathsf{r}']$
Form "interval" partition $\mathcal{P}_P$ of $\{1, \ldots, |\mathcal{R}|\}$ with equivalence relation defined by $j \equiv j' \Leftrightarrow Q_P[j] = Q_P[j + i]$ for all $i \in [0, j' - j]$
Order $\mathcal{P}_P = ([J_1], \ldots, [J_{|\mathcal{P}_P|}])$ by $[J] < [J']$ iff $j < j'$ for all $j \in [J]$ and all $j' \in [J']$
for all $[R] \in \mathcal{P}_R$ **do**
  $\hat{k}_{\min} \leftarrow \min\{k \in \{1, \ldots, |\mathcal{P}_P|\} \text{ such that } Q_R[R] = Q_P[J_k]\}$
  $\hat{k}_{\max} \leftarrow \max\{k \in \{1, \ldots, |\mathcal{P}_P|\} \text{ such that } Q_R[R] = Q_P[J_k]\}$
  if $\hat{k}_{\max} - \hat{k}_{\min} + N - |\mathcal{P}_P| > \delta N$ **then**
    return $\perp$
  for all $\mathsf{r} \in [R]$ **do**
    $\widehat{\mathsf{val}}_{\mathsf{min}}(\mathsf{r}) \leftarrow \hat{k}_{\min}$
    $\widehat{\mathsf{val}}_{\mathsf{max}}(\mathsf{r}) \leftarrow \hat{k}_{\max} + N - |\mathcal{P}_P|$
return $\widehat{\mathsf{val}}_{\mathsf{min}}, \widehat{\mathsf{val}}_{\mathsf{max}}$

3.3.1 Leakage optimality

The exact reconstruction algorithm in the previous section was leakage-optimal. We can define a similar notion of leakage-optimality for a diameter-δ approximate reconstruction algorithm: such an algorithm must succeed whenever any other *correct* approximate reconstruction algorithm succeeds on the same leakage. A correct diameter-δ approximate reconstruction algorithm is one that, for each possible input, either (a) with probability 1, returns $\perp$ (failure), or (b) with probability 1, returns maps $\widehat{\mathsf{val}}_{\mathsf{min}}$ and $\widehat{\mathsf{val}}_{\mathsf{max}}$ such that the map val defining any database consistent with the given leakage satisfies $\mathsf{val}(\mathsf{r}) \in \{\widehat{\mathsf{val}}_{\mathsf{min}}(\mathsf{r}), \ldots, \widehat{\mathsf{val}}_{\mathsf{max}}(\mathsf{r})\}$ and $\widehat{\mathsf{val}}_{\mathsf{max}}(\mathsf{r}) - \widehat{\mathsf{val}}_{\mathsf{min}}(\mathsf{r}) \leq \delta N$ for all records (success). Unlike the case for exact reconstruction, a correct algorithm here does not necessarily return $\perp$ when its input does not specify a *unique* database: of course, another database whose records have values at most δN away from their original values could still generate the same leakage.

In the full version of our paper [47, p. 31], we stated "Although we do not prove it, it is also clear that the [approximate reconstruction] algorithm is [leakage-optimal]." In general, however, this is not true: the characteristic feature of the approximation

algorithm—that it assigns ranges of values to records—combined with the adversary's knowledge that the data is dense allows it to do better in some cases. Consider the following example with $N = 10$ and 3 queries.



Running Algorithm 3.2 yields a partition of positions of size $|\mathcal{P}_P| = 3$. Since each class of records has only one corresponding class of positions, $\hat{k}_{\min} = \hat{k}_{\max}$ for each class of records. Records with values 1–3 will be assigned the range of values $[1, 8]$, records with values 4–7 will be assigned values $[2, 9]$, and records with values 8–10 will be assigned $[3, 10]$. Thus, the approximation is correct to within a diameter of $\delta \geq 0.7$ for any dense database with $N = 10$, regardless of the total number of records.

Suppose the database has 10 records, one with each value. Our attack is for dense databases and we assume the adversary knows the data is dense: as soon as it sees that there are 10 records and $N = 10$ possible values, it can deduce that there is exactly one record with each value. Given the same leakage as previously, it has enough information to assign the records with values 1–3 the range of values $[1, 3]$; records with values 4–7, $[4, 7]$; and records with values 8–10, $[8, 10]$, which corresponds to approximate reconstruction within diameter $\delta = 0.3$. This same counterexample extends to more than 10 records: if none of the three partitions has more than 7 records, an adversary will be able to reconstruct values within diameter $\delta < 0.7$. Such cases prevent Algorithm 3.2 from being leakage-optimal. The exact reconstruction algorithm, from Section 3.2 (Algorithm 3.1), however, does not have this weakness since the question of its success is binary: records either do or do not have the same values.

3.3.2 Query analysis

The design of Algorithm 3.2 is independent of the query distribution. To estimate the expected number of required queries, however, it is necessary to make an assumption about the query distribution. Like in the last section, suppose all $N(N+1)/2$ ranges occur with uniform probability. Then, the expected number of required queries for the algorithm to succeed is $\mathcal{O}(N \log 1/\delta)$. Using the following proof technique, we require that $\delta N \geq 4$ (otherwise, exact reconstruction is achieved). Without loss of generality, we can assume that δN is an integer.

Theorem 3.5. *Let N be the number of possible values a record can have, let $\delta \in [4/N, 1 - 1/N)$ be the targeted diameter, and let DB be a set of records where each value appears in at least one record. Then, the expected number of uniformly random queries until Algorithm 3.2 succeeds for $N \geq 4$ is at most $(N+1)\left(\frac{3}{2}\log(2/\delta) + 4\right)$.*

Proof. Let $\{[a_i, b_i]\}_{i=1}^{\mathsf{n_q}}$ be the set of sampled queries and $\{(x_i, y_i, \mathsf{R}_i)\}_{i=1}^{\mathsf{n_q}}$ be the corresponding leakage. Algorithm 3.2 succeeds iff every class $[R]$ in the partition of records $\mathcal{P}_R$ is assigned an interval of width at most δN.

Claim 3.6. *The success of Algorithm 3.2 is implied by the following three events:*

1. *For $N/2 - \delta N/4$ distinct values v in $\{1, \ldots, N/2\}$, there is at least one query with $a = v$ and $b \geq N/2 + 1$.*
2. *For $N/2 - \delta N/4$ distinct values v in $\{N/2 + 1, \ldots, N\}$, there is at least one query with $a \leq N/2$ and $b = v$.*
3. *There are two queries whose union is $[N/2 + 1, N]$.*

Proof of Claim 3.6. Recall that each class of records $[R]$ is assigned an interval of width $\hat{k}_{\max} - \hat{k}_{\min} + N - |\mathcal{P}_P|$, where $\hat{k}_{\min}$ and $\hat{k}_{\max}$ are the smallest and largest indices of classes of positions in $\mathcal{P}_P$ that matched the same set of queries as $[R]$. To prove this claim, we will show that the three stated events imply that $N - |\mathcal{P}_P| \leq \delta N/2$ and $\hat{k}_{\max} - \hat{k}_{\min} \leq \delta N/2$ for all classes of records.

First, consider the number of elements in the optimized list representation of Q_P (introduced in Section 3.2.3). This list is initially $((0, 0^{\ell_{\mathcal{H}}}), (|\mathcal{R}| + 1, 0^{\ell_{\mathcal{H}}}))$. Event 1 results in the creation of at least $N/2 - \delta N/4$ items of the form $(\mathsf{rank}(v-1) + 1, \cdot)$, for $v \in \{1, \ldots, N/2\}$. Similarly, Event 2 adds at least $N/2 - \delta N/4 - 1$ items of the form $(\mathsf{rank}(v) + 1, \cdot)$, for $v \in \{N/2 + 1, \ldots, N\}$. (The subtracted 1 is due to the case $v = N$, which already has a corresponding entry.) Event 3 results in the addition of a point $(\mathsf{rank}(N/2) + 1, \cdot)$. Thus, the total number of points in the list Q_P is at least $2 + N - \delta N/2$, which corresponds to at least $N - \delta N/2$ classes of positions in $\mathcal{P}_P$. That is, $N - |\mathcal{P}_P| \leq \delta N/2$.

Next, we show that $\hat{k}_{\max} - \hat{k}_{\min} \leq \delta N/2$ for all classes of records by proving something slightly stronger: if two records are in the same class of records in $\mathcal{P}_R$, then the difference of their values is no more than $\delta N/2$. This implies the desired property since $\hat{k}_{\min}$ and $\hat{k}_{\max}$ refer to indices in the list of classes of positions, and each class of positions corresponds to at least one different value.

Let r and r' be two records with values v and v' such that $|v - v'| > \delta N/2$. We will prove that they are in different classes of records. If the values are in different halves (i.e., one is at most $N/2$ and the other is at least $N/2 + 1$), then Event 3 guarantees that they cannot be in the same class of records, so they must be in the same half.

Suppose that they are in the first half. If there is at least one value between them that has a query of the type in Event 1, then they must be in different classes because this query will match only the greater of the two values. Similarly, if the greater value has a corresponding query of the type in Event 1, then it will also be in a different class than the lesser value. The only remaining case is that the greater value and all values between v and v' do not correspond to queries of the type in Event 1. However, since there are at most $\delta N/4$ such values with no corresponding queries, the difference $|v - v'|$ would then be at most $\delta N/4 < \delta N/2$, a contradiction. The case where the two values are in the second half is similar. This concludes the proof of Claim 3.6. □

We now continue with the proof of Theorem 3.5 and consider the number of expected queries until Events 1, 2, and 3 occur, starting with Events 1 and 2.

Claim 3.7. *The expected number of queries until Events 1 and 2 occur is at most* $\frac{3}{2}(N+1)\log(2/\delta)$.

Proof of Claim 3.7. There are $N^2/4$ candidate *bridging* queries that are relevant for Events 1 and 2—those with $a \leq N/2$ and $b \geq N/2 + 1$ (which "bridge" the two halves). First, we use a coupon collector-style analysis to bound the expected number of uniformly sampled bridging queries until Events 1 and 2 occur, then we determine the expected number of draws from *all* queries until this many bridging queries occur.

Event 1 occurs iff there are $N/2 - \delta N/4$ distinct left endpoints a in the set of bridging queries; Event 2, the same number of distinct right endpoints b. Bridging queries have the property that left and right endpoint values are independent: $\text{Prob}[a = v] = \text{Prob}[b = v'] = (N/2)/(N^2/4) = 1/(N/2)$ and

$$\text{Prob}[a = v \,|\, b = v'] := \frac{\text{Prob}[a = v \wedge b = v']}{\text{Prob}[b = v']} = \frac{1/(N^2/4)}{1/(N/2)} = 1/(N/2) = \text{Prob}[a = v]\,.$$

Therefore, Events 1 and 2 are independent when drawing from the set of bridging queries, and they occur with the same probability, by symmetry.

Let T_i^L be a random variable representing the number of bridging queries drawn to get the ith distinct left endpoint after the $(i-1)$st left endpoint (where $T_0^L := 0$ for convenience). Also let the random variable $T^L := \sum_{i=1}^{N/2-\delta N/4} T_i^L$ represent the total

number of bridging queries required until Event 1. Let T^R be the identically defined random variable representing the number of bridging queries drawn until Event 2 occurs. To bound the expected number of queries until both Events 1 and 2 occur, we must bound $\mathbb{E}\big[\max\{T^L, T^R\}\big]$. By the definitions of T^L and T^R, we have

$$\begin{aligned}\max\{T^L, T^R\} &= \max\left\{\sum_{i=1}^{N/2-\delta N/4} T_i^L, \sum_{i=1}^{N/2-\delta N/4} T_i^R\right\} \\ &\leq \sum_{i=1}^{N/2-\delta N/4} \max\left\{T_i^L, T_i^R\right\}. \end{aligned} \tag{3.2}$$

Let $T_i^{\max} \coloneqq \max\left\{T_i^L, T_i^R\right\}$ be a random variable representing the maximum of T_i^L and T_i^R. Since T_i^L and T_i^R are independent and identically distributed, we have

$$\begin{aligned}\text{Prob}[T_i^{\max} \geq x] &= 1 - \text{Prob}[T_i^{\max} \leq x-1] \\ &= 1 - \text{Prob}\big[T_i^L \leq x-1\big] \cdot \text{Prob}\big[T_i^R \leq x-1\big] \\ &= 1 - \big(\text{Prob}\big[T_i^L \leq x-1\big]\big)^2 \\ &= 1 - \big(1 - \text{Prob}\big[T_i^L \geq x\big]\big)^2.\end{aligned}$$

For a fixed value v, the number of bridging queries $[a, b]$ with $a = v$ is $N/2$. Since queries are drawn uniformly at random and there are initially $N/2$ candidates for v, each T_i^L is a geometric random variable with success probability

$$p_i = \frac{(N/2)(N/2+1-i)}{N^2/4} = \frac{N/2+1-i}{N/2}.$$

Therefore, $\text{Prob}\big[T_i^L \geq x\big] = (1-p_i)^{x-1}$.

Substituting this term into the equation for $\text{Prob}[T_i^{\max} \geq x]$ yields

$$\begin{aligned}\text{Prob}[T_i^{\max} \geq x] &= 1 - \big(1 - (1-p_i)^{x-1}\big)^2 \\ &= 1 - \Big(1 - 2(1-p_i)^{x-1} + (1-p_i)^{2(x-1)}\Big) \\ &= 2(1-p_i)^{x-1} - (1-p_i)^{2(x-1)}.\end{aligned}$$

Now, by Formula A.4, the expected value of $T_i^{\max}$ is

$$\begin{aligned}
\mathbb{E}[T_i^{\max}] &= \sum_{x=1}^{\infty} \text{Prob}[T_i^{\max} \geq x] \\
&= \sum_{x=1}^{\infty} \left(2(1-p_i)^{x-1} - (1-p_i)^{2(x-1)}\right) \\
&= \sum_{x=0}^{\infty} \left(2(1-p_i)^{x} - (1-p_i)^{2x}\right) \\
&= \frac{2}{1-(1-p_i)} - \frac{1}{1-(1-p_i)^2} && \text{(Formula A.5)} \\
&= \frac{2}{p_i} - \frac{1}{p_i(2-p_i)} \\
\mathbb{E}[T_i^{\max}] &= \frac{1}{p_i}\left(2 - \frac{1}{2-p_i}\right).
\end{aligned}$$

Recall that we must bound $\mathbb{E}\big[\max\{T^L, T^R\}\big]$. By applying linearity of expectation to Equation 3.2 and combining it with the expected value of $T_i^{\max}$, we obtain

$$\mathbb{E}\big[\max\{T^L, T^R\}\big] \leq \sum_{i=1}^{N/2-\delta N/4} \mathbb{E}[T_i^{\max}] = \sum_{i=1}^{N/2-\delta N/4} \frac{1}{p_i}\left(2 - \frac{1}{2-p_i}\right).$$

First, note that $\sum_{i=1}^{N/2-\delta N/4} \frac{1}{p_i}$ is simply the expected value of T_L (or T_R), the total number of bridging queries until Event 1 (or Event 2), due to linearity of expectation and the fact that $T_i^L \sim \mathsf{Geo}\,(p_i)$:

$$\mathbb{E}\big[T^L\big] = \sum_{i=1}^{N/2-\delta N/4} \mathbb{E}\big[T_i^L\big] = \sum_{i=1}^{N/2-\delta N/4} \frac{1}{p_i}.$$

Recalling that $p_i = \frac{N/2+1-i}{N/2}$ and applying Bound A.3, we can bound it as follows:

$$\sum_{i=1}^{N/2-\delta N/4} \frac{1}{p_i} = \sum_{i=0}^{N/2-\delta N/4-1} \frac{N/2}{N/2-i} = (N/2)\left(\mathsf{H}_{N/2} - \mathsf{H}_{\delta N/4}\right) \leq (N/2)\left(\log(2/\delta)\right).$$

Using $p_i > 0$, we obtain $\left(2 - \frac{1}{2-p_i}\right) < 3/2$, from which we may conclude that

$$\mathbb{E}\big[\max\{T^L, T^R\}\big] < (3N/4)\log(2/\delta).$$

Finally, we compute an upper bound on the expected number of uniformly random queries drawn until $\mathbb{E}\big[\max\{T^L, T^R\}\big]$ bridging queries occur. Since there are $N^2/4$ bridging queries, one occurs with probability $p_{\mathsf{bridge}} = (N^2/4)/(N(N+1)/2) = N/(2(N+1))$. Let T_{bridge} be the random variable representing the number of uniformly drawn range queries until $\mathbb{E}\big[\max\{T^L, T^R\}\big]$ bridging queries occur. T_{bridge} has

a negative binomial distribution with parameters $\mathbb{E}\left[\max\{T^L, T^R\}\right]$ and p_{bridge}, so its expected value $\mathbb{E}[T_{\mathsf{bridge}}]$ is

$$\frac{\mathbb{E}\left[\max\{T^L, T^R\}\right]}{p_{\mathsf{bridge}}} < \frac{(3N/4)\log(2/\delta)}{N/(2(N+1))} = \frac{(3N/2)\log(2/\delta)}{1/(N+1)} = \frac{3}{2}(N+1)\log(2/\delta).$$

This concludes the proof of Claim 3.7. □

Finally, returning to the proof of Theorem 3.5, we consider the expected number of queries until Event 3 occurs. Like in the proof of Theorem 3.2, we consider one particular way in which two queries' union could be $[N/2+1, N]$: there were two queries of the form $[N/2+1, b]$ and $[a, N]$, where in the "left" queries, $b \in \{\lfloor 3N/4 \rfloor, \ldots, N\}$ and in the "right" queries, $a \in \{N/2+1, \ldots, \lfloor 3N/4 \rfloor + 1\}$. The probability that any query is a "left" (or "right") query is at least $\frac{N/4}{N(N+1)/2} = 1/(2(N+1))$, so the number of queries required until a "left" query (or "right" query) has been drawn is distributed geometrically with probability of success at least $1/(2(N+1))$. The expected number of draws till multiple events occur is at most the sum of the expected number of draws till each individual event occurs, so Event 3 occurs within at most $4(N+1)$ drawn queries, and the expected number of queries until all three events occur (implying the success of Algorithm 3.2) is at most

$$\frac{3}{2}(N+1)\log(2/\delta) + 4(N+1) = (N+1)\left(\frac{3}{2}\log(2/\delta) + 4\right). \qquad \square$$

Theorem 3.5 shows that diameter-δ approximate reconstruction takes $\mathcal{O}(N\log(1/\delta))$ queries, fewer than the $\mathcal{O}(N\log N)$ queries required for exact reconstruction.

3.4 Inferred reconstruction

The two previous reconstruction algorithms guarantee correctness of the reconstructed values: in the exact reconstruction algorithm, $\mathsf{val}(\mathsf{r}) = \widehat{\mathsf{val}}(\mathsf{r})$ for all records, and in the approximate reconstruction algorithm, $\mathsf{val}(\mathsf{r}) \in \{\widehat{\mathsf{val}}_{\mathsf{min}}(\mathsf{r}), \ldots, \widehat{\mathsf{val}}_{\mathsf{max}}(\mathsf{r})\}$. This section examines a heuristic algorithm that provides no correctness guarantees, but uses additional input to *infer* record values given leakage from fewer queries. The additional input is the approximate distribution of values in the database, which we call an auxiliary distribution. Like the first two attacks, this one also does not depend on the query distribution being uniform, but unlike the first two attacks, it does not require the data to be dense. Simply put, given knowledge of the distribution of

values, and observing rank and access pattern leakage, it is possible to quickly and roughly reconstruct values.

The auxiliary distribution. In many real scenarios, the distribution of values may be predictable. For instance, the values might represent the age of patients in a hospital or salaries in a personnel database. The availability of such a distribution is not a new assumption; previous statistical inference attacks on encrypted databases have used it [62, 7].

Reasoning behind the algorithm The idea behind this algorithm is to use the auxiliary distribution to map positions to values. First, rank and access pattern leakage are used to partition positions and records into classes: each class of records corresponds to at least one class of positions, like in the exact and approximate algorithms. The next step is similar to the approximate reconstruction algorithm, but instead of assigning a range of values to each class of records, their range of *positions* is used. The endpoints of these ranges are mapped to the most *likely* values according to the auxiliary distribution, and then a point estimate is assigned to each class of records by computing the expected value of the auxiliary distribution between those most likely endpoints. These ranges of positions have boundaries that correspond to ranks.

Algorithm details. The pseudocode in Algorithm 3.3 describes the details of this heuristic, inferred reconstruction attack. It still assumes that the total set of record identifiers and number of possible values, N, is known ahead of time. The algorithm forms and then matches partitions of records and positions in the same way as in the approximate reconstruction attack.

For each class of records, it finds the smallest and largest *positions* that matched the same set of queries as this class (lines 11–12). The smallest position, $\hat{\jmath}_{\min}$, must be equal to $1+\mathsf{rank}(a)$ for some unknown value a, while the largest position, $\hat{\jmath}_{\max}$, must be equal to $\mathsf{rank}(b)$ for some value b. Using the cumulative auxiliary distribution, $\mathsf{F}(x) \coloneqq \text{Prob}[X \leq x]$, it is possible to find the most likely pre-images of these two ranks. In general, the rank of a value a can be interpreted as a binomial random variable with parameters $|\mathcal{R}|$ and $\mathsf{F}(a)$: the probability that $\hat{\jmath} = \mathsf{rank}(a)$ is $\binom{|\mathcal{R}|}{\hat{\jmath}}(\mathsf{F}(a))^{\hat{\jmath}}(1 - \mathsf{F}(a))^{|\mathcal{R}|-\hat{\jmath}}$. Applying the technique of maximum likelihood estimation (MLE), the most likely value of a is

$$\arg\max_a \binom{|\mathcal{R}|}{\hat{\jmath}}(\mathsf{F}(a))^{\hat{\jmath}}(1 - \mathsf{F}(a))^{|\mathcal{R}|-\hat{\jmath}} = \arg\max_a \hat{\jmath}\log\mathsf{F}(a) + (|\mathcal{R}| - \hat{\jmath})\log(1 - \mathsf{F}(a)).$$

Algorithm 3.3 Inferred reconstruction attack with auxiliary distribution and rank and access pattern leakage

Input: query leakage $\{(x_i, y_i, \mathsf{R}_i)\}_{i=1}^{\mathsf{n_q}}$, auxiliary distribution $\mathsf{D_{aux}}$ with pmf f and cdf F.
Output: $\perp$ or map $\widehat{\mathsf{val}} : \mathcal{R} \to \{1, \ldots, N\}$.

1: $\widehat{\mathsf{val}}$, Q_R, $Q_P \leftarrow$ empty maps
2: **for all** $i \in \{1, \ldots, \mathsf{n_q}\}$ **do**
3: **for all** $\mathsf{r} \in \mathsf{R}_i$ **do**
4: $Q_R[\mathsf{r}] \leftarrow Q_R[\mathsf{r}] \cup \{i\}$
5: **for all** $j \in \{x_i + 1, \ldots, y_i\}$ **do**
6: $Q_P[j] \leftarrow Q_P[j] \cup \{i\}$
7: Form partition $\mathcal{P}_R$ of $\mathcal{R}$ with equivalence relation defined by $\mathsf{r} \equiv \mathsf{r}' \Leftrightarrow Q_R[\mathsf{r}] = Q_R[\mathsf{r}']$
8: Form "interval" partition $\mathcal{P}_P$ of $\{1, \ldots, |\mathcal{R}|\}$ with equivalence relation defined by $j \equiv j' \Leftrightarrow Q_P[j] = Q_P[j + i]$ for all $i \in [0, j' - j]$
9: Order $\mathcal{P}_P = ([J_1], \ldots, [J_{|\mathcal{P}_P|}])$ by $[J] < [J']$ iff $j < j'$ for all $j \in [J]$ and all $j' \in [J']$
10: **for all** $[R] \in \mathcal{P}_R$ **do**
11: $\hat{\jmath}_{\min} \leftarrow \min\{j \in [J_{\hat{k}_{\min}}]\}$ where
 $\hat{k}_{\min} = \min\{k \in \{1, \ldots, |\mathcal{P}_P|\} : Q_R[R] = Q_P[J_k]\}$
12: $\hat{\jmath}_{\max} \leftarrow \max\{j \in [J_{\hat{k}_{\max}}]\}$ where
 $\hat{k}_{\max} = \max\{k \in \{1, \ldots, |\mathcal{P}_P|\} : Q_R[R] = Q_P[J_k]\}$
13: $\hat{a} \leftarrow \arg\max_{a \in \{1,\ldots,N\}} \hat{\jmath}_{\min} \log \mathsf{F}(a) + (|\mathcal{R}| - \hat{\jmath}_{\min}) \log(1 - \mathsf{F}(a))$
14: $\hat{b} \leftarrow \arg\max_{b \in \{1,\ldots,N\}} \hat{\jmath}_{\max} \log \mathsf{F}(b) + (|\mathcal{R}| - \hat{\jmath}_{\max}) \log(1 - \mathsf{F}(b))$
15: $\hat{v} \leftarrow \hat{b}$ if $\hat{a} = \hat{b}$, else $\left(\sum_{i=a+1}^{\hat{b}} i \cdot \mathsf{f}(i)\right) / (\mathsf{F}(\hat{b}) - \mathsf{F}(\hat{a}))$
16: **for all** $\mathsf{r} \in [R]$ **do**
17: $\widehat{\mathsf{val}}(\mathsf{r}) \leftarrow \hat{v}$
18: **return** $\widehat{\mathsf{val}}$

Without the restriction of taking the maximum over the N possible values, and instead treating $\mathsf{F}(a)$ as any probability p in $[0, 1]$, the log-likelihood function would be $\hat{\jmath} \log p + (|\mathcal{R}| - \hat{\jmath}) \log(1 - p)$, and the maximum would occur when its derivative with respect to p is 0:

$$0 = \frac{\hat{\jmath}}{p} - \frac{|\mathcal{R}| - \hat{\jmath}}{1 - p} \qquad \Leftrightarrow \qquad p\,|\mathcal{R}| - p\hat{\jmath} = \hat{\jmath} - p\hat{\jmath} \qquad \Leftrightarrow \qquad p = \frac{\hat{\jmath}}{|\mathcal{R}|}.$$

Therefore, the value a that maximizes the likelihood should be close to $\hat{\jmath}/\,|\mathcal{R}|$. This method is used to determine the most likely values $\hat{a}$ and $\hat{b}$ corresponding to $\hat{\jmath}_{\min}$ and $\hat{\jmath}_{\max}$ (lines 13–14). The next step (line 15) is to compute the expected value $\hat{v}$ of the auxiliary distribution $\mathsf{D_{aux}}$ restricted to the interval $(\hat{a}, \hat{b}]$:

$$\mathbb{E}\Big[v \,|\, v \in (\hat{a}, \hat{b}]\Big] = \begin{cases} \hat{b} & \text{if } \hat{a} = \hat{b}, \\ \frac{\sum_{i=a+1}^{\hat{b}} i \cdot \mathsf{f}(i)}{\mathsf{F}(\hat{b}) - \mathsf{F}(\hat{a})} & \text{else.} \end{cases}$$

Every record in the current class of records is assigned this estimated value, concluding the algorithm.

3.4.1 Experimental evaluation

This attack is less suited to rigorous analysis than the previous two attacks. First, its performance depends on the accuracy of the auxiliary distribution $\mathsf{D}_{\mathsf{aux}}$. Second, although ranks are inverted to their most likely values, this step is done independently for each endpoint. In addition to being rather crude, this method permits two different ranks to be assigned the same value. Therefore, we evaluate the performance of this attack experimentally.

Design. We simulate queries on `AGE` values from patient records of the 200 largest hospitals in the 2009 HCUP data (q.v. Section 1.3). These 200 hospitals had between 13 000 and 122 000 records each, 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 $\mathsf{D}_{\mathsf{aux}}$ was the aggregate of the 200 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. For example, while some of the largest 200 hospitals' records corresponded to age 114, none corresponded to age 113. Further, each hospital's data is not necessarily dense with respect to the auxiliary distribution. For instance, in all of the largest 200 hospitals' records, only 6 records were for patients with age 119.

Evaluation. 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. This error ϵ is different than the diameter δ in the approximate attack. There, each record's value was guaranteed to be in some interval of width δN; here, there is no guarantee that a record's value is even within the interval $(\hat{a}, \hat{b}]$ computed on lines 13–14, so we simply measure the error as $\epsilon N = |\hat{v} - \mathsf{val}(\mathsf{r})|$. Since ages are in the range $[0, 124]$, the margin of error for all records cannot be higher than 124, which corresponds to $\epsilon = 124/125$. Perfect reconstruction corresponds to a vertical line at $\epsilon = 0$. Successful attacks have steeply rising curves that reach a fraction of records $y = 1$ for small values of the error ϵ.

Since we assume that the set of record identifiers is known, even records that have not matched any queries can be assigned a value estimate by matching them with the positions that have also not matched any queries. (In Section 3.4.2, we evaluate the attack without the assumption that the set of all record identifiers is known ahead of

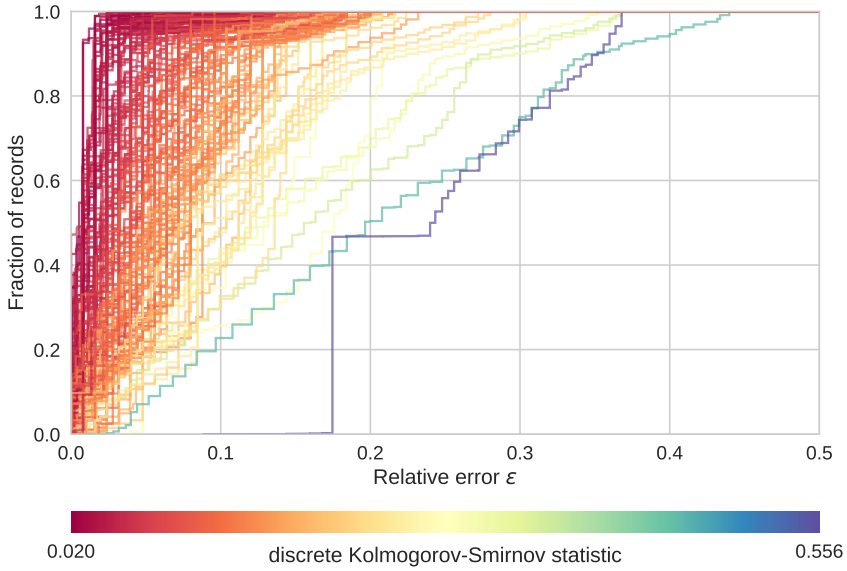


Figure 3.2: Fraction of records recovered within ϵ of true `AGE` value as number of observed queries tends to infinity, for largest 200 hospitals in 2009 HCUP data.

time, in which case we set ϵ to 1 for records that have not been seen yet.) We did not round the point guesses for each class of records ($\hat{v}$ on line 15) to the nearest integer.

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 specialized departments, e.g., neonatal, pediatric, or geriatric. Figure 3.2 shows the *asymptotic* success of the attack for each of the 200 hospitals: 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 color-code each hospital's curve in Figure 3.2 with the discrete Kolmogorov–Smirnov (K-S) statistic for the per-hospital and aggregate distributions. The K-S statistic is a cumulative goodness-of-fit test; for two discrete distributions, it is the maximum absolute difference between their cumulative distribution functions: $\mathsf{KS}(\mathsf{F}, \mathsf{F}') \coloneqq \max_{v \in \{1,\ldots,N\}} |\mathsf{F}(v) - \mathsf{F}'(v)|$. 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. Figure 3.2 illustrates the importance of the closeness of the auxiliary distribution to

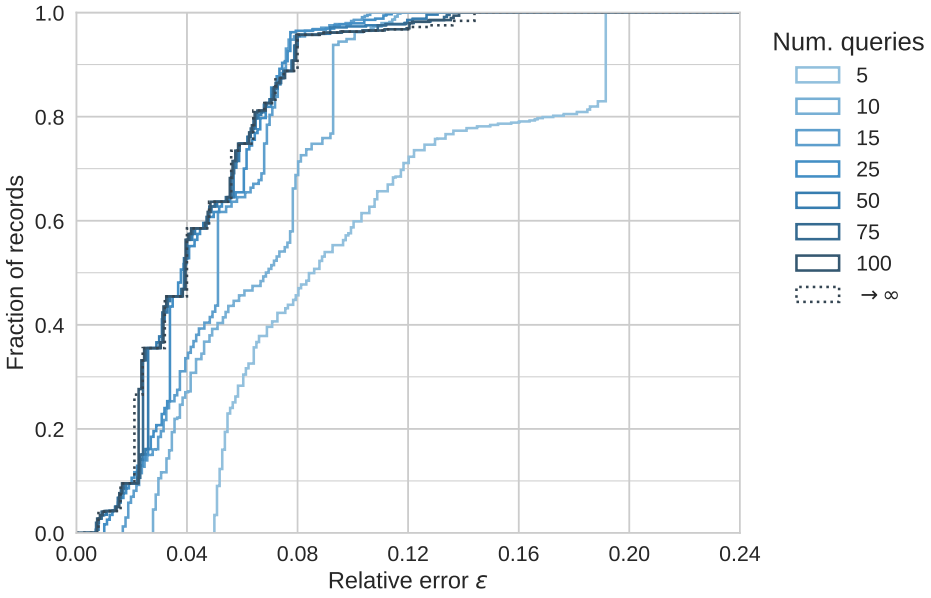
the actual distribution: for small K-S values (encoded in dark red, in the upper left), the algorithm generally performs better, recovering more records with a smaller relative error. Other statistics seemed less appropriate since they were not cumulative and therefore did not capture the ordering of the values. The Kullback–Leibler divergence is not suitable for comparing distributions having different support (which was the case for most hospitals; not all distinct ages from the auxiliary distribution appeared in each hospital's records). The Chi-Square goodness-of-fit test is not suited to small (conventionally, less than 5) expected counts, which was the case for most ages above 110.

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).

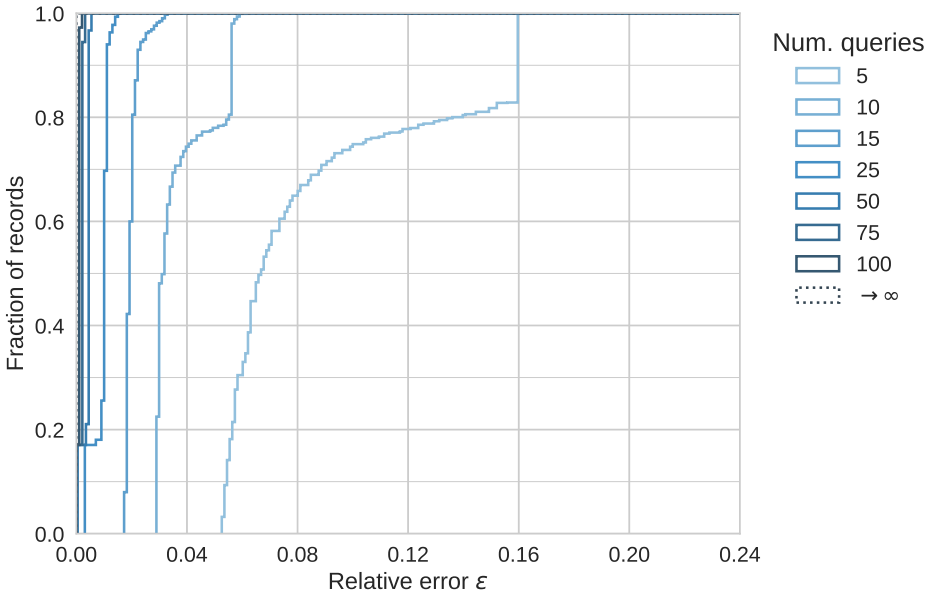
Required number of observed queries. Figure 3.3a shows the success of the attack on this particular hospital's data, averaged over 1000 experiments, with values assigned to records after 5, 10, 15, 25, 50, 75, and 100 queries. Figure 3.3b shows what its success would be if the auxiliary information were perfect. Even with this simple heuristic attack and a rough auxiliary distribution, the number of queries required to reconstruct most of the database is relatively small.

Recall from Corollary 3.4 in Section 3.2 that the expected number of uniformly distributed queries for an *exact* reconstruction attack can be as high as $N \log N + N + 6.6\sqrt{N} + 7$, which is about 809 for $N = 125$. Also recall from Section 3.3 that diameter-δ approximate reconstruction means that each record is assigned an interval of width at most δN. We could extend this attack by assigning a point guess equal to the interval's median value. For the sake of comparison with the relative error ϵ in this section, such a median point guess would correspond to $\delta = 2\epsilon$, which has an expected number of required queries as high as $(N + 1) \cdot (\log(1/\epsilon) + 4)$ (q.v. Theorem 3.5). For $N = 125$, recovering all records' values within 40 years ($\epsilon = 0.32$) requires up to 648 queries, within 20 years ($\epsilon = 0.16$) requires up to 735 queries, and within 15 years ($\epsilon = 0.12$) requires up to 771 queries.

With an auxiliary distribution, however, 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, it can guess the ages of 55% of records within 5 years ($\epsilon = 0.04$) After 50 queries, it can guess the ages of 35% of records within 3 years ($\epsilon = 0.024$). After 100 queries, the success of the attack is restricted only by the accuracy of the auxiliary data's distribution.



(a) Approximate auxiliary information



(b) Exact auxiliary information

Figure 3.3: Fraction of records reconstructed within relative additive factor ϵ of true `AGE` value using Algorithm 3.3 for one hospital, averaged over 1000 experiments.

3.4.2 Relaxing assumptions

This attack makes two non-trivial assumptions in addition to knowledge of an auxiliary distribution: first, that the total number of records is known, and second, that the set of all record identifiers is known. We briefly discuss the impact of removing these two assumptions.

Estimating the total number of records. For some distributions, approximating the total number of records is easy with only a few queries. Our approach comprises two steps: (i) computing the expected value of the maximum right endpoint b in any query so far, then (ii) translating the maximum observed rank and the cumulative auxiliary distribution to arrive at an estimate $\tilde{\mathsf{n}_\mathsf{r}}$ of the total number of records.

Let $\mathsf{B}_{\max}$ be a random variable representing the maximum right endpoint of $\mathsf{n_q}$ uniformly sampled range queries, $\{[a_i, b_i]\}_{i=1}^{\mathsf{n_q}}$. The number of distinct queries that have right endpoint at most $b_{\max}$ is $\sum_{b=1}^{b_{\max}} b = b_{\max}(b_{\max}+1)/2$, so the probability that all $\mathsf{n_q}$ right endpoints are at most $b_{\max}$ is $\mathrm{Prob}[\mathsf{B}_{\max} \leq b_{\max}] = \left(\frac{b_{\max}(b_{\max}+1)}{N(N+1)}\right)^{\mathsf{n_q}}$. The maximum right endpoint after $\mathsf{n_q}$ queries is $b_{\max}$ iff $b_i \leq b_{\max}$ for all $i \in \{1, \ldots, \mathsf{n_q}\}$, but not $b_i \leq b_{\max} - 1$ for all i. Therefore, the probability that the maximum right endpoint is $b_{\max}$ after $\mathsf{n_q}$ queries is

$$\begin{aligned}\mathrm{Prob}[\mathsf{B}_{\max} = b_{\max}] &= \left(\frac{b_{\max}(b_{\max}+1)}{N(N+1)}\right)^{\mathsf{n_q}} - \left(\frac{(b_{\max}-1)b_{\max}}{N(N+1)}\right)^{\mathsf{n_q}} \\ &= \frac{(b_{\max}(b_{\max}+1))^{\mathsf{n_q}} - ((b_{\max}-1)b_{\max})^{\mathsf{n_q}}}{(N(N+1))^{\mathsf{n_q}}}.\end{aligned}$$

The expected value $\mathbb{E}[\mathsf{B}_{\max}]$ is then equal to

$$\begin{aligned}
&= \sum_{b_{\max}=1}^{N} b_{\max}\left(\frac{(b_{\max}(b_{\max}+1))^{\mathsf{n_q}} - ((b_{\max}-1)b_{\max})^{\mathsf{n_q}}}{(N(N+1))^{\mathsf{n_q}}}\right) \\
&= \frac{\left(\sum_{b_{\max}=1}^{N} b_{\max}\left(b_{\max}(b_{\max}+1)\right)^{\mathsf{n_q}} - \sum_{b_{\max}=2}^{N} b_{\max}\left((b_{\max}-1)b_{\max}\right)^{\mathsf{n_q}}\right)}{(N(N+1))^{\mathsf{n_q}}} \\
&= \frac{\left(\sum_{b_{\max}=1}^{N} b_{\max}\left(b_{\max}(b_{\max}+1)\right)^{\mathsf{n_q}} - \sum_{b_{\max}=1}^{N-1}(b_{\max}+1)\left(b_{\max}(b_{\max}+1)\right)^{\mathsf{n_q}}\right)}{(N(N+1))^{\mathsf{n_q}}} \\
&= \frac{\left(N(N(N+1))^{\mathsf{n_q}} - \sum_{b_{\max}=1}^{N-1}\left(b_{\max}(b_{\max}+1)\right)^{\mathsf{n_q}}\right)}{(N(N+1))^{\mathsf{n_q}}} \\
&= N - \frac{1}{(N(N+1))^{\mathsf{n_q}}}\sum_{b_{\max}=1}^{N-1}\left(b_{\max}(b_{\max}+1)\right)^{\mathsf{n_q}}.
\end{aligned}$$

Let the observed rank leakage from the $\mathsf{n_q}$ queries be $\{(x_i, y_i)\}_{i=1}^{\mathsf{n_q}}$. For step (ii), we guess that the maximum observed rank, $y_{\max} \coloneqq \max\{y_i\}_{i=1}^{\mathsf{n_q}}$, corresponds to

the expected maximum right endpoint, $\mathbb{E}[\mathsf{B}_{\max}]$. Since, by definition, rank is the number of records with value less than or equal to any particular value, it should be proportional to the fraction of records with value less than or equal to that particular value, i.e., $\mathbb{E}[\mathsf{rank}(x)] = \mathsf{F}(x) \cdot \mathsf{n_r}$. Thus, we obtain an estimate for the total number of records:

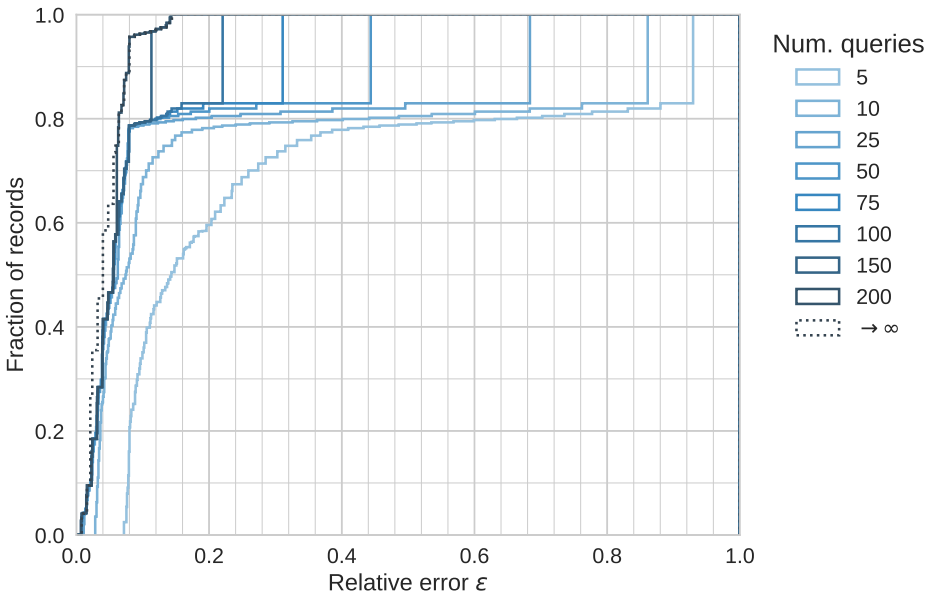
$$\tilde{\mathsf{n}}_\mathsf{r} \coloneqq \frac{y_{\max}}{\mathsf{F}_{\mathsf{aux}}\left(\lfloor \mathbb{E}[\mathsf{B}_{\max}] \rceil\right)}.$$

For uniformly random range queries with $N = 125$, the expected maximum right endpoint $\mathbb{E}[\mathsf{B}_{\max}]$ is 118.6 after just 10 queries, 122.0 after 25 queries, 123.2 after 50 queries, and 123.6 after 75 queries.

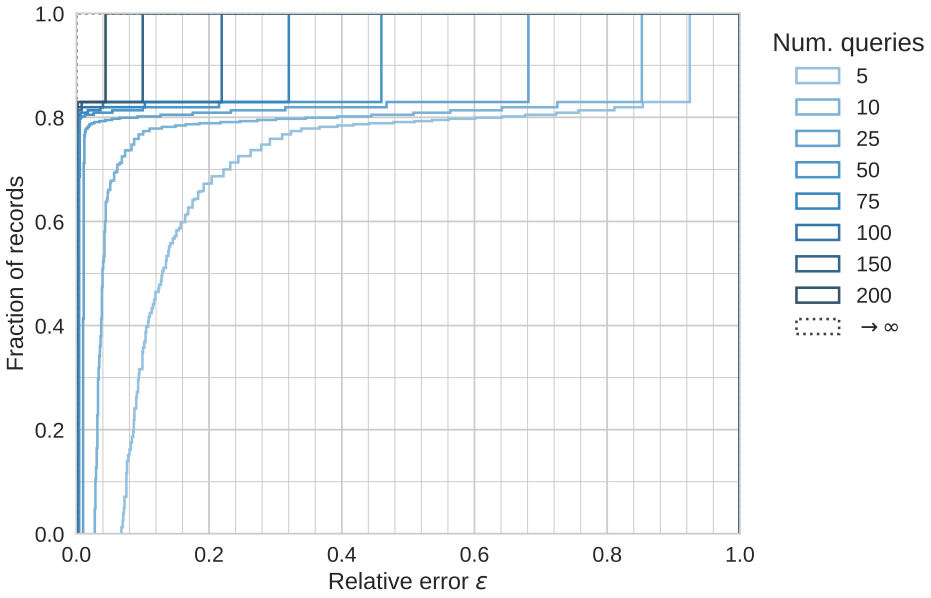
Returning to our experiment, we see that this heuristic for guessing the total number of records works well for the age dataset because it takes few queries for the maximum rank value to be observed: ages above 110 are infrequent, so it is unlikely to take more than 10 queries to observe the maximum rank value, $\mathsf{rank}(N)$, i.e., the number of records. For other distributions, perhaps the minimum or the mean rank value would be more suitable. Since so few queries are required to estimate the total number of records $\mathsf{n_r}$ given an auxiliary distribution, we are confident that removing this assumption would not have significantly decreased the attack's success in our experiments.

Known set of record identifiers. We have assumed that the adversary knows the set of all record identifiers; otherwise, it could not "reconstruct" any records that have not matched at least one query. Figure 3.4 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 (3.4a) and exact auxiliary information (3.4b). Records that have not matched any query so far are assigned an error of $\epsilon = 1$.

In this particular experiment, the most significant phenomenon arises from the fact that 17% of our chosen hospital's records are for patients with age 0. 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 Figure 3.4. Because we average the error over a large number of experiments, the horizontal position of the vertical line at the top of each curve reflects the probability that a query covering the value 0 was issued. If the value 0 has been queried, the corresponding



(a) Approximate auxiliary information



(b) Exact auxiliary information

Figure 3.4: Fraction of records reconstructed within relative additive factor ϵ of true `AGE` value using Algorithm 3.3 for one hospital, averaged over 1000 experiments, without assumption that set of record identifiers is known.

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.

If the adversary knows the set of all record identifiers, then it recovers all records within an error of about $\epsilon = 0.19$ after seeing only 5 queries 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 Figure 3.4, is significantly worse than in the case where record identifiers are known. When comparing these graphs to Figure 3.3, 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.

3.5 Conclusions

Many schemes that support range queries on outsourced data can reveal to a passive adversary which records matched the query (the access pattern) and the rank of the endpoints, i.e., how many records in the database have value less than them. In this chapter, we examined how to exploit such leakage to reconstruct the database, either exactly determining each record's value, or narrowing its value down to an interval of width δN.

Our attacks are computationally efficient and do not require queries to have any particular distribution. To analyze how many queries an adversary would need to observe before the attacks work, though, we needed to make an assumption about their distribution. With uniformly random queries, exact reconstruction takes $\mathcal{O}(N \log N)$ queries, while diameter-δ approximate reconstruction takes $\mathcal{O}(N \log(1/\delta))$ queries. These attacks are well suited to dense databases with a not-too-large number of possible values N, since observing enough queries might be impractical otherwise.

One way to avoid the requirement that data be dense is to use an estimate of the data's distribution. We also evaluated an inference attack that—when the estimated distribution was close to the true distribution—was able to reconstruct a large fraction of records to within a small additive error. Additional query leakage could also provide a way to sidestep the requirement that data be dense—for instance, if queries had a separate identifier that leaked when two queries had the same endpoints, this information could be used to estimate the distance between records.

Chapter 4

Access pattern attacks

Background. The last chapter built on KKNO's generic attacks [42] on range query leakage by considering an additional type of leakage common to many range query schemes. We derived attacks that exploit this extra type of leakage and achieve reconstruction with leakage from even fewer uniformly random queries: $\mathcal{O}(N \log N)$ versus $\mathcal{O}\big(N^2 \log N\big)$ for exact reconstruction of dense data.

In this chapter, we return to KKNO's original setting of just access pattern leakage. With Brice Minaud and Kenny Paterson, we observed that with a different analysis and algorithm, it is possible to achieve exact reconstruction on a dense database in only $\mathcal{O}(N \log N)$ uniformly random queries. We published our results in the paper "Improved Reconstruction Attacks on Encrypted Data Using Range Query Leakage" [47, 48], along with an approximate reconstruction attack. These results are only for dense databases, however, and the problem of reconstruction from access pattern leakage on sparse databases remained tempting, despite KKNO having proven a lower bound of $\Omega\big(N^4\big)$ uniformly random queries for exact reconstruction on sparse databases [42].

With Paul Grubbs, Brice Minaud, and Kenny Paterson, we tackled the problem of reconstruction of non-dense databases using a completely new approach and goal: approximate *ordered* reconstruction (AOR). The resulting attack's query complexity depends only on the relative desired precision, not the domain size N. This result was included in our paper "Learning to Reconstruct: Statistical Learning Theory and Encrypted Database Attacks." It was published at IEEE S&P 2019 [28] and its full version appears on the IACR's Cryptology ePrint Archive [29]. My main contribution to this paper was designing and implementing evaluations of the ϵ-AOR (here in

Section 4.3.4). This chapter also includes my own analysis of the VC dimension of the concept space $(\mathsf{X}_{\text{ranges}}, \mathbb{C}_{\text{ends}})$ and my own proof of complexity of the AOR attack.

Introduction. In the last chapter, we presented and analyzed database reconstruction attacks exploiting rank and access pattern leakage from range queries on dense databases. Recall the main idea of the first two attacks: records are partitioned according to which set of queries they matched (using access pattern leakage), then matched to classes of positions and sorted (using rank leakage). In this chapter, we consider a stronger class of attacks that use *only* access pattern leakage.

Reconstruction up to reflection. Using only access pattern leakage, without assuming any particular query or data distribution, it is impossible to reconstruct all records' values with probability 1, as the following example illustrates. Consider any database DB with values in $[1, N]$. The access pattern leakage of any query $[a, b]$ on DB is identical to the leakage of the query $[N+1-b, N+1-a]$ on a different database, DB', constructed from DB by mapping the value of every record, $\mathsf{val}(\mathsf{r})$, to $N+1-\mathsf{val}(\mathsf{r})$. If the query distribution were known, it would be possible to break the symmetry using the query frequency information. For example, if 1 is more likely to be a query endpoint than N, it would be possible to distinguish records having these two values by counting how many queries match the records with value 1 and how many match the records with value N. If the data distribution were partly known, then the symmetry could also be broken. For example, if more records are expected to have value a than $N+1-a$ (for any a other than the midpoint, $(N+1)/2$), then the number of records with these values can determine the correct order. The attacks in this section will reconstruct all records' values up to *global* reflection.

Chapter overview. We begin by presenting variants of the attacks in the last chapter that we adapted to eschew rank leakage. The exact reconstruction attack in this chapter (Algorithm 4.1 in Section 4.1) builds a partition of records in the same way as Algorithm 3.1 from Section 3.2, but sorts its elements differently. The approximate reconstruction attack in this chapter (Algorithm 4.2 in Section 4.2) takes a slightly different approach from the previous algorithms, but its success depends on similar events. In Section 4.3, we introduce an entirely new approach to access pattern attacks on range queries using a specialized data structure and tools from statistical learning theory. With this new approach, we can remove the requirement that the database be dense and achieve a new kind of reconstruction: ϵ-approximate order reconstruction (ϵ-AOR).

4.1 Exact reconstruction for dense data

Like Algorithm 3.1 in the previous chapter, the goal of this algorithm is to find a value mapping $\widehat{\mathsf{val}}$ that assigns the correct value to each record in the database. However, as mentioned in this chapter's introduction, without rank leakage or some other information about query or data distribution, we can reconstruct values only up to global reflection. The symbol "$\updownarrow$" denotes this restriction in the overview of attacks given in Table 1.2 in Section 1.2.

Reasoning behind the algorithm. In Algorithm 3.1, records were partitioned according to which set of queries they matched (using the access pattern leakage), then matched to classes of positions and sorted (using rank leakage). Without using rank leakage, however, it is still possible to sort the classes of records to assign them values from 1 to N. Suppose that sufficiently many queries have been observed to partition the records into N classes. Then, any query that matches more than one of these classes provides some information about their orders: their values must be in a continuous sub-interval of $[1, N]$ since they were chosen precisely based on their values being in that sub-interval. Similarly, if the sets of classes matched by two queries intersect, then the union of these sets must also be in a continuous sub-interval of $[1, N]$; the union of two overlapping intervals is also an interval.

We will use access pattern leakage to sort the N classes as follows. First, we identify one of the classes that corresponds to the value 1 or N—an initial "endpoint class." Since our target is full reconstruction up to global reflection, we can assign the records in this class the value 1. Next, we iteratively identify the classes of records with value 2, 3, and so on by finding unions of queries that match the endpoint class and 1 more class, and 2 more classes, and so on, until all other classes have been successively identified.

Algorithm details. The pseudocode in Algorithm 4.1 describes the details of the algorithm. First, the partition of records $\mathcal{P}_R$ is built as in the full reconstruction attack with rank and access pattern leakage, i.e., by grouping together records that matched exactly the same set of queries (lines 2–5). If the number of equivalence classes of records, $|\mathcal{P}_R|$, is not the number of possible values a record can have, N, then it is not possible to assign a value to each record—either not enough queries have been observed yet, or not every value appears in at least one record—so the algorithm aborts (line 7). If $|\mathcal{P}_R| = N$, then the algorithm proceeds by trying to sort the classes.

The first step in sorting the classes is identifying a class of records with value 1 or N by building a maximal strict subset of all records, R_{mss}. The algorithm picks (at random) a query that did not match all records and initializes R_{mss} to this set (line 8). Then, until it is no longer possible, it expands R_{mss} with overlapping query leakage sets that are not entirely contained in the current maximal strict subset and whose union with the current maximal strict subset is not all records (lines 9–10). When augmenting R_{mss} in this way is no longer possible, if the records that are *not* in this maximal strict subset, $\mathcal{R} \setminus R_{\text{mss}}$, belong to multiple classes of records in the partition of records, then the algorithm has failed (line 12). If the excluded records all belong to the same class, then they must be the records with value 1, up to reflection, and are assigned this value (line 14) and recorded in the book-keeping map $R_{\leq}$ (line 15). As the algorithm progresses, $R_{\leq}[j]$ will eventually be a map from integer $j \in [1, N]$ to the records with value less than or equal to j.

Next, starting with $j = 1$, the algorithm tries to identify records with value less than or equal to $j + 1$. At each iteration, it forms a set T by intersecting all query leakage sets R_i that overlap with, but are not entirely contained in, $R_{\leq}[j]$ and then removing $R_{\leq}[j]$ from this set (line 17). Then, it also removes from this set T any query leakage set that partially overlaps it "on the right." It does this by removing the records in any query leakage set R_i that intersects T but does not contain all of T, and does not intersect the previously identified records $R_{\leq}[j]$ (line 19). If, after this trimming of the set T, the records in it belong to multiple classes in the partition of records, then the algorithm has failed (line 21). However, if they all belong to the same class, then these records are assigned the value $j + 1$ (line 23), the book-keeping structure $R_{\leq}$ is updated (line 24), and the algorithm increments j until it equals $N - 1$. If the algorithm completes when $j = N - 1$, then it has succeeded and it returns the record-to-value mapping $\widehat{\mathsf{val}}$.

4.1.1 Leakage optimality

Like Algorithm 3.1 in the previous chapter, Algorithm 4.1 is leakage-optimal: whenever any other correct algorithm succeeds on the same leakage, Algorithm 4.1 also succeeds.

Proposition 4.1. *Let A be any correct exact reconstruction algorithm that takes as input the access pattern leakage from some range queries on a set of records $\mathcal{R}$ that have integer values between 1 and N. Then, whenever A succeeds on a particular input, so does Algorithm 4.1.*

Algorithm 4.1 Full reconstruction attack with access pattern leakage

Input: query leakage $\{\mathsf{R}_i\}_{i=1}^{\mathsf{n_q}}$.

Output: $\perp$ or map $\widehat{\mathsf{val}} : \mathcal{R} \to [1, N]$ such that either $\forall \mathsf{r} \in \mathcal{R}$, $\widehat{\mathsf{val}}(\mathsf{r}) = \mathsf{val}(\mathsf{r})$, or $\forall \mathsf{r} \in \mathcal{R}$, $\widehat{\mathsf{val}}(\mathsf{r}) = N + 1 - \mathsf{val}(\mathsf{r})$.

1: $\widehat{\mathsf{val}}$, Q_R, $R_{\leq} \leftarrow$ empty maps
2: **for all** $i \in \{1, \ldots, \mathsf{n_q}\}$ **do**
3: **for all** $\mathsf{r} \in \mathsf{R}_i$ **do**
4: $Q_R[\mathsf{r}] \leftarrow Q_R[\mathsf{r}] \cup \{i\}$
5: Form partition $\mathcal{P}_R$ of $\mathcal{R}$ with equivalence relation defined by $\mathsf{r} \equiv \mathsf{r}' \Leftrightarrow Q_R[\mathsf{r}] = Q_R[\mathsf{r}']$
6: **if** $|\mathcal{P}_R| \neq N$ **then**
7: **return** $\perp$
8: $R_{\text{mss}} \xleftarrow{\$} \{\mathsf{R}_i : |\mathsf{R}_i| < |\mathcal{R}|\}$
9: **while** $\exists \mathsf{R}_i : \mathsf{R}_i \cap R_{\text{mss}} \neq \emptyset$, $\mathsf{R}_i \not\subseteq R_{\text{mss}}$, and $\mathsf{R}_i \cup R_{\text{mss}} \subsetneq \mathcal{R}$ **do**
10: $R_{\text{mss}} \leftarrow R_{\text{mss}} \cup \mathsf{R}_i$
11: **if** $\nexists [R_k] \in \mathcal{P}_R : \{\mathcal{R} \setminus R_{\text{mss}}\} = \{\mathsf{r} \in \mathcal{R} : \mathsf{r} \in [R_k]\}$ **then**
12: **return** $\perp$
13: **for all** $\mathsf{r} \in \mathcal{R} \setminus R_{\text{mss}}$ **do**
14: $\widehat{\mathsf{val}}(\mathsf{r}) \leftarrow 1$
15: $R_{\leq}[1] \leftarrow \mathcal{R} \setminus R_{\text{mss}}$
16: **for all** $j \in \{1, \ldots, N-1\}$ **do**
17: $T \leftarrow \{\cap \{\mathsf{R}_i : \mathsf{R}_i \cap R_{\leq}[j] \neq \emptyset, \mathsf{R}_i \not\subseteq R_{\leq}[j]\}\} \setminus R_{\leq}[j]$
18: **while** $\exists \mathsf{R}_i : \mathsf{R}_i \cap T \neq \emptyset, \mathsf{R}_i \not\subseteq T \cup R_{\leq}[j], T \setminus \mathsf{R}_i \neq \emptyset$ **do**
19: $T \leftarrow T \setminus \mathsf{R}_i$
20: **if** $\nexists [R_k] \in \mathcal{P}_R : T = \{\mathsf{r} \in \mathcal{R} : \mathsf{r} \in [R_k]\}$ **then**
21: **return** $\perp$
22: **for all** $\mathsf{r} \in T$ **do**
23: $\widehat{\mathsf{val}}(\mathsf{r}) \leftarrow j + 1$
24: $R_{\leq}[j+1] \leftarrow R_{\leq}[j] \cup T$
25: **return** $\widehat{\mathsf{val}}$

Proof. Suppose that Algorithm 4.1 did not succeed. This failure must have occurred on line 7, 12, or 21. We show that in all cases, no other correct algorithm could have succeeded with probability 1 on the same input.

Suppose the failure occurred on line 7, so the number of groups in the partition of records $\mathcal{P}_R$ is strictly less than N. The same reasoning as in Proposition 3.1 applies: either all of the records in each class have the same value—in which case the database was not dense—or there is at least one class containing records with two or more different values—in which case we can construct a different (up to reflection) database that yields the same leakage. In either case, no other correct algorithm can succeed with probability 1 on this input.

Suppose the failure occurred on line 12. We will show that full reconstruction is impossible by changing the values of records in some classes without changing the leakage (up to reflection). At this point, the records are partitioned into N classes, each of which corresponds to records with one value. Since the fail condition was

triggered, the complement of the maximal strict subset R_{mss} contains at least two classes, and there can be no query that overlaps these classes yet does not contain all of $\mathcal{R} \setminus R_{\text{mss}}$—such a query would allow expanding R_{mss}. Then, consider the value mapping val' that coincides with the true value mapping val for all records in R_{mss}, but reverses the order of points in its complement, $\mathcal{R} \setminus R_{\text{mss}}$. This new value mapping cannot correspond to a reflection of the original values: $\mathcal{R} \setminus R_{\text{mss}}$ contains at least two classes and their relative ordering is changed, while the ordering of these classes relative to those in R_{mss} is unchanged. However, this new value mapping yields the same leakage as the actual one; the only type of query that could distinguish these value mappings is one that would add a class of points to R_{mss}. Thus, full reconstruction is impossible.

Lastly, suppose Algorithm 4.1 failed on line 21. At this point, the set T must contain at least two classes of records. Since the fail condition was triggered, there can be no query that overlaps but does not contain all of T, and contains records outside of T. We can thus construct a new value mapping by reversing the values of records in $R_{\leq}[j] \cup T$ and keeping the value of all other records the same. By construction, $R_{\leq}[j]$ and $R_{\leq}[j] \cup T$ form initial (up to reflection) segments of values 1 through N. Therefore, the only type of query that could distinguish this new value mapping from the true value mapping is one that overlaps a strict subset of T and at least part of $\mathcal{R} \setminus \{R_{\leq}[j] \cup T\}$, which is exactly the type of query that would remove a class of points from T. Therefore, full reconstruction by any correct algorithm is impossible.

Since, for each of the three points where Algorithm 4.1 can fail, any other correct algorithm would fail, Algorithm 4.1 is leakage-optimal. □

4.1.2 Query analysis

The leakage optimality of Algorithm 4.1 means that if it fails, full reconstruction is impossible: this algorithm requires at most as many queries as any other correct algorithm. The actual number of queries required, however, depends on the query distribution. Again, we analyze the number of required queries in the case where all $N(N+1)/2$ ranges occur with uniform probability. In this case, as the following theorem shows, the expected number of required queries for the algorithm to succeed is $\mathcal{O}(N \log N)$.

Theorem 4.2. *Let N be the number of possible values a record can have, and let* DB *be a set of records where each value appears in at least one record, i.e., the database is dense. Then, the probability that Algorithm 4.1 fails given the leakage from $\mathsf{n_q}$ queries drawn uniformly at random is at most $N \cdot e^{-\mathsf{n_q}/(N+1)} + 4 \cdot e^{-\mathsf{n_q}/(2N+2)}$.*

Proof. Let $\{[a_i, b_i]\}_{i=1}^{\mathsf{n_q}}$ be the set of sampled queries and $\{\mathsf{R}_i\}_{i=1}^{\mathsf{n_q}}$ be the corresponding leakage. For ease of exposition and avoidance of rounding functions, we assume that N is an even integer. We begin by proving the following claim. Recall that the length of a query is the number of values in it; the length of $[a, b]$ is $b - a + 1$.

Claim 4.3. *Algorithm 4.1 succeeds if*

1. *There is at least one query $[a, b]$ with $a = v$ and length at least 2 for each v from 1 to $N/2$.*
2. *There is at least one query $[a, b]$ with $b = v$ and length at least 2 for each v from $N/2 + 1$ to N.*
3. *There are two overlapping queries whose union is $[1, N/2]$.*
4. *There are two overlapping queries whose union is $[N/2 + 1, N]$.*

Proof of Claim 4.3. We show that when these four events hold, none of the failure conditions of Algorithm 4.1 can occur. Suppose the failure occurred on line 7, where the number of groups in the partition of records $\mathcal{P}_R$ is strictly less than N. In Claim 3.3 in the proof of Theorem 3.2, we defined similar events that guaranteed that the partition (which was formed in exactly the same way, with just access pattern leakage) had size N. Events 1, 2, and 3 here imply those events, therefore Algorithm 4.1 cannot fail here either.

Next, consider the failure at line 12. By its construction, the set R_{mss} holds some set of records whose values are in an unknown range, $[a_{\text{mss}}, b_{\text{mss}}]$, and there is no query overlapping R_{mss} whose union with R_{mss} is not the full set of records, $\mathcal{R}$. We first show that $a_{\text{mss}} \leq 2$. On the contrary, if $a_{\text{mss}} > N/2 + 1$, then the two overlapping queries from Event 4 could be used to extend R_{mss}. If $a_{\text{mss}} \in [3, N/2 + 1]$, then Event 1 with $a = v = a_{\text{mss}} - 1$ could be used to extend R_{mss}. Therefore, a_{mss} must be either 1 or 2. Similarly, using Events 2 and 3, we can deduce that b_{mss} must be either $N - 1$ or N, leaving four possible ranges that could be $[a_{\text{mss}}, b_{\text{mss}}]$. We can eliminate $[1, N]$ since $R_{\text{mss}} \subsetneq \mathcal{R}$ by construction, and we can eliminate $[2, N - 1]$ since Event 3 (or Event 4) could be used to extend R_{mss}. Therefore, the only two possibilities for the range of points corresponding to R_{mss} are $[1, N - 1]$ and $[2, N]$, leaving $\mathcal{R} \setminus R_{\text{mss}}$ to equal a single class of records in the partition of records, $\mathcal{P}_R$.

Lastly, consider the failure condition at line 21. At this point, $R_{\leq}[j]$ contains all records whose value (up to reflection) is less than or equal to j—there were N classes of records, the algorithm identified one of the extreme ones in $\mathcal{R} \setminus R_{\text{mss}}$, and in each iteration of the loop, the next successive class of records was identified. T is the

intersection of all queries overlapping but not contained in $R_{\leq}[j]$, minus $R_{\leq}[j]$ itself. Because of this construction, T contains exactly the records with values in the range $[j+1, b_T]$ for some unknown b_T. We will show that in every iteration j, b_T must be $j+1$ and therefore the fail condition cannot be triggered.

First, suppose $j+1 \leq N/2$. Event 3 means that the intersection of queries initially forming T must have values that are a subset of $[1, N/2]$, so $b_T \geq N/2+1$ is impossible. Also, b_T cannot be in $[j+2, N/2]$ because the query from Event 1 with left endpoint equal to b_T was subtracted from T in the previous loop. Next, consider $j+1 \geq N/2+1$. In this case, the query from Event 2 with right endpoint equal to $j+1$ means that the intersection of queries initially forming T contains values that are at most $j+1$. In either case, the right endpoint b_T of T must be $j+1$, so T is the set of records with value $j+1$, and the fail condition does not occur. Thus, when the four events occur, Algorithm 4.1 succeeds, ending the proof of Claim 4.3. □

Let us now continue with the proof of Theorem 4.2 by bounding the probabilities of these four events. Events 1 and 2 occur with the same probability by reflection, so consider only Event 1 for now. The probability that Event 1 does not occur, i.e., that at least one value $v \in \{1, \ldots, N/2\}$ does *not* arise as the left endpoint of any query that has length at least 2, is

$$\begin{aligned}
\text{Prob}[\neg\text{Event 1}] &\leq \sum_{v=1}^{N/2} \text{Prob}[a_i \neq v \ \forall i : |b_i - a_i| \geq 1] && \text{(union bound)} \\
&= \sum_{v=1}^{N/2} (1 - \text{Prob}[a_i = v, b_i \geq v+1])^{\mathsf{n_q}} \\
&= \sum_{v=1}^{N/2} \left(1 - \frac{N-v}{N(N+1)/2}\right)^{\mathsf{n_q}} && \text{(uniformity of queries)} \\
&\leq \sum_{v=1}^{N/2} \left(1 - \frac{1}{N+1}\right)^{\mathsf{n_q}} && (N - v \geq N - N/2 \geq N/2) \\
&\leq N/2 \left(1 - \frac{1}{N+1}\right)^{\mathsf{n_q}} \\
&\leq (N/2)\ e^{-\mathsf{n_q}/(N+1)} && \text{(Bound A.7).}
\end{aligned}$$

Next, consider Events 3 and 4, which also occur with the same probability due to reflection symmetry. Note that Event 4 is similar to Event 3 from the proof of Theorem 3.2: here, we have the additional restriction that the two queries overlap. We consider only pairs of queries having the form $[N/2+1, b]$ and $[a, N]$, where in the "left" queries, $b \in \{\lfloor 3N/4 \rfloor + 1, \ldots, N\}$ and in the "right" queries, $a \in \{N/2 + 1, \ldots, \lfloor 3N/4 \rfloor + 1\}$.

Then, the union of any "left" query and any "right" query is $\{N/2+1, \ldots, N\}$, and they intersect at least at $\lfloor 3N/4 \rfloor + 1$. The number of possible values of b in a "left" query is $N - \lfloor 3N/4 \rfloor \geq N - (3N/4) = N/4$, while there are $\lfloor 3N/4 \rfloor + 1 - N/2 \geq (3N/4 - 1) + 1 - (N/2) \geq N/4$ options for a in a "right" query. Therefore, the probability that any single query is *not* a "left" query (or not a "right" query) is at most $1 - \frac{N/4}{N(N+1)/2} = 1 - \frac{1}{2(N+1)}$ and the probability that Event 4 does not occur is

$$\begin{aligned}
\text{Prob}[\neg\text{Event 4}] &\leq \text{Prob}[\text{no "left"} \cup \text{ no "right"}] \\
&\leq \text{Prob}[\text{no "left"}] + \text{Prob}[\text{no "right"}] && \text{(union bound)} \\
&\leq 2\left(1 - \frac{1}{2(N+1)}\right)^{\mathsf{n_q}} && \text{(independence of queries)} \\
&\leq 2\left(e^{-\frac{1}{2(N+1)}}\right)^{\mathsf{n_q}} && \text{(Bound A.7).}
\end{aligned}$$

Finally, recalling that Events 1 and 2 occur with the same probability, and Events 3 and 4 occur with the same probability, we can apply a union bound to conclude that the probability that Algorithm 4.1 fails is at most

$$\begin{aligned}
&\text{Prob}[\neg\text{Event 1}, \neg\text{Event 2}, \neg\text{Event 3, or } \neg\text{Event 4}] \\
&\leq 2 \cdot \text{Prob}[\neg\text{Event 1}] + 2 \cdot \text{Prob}[\neg\text{Event 4}] \\
&\leq N \cdot e^{-\mathsf{n_q}/(N+1)} + 4 \cdot e^{-\mathsf{n_q}/(2N+2)}.
\end{aligned}$$

□

Therefore, given *only* the access pattern leakage from $\mathsf{n_q}$ uniformly sampled queries, Algorithm 4.1 succeeds in reconstructing the values of all records up to reflection with probability at least $1 - N \cdot e^{-\mathsf{n_q}/(N+1)} - 4 \cdot e^{-\mathsf{n_q}/(2N+2)}$. Next, we determine an upper bound on the expected number of uniformly random queries until Algorithm 4.1 succeeds.

Corollary 4.4. *The expected number of uniformly sampled queries until Algorithm 4.1 succeeds is at most* $N \log N + \mathcal{O}(N)$. *Specifically, for* $N \geq 4$, *it is at most* $(N+1)\log N + N + 8\sqrt{N} + 9$.

Proof. Let X be a random variable representing the number of queries (sampled uniformly at random) until the algorithm succeeds. By Theorem 4.2, we have $\text{Prob}[\mathsf{X} \leq x] \geq 1 - N \cdot e^{-x/(N+1)} - 4 \cdot e^{-x/(2N+2)}$, or, written as an inverse cumulative distribution function,

$$\text{Prob}[\mathsf{X} \geq x] \leq N \cdot e^{-(x-1)/(N+1)} + 4 \cdot e^{-(x-1)/(2N+2)}.$$

Then, applying Formula A.4, we obtain that the expected value is

$$\begin{aligned}\mathbb{E}[\mathsf{X}] &= \sum_{x=1}^{\lceil (N+1)\log N\rceil} \text{Prob}[\mathsf{X} \geq x] + \sum_{x=\lceil (N+1)\log N\rceil+1}^{\infty} \text{Prob}[\mathsf{X} \geq x] \\ &\leq (N+1)\log N + 1 + \left(\sum_{x=\lceil (N+1)\log N\rceil}^{\infty} N \cdot e^{-x/(N+1)}\right) \\ &\quad + \left(\sum_{x=\lceil (N+1)\log N\rceil}^{\infty} 4 \cdot e^{-x/(2N+2)}\right).\end{aligned}$$

Following similar steps as in the proof of Corollary 3.4, the first sum satisfies

$$\begin{aligned}\sum_{x=\lceil (N+1)\log N\rceil}^{\infty} N \cdot e^{-x/(N+1)} &= N \cdot \left(\frac{e^{-\lceil (N+1)\log N\rceil/(N+1)}}{1-e^{-1/(N+1)}}\right) && \text{(Formula A.6)} \\ &\leq N \cdot \left(\frac{e^{-\log N}}{1-e^{-1/(N+1)}}\right) \\ &= \frac{1}{1-e^{-1/(N+1)}} \\ &\leq N+2 && \text{(Bound A.10).}\end{aligned}$$

Similarly, the second sum satisfies

$$\begin{aligned}\sum_{x=\lceil (N+1)\log N\rceil}^{\infty} 4 \cdot e^{-x/(2N+2)} &= 4 \cdot \sum_{x=\lceil (N+1)\log N\rceil}^{\infty} e^{-x/(2N+2)} \\ &= 4 \cdot \left(\frac{e^{-\lceil (N+1)\log N\rceil/(2N+2)}}{1-e^{-1/(2N+2)}}\right) && \text{(Formula A.6)} \\ &\leq 4 \cdot \left(\frac{e^{-((N+1)\log N)/(2(N+1))}}{1-e^{-1/(2N+2)}}\right) \\ &= 4 \cdot \left(\frac{e^{-(\log N)/2}}{1-e^{-1/(2N+2)}}\right) \\ &= \frac{4}{\sqrt{N}} \cdot \left(\frac{1}{1-e^{-1/(2N+2)}}\right) \\ &\leq \frac{4}{\sqrt{N}} \cdot (2N+3) && \text{(Bound A.10)} \\ &= 8\sqrt{N} + \frac{12}{\sqrt{N}}.\end{aligned}$$

Combining these 3 parts of the sum, we obtain

$$\begin{aligned}\mathbb{E}[\mathsf{X}] &\leq (N+1)\log N + 1 + N + 2 + 8\sqrt{N} + \frac{12}{\sqrt{N}} \\ &\leq (N+1)\log N + N + 8\sqrt{N} + 9 && \text{for } N \geq 4. \qquad \square\end{aligned}$$

Thus, the expected number of queries to achieve full database reconstruction up to reflection with Algorithm 4.1 using only access pattern leakage from uniformly random queries is at most $N \log N + \mathcal{O}(N)$. Recall that Algorithm 3.1 in Section 3.2 had the same expected asymptotic query complexity despite using access pattern leakage *and* rank leakage. These results indicate that for range queries, rank leakage is somewhat superfluous; its only advantage asymptotically is breaking the symmetry between k and $N + 1 - k$.

4.2 Approximate reconstruction for dense data

As in the previous chapter, where both rank and access pattern leakage were available, it is also possible here, where only access pattern leakage is available, to gain partial information about values with fewer queries than for exact reconstruction. However, the algorithm in this section takes a different approach from Algorithm 3.2. Recall that the output of Algorithm 3.2 is two maps, $\widehat{\mathsf{val}}_{\mathsf{min}}$ and $\widehat{\mathsf{val}}_{\mathsf{max}}$, that assign a range of values to each record, and that different records can be assigned ranges of different widths. In this section, our algorithm assigns a range of values of the same size to each record; it will output a single value $\widehat{\mathsf{val}}$ for each record. To reflect this difference, we call the goal in this section ϵ-*approximate reconstruction*. Instead of assigning a range of values of diameter at most δN to each record, we try to assign a point guess that is less than ϵN away from the record's true value. Throughout, we assume ϵN is an integer, and we assign only point guesses that are integers (since all values are integers too).

Reasoning behind the algorithm. This algorithm takes a different approach from the previous one. Recall that when the number of classes of records in the partition of records $\mathcal{P}_R$ is strictly less than N, the values corresponding to some of these classes may not be intervals. When rank leakage was available, this limitation was tolerable: classes corresponding to non-continuous sets of records were identifiable. Now, however, a different approach ensures that the values corresponding to any sets of records we consider form an interval.

The algorithm has three main steps: splitting the records into two overlapping sets such that each set corresponds to a continuous interval of values, grouping and sorting the records within each set, and finally assigning values to each of the sorted groups. The assigned values depend on two properties: the index of the group in the sorted list of all groups and the number of groups relative to the number of possible values, N. Suppose there are $g < N$ sorted groups, $(R_1, R_2, \ldots, R_g)$. Our algorithm ensures

that all records with the same value are in the same group, so there are $N - g$ values that may not have their own group. Therefore, the records in group R_i can have any value in the range $[i, i + N - g]$. If $N - g$ is even, the point guess corresponding to the midpoint of this interval would be $(i + i + N - g)/2 = i + (N - g)/2$, otherwise it can be either $i + \lfloor (N - g)/2 \rfloor$ or $i + \lceil (N - g)/2 \rceil$. In all cases, the midpoint is at most $\lceil (N - g)/2 \rceil$ away from any value in the interval. Therefore, to achieve ϵ-approximate reconstruction, we require $\lceil (N - g)/2 \rceil < \epsilon N$, or that the number of groups satisfies $g > N - 2\epsilon N$.

Algorithm details. The first step is splitting the records into two "halves." Let r be any record in the database (line 2). (If splitting into halves does not succeed with this record, then another will be chosen.) Let M_{r} be the intersection of all queries containing r (line 3). Since an intersection of intervals is also an interval, the values corresponding to the records in M_{r} form a (continuous) interval. The goal of the splitting step is to find two overlapping sets of records L and R such that their union is the set of all records ($L \cup R = \mathcal{R}$), their intersection is M_{r} ($L \cap R = M_{\mathsf{r}}$), and the values of records in each L and R form continuous intervals.

First, we find a "left" query R_L and a "right" query R_R whose intersection is M_{r} and whose union $\mathsf{R}_L \cup \mathsf{R}_R$ is of maximal size among all such queries (line 4). Next, we find another "left" query $\mathsf{R}_{L'}$ that overlaps R_L and whose intersection with R_L is contained in $\mathsf{R}_L \setminus M_{\mathsf{r}}$, and whose union with R_L is maximal among all such queries (line 5). Similarly, we find another "right" query $\mathsf{R}_{R'}$ that overlaps R_R, whose intersection with R_R is contained in $\mathsf{R}_R \setminus M_{\mathsf{r}}$, and whose union with R_R is of maximal size (line 6). If the union of these four sets $\mathsf{R}_{L'} \cup \mathsf{R}_L \cup \mathsf{R}_R \cup \mathsf{R}_{R'}$ is the set of all records $\mathcal{R}$, then we go on to build $L = \mathsf{R}_{L'} \cup \mathsf{R}_L$ and $R = \mathsf{R}_R \cup \mathsf{R}_{R'}$ (lines 8 and 9). These are the two "halves," concluding the first step.

Next, we sort the records in each half and use M_{r} to bridge the sorted groups. To sort the left half, we look for queries that contain M_{r} and contain some records outside of the right half R, i.e., records that are only in the left half L. These queries can be ordered with respect to inclusion based on the records in L that they contain. Let C_L be this set of queries, where each query's record set is replaced with its intersection with $L \setminus M_{\mathsf{r}}$ and each set is numbered according to inclusion (line 10). We do the same process to obtain an ordered set C_R of sets of records, sorted by inclusion, in $R \setminus M_{\mathsf{r}}$ (line 12). Now, we can form $|C_L| + 1 + |C_R|$ disjoint groups of records based on which of the sorted elements of C_L or C_R they appeared in first (i.e., of the smallest size), or whether they are in M_{r}. If the number of disjoint groups of records is strictly greater than $N - 2\epsilon N$, then we assign values to the records in each of these sets as

explained in the introduction to this section (lines 16–22). Otherwise, we try again with a different record on line 3. If all records have been tested and the condition on line 15 was not satisfied for any of them, then the algorithm fails and returns $\perp$.

Algorithm 4.2 ϵ-Approximate reconstruction attack with access pattern leakage

Input: real $\epsilon \in [2/N, 1)$, query leakage $\{(\mathsf{R}_i)\}_{i=1}^{\mathsf{n_q}}$.

Output: $\perp$ or map $\widehat{\mathsf{val}} : \mathcal{R} \to \{1, \ldots, N\}$ such that for all records $\mathsf{r} \in \mathcal{R}$, $\left|\mathsf{val}(\mathsf{r}) - \widehat{\mathsf{val}}(\mathsf{r})\right| < \epsilon N$ or for all records $\mathsf{r} \in \mathcal{R}$, $\left|\mathsf{val}(\mathsf{r}) - (N + 1 - \widehat{\mathsf{val}}(\mathsf{r}))\right| < \epsilon N$.

1: $\widehat{\mathsf{val}} \leftarrow$ empty map
2: **for all** $\mathsf{r} \in \mathcal{R}$ **do**
3: $\quad M_{\mathsf{r}} \leftarrow \cap\{\mathsf{R}_i : \mathsf{r} \in \mathsf{R}_i\}$
4: $\quad$ Find $(\mathsf{R}_L, \mathsf{R}_R)$ such that $\mathsf{R}_L \cap \mathsf{R}_R = M_{\mathsf{r}}$ and $|\mathsf{R}_L \cup \mathsf{R}_R|$ is maximal
5: $\quad$ Find $\mathsf{R}_{L'}$ such that $\mathsf{R}_{L'} \cap \mathsf{R}_L \neq \emptyset$, $\mathsf{R}_{L'} \cap \mathsf{R}_L \subseteq \mathsf{R}_L \setminus M_{\mathsf{r}}$ and $|\mathsf{R}_{L'} \cup \mathsf{R}_L|$ is maximal
6: $\quad$ Find $\mathsf{R}_{R'}$ such that $\mathsf{R}_{R'} \cap \mathsf{R}_R \neq \emptyset$, $\mathsf{R}_{R'} \cap \mathsf{R}_R \subseteq \mathsf{R}_R \setminus M_{\mathsf{r}}$ and $|\mathsf{R}_{R'} \cup \mathsf{R}_R|$ is maximal
7: $\quad$ **if** $\mathsf{R}_{L'} \cup \mathsf{R}_L \cup \mathsf{R}_R \cup \mathsf{R}_{R'} = \mathcal{R}$ **then**
8: $\quad\quad L \leftarrow \mathsf{R}_{L'} \cup \mathsf{R}_L$
9: $\quad\quad R \leftarrow \mathsf{R}_R \cup \mathsf{R}_{R'}$
10: $\quad\quad C_L \leftarrow \{\mathsf{R}_i \setminus R : M_{\mathsf{r}} \subseteq \mathsf{R}_i, \mathsf{R}_i \setminus R \neq \emptyset\}$
11: $\quad\quad C_L[1] \subset C_L[2] \subset \cdots \subset C_L[|C_L|] \leftarrow C_L$
12: $\quad\quad C_R \leftarrow \{\mathsf{R}_i \setminus L : M_{\mathsf{r}} \subseteq \mathsf{R}_i, \mathsf{R}_i \setminus L \neq \emptyset\}$
13: $\quad\quad C_R[1] \subset C_R[2] \subset \cdots \subset C_R[|C_R|] \leftarrow C_R$
14: $\quad\quad g \leftarrow |C_L| + |C_R| + 1$
15: $\quad\quad$ **if** $g > N - 2\epsilon N$ **then**
16: $\quad\quad\quad$ **for all** $\mathsf{r} \in \mathcal{R}$ **do**
17: $\quad\quad\quad\quad$ **if** $\mathsf{r} \in L$ **then**
18: $\quad\quad\quad\quad\quad \widehat{\mathsf{val}}(\mathsf{r}) \leftarrow |C_L| + 1 - \min\{j : \mathsf{r} \in C_L[j]\} + \lfloor (N - g)/2 \rfloor$
19: $\quad\quad\quad\quad$ **else if** $\mathsf{r} \in M_{\mathsf{r}}$ **then**
20: $\quad\quad\quad\quad\quad \widehat{\mathsf{val}}(\mathsf{r}) \leftarrow |C_L| + 1 + \lfloor (N - g)/2 \rfloor$
21: $\quad\quad\quad\quad$ **else**
22: $\quad\quad\quad\quad\quad \widehat{\mathsf{val}}(\mathsf{r}) \leftarrow |C_L| + 1 + \min\{j : \mathsf{r} \in C_R[j]\} + \lfloor (N - g)/2 \rfloor$
23: $\quad\quad\quad\quad$ **return** $\widehat{\mathsf{val}}$
24: **return** $\perp$

4.2.1 Query analysis

We now show that Algorithm 4.2 succeeds with access pattern leakage from at most $N \log(1/\epsilon) + \mathcal{O}(N)$ queries sampled uniformly at random.

Theorem 4.5. *Let $N \geq 4$ be the number of possible values a record can have, let $\epsilon \in [2/N, 1 - 1/N)$ be the targeted precision, and let DB be a set of records where each value appears in at least one record, i.e., DB is dense. Then, the expected number of queries drawn uniformly at random until Algorithm 4.2 succeeds is at most $2(N + 1)\log(1/\epsilon) + 8N + 4(1 + 1/\epsilon)$.*

Proof. To avoid cluttered notation due to rounding, we assume N is a multiple of 4 and $\epsilon N/2$ is an integer. Let $\{[a_i, b_i]\}_{i=1}^{\mathsf{n_q}}$ be the set of sampled queries and $\{\mathsf{R}_i\}_{i=1}^{\mathsf{n_q}}$ be the corresponding leakage.

Claim 4.6. *Algorithm 4.2 succeeds if the following four events occur:*

1. *For $N/2 - \epsilon N/2$ distinct values v in $\{1, \ldots, N/2\}$, there is at least one query with $a = v$ and $b \geq N/2 + 1$.*
2. *For $N/2 - \epsilon N/2$ distinct values v in $\{N/2+1, \ldots, N\}$, there is at least one query with $a \leq N/2$ and $b = v$.*
3. *There are two overlapping queries whose union is $[1, N/2 + 1]$.*
4. *There are two overlapping queries whose union is $[N/2 + 1, N]$.*

Proof of Claim 4.6. Suppose that all four events occurred. Algorithm 4.2 can fail only if no suitable record r was found in the loop on line 2. We will show that if r is any record with value $N/2 + 1$, then the algorithm succeeds. First, Events 3 and 4 imply that M_{r} contains only records with value $N/2 + 1$, and that the four sets of queries on line 7 indeed cover all records $\mathcal{R}$. Event 1 implies that $|C_L| \geq (1 - \epsilon)N/2$, and Event 2 that $|C_R| \geq (1 - \epsilon)N/2 - 1$ (because Event 2 may include a query with $b = N/2 + 1$). Therefore, $|C_L| + |C_R| + 1 \geq (1 - \epsilon)N$, and the final check at line 15 succeeds, concluding the proof of Claim 4.6. □

Returning to the proof of Theorem 4.5, the expected number of queries until all events occur is at most the sum of the expected numbers of queries until each event occurs, so we look at the events separately. Events 1 and 2 occur with the same probability due to reflection, so consider only Event 1 for now. Let T_i^L be a random variable representing the number of queries drawn to get the ith distinct left endpoint in $[1, N/2]$ of a query with right endpoint at least $N/2 + 1$ after the $(i - 1)$st left endpoint (where $T_0^L \coloneqq 0$ for convenience). Since queries are drawn uniformly at random and there are initially $N/2$ candidates for v and always $N/2$ for each right endpoint, each T_i^L is a geometric random variable with success probability $p_i \coloneqq \frac{(N/2)(N/2-i)}{N(N+1)/2} = \frac{N/2-i}{N+1}$. Let the random variable $T^L \coloneqq \sum_{i=1}^{(1-\epsilon)N/2} T_i^L$ represent the total number of queries until Event 1 occurs. Since it is a sum of independent random variables, we use linearity of expectation to upper bound its expected value:

$$\begin{aligned}
\mathbb{E}\left[T^L\right] &= \sum_{i=1}^{(1-\epsilon)N/2} \mathbb{E}\left[T_i^L\right] = \sum_{i=1}^{(1-\epsilon)N/2} \frac{N+1}{N/2-i} = (N+1) \sum_{i=\epsilon N/2}^{N/2-1} \frac{1}{i} \\
&= (N+1)\left(\mathsf{H}_{N/2-1} - \mathsf{H}_{\epsilon N/2-1}\right) \\
&\leq (N+1)\left(\log \frac{1}{\epsilon} - \frac{2}{N}(1 - 1/\epsilon)\right) && \text{(Bound A.3)} \\
&\leq (N+1) \log \frac{1}{\epsilon} - 2(1 - 1/\epsilon).
\end{aligned}$$

The expected number of queries until Events 1 and 2 occur is therefore at most $2(N+1)\log(1/\epsilon) - 4(1-1/\epsilon)$.

Next, consider Events 3 and 4. We use the same technique as we did for Events 3 and 4 in the proof of Theorem 4.2: we consider only particular types of overlapping queries whose union is the given range. Here, Event 3 is slightly different—the overlapping queries' union must be $[1, N/2+1]$ instead of $[1, N/2]$—but this wider range translates to a higher probability of getting the desired type of query, so the expected number of queries until Event 3 occurs is at most the expected number of queries until Event 4 occurs. In Event 4, there are at least $N/4$ "left" queries and $N/4$ "right" queries that overlap by at least 1 value, $3N/4$. Since queries are drawn uniformly at random, the number of queries until a "left" (or "right") query is drawn is distributed geometrically with probability of success at least $(N/4)/(N(N+1)/2) = 1/(2(N+1))$. Hence, the expected number of queries until both Events 3 and 4 occur is at most

$$2 \cdot (2 \cdot (2(N+1))) = 8(N+1).$$

Combining this bound with the one for Events 1 and 2, we conclude that the expected number of queries until Algorithm 4.2 succeeds is at most

$$2(N+1)\log(1/\epsilon) - 4(1-1/\epsilon) + 8(N+1) = 2(N+1)\log(1/\epsilon) + 8N + 4(1+1/\epsilon),$$

concluding the proof of Theorem 4.5. □

Thus, the expected number of queries to achieve ϵ-approximate database reconstruction up to reflection with Algorithm 4.2 using only access pattern leakage is at most $N\log(1/\epsilon)+\mathcal{O}(N)$. As in the case for full reconstruction, the approximate reconstruction algorithm that uses rank leakage (Algorithm 3.2 in Section 3.3) has the same asymptotic query complexity—again, the only advantage of rank leakage asymptotically appears to be breaking symmetry.

4.3 Approximate ordered reconstruction for sparse data

All attacks so far require leakage from a number of queries that is at least linear in N, the number of possible values a record can take. They also require the database to be dense. As N increases, so must the number of records in the database to maintain density. In this section, we develop and analyze an approximate reconstruction attack

that succeeds even when not every value from 1 to N appears in the database. Further, the number of queries required depends only on the target precision, not on N.

The attack uses a special data structure—a PQ tree—to maintain information from range query leakage. We analyze it using properties of *concept spaces* over the set of all possible range queries, and apply results from learning theory that rely on the *Vapnik–Chervonenkis (VC) dimension* of the concept space, which roughly measures how complex it is.

The type of approximate reconstruction targeted by this attack is different from all previous attacks; instead, the structure of the output naturally arises from using a PQ tree. Specifically, this attack targets ϵ-*approximate ordered reconstruction* (ϵ-AOR) by grouping records together whose values are close and then sorting these groups. This goal is achieved when an algorithm outputs some number k of disjoint sets of records, $A_1, \ldots, A_k$, such that the diameter of each set, $\mathsf{diam}(A_i)$, is strictly less than ϵN, the sets are ordered correctly up to reflection (i.e., either $A_1 \overset{\text{val}}{<} \cdots \overset{\text{val}}{<} A_k$ or $A_1 \overset{\text{val}}{>} \cdots \overset{\text{val}}{>} A_k$), and every record whose value is further than ϵN from the endpoints 1 and N is belongs to one of the k sets.

Section overview. We first introduce PQ trees and the relevant concepts from VC theory in Sections 4.3.1 and 4.3.2. Then, in Section 4.3.3, we present our ϵ-approximate ordered reconstruction algorithm and analyze its query complexity. We experimentally evaluate the bound in Section 4.3.4.

4.3.1 PQ trees

PQ trees were discovered nearly 45 years ago [14]. They have been used as tools to solve problems like planarity testing for graphs and the consecutive ones problem. A PQ tree can compactly represent a set of *orderings* (permissible permutations) over some base set of elements and can be efficiently updated after learning about a new set of "consecutive" elements, referred to as an *interval*. We will use PQ trees to encode orderings of records compatible with the given access pattern leakage. The idea of using a PQ tree to order records in a database is not new [19], but it was not previously used to reconstruct values in a database.

Structure and interpretation. A PQ tree is a rooted tree with three types of nodes. The leaves are the elements of the base set—the records, in our setting. A PQ tree has two types of internal (i.e., non-leaf) nodes that dictate which orderings of their children are possible: P nodes and Q nodes. The children of a P node can be reordered in any way: if a P node has k children, then there are $k!$ possible orderings

of its children. The children of a Q node, on the other hand, can only be reflected; there are two possible orderings of its children. When a node has exactly two children, that node can be either a P node or a Q node. A node with one child can always be replaced by that child.

PQ trees support two operations: Build and Update. The Build operation takes a base set and outputs a PQ tree T consisting of a root P node and one leaf for each of the elements in the base set. The Update operation takes a PQ tree T and an *interval*, or subset of elements in the base set. (In our application, the intervals will be sets of records matched by queries.) It returns $\perp$ if the interval is not a subset of the tree's leaves, or if none of the orderings compatible with the tree allow the elements in the supplied interval to be contiguous. Otherwise, it modifies the structure of the PQ tree T precisely to reduce the set of orderings (compatible permutations) such that the elements in the provided set are always contiguous. If the elements in the provided interval are already contiguous in all compatible orderings, then the tree is not modified. The most restricted set of orderings that can be represented by a PQ tree is 2, in which case the root is a Q node with all elements in the base set as children.

Notation. If T is a PQ tree, we use root(T) to denote its root. PQ trees are commonly depicted with the root at the top, growing down. The children, descendants, parent, and ancestors of the node node are defined as follows:

- children(node): The set of direct descendants of the node node, empty iff node is a leaf.
- descendants(node): The set of node's children, its children's children, and so on (until the leaves), or node itself if it is a leaf.
- parent(node): The direct ancestor of the node node, empty iff node is the root.
- ancestors(node): The set of node's parent, its parent's parent, and so on (until the root).

Two more terms will also be helpful in dealing with the nodes in PQ trees:

- leaves(node): The subset of descendants of node that are leaves (i.e., the leaves of the subtree rooted at node). We may say that a leaf "belongs" to node if it is in the set leaves(node).
- lca(A): The lowest common ancestor of all nodes in the set A (i.e., the node furthest from the root that contains all nodes in A as descendants).

The symbols $\prec$ and $\succ$ (or $\preceq$ and $\succeq$, if repeated elements are allowed) denote an order relation between the leaves of a PQ tree T. We say that an ordering $\prec$ on the leaves of a PQ tree T is *compatible* with T if there is a way to re-order the children of the internal nodes, following rules about Q nodes' children (since P nodes' children may be in any order), such that the leaves, when read from left to right, are in the order dictated by $\prec$. Looking ahead, we will reason about which relations hold between at least three leaves of a PQ tree for all compatible orderings $\prec$. (Since the internal nodes in a PQ tree allow either reflections or arbitrary permutations of their children, it is always true that any two leaves a and b satisfy $a \prec b$ for some compatible ordering $\prec$.) For example, we will prove results about what it means when three leaves a, b, and c satisfy $a \prec b \prec c$ for all orderings compatible with a PQ tree T.

Useful properties. PQ trees have some appealing properties, which we state here, but do not prove. The size of any PQ tree on a base set of k elements is linear in k. The complexity of the Update procedure on a PQ tree is linear in the size of its base set. The order in which a PQ tree is updated on various intervals does not matter; the resulting tree represents the same set of orderings.

It is also possible to account for the case where not all elements of the base set are known in advance. A special leaf node can be added during the Build procedure, corresponding to the undiscovered elements. When running the Update procedure, if the interval contains a new element, then it is added as a sibling of this special leaf node (i.e., as a child of the special leaf node's parent).

PQ trees and AOR. Recall the goal of approximate ordered reconstruction: grouping together records with values that are close, and sorting these groups. Looking ahead to Section 4.3.3, our algorithm will simply build a PQ tree on the base set of all records, then run the Update procedure on each set of records in the observed access pattern leakage.

Access pattern leakage from range queries on a database can be interpreted exactly as *intervals* in the context of PQ trees: the records whose values fall in a range $[a, b]$ are precisely those that are contiguous when the set of all records is sorted by value. Consider how a PQ tree built on the base set of all records would look after running the Update procedure on the access pattern leakage of *all* possible range queries on a dense database. Its root would be a Q node and it would have N children, one for each value from 1 to N. If two records have the same value, then they must match exactly the same set of queries; there can be no range query that matches one but not the other. If a particular value appears in only one record, then it is a direct child of

the root Q node. Otherwise, all records with that particular value are children of a P node, and thus are grandchildren of the root Q node. This PQ tree is as reduced as it can be; the records have been grouped by value and ordered (up to reflection). When the database is dense, the first child of the Q node corresponds to the record(s) with value 1; the second, the record(s) with value 2; and so on.

We now state and prove some results about PQ trees that will be useful in the proof that our algorithm achieves ϵ-approximate order reconstruction. The first one says that if for all orderings compatible with a given PQ tree, the relative order of a set of elements is fixed up to reflection, then they must all belong to different children of their lowest common ancestor.

Lemma 4.7. *Let* $\mathsf{r}_1, \ldots, \mathsf{r}_k$ *be any* $k \geq 3$ *leaves in a PQ tree* T*, and let* $L = \mathsf{lca}(\mathsf{r}_1, \ldots, \mathsf{r}_k)$ *be their lowest common ancestor. Suppose that for all orderings compatible with* T*, either* $\mathsf{r}_1 \prec \mathsf{r}_2 \prec \cdots \prec \mathsf{r}_k$ *or* $\mathsf{r}_k \prec \mathsf{r}_{k-1} \prec \cdots \prec \mathsf{r}_1$*. Then, each leaf* r_i *belongs to a different child of the lowest common ancestor* L*, and* L *is a Q node.*

Proof. Consider any two leaves r_i and r_j and suppose, towards a contradiction, that they belong to the same child, C, of their lowest common ancestor L. Regardless of whether C is a P node or a Q node, it allows two relative orderings: $\mathsf{r}_i \prec \mathsf{r}_j$ and $\mathsf{r}_i \succ \mathsf{r}_j$. Consider now any other leaf r_h different from r_i and r_j, which must exist for $k \geq 3$. Without loss of generality, we can assume that the leaf r_h does not belong to C. (If all leaves belonged to the same child of L, then that child, not L, would be their lowest common ancestor.) Then, two relative orderings of C and r_h are possible: $C \prec \mathsf{r}_h$ and $C \succ \mathsf{r}_h$. However, within C, both $\mathsf{r}_i \prec \mathsf{r}_j$ and $\mathsf{r}_j \prec \mathsf{r}_i$ are possible, yielding two orders supposedly compatible with T that are not reflections of each other: $\mathsf{r}_i \prec \mathsf{r}_j \prec \mathsf{r}_h$ and $\mathsf{r}_j \prec \mathsf{r}_i \prec \mathsf{r}_h$. Therefore, the assumption that r_i and r_j belong to the same child of L must be false. Since r_i and r_j were arbitrary leaves, this implies that all k leaves belong to different children of L. Therefore, L has at least k children, and since $k \geq 3$, the relative ordering of the leaves cannot be fixed up to reflection unless L is a Q node. □

A corollary of this result is that the lowest common ancestor of any two of the leaves r_i and r_j in the statement of Lemma 4.7 must be L, the lowest common ancestor of the three leaves. By definition of the lowest common ancestor, it cannot be an ancestor of L, and if it were a child of L, then the two leaves would have to belong to that same child.

The next useful result is for a similar setting, where we consider groups A_i of leaves instead of individual leaves themselves. It expresses the fact that the sets A_i must also correspond to different children of their lowest common ancestor L.

Lemma 4.8. *Let $A_1, \ldots, A_k$ be $k \geq 3$ non-overlapping sets of leaves in a PQ tree T, let $A \coloneqq \cup_{i=1}^{k} A_i$ be their union, and let $L \coloneqq \mathsf{lca}(A)$ be their lowest common ancestor. Suppose that for all orderings compatible with T, either $A_1 \prec A_2 \prec \cdots \prec A_k$ or $A_k \prec A_{k-1} \prec \cdots \prec A_1$. Then for every child C of L, either $\mathsf{leaves}(C) \cap A = \emptyset$, or there exists a single set of leaves A_i such that $\mathsf{leaves}(C) \cap A_i \neq \emptyset$.*

Proof. Let C be a child of L and suppose $\mathsf{leaves}(C) \cap A \neq \emptyset$. Suppose, towards a contradiction, that the leaves of C overlap with the two sets A_{i_C} and $A_{i_{C'}}$ (and maybe others). Let r_C and $\mathsf{r}_{C'}$ be leaves in the intersections $\mathsf{leaves}(C) \cap A_{i_C}$ and $\mathsf{leaves}(C) \cap A_{i_{C'}}$ respectively.

Consider now another leaf r_h different from r_C and $\mathsf{r}_{C'}$. Without loss of generality, we can assume that the leaf r_h belongs to a different child of L than C, say C_h (otherwise C would be the lowest common ancestor, not L). Then, two relative orderings of C and r_h are possible: $C \prec \mathsf{r}_h$ and $C \succ \mathsf{r}_h$. However, within C, both $\mathsf{r}_C \prec \mathsf{r}_{C'}$ and $\mathsf{r}_C \succ \mathsf{r}_{C'}$ are possible, yielding two orders supposedly compatible with T: $\mathsf{r}_C \prec \mathsf{r}_{C'} \prec \mathsf{r}_h$ and $\mathsf{r}_{C'} \prec \mathsf{r}_C \prec \mathsf{r}_h$. Since $\mathsf{r}_C \in A_{i_C}$, $\mathsf{r}_{C'} \in A_{i_{C'}}$, and $\mathsf{r}_h \in A_{C_h}$, these two orderings on elements correspond to orderings on the sets $\{A_{i_C}, A_{i_{C'}}, A_{C_h}\}$ that are not reflections of each other. This contradiction implies that the leaves of any child C having a non-empty intersection with A must intersect with a single set of leaves A_i. □

The next result shows that after updating a PQ tree with two intervals of a particular form, the relative order of three elements is fixed in all orderings compatible with the tree.

Lemma 4.9. *Let r_1, r_2, and r_3 be three distinct elements in the base set, and let T be a PQ tree built on this base set. Suppose the tree T was updated with two intervals R and R' such that*

- *r_1 and r_2 are in R, but r_3 is not in R, and*
- *r_2 and r_3 are in R', but r_1 is not in R'.*

Then, in all orderings $\prec$ compatible with the PQ tree T, the relative order of r_1, r_2, and r_3 is fixed up to reflection as $\mathsf{r}_1 \prec \mathsf{r}_2 \prec \mathsf{r}_3$.

Proof. Let $\prec$ be any ordering compatible with the PQ tree. From the interval R, we know that r_3 cannot be between r_1 and r_2; it must be the case that either (i) $\mathsf{r}_3 \prec \mathsf{r}_1, \mathsf{r}_2$

or (ii) $\mathsf{r}_3 \succ \mathsf{r}_1, \mathsf{r}_2$. Similarly, from the interval R', we learn that r_1 cannot be between in r_2 and r_3; it must be the case that either (iii) $\mathsf{r}_1 \succ \mathsf{r}_2, \mathsf{r}_3$ or (iv) $\mathsf{r}_1 \prec \mathsf{r}_2, \mathsf{r}_3$. We now consider all four combinations of conditions:

- (i) and (iii) yield the possible ordering $\mathsf{r}_3 \prec \mathsf{r}_2 \prec \mathsf{r}_1$.
- (i) and (iv) yield a contradiction since $\mathsf{r}_3 \prec \mathsf{r}_1$ and $\mathsf{r}_1 \prec \mathsf{r}_3$ cannot both be true.
- (ii) and (iii) yield a contradiction since $\mathsf{r}_3 \succ \mathsf{r}_1$ and $\mathsf{r}_1 \succ \mathsf{r}_3$ cannot both be true.
- (ii) and (iv) yield the possible ordering $\mathsf{r}_3 \succ \mathsf{r}_2 \succ \mathsf{r}_1$.

Since the only two possibilities are $\mathsf{r}_3 \prec \mathsf{r}_2 \prec \mathsf{r}_1$ and $\mathsf{r}_3 \succ \mathsf{r}_2 \succ \mathsf{r}_1$, the order of the three records is fixed up to reflection. □

Lemma 4.9 is easily extended to the case where we consider sets of elements instead of single elements; the proof proceeds nearly identically. Recall that for two sets R and R', $R \prec R'$ means that for all $\mathsf{r} \in R$ and all $\mathsf{r}' \in R'$, we have $\mathsf{r} \prec \mathsf{r}'$.

Corollary 4.10. *Let R_1, R_2, and R_3 be disjoint, non-empty subsets of elements from the base set, and let T be a PQ tree built on this base set. Suppose the tree T was updated with two intervals R and R' such that*

- *$(R_1 \cup R_2) \subseteq \mathsf{R}$, but $R_3 \cap \mathsf{R} = \emptyset$, and*
- *$(R_2 \cup R_3) \subseteq \mathsf{R}'$, but $R_1 \cap \mathsf{R}' = \emptyset$.*

Then, in all orderings $\prec$ compatible with the PQ tree T, the relative order of R_1, R_2, and R_3 is fixed up to reflection as $R_1 \prec R_2 \prec R_3$.

We will use Lemma 4.9 and Corollary 4.10 extensively in the proof that our AOR algorithm succeeds after a certain number of queries. Another lemma that will be useful is the following, which proves that if the relative order of three elements is fixed in all orderings compatible with a PQ tree, then it is possible to "lift" the ordering to intervals containing them.

Lemma 4.11. *Let r_1, r_2, and r_3 be three distinct elements in the base set, and let T be a PQ tree built from this base set. Suppose that the tree T was updated with three disjoint intervals R_1, R_2, and R_3 whose intersections with the three elements $\{\mathsf{r}_1, \mathsf{r}_2, \mathsf{r}_3\}$ are exactly r_1, r_2, and r_3 respectively (e.g., $\mathsf{r}_1 \in R_1$, but $\{\mathsf{r}_2, \mathsf{r}_3\} \cap R_1 = \emptyset$). If for all orderings $\prec$ compatible with the tree T, $\mathsf{r}_1 \prec \mathsf{r}_2 \prec \mathsf{r}_3$ (up to reflection), then for all orderings $\prec$ compatible with the tree, $R_1 \prec R_2 \prec R_3$ (up to reflection) as well.*

Proof. Let $L \coloneqq \mathsf{lca}(R_1, R_2, R_3)$ be the lowest common ancestor of the elements in R_1, R_2, and R_3. By definition of *lowest* common ancestor, R_1, R_2, and R_3 cannot

all belong to the same child of L (otherwise that child would be their lowest common ancestor).

We first prove that each child of L can contain elements from at most one of the sets R_i. Suppose for a contradiction that a child C of L contains at least one element from multiple sets, say at least one from R_1 and at least one from R_2. Since the tree was updated for the interval R_1, the elements of R_1 are contiguous in all compatible orderings, and no elements of R_2 or R_3 can come between them in any compatible orderings (by contiguity and disjointness of the three sets R_i). Given that L's child C contains elements from both R_1 and R_2, the elements of R_1 cannot be contiguous unless they all belong to the same child C—otherwise, there would exist a compatible ordering where the elements of R_1 that belong to C and those that belong to another child may be separated by an element of R_2 (regardless of the node type of C). Similar reasoning implies that all elements in R_2 must also belong to the same child C as all of the elements in R_1. Now, because $\mathsf{r}_1 \in R_1$ and $\mathsf{r}_2 \in R_2$, they must both belong to the child C. However, by hypothesis, $\mathsf{r}_1 \prec \mathsf{r}_2 \prec \mathsf{r}_3$ in all orderings compatible with the PQ tree, and by implication of Lemma 4.7, the lowest common ancestor of any two of these three elements is also the lowest common ancestor of all three. Since r_1 and r_2 both belong to C, their lowest common ancestor must be C or a descendant of C, which implies that r_3 must also belong to C. Applying the same reasoning we applied to elements in R_1 and R_2, if r_3 belongs to the child C of L, then so must all other elements of R_3. However, this contradicts the definition of lowest common ancestor. Therefore, each child C of L must contain elements from at most one of the intervals R_1, R_2, and R_3.

Now, L has at least three different children, and the elements r_1, r_2, and r_3 each belong to different children, so L is their lowest common ancestor too, and, by Lemma 4.7, it is a Q node. Lastly, because the tree T was updated with the intervals R_1, R_2, and R_3, we know that the elements in each set are contiguous in all compatible orderings. Therefore, the children of L having a non-empty intersection with R_1 (or R_2, or R_3) are contiguous in all compatible orderings, so an ordering on R_1, R_2, and R_3 is well defined. The last step is to prove that these three sets are ordered correctly. Since L is a Q node, three relative orders (up to reflection) are possible: (i) $R_1 \prec R_2 \prec R_3$, (ii) $R_2 \prec R_1 \prec R_3$, and (iii) $R_1 \prec R_3 \prec R_2$. By hypothesis, $\mathsf{r}_1 \prec \mathsf{r}_2 \prec \mathsf{r}_3$ in all compatible orderings, therefore (ii) and (iii) are eliminated, concluding the proof. □

The proof of Lemma 4.11 does not assume that any of the intervals R_1, R_2, or R_3 contains more than one element. Note that running the Update procedure on a singleton interval is a null operation, in the sense that it has no effect on the structure

of the tree, so we may always assume that the Update procedure of a PQ tree was called on any interval containing a single element of its base set. This implies the following result:

Corollary 4.12. *Let* r_1, r_2, *and* r_3 *be three distinct elements in the base set, and let* T *be a PQ tree built from this base set. Let* R_1, R_2, *and* R_3 *be three disjoint subsets of the base set whose intersections with the three elements* $\{\mathsf{r}_1, \mathsf{r}_2, \mathsf{r}_3\}$ *are exactly* r_1, r_2, *and* r_3 *respectively. Suppose that for each set* R_i, *either* $R_i = \{\mathsf{r}_i\}$ *or the tree* T *was updated with the interval* R_i. *If for all orderings* $\prec$ *compatible with the tree* T, $\mathsf{r}_1 \prec \mathsf{r}_2 \prec \mathsf{r}_3$ *(up to reflection), then for all orderings* $\prec$ *compatible with the tree,* $R_1 \prec R_2 \prec R_3$ *(up to reflection) as well.*

4.3.2 Background on statistical learning theory

Ground sets and concepts. The terminology we use in this section is mainly from Mitzenmacher and Upfal's 2017 textbook [56]. Let X be a finite set of elements. In our application, the particular ground set we use is $\mathsf{X}_{\text{ranges}} \coloneqq \{[a, b] : 1 \le a \le b \le N\}$, the set of all possible range queries on values 1 to N. A *concept* C is a subset of X. For example, we can define a concept for each value i from 1 to N as the set of ranges matching i, or for each pair of values $i < j$ as the set of ranges whose left endpoint is in $[i, j]$. Any concept C can be viewed as a function $\mathsf{X} \to \{0, 1\}$ (its characteristic function): the function's output on input $x \in \mathsf{X}$ is 1 if $x \in C$ and 0 otherwise. Given a probability distribution D on the set X, let f_D represent the probability mass function. The probability $\text{Prob}_\mathsf{D}[C]$ of a concept C is equal to the probability that a single element of X sampled according to D is in C, i.e., $\text{Prob}_\mathsf{D}[C] \coloneqq \sum_{c \in C} \mathsf{f}_\mathsf{D}(c)$.

Concept spaces. A *concept space* is a pair $(\mathsf{X}, \mathbb{C})$ where $\mathbb{C}$ is a set of concepts (subsets) of X. Given a concept space $(\mathsf{X}, \mathbb{C})$ and a sample S of elements drawn from X according to D, we may ask whether every concept in $\mathbb{C}$ with some not-too-small probability occurs in the sample S. Answering this question involves analyzing an object called an ϵ-*net* [34].

ϵ-Nets. A subset $S \subseteq \mathsf{X}$ is an ϵ-*net* for the concept space $(\mathsf{X}, \mathbb{C})$ with respect to the distribution D if for every concept $C \in \mathbb{C}$ having $\text{Prob}_\mathsf{D}[C] \ge \epsilon$, the intersection $S \cap C$ is non-empty. Informally, a sample S is an ϵ-net iff every concept of probability at least ϵ occurs in the sample. One way to analyze when a subset $S \subseteq \mathsf{X}$ is an ϵ-net is to characterize the complexity of the concept space. 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 finite concept space $(\mathsf{X}, \mathbb{C})$ and a 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): $\mathbb{C}_S \coloneqq \{C \cap S\}_{C \in \mathbb{C}}$. The size of this set is called the *index of* $\mathbb{C}$ *with respect to* S, $\Delta_{\mathbb{C}}(S)$:

$$\Delta_{\mathbb{C}}(S) \coloneqq |\mathbb{C}_S| = |\{C \cap S : C \in \mathbb{C}\}| .$$

The index of a concept space relative to a set S is at most the number of possible subsamples, $2^{|S|}$, and, when $\mathbb{C}$ is finite (which is always the case in this thesis), it is at most $|\mathbb{C}|$.

Shattering and VC dimension. 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., if $\Delta_{\mathbb{C}}(S) = 2^{|S|}$. The *VC dimension*, denoted by d, of a concept space $(\mathsf{X}, \mathbb{C})$ is the largest size of a sample $S \subseteq \mathsf{X}$ 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 $\mathbb{C}$. Proving that the VC dimension of a concept space is at least a given number is easy; it suffices to give an example of a sample of that size that is shattered by the concept space. On the other hand, proving that the VC dimension of a concept space is less than a given number is harder; it requires proving that *any* sample of that size is not shattered by the concept class. VC dimension is an indicator of the complexity of a concept space. Related to VC dimension is the *growth function* $m_{\mathbb{C}}(n)$ 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: $m_{\mathbb{C}}(n) \coloneqq \max_{S \subseteq \mathsf{X} : |S| = n} \Delta_{\mathbb{C}}(S)$. The VC dimension, then, is the largest value of n for which the growth function equals 2^n. Knowing an upper bound on the growth function can also give an upper bound on the VC dimension.

One of the fundamental results of VC theory is that knowing the VC dimension of a concept space is sufficient to determine an upper bound on the growth function: either the VC dimension d is infinite or the growth function is bounded by $\sum_{i=0}^{d} \binom{n}{i}$.

Sauer's Lemma [75]. *Let* $(\mathsf{X}, \mathbb{C})$ *be a concept space having finite VC dimension* d*. Then, the growth function satisfies* $m_{\mathbb{C}}(n) \leq \sum_{i=0}^{d} \binom{n}{i}$.

Consider the ground set $\mathsf{X}_{\text{ranges}}$ of all $N(N+1)/2$ ranges in $[1, N]$, and the concept space $\mathbb{C}_{\text{point}} \coloneqq \{A_k : k \in [1, N]\}$ of N classes, where the concept $A_k \coloneqq \{[a, b] : k \in [a, b]\}$ is the set of ranges containing value k. Then, as the next lemma establishes, the VC dimension of this concept space is 2.

Lemma 4.13. $(\mathsf{X}_{ranges}, \mathbb{C}_{point})$ *has growth function* $2n$ *and VC dimension 2.*

Proof. Let S be any sample of n ranges, say $S = \{[a_1, b_1], \ldots, [a_n, b_n]\}$. Consider the set of points Y comprising 1, $N+1$, all of the left endpoints of ranges in S, and 1 more

than each right endpoint of ranges in S: $Y \coloneqq \{1, N+1\} \cup \{a_i\}_{1 \le i \le n} \cup \{b_i + 1\}_{1 \le i \le n}$. It contains some $\ell \le 2n+2$ elements, $y_1 < \ldots < y_\ell$, where $y_1 = 1$ and $y_\ell = N+1$. By adding $b_i + 1$ instead of b_i, we obtain the property that any $y \in Y$ matches a different subset of ranges in S than $y - 1$. The point $N + 1$ is added to Y for convenience only. Whenever two distinct integers k_1 and k_2 lie in the same interval $[y_i, y_{i+1})$, they must match exactly the same subset of ranges in S, since all ranges match both or neither of them. Formally, in terms of the concepts A_i, we have $S \cap A_{k_1} = S \cap A_{k_2}$. Therefore, the concepts corresponding to values in an interval $[y_i, y_{i+1})$ contribute at most 1 different induced subsample of S. Given that ℓ points create $\ell - 1$ intervals, the growth function is at most $\ell - 1 \le 2n + 1$.

If $\ell \le 2n + 1$, then $\ell - 1 \le 2n$, so we are done. If $\ell = 2n + 2$, then, because $|S| = n$, all left endpoints a_i must be strictly greater than $y_1 = 1$ and all right endpoints $b_i + 1$ must be strictly less than $y_\ell = N + 1$. In this case, for any $k_1 \in [y_1, y_2) = [1, y_2)$ and $k_2 \in [y_{\ell-1}, y_\ell) = [y_{\ell-1}, N + 1)$, $S \cap A_{k_1} = S \cap A_{k_2} = \emptyset$, so the growth function (i.e., the maximum index of $\mathbb{C}_{\text{point}}$ over a sample of size n) is again at most $2n$.

We now show this bound is tight for $N \ge 2n$ by constructing a set S of n ranges such that $S \cap A_k$ takes $2n$ distinct values as k spans $[1, N]$. Consider the set of ranges $S \coloneqq \{[1, n], [2, n + 1], \ldots, [n, 2n - 1]\}$. For $N \ge 2n$, the resulting set Y is $\{1, 2, \ldots, 2n, N + 1\}$, with $|Y| = \ell = 2n + 1$. Each of the sets A_k for $k = 1, \ldots, 2n$ induces a distinct subsample of S: for k in $\{1, \ldots, n\}$, the induced subsample $S \cap A_k$ is $\{[1, n], \ldots, [k, n+k-1]\}$, while for $k \in \{n+1, \ldots, 2n-1\}$, it is $\{[k-n+1, k], \ldots, [n, 2n-1]\}$, and for $k = 2n$, it is the empty subsample $\emptyset$.

Hence, the growth function of $(\mathsf{X}_{\text{ranges}}, \mathbb{C}_{\text{point}})$ is $2n$ for $N \ge 2n$. Since $\mathbb{C}_{\text{point}}$ induces at most $2n$ subsamples in a sample of size n, and shattering a sample requires 2^n subsamples (and $2^n > 2n$ for $n > 2$), the size of the largest sample that can be shattered, and thus the VC dimension of this concept space, is 2. □

In the analysis of our ϵ-AOR attack in Section 4.3, we use a slightly more complex concept space over the ground set of ranges $\mathsf{X}_{\text{ranges}}$, where the set of concepts is defined as $\mathbb{C}_{\text{ends}} \coloneqq \{A_{s,t,u,v} : 1 \le s < t \le u < v \le N\}$, where the concept $A_{s,t,u,v} \coloneqq \{[a, b] : a \in [s, t] \text{ and } b \in [u, v]\}$. Its concept classes include ranges whose left and right endpoints fall in some particular intervals. To analyze its VC dimension, we use the following result characterizing concept spaces built from other concept spaces.

Concept class construction lemma [45, Lemma 9.7]. *Let* $(\mathsf{X}, \mathbb{C})$ *and* $(\mathsf{X}, \mathbb{C}')$ *be two concept spaces over the same ground set with VC dimensions* d *and* d' *respectively.*

Then, the concept space $(\mathsf{X}, \mathbb{C} \cap \mathbb{C}')$*, where* $\mathbb{C} \cap \mathbb{C}' \coloneqq \{C \cap C' : C \in \mathbb{C}, C' \in \mathbb{C}'\}$*, has VC dimension at most* $d + d' - 1$*.*

The set of concepts of interest, $\mathbb{C}_{\text{ends}}$, can be written as the intersection of the following two concept spaces:

$$\mathbb{C}_{\text{L}} \coloneqq \{A_{s,t} : 1 \leq s < t \leq N\} \text{ where } A_{s,t} \coloneqq \{[a,b] : a \in [s,t]\}, \text{ and}$$
$$\mathbb{C}_{\text{R}} \coloneqq \{A_{u,v} : 1 \leq u < v \leq N\} \text{ where } A_{u,v} \coloneqq \{[a,b] : b \in [u,v]\}.$$

We use the Concept class construction lemma to determine an upper bound on the VC dimension of $(\mathsf{X}_{\text{ranges}}, \mathbb{C}_{\text{ends}})$. Consider $\mathbb{C}_{\text{L}}$, the set of concepts corresponding to ranges whose left endpoints fall in some given range. The following lemma proves that its VC dimension is at most 2:

Lemma 4.14. *The growth function of* $(\mathsf{X}_{ranges}, \mathbb{C}_L)$ *is at most* $n^2/2 + n/2 + 1$ *and its VC dimension is at most 2.*

Proof. Let S be any sample of n ranges, say $S = \{[a_1, b_1], \ldots, [a_n, b_n]\}$. Consider the set of points $Y \coloneqq \{1, N+1\} \cup \{a_i\}_{1 \leq i \leq n}$. It contains some $\ell \leq n + 2$ elements, $y_1 < \ldots < y_\ell$, where $y_1 = 1$ and $y_\ell = N + 1$, and splits $[1, N+1]$ into $\ell - 1 \leq n + 1$ intervals. Every non-empty intersection of S with the concepts $A_{s,t}$ of $\mathbb{C}_{\text{L}}$ can be characterized by which intervals $(y_i, y_{i+1}]$, $[y_j, y_{j+1})$, with $i \neq j$, the endpoints s and t fall in. Indeed, where exactly the minimum left endpoint s falls in the range $(y_i, y_{i+1}]$ does not matter since, by definition, no query in S had a left endpoint in (y_i, y_{i+1}). Similarly, where the maximum left endpoint t falls in the range (y_j, y_{j+1}) does not matter since no query in S had a left endpoint in (y_j, y_{j+1}). When $i = j$, the representative endpoints s and t must both be in (y_i, y_{i+1}), which results in an empty intersection with S since no queries have left endpoints in this interval.

Since there are $\ell - 1 \leq n+1$ intervals in which we can pick representative values of s and t, the number of different (empty or non-empty) intersections of S with the concepts of $\mathbb{C}_{\text{L}}$—and therefore the growth function—is at most $\binom{\ell-1}{2} + 1 \leq n(n+1)/2 + 1$. Since $2^n > n(n+1)/2 + 1$ for any integer $n > 2$, and they are equal for $n = 2$, we may conclude that the VC dimension of $(\mathsf{X}_{\text{ranges}}, \mathbb{C}_{\text{L}})$ is at most 2. □

Using a nearly identical proof, we conclude that the VC dimension of $\mathbb{C}_{\text{R}}$ is also at most 2. Thus, we are able to prove that the VC dimension of $\mathbb{C}_{\text{ends}}$ is at most 3. Since $\mathbb{C}_{\text{L}} \cap \mathbb{C}_{\text{R}} = \mathbb{C}_{\text{ends}}$, the result follows directly from the Concept class construction lemma.

Corollary 4.15. *The VC dimension of* $(\mathsf{X}_{ranges}, \mathbb{C}_{ends})$ *is at most 3.*

Haussler and Welzl, who introduced ϵ-nets, derived an upper bound on the sample size required to obtain an ϵ-net in the case where the distribution over X is uniform [34, Thm. 3.7]. Later work extended this bound to arbitrary distributions D over X:

ϵ-Net Theorem [56]. *Let* $(\mathsf{X}, \mathbb{C})$ *be a concept space with growth function* $m_{\mathbb{C}}(n)$ *and VC dimension* d*. Let* D *be a probability distribution on* X *and let* S *be a set of size* n *drawn from* X *according to* 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 drawn according to D *is an* ϵ*-net with probability at least* $1 - \delta$*. Specifically, a sample drawn according to* D *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.* proved a lower bound [22, Cor. 5] on the number of samples needed to obtain an ϵ-net with probability at least $1 - \delta$.

Theorem [22, 56]. *Let* $(\mathsf{X}, \mathbb{C})$ *be a concept space of VC dimension* d*. Let* D *be a probability distribution on* X *and let* S *be a sample drawn from* X *according to* D*. Let* $\epsilon > 0$ *and* $\delta > 0$*. Suppose* S *is an* ϵ*-net with probability at least* $1 - \delta$*. Then,* $|S| = \Omega\left(\frac{d}{\epsilon} + \frac{1}{\epsilon}\log\frac{1}{\delta}\right)$.

We now have the necessary results to analyze our algorithm, presented in the next section.

4.3.3 ϵ-Approximate ordered reconstruction algorithm

The ϵ-AOR algorithm targets ϵ-approximate ordered reconstruction: it groups records with close values into sets and orders these sets, up to reflection. The idea behind the algorithm is simple: build a PQ tree whose leaves are records and run the Update procedure on the access pattern leakage from every query. In this way, the number of possible orderings of the records is gradually restricted. Once all queries have been processed, the algorithm looks for a Q node that has the majority of leaves (records) below it. Since the possible orderings of a Q node's children are fixed up to reflection, the records can be grouped according to which children of the Q node they belong to. The sets A_i output by the algorithm correspond to the leaves of the children of this Q node.

Algorithm 4.3 Approximate ordered reconstruction attack with access pattern leakage

Input: query leakage $\{(\mathsf{R}_i)\}_{i=1}^{\mathsf{n_q}}$.
Output: $\perp$ or non-overlapping sets of records $A_1, \ldots, A_k$.

1: $\mathsf{T} \leftarrow \mathsf{Build}(\mathcal{R})$
2: **for all** $i \in \{1, \ldots, \mathsf{n_q}\}$ **do**
3: $\quad \mathsf{T.Update}(\mathsf{R}_i)$
4: $\mathsf{currNode} \leftarrow \mathsf{root}(\mathsf{T})$
5: **while** $\exists C \in \mathsf{children}(\mathsf{currNode})$ with $|\mathsf{leaves}(C)| > \mathsf{n_r}/2$ **do**
6: $\quad \mathsf{currNode} \leftarrow C$
7: **if** currNode is a P node and $|\mathsf{children}(\mathsf{currNode})| > 2$ **then**
8: $\quad$ **return** $\perp$
9: $C_1, \ldots, C_k \leftarrow \mathsf{children}(\mathsf{currNode})$
10: **for all** $i \in \{1, \ldots, k\}$ **do**
11: $\quad A_i \leftarrow \mathsf{leaves}(C_i)$
12: **return** $A_1, \ldots, A_k$

Algorithm details. The algorithm begins by building a PQ tree that initially consists of a single P node with as many children as there are records (line 1). It then processes the access pattern leakage from each query using the PQ tree's update procedure to restrict the set of orderings it encodes (lines 2–3). To find the Q node whose children's leaves will form the sets A_i output by the record, we first find the deepest (i.e., closest to the root) node in the tree that is an ancestor of the majority of records (lines 4–6). If this node is not a Q node (which is equivalent to a P node when it has 2 children), then the algorithm fails and returns $\perp$ (line 8). Otherwise, the leaves of each of this node's children (in order) form the sets $A_1, \ldots, A_k$ returned by the algorithm (lines 9–12).

While this algorithm itself is simple, its analysis is less so. The main idea is to observe enough queries that all records having value further than ϵN from the endpoints are partitioned into groups of small diameter (close to the target precision, ϵN), and that the order of these groups is known.

Necessary properties. Our analysis requires the records in the database DB to have three properties. First, we require that there are at least two records, r_a and r_b, which we sometimes refer to as *anchor records*, whose values are in the range $(N/4, 3N/4)$ and are at least $N/3$ apart:

Property 1. There exist two records $\mathsf{r}_a, \mathsf{r}_b \in \mathsf{DB}$ such that $\mathsf{val}(\mathsf{r}_a) \in (N/4, 3N/4)$, $\mathsf{val}(\mathsf{r}_b) \in (N/4, 3N/4)$, and $|\mathsf{val}(\mathsf{r}_b) - \mathsf{val}(\mathsf{r}_a)| > N/3$.

Second, we require that there are at least three records, r_c, r_d, and r_e (which could be r_a and/or r_b), whose values are in the range $(\epsilon N, N + 1 - \epsilon N)$, and whose values are at least ϵN apart from each other:

Property 2. There exist three records $\mathsf{r}_c, \mathsf{r}_d, \mathsf{r}_e \in \mathsf{DB}$ such that $\mathsf{val}(\mathsf{r}) \in (\epsilon N, N+1-\epsilon N)$ for each $\mathsf{r} \in \{\mathsf{r}_c, \mathsf{r}_d, \mathsf{r}_e\}$ and $\min_{(\mathsf{r},\mathsf{r}') \subset \{\mathsf{r}_c,\mathsf{r}_d,\mathsf{r}_e\}} |\mathsf{val}(\mathsf{r}) - \mathsf{val}(\mathsf{r}')| > \epsilon N$.

Properties 1 and 2 are used to guarantee the existence of a Q node whose children's leaves will be the sets of records output by the algorithm.

Third, we require that the values in the database are not concentrated at the endpoints 1 and N, or in a range of length ϵN:

Property 3. The number of records with values in the range $(\epsilon N, N+1-\epsilon N)$, is strictly greater than $\mathsf{n}_\mathsf{r}/2$, and there is no i such that the number of records with values in the range $(i, i+\epsilon N)$ is strictly greater than $\mathsf{n}_\mathsf{r}/2$. Property 3 is used to ensure that the Q node is the deepest one in the PQ tree.

The "ϵ" in ϵ-net. Our proof uses the concept space $(\mathsf{X}_{\text{ranges}}, \mathbb{C}_{\text{ends}})$, introduced in Section 4.3.2, whose VC dimension is at most 3 (q.v. Corollary 4.15). Recall that a concept $A_{s,t,u,v} \in \mathbb{C}_{\text{ends}}$ is the set of all ranges whose left endpoint is in $[s,t]$ and whose right endpoint is in $[u,v]$. Under the uniform distribution on $\mathsf{X}_{\text{ranges}}$, the probability of concept $A_{s,t,u,v}$ is

$$\frac{(t-s+1)(v-u+1)}{N(N+1)/2}.$$

In particular, if $t-s+1 \geq \lfloor \epsilon N/2 \rfloor$ and $v-u+1 \geq N/4$, then the probability of concept $A_{s,t,u,v}$ is at least

$$\begin{aligned}
\frac{\lfloor \epsilon N/2 \rfloor \cdot N/4}{N(N+1)/2} = \frac{\lfloor \epsilon N/2 \rfloor}{2(N+1)} \geq \frac{(\epsilon N - 1)/2}{2(N+1)} &= \frac{\epsilon - 1/N}{4 + 4/N} \\
&\geq \frac{\epsilon - 1/4}{5} \qquad (N \geq 4) \\
&= \epsilon/5 - 1/20.
\end{aligned}$$

Our analysis uses ϵ-nets with "ϵ" equal to $\epsilon/5 - 1/20$.

Theorem 4.16. *Let N be the number of possible values a record can have, and let DB be a set of records satisfying Properties 1, 2, and 3. Let $\epsilon \in (1/N, 1/4)$ be the desired precision. Then, given the leakage from at least $\mathcal{O}\left(\frac{1}{\epsilon}\log\frac{1}{\epsilon} + \frac{1}{\epsilon}\log\frac{1}{\delta}\right)$ range queries drawn uniformly at random, the probability that the output of Algorithm 4.3 does not achieve ϵ-approximate ordered reconstruction is at most δ.*

Proof. Suppose, for convenience and to avoid cluttered notation, that ϵN is an integer (so $\epsilon N \geq 2$). Our proof uses concepts from $(\mathsf{X}_{\text{ranges}}, \mathbb{C}_{\text{ends}})$ with probability at least $\epsilon/5 - 1/20$. All concepts $A_{s,t,u,v}$ with $t-s+1 \geq \lfloor \epsilon N/2 \rfloor$ and $v-u+1 \geq N/4$ (or *vice*

versa) have at least this probability. We proceed by showing how we can use such concepts to guarantee success of the algorithm. By the ϵ-net theorem in Section 4.3.2, a uniformly random sample of $\mathcal{O}\left(\frac{1}{\epsilon}\log\frac{1}{\epsilon}+\frac{1}{\epsilon}\log\frac{1}{\delta}\right)$ queries is an $(\epsilon/5-1/20)$-net for $(\mathsf{X}_{\text{ranges}}, \mathbb{C}_{\text{ends}})$ with probability at least $1-\delta$.

Let r_a and r_b be the two records referred to in the statement of Property 1. Recall that their values are in the interval $(N/4, 3N/4)$, and that their difference $|\mathsf{val}(\mathsf{r}_b) - \mathsf{val}(\mathsf{r}_a)|$ is greater than $N/3$. Without loss of generality, suppose that $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b$. We demonstrate in a series of steps that if the set of queried ranges is an $(\epsilon/5-1/20)$-net, then Algorithm 4.3 succeeds except with probability δ.

Step 1. Let r be any record whose value is at least $\lfloor \epsilon N/2 \rfloor$ away from $\mathsf{val}(\mathsf{r}_a)$, $\mathsf{val}(\mathsf{r}_b)$, and the endpoints 1 and N. (Such a record exists by Property 3.) We show that in all orderings $\prec$ compatible with the PQ tree T built in Algorithm 4.3, the relative order of r_a, r_b, and r is correct up to reflection. By Lemma 4.9, one way to show that the three records are correctly ordered is to demonstrate that there have been range queries matching two different subsets of size two of the set $\{\mathsf{r}, \mathsf{r}_a, \mathsf{r}_b\}$. Crucially, these will be queries that belong to concept classes whose probabilities are at least $\epsilon/5 - 1/20$. We consider the following three cases, based on the value of r relative to the values of r_a and r_b.

- Case 1: $\mathsf{r} \overset{\mathsf{val}}{<} \mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b$. Consider the following queries:

  i. left endpoint in $[1, \lfloor \epsilon N/2 \rfloor]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_a), \mathsf{val}(\mathsf{r}_b) - 1]$ (interval of length $\mathsf{val}(\mathsf{r}_b) - \mathsf{val}(\mathsf{r}_a) \geq N/3$ by assumption in Property 1). This query matches r and r_a, but does not match r_b.

  ii. left endpoint in $[\mathsf{val}(\mathsf{r}_a) - \lfloor \epsilon N/2 \rfloor + 1, \mathsf{val}(\mathsf{r}_a)]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_b), N]$ (interval of length $N - \mathsf{val}(\mathsf{r}_b) + 1 \geq N/4$ by assumption in Property 1). This query does not match r, but matches r_a and r_b.

  By Lemma 4.9 on queries i and ii, $\mathsf{r} \prec \mathsf{r}_a \prec \mathsf{r}_b$ for all compatible orderings $\prec$.

- Case 2: $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r} \overset{\mathsf{val}}{<} \mathsf{r}_b$. Consider the following queries:

  iii. left endpoint in $[1, \mathsf{val}(\mathsf{r}_a)]$ (interval of length $\mathsf{val}(\mathsf{r}_a) \geq N/4$ by assumption), and right endpoint in $[\mathsf{val}(\mathsf{r}_b) - \lfloor \epsilon N/2 \rfloor, \mathsf{val}(\mathsf{r}_b) - 1]$ (interval of length $\lfloor \epsilon N/2 \rfloor$). This query matches r_a and r, but does not match r_b.

  iv. left endpoint in $[\mathsf{val}(\mathsf{r}_a) + 1, \mathsf{val}(\mathsf{r}_a) + \lfloor \epsilon N/2 \rfloor]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_b), N]$ (interval of length $N - \mathsf{val}(\mathsf{r}_b) + 1 \geq N/4$ by assumption). This query does not match r_a, but matches r and r_b.

By Lemma 4.9 on queries iii and iv, $\mathsf{r}_a \prec \mathsf{r} \prec \mathsf{r}_b$ for all compatible orderings $\prec$.

- Case 3: $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b \overset{\mathsf{val}}{<} \mathsf{r}$. By reflection symmetry of r_a and r_b, this case is very similar to Case 1.

This concludes Step 1: each record sufficiently far in value from the endpoints and the anchor elements r_a and r_b is in the correct *region* relative to these anchor records. The three possible regions are (i) less than both r_a and r_b, (ii) between r_a and r_b, and (iii) greater than both r_a and r_b. Next, we show that any *two* records sufficiently far from the endpoints, anchor elements, and each other are ordered relatively correctly.

Step 2. Let r and r' be any two records whose values are at least $\lfloor \epsilon N/2 \rfloor$ away from $\mathsf{val}(\mathsf{r}_a)$, $\mathsf{val}(\mathsf{r}_b)$, 1, N, and each other. We show that in all orderings $\prec$ compatible with the PQ tree, the relative order of r_a, r_b, r, and r' is correct up to reflection.

First, if r and r' are not in the same region relative to r_a and r_b, then the result from the previous step is sufficient to guarantee that all four records are ordered correctly. Therefore, we can assume that both r and r' are $\overset{\mathsf{val}}{<} \mathsf{r}_a$, both are between r_a and r_b, or both are $\overset{\mathsf{val}}{>} \mathsf{r}_b$ (for $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b$). Since records r and r' are interchangeable, we consider only the three cases with $\mathsf{r} \overset{\mathsf{val}}{<} \mathsf{r}'$ (again, for $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b$). In this step, we use both Lemma 4.9 and Corollary 4.10. We consider the following three cases, based on which region relative to r_a and r_b the two records r and r' are in.

- Case 1: $\mathsf{r} \overset{\mathsf{val}}{<} \mathsf{r}' \overset{\mathsf{val}}{<} \mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b$. Consider the following queries:

  i. left endpoint in $[1, \lfloor \epsilon N/2 \rfloor]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_a), \mathsf{val}(\mathsf{r}_b) - 1]$ (interval of length $\mathsf{val}(\mathsf{r}_b) - \mathsf{val}(\mathsf{r}_a) \geq N/3$). This query matches $\{\mathsf{r}, \mathsf{r}', \mathsf{r}_a\}$, but does not match r_b.

  ii. left endpoint in $[\mathsf{val}(\mathsf{r}') - \lfloor \epsilon N/2 \rfloor + 1, \mathsf{val}(\mathsf{r}')]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_b), N]$ (interval of length $N - \mathsf{val}(\mathsf{r}_b) + 1 \geq N/4$). This query does not match r, but matches $\{\mathsf{r}', \mathsf{r}_a, \mathsf{r}_b\}$.

  By Corollary 4.10 on queries i and ii, $\mathsf{r} \prec \{\mathsf{r}', \mathsf{r}_a\} \prec \mathsf{r}_b$ for all compatible orderings $\prec$. By Step 1, $\mathsf{r}' \prec \mathsf{r}_a \prec \mathsf{r}_b$ for all compatible orderings, so the four records are correctly ordered.

- Case 2: $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r} \overset{\mathsf{val}}{<} \mathsf{r}' \overset{\mathsf{val}}{<} \mathsf{r}_b$. Consider the following queries:

  iii. left endpoint in $[1, \mathsf{val}(\mathsf{r}_a)]$ (interval of length $\mathsf{val}(\mathsf{r}_a) \geq N/4$), and right endpoint in $[\mathsf{val}(\mathsf{r}), \mathsf{val}(\mathsf{r}) + \lfloor \epsilon N/2 \rfloor - 1]$ (interval of length $\lfloor \epsilon N/2 \rfloor$). This query matches $\{\mathsf{r}_a, \mathsf{r}\}$, but does not match $\{\mathsf{r}', \mathsf{r}_b\}$.

iv. left endpoint in $[\mathsf{val}(\mathsf{r}_a)+1, \mathsf{val}(\mathsf{r}_a)+\lfloor \epsilon N/2 \rfloor]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_b), N]$ (interval of length $N - \mathsf{val}(\mathsf{r}_b) + 1 \geq N/4$). This query does not match r_a, matches $\{\mathsf{r}, \mathsf{r}'\}$, and does not match r_b.

By Lemma 4.9 on queries iii and iv, $\mathsf{r}_a \prec \mathsf{r} \prec \mathsf{r}'$ for all compatible orderings $\prec$. By Step 1, $\mathsf{r}_a \prec \{\mathsf{r}, \mathsf{r}'\} \prec \mathsf{r}_b$ for all compatible orderings, so the four records are correctly ordered.

- Case 3: $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b \overset{\mathsf{val}}{<} \mathsf{r} \overset{\mathsf{val}}{<} \mathsf{r}'$. By reflection symmetry of r_a and r_b, this case is very similar to Case 1.

In all cases, the four records are correctly ordered relative for all compatible orderings.

Step 3. Let r, r', and r'' be any three records whose values are at least ϵN away from each other, 1, and N. In this step, we show that in all orderings $\prec$ compatible with the PQ tree, the relative order of r, r', and r'' is correct up to reflection. Suppose, without loss of generality, that their true order is $\mathsf{r} \overset{\mathsf{val}}{<} \mathsf{r}' \overset{\mathsf{val}}{<} \mathsf{r}''$ (for $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b$). We consider 3 different cases, based on how the records' values relate to the anchor records' values $\mathsf{val}(\mathsf{r}_a)$ and $\mathsf{val}(\mathsf{r}_b)$: they contain both anchor records' values, they are all far from the anchor records' values, or at least one of the three records is close to at least one of the anchor records' values.

- Case 1: two of the records' values are equal to $\mathsf{val}(\mathsf{r}_a)$ and $\mathsf{val}(\mathsf{r}_b)$. Since records with the same value are indistinguishable, by Step 1, the three records are correctly ordered.
- Case 2: all three of the records' values are at least $\lfloor \epsilon N/2 \rfloor$ away from $\mathsf{val}(\mathsf{r}_a)$ and $\mathsf{val}(\mathsf{r}_b)$. Consider applying Step 2 three times, with $\{\mathsf{r}, \mathsf{r}'\}$, with $\{\mathsf{r}, \mathsf{r}''\}$, and with $\{\mathsf{r}', \mathsf{r}''\}$. The ordering of r, r', and r'' relative to each other and r_a and r_b is completely (and correctly) determined up to reflection.
- Case 3: at least one of the records' values is within $\lfloor \epsilon N/2 \rfloor$ of an anchor record's value: let r_{i^*} be this record and let r_j and r_k be the other two records. (One of them may also be close to the other anchor record.) Without loss of generality, we may consider only the case where r_{i^*} is close to the anchor record r_a. (By reflection symmetry, the same arguments can be used to show the records are correctly ordered when r_{i^*} is close to r_b.)

  We subdivide this case into $\binom{3}{2} = 6$ subcases, depending on which regions (relative to r_a and r_b) r_j and r_k are in, e.g., both less than r_a, or one less than r_a and the other between r_a and r_b. For each subcase, we prove that $\{\mathsf{r}_{i^*}, \mathsf{r}_j, \mathsf{r}_k, \} = \{\mathsf{r}, \mathsf{r}', \mathsf{r}''\}$ are always correctly ordered if the set of queries is an $(\epsilon/5 - 1/20)$-net.

Subcase 1. both r_j and r_k are less than r_a. Without loss of generality, suppose $\mathsf{r}_j \overset{\mathsf{val}}{<} \mathsf{r}_k$, so the correct order of the records is $\mathsf{r}_j \overset{\mathsf{val}}{<} \mathsf{r}_k \overset{\mathsf{val}}{<} \mathsf{r}_{i^*}$ (assuming $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b$). Since r_j and r_k are at least ϵN apart from each other and r_{i^*}, which is close to r_a, both r_j and r_k are far enough from r_a for the result from Step 2 to apply: for all orderings compatible with the PQ tree, $\mathsf{r}_j \prec \mathsf{r}_k \prec \mathsf{r}_a \prec \mathsf{r}_b$. It therefore remains to prove that r_{i^*} is ordered correctly relative to $\{\mathsf{r}_j, \mathsf{r}_k\}$.

Consider the following queries:

i. left endpoint in $[\mathsf{val}(\mathsf{r}_k) + 1, \mathsf{val}(\mathsf{r}_k) + \lfloor \epsilon N/2 \rfloor]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_b), N]$ (interval of length at least $N/4$). It does not match $\{\mathsf{r}_j, \mathsf{r}_k\}$, but does match $\{\mathsf{r}_{i^*}, \mathsf{r}_a, \mathsf{r}_b\}$.

ii. left endpoint in $[\mathsf{val}(\mathsf{r}_j) + 1, \mathsf{val}(\mathsf{r}_j) + \lfloor \epsilon N/2 \rfloor]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_a), \mathsf{val}(\mathsf{r}_b) - 1]$ (interval of length at least $N/3$). It does not match r_j, does match $\{\mathsf{r}_k, \mathsf{r}_a\}$, maybe matches r_{i^*}, and does not match r_b. (Since we know only that r_{i^*} and r_a are close, it is possible that $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_{i^*}$ and hence possible that the right endpoint of this query is between $\mathsf{val}(\mathsf{r}_a)$ and $\mathsf{val}(\mathsf{r}_{i^*})$, in which case it does not match r_{i^*}.)

We consider two cases based on whether query ii matched r_{i^*} or not:

– If query ii matches r_{i^*}, so matches $\{\mathsf{r}_k, \mathsf{r}_a, \mathsf{r}_{i^*}\}$, then by Corollary 4.10 on queries ii and i, $\mathsf{r}_k \prec \{\mathsf{r}_a, \mathsf{r}_{i^*}\} \prec \mathsf{r}_b$ for all compatible orderings $\prec$. Since $\mathsf{r}_j \prec \mathsf{r}_k \prec \mathsf{r}_a$ by Step 2, the three records are correctly ordered.

– If query ii does not match r_{i^*}, so matches $\{\mathsf{r}_k, \mathsf{r}_a\}$, then again by Corollary 4.10 on queries ii and i, $\mathsf{r}_k \prec \mathsf{r}_a \prec \{\mathsf{r}_{i^*}, \mathsf{r}_b\}$ for all compatible orderings $\prec$. Since $\mathsf{r}_j \prec \mathsf{r}_k \prec \mathsf{r}_a$ by Step 2, the three records are again correctly ordered.

This concludes the first of six subcases.

Subcase 2. One of $\{\mathsf{r}_j, \mathsf{r}_k\}$ is less than r_a, and the other is between r_a and r_b in value. Without loss of generality, suppose $\mathsf{r}_j \overset{\mathsf{val}}{<} \mathsf{r}_k$, so the correct order of the records up to reflection is $\mathsf{r}_j \overset{\mathsf{val}}{<} \mathsf{r}_{i^*} \overset{\mathsf{val}}{<} \mathsf{r}_k$ (assuming $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b$). In this case, neither r_j nor r_k can be close to r_a (since r_{i^*} is), but r_k may be close to r_b. By Step 1, $\mathsf{r}_j \prec \mathsf{r}_a \prec \mathsf{r}_b$ for all compatible orderings $\prec$.

Consider the following queries:

i. left endpoint in $[\mathsf{val}(\mathsf{r}_j) + 1, \mathsf{val}(\mathsf{r}_j) + \lfloor \epsilon N/2 \rfloor]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_b), N]$ (interval of length at least $N/4$). It does not match r_j, but does match $\{\mathsf{r}_a, \mathsf{r}_{i^*}, \mathsf{r}_k, \mathsf{r}_b\}$.

ii. left endpoint in $[1, \mathsf{val}(\mathsf{r}_a)]$ (interval of length at least $N/4$), and right endpoint in $[\mathsf{val}(\mathsf{r}_k) - \lfloor \epsilon N/2 \rfloor, \mathsf{val}(\mathsf{r}_k) - 1]$ (interval of length $\lfloor \epsilon N/2 \rfloor$). It maybe matches r_j, matches $\{\mathsf{r}_a, \mathsf{r}_{i^*}\}$, and does not match r_k or r_b.

iii. left endpoint in $[\mathsf{val}(\mathsf{r}_k) - \lfloor \epsilon N/2 \rfloor + 1, \mathsf{val}(\mathsf{r}_k)]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_b), N]$ (interval of length at least $N/4$). It does not match $\{\mathsf{r}_j, \mathsf{r}_a, \mathsf{r}_{i^*}\}$, and does match $\{\mathsf{r}_k, \mathsf{r}_b\}$.

Query ii may or may not match r_j. We consider these two cases:

– If query ii matches r_j, so matches $\{\mathsf{r}_j, \mathsf{r}_a, \mathsf{r}_{i^*}\}$, then by Corollary 4.10 on queries ii and i, $\mathsf{r}_j \prec \{\mathsf{r}_a, \mathsf{r}_{i^*}\} \prec \{\mathsf{r}_k, \mathsf{r}_b\}$ for all compatible orderings $\prec$, so the records are correctly ordered.

– If query ii does not match r_j, so matches $\{\mathsf{r}_a, \mathsf{r}_{i^*}\}$, by Corollary 4.12 on queries ii and iii, we can "lift" $\mathsf{r}_j \prec \mathsf{r}_a \prec \mathsf{r}_b$ to $\mathsf{r}_j \prec \{\mathsf{r}_a, \mathsf{r}_{i^*}\} \prec \{\mathsf{r}_k, \mathsf{r}_b\}$ for all compatible orderings $\prec$, so the three records are correctly ordered.

This concludes the second of six subcases.

Subcase 3. One of $\{\mathsf{r}_j, \mathsf{r}_k\}$ is less than r_a, and the other is greater than r_b. Without loss of generality, suppose $\mathsf{r}_j \overset{\mathsf{val}}{<} \mathsf{r}_k$ (for $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b$), so the correct order of the records up to reflection is $\mathsf{r}_j \overset{\mathsf{val}}{<} \mathsf{r}_{i^*} \overset{\mathsf{val}}{<} \mathsf{r}_k$ (again, assuming $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b$). Then, r_j cannot be close to r_a (since r_{i^*} is), but r_k may be close to r_b. By Step 1, $\mathsf{r}_j \prec \mathsf{r}_a \prec \mathsf{r}_b$ for all compatible orderings $\prec$. In this subcase, we divide the analysis by which side of the anchor record r_a the record r_{i^*} is on.

First, suppose $\mathsf{r}_{i^*} \overset{\mathsf{val}}{<} \mathsf{r}_a$. Consider the following two queries:

i. left endpoint in $[\mathsf{val}(\mathsf{r}_{i^*}) - \lfloor \epsilon N/2 \rfloor, \mathsf{val}(\mathsf{r}_{i^*} - 1]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_a), \mathsf{val}(\mathsf{r}_b) - 1]$ (interval of length at least $N/3$). It matches only $\{\mathsf{r}_{i^*}, \mathsf{r}_a\}$, and none of the other records.

ii. left endpoint in $[\mathsf{val}(\mathsf{r}_a) + 1, \mathsf{val}(\mathsf{r}_b)]$ (interval of length at least $N/3$), and right endpoint in $[\mathsf{val}(\mathsf{r}_k), \mathsf{val}(\mathsf{r}_k) + \lfloor \epsilon N/2 \rfloor - 1]$ (interval of length $\lfloor \epsilon N/2 \rfloor$). It matches only $\{\mathsf{r}_b, \mathsf{r}_k\}$, and none of the other records.

By Corollary 4.12 on these two queries, we may then "lift" $\mathsf{r}_j \prec \mathsf{r}_a \prec \mathsf{r}_b$ to $\mathsf{r}_j \prec \{\mathsf{r}_{i^*}, \mathsf{r}_a\} \prec \{\mathsf{r}_b, \mathsf{r}_k\}$, thus the three records are ordered correctly by all compatible orderings.

Otherwise, $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_{i^*}$. In this case, the true ordering is $\mathsf{r}_j \overset{\mathsf{val}}{<} \mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_{i^*} \overset{\mathsf{val}}{<} \mathsf{r}_b \overset{\mathsf{val}}{<} \mathsf{r}_k$. No query with one endpoint in an interval of length $\lfloor \epsilon N/2 \rfloor$ and the other endpoint in an interval of length at least $N/4$ is guaranteed to always contain a particular

subset of the five records, so we consider more complex cases. Our case-by-case analysis involves the following three queries:

iii. left endpoint in $[1, \mathsf{val}(\mathsf{r}_a)]$ (interval of length at least $N/4$), and right endpoint in $[\mathsf{val}(\mathsf{r}_b) - \lfloor \epsilon N/2 \rfloor, \mathsf{val}(\mathsf{r}_b) - 1]$. It may match r_j, matches $\{\mathsf{r}_a, \mathsf{r}_{i^*}\}$, and does not match $\{\mathsf{r}_b, \mathsf{r}_k\}$.

iv. left endpoint in $[\mathsf{val}(\mathsf{r}_a) - \lfloor \epsilon N/2 \rfloor + 1, \mathsf{val}(\mathsf{r}_a)]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_b), N]$ (interval of length at least $N/4$). It does not match r_j, matches $\{\mathsf{r}_a, \mathsf{r}_{i^*}, \mathsf{r}_b\}$, and may match r_k.

v. left endpoint in $[\mathsf{val}(\mathsf{r}_a) + 1, \mathsf{val}(\mathsf{r}_b)]$ (interval of length at least $N/3$), and right endpoint in $[\mathsf{val}(\mathsf{r}_k), \mathsf{val}(\mathsf{r}_k) + \lfloor \epsilon N/2 \rfloor - 1]$ (interval of length $\lfloor \epsilon N/2 \rfloor$). It does not match $\{\mathsf{r}_j, \mathsf{r}_a\}$, may match r_{i^*}, and matches $\{\mathsf{r}_b, \mathsf{r}_k\}$.

We divide the analysis for part two ($\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_{i^*}$) into four cases based on whether query iii matches r_j, and query iv matches r_k or query v matches r_{i^*}:

- If query iii matches r_j, so matches $\{\mathsf{r}_j, \mathsf{r}_a, \mathsf{r}_{i^*}\}$, and query iv matches r_k, so matches $\{\mathsf{r}_a, \mathsf{r}_{i^*}, \mathsf{r}_b, \mathsf{r}_k\}$, then by Corollary 4.10 on queries iii and iv, $\mathsf{r}_j \prec \{\mathsf{r}_a, \mathsf{r}_{i^*}\} \prec \{\mathsf{r}_b, \mathsf{r}_k\}$ for all compatible orderings, as required.
- If query iii matches r_j, so matches $\{\mathsf{r}_j, \mathsf{r}_a, \mathsf{r}_{i^*}\}$, and query iv does not match r_k, so matches $\{\mathsf{r}_a, \mathsf{r}_{i^*}, \mathsf{r}_b\}$, then by Corollary 4.10 on queries iii and iv, $\mathsf{r}_j \prec \{\mathsf{r}_a, \mathsf{r}_{i^*}\} \prec \mathsf{r}_b$. It therefore remains to show that r_k is correctly ordered relative to the other two records.

  If query v matches r_{i^*}, so matches $\{\mathsf{r}_{i^*}, \mathsf{r}_b, \mathsf{r}_k\}$, then by Corollary 4.10 on queries iii and v, $\{\mathsf{r}_j, \mathsf{r}_a\} \prec \mathsf{r}_{i^*} \prec \{\mathsf{r}_b, \mathsf{r}_k\}$ for all compatible orderings.

  If query v does not match r_{i^*}, so matches $\{\mathsf{r}_b, \mathsf{r}_k\}$, then by Corollary 4.10 on queries iv and v, $\{\mathsf{r}_a, \mathsf{r}_{i^*}\} \prec \mathsf{r}_b \prec \mathsf{r}_k$, as required to show that r_k is correctly ordered relative to the other two records. Thus the three records are correctly ordered in all compatible orderings.
- If query iii does not match r_j, so matches $\{\mathsf{r}_a, \mathsf{r}_{i^*}\}$, and query v matches r_{i^*}, so matches $\{\mathsf{r}_{i^*}, \mathsf{r}_b, \mathsf{r}_k\}$, then by Corollary 4.10 on queries iii and v, $\mathsf{r}_a \prec \mathsf{r}_{i^*} \prec \{\mathsf{r}_b, \mathsf{r}_k\}$ for all compatible orderings. Recalling that by Step 1, $\mathsf{r}_j \prec \mathsf{r}_a \prec \mathsf{r}_b$, we conclude that the three records are correctly ordered.
- If query iii does not match r_j, so matches $\{\mathsf{r}_a, \mathsf{r}_{i^*}\}$, and query v does not match r_{i^*}, so matches $\{\mathsf{r}_b, \mathsf{r}_k\}$, then we may apply Corollary 4.12 to queries iii and v to "lift" $\mathsf{r}_j \prec \mathsf{r}_a \prec \mathsf{r}_b$ to $\mathsf{r}_j \prec \{\mathsf{r}_a, \mathsf{r}_{i^*}\} \prec \{\mathsf{r}_b, \mathsf{r}_k\}$, thus the three records are correctly ordered.

This concludes the third of six subcases.

Subcase 4. Both r_j and r_k are between the anchor records r_a and r_b. Without loss of generality, suppose $\mathsf{r}_j \overset{\mathsf{val}}{<} \mathsf{r}_k$ (for $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b$), so the correct order of the records up to reflection is $\mathsf{r}_{i^*} \overset{\mathsf{val}}{<} \mathsf{r}_j \overset{\mathsf{val}}{<} \mathsf{r}_k$ (again, for $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b$). By Step 1, $\mathsf{r}_a \prec \mathsf{r}_j \prec \mathsf{r}_b$ for all compatible orderings $\prec$, since r_j cannot be close to r_a.

Consider the following three queries:

i. left endpoint in $[\mathsf{val}(\mathsf{r}_j) - \lfloor \epsilon N/2 \rfloor + 1, \mathsf{val}(\mathsf{r}_j)]$ (interval of length $\lfloor \epsilon N/2 \rfloor$) and right endpoint in $[\mathsf{val}(\mathsf{r}_b), N]$ (interval of length at least $N/4$). It does not match r_a or r_{i^*}, and does match $\{\mathsf{r}_j, \mathsf{r}_k, \mathsf{r}_b\}$.

ii. left endpoint in $[1, \mathsf{val}(\mathsf{r}_a)]$ (interval of length at least $N/4$), and right endpoint in $[\mathsf{val}(\mathsf{r}_j), \mathsf{val}(\mathsf{r}_j) + \lfloor \epsilon N/2 \rfloor - 1]$ (interval of length $\lfloor \epsilon N/2 \rfloor$). It maybe matches r_{i^*}, matches $\{\mathsf{r}_a, \mathsf{r}_j\}$, and does not match r_b.

iii. left endpoint in $[1, \lfloor \epsilon N/2 \rfloor]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_a), \mathsf{val}(\mathsf{r}_b) - 1]$ (interval of length at least $N/3$). It matches $\{\mathsf{r}_a, \mathsf{r}_{i^*}\}$, maybe matches r_j or r_k, and does not match r_b.

If query ii does match r_{i^*}, so matches $\{\mathsf{r}_{i^*}, \mathsf{r}_a, \mathsf{r}_j\}$, then by Corollary 4.10 on queries ii and i, $\{\mathsf{r}_{i^*}, \mathsf{r}_a\} \prec \mathsf{r}_j \prec \{\mathsf{r}_k, \mathsf{r}_b\}$ for all compatible orderings $\prec$, so the records are correctly ordered.

If query ii does not match r_{i^*}, so matches just $\{\mathsf{r}_a, \mathsf{r}_j\}$, then it must be the case that r_{i^*} is less than r_a (assuming $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b$). By Corollary 4.10 on queries ii and i, $\mathsf{r}_a \prec \mathsf{r}_j \prec \{\mathsf{r}_k, \mathsf{r}_b\}$ for all compatible orderings $\prec$, so it remains to prove that r_{i^*} is correctly ordered relative to r_j and r_k. We now consider the three possibilities for query iii, based on which of the "maybe" elements it matched.

- If query iii matches neither r_j nor r_k, only $\{\mathsf{r}_{i^*}, \mathsf{r}_a\}$, then by Lemma 4.9 on queries iii and ii, $\mathsf{r}_{i^*} \prec \mathsf{r}_a \prec \mathsf{r}_j$ for all compatible orderings $\prec$, so r_{i^*}'s relative order is correct, as required to prove that the records are correctly ordered.
- If query iii matches r_j but not r_k, so matches $\{\mathsf{r}_{i^*}, \mathsf{r}_a, \mathsf{r}_j\}$, then by Corollary 4.10 on queries iii and i, $\{\mathsf{r}_{i^*}, \mathsf{r}_a\} \prec \mathsf{r}_j \prec \{\mathsf{r}_k, \mathsf{r}_b\}$ for all compatible orderings $\prec$, so the records are correctly ordered.
- Lastly, if query iii matches both r_j and r_k, so $\{\mathsf{r}_{i^*}, \mathsf{r}_a, \mathsf{r}_j, \mathsf{r}_k\}$, then by Corollary 4.10 on queries iii and i, $\{\mathsf{r}_{i^*}, \mathsf{r}_a\} \prec \{\mathsf{r}_j, \mathsf{r}_k\} \prec \mathsf{r}_b$ for all compatible orderings. Thus, r_{i^*} is correctly ordered with respect to r_j and r_k, as required, so the three records are correctly ordered.

This concludes the fourth of six subcases.

Subcase 5. One of $\{\mathsf{r}_j, \mathsf{r}_k\}$ is between the anchor records r_a and r_b, and the other is greater than r_b. Without loss of generality, suppose r_j is the record between r_a and r_b, so the correct order of the records up to reflection is $\mathsf{r}_{i^*} \overset{\mathsf{val}}{<} \mathsf{r}_j \overset{\mathsf{val}}{<} \mathsf{r}_k$ (for $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b$). It is possible that one of $\{\mathsf{r}_j, \mathsf{r}_k\}$ is close to r_b, but not both, since they are at least ϵN apart. We therefore separate further into the following four cases based on which side of the anchor record r_a the record r_{i^*} is on, and which one of $\{\mathsf{r}_j, \mathsf{r}_k\}$ is not close to r_b.

– $\mathsf{r}_a \overset{\mathsf{val}}{\leq} \mathsf{r}_{i^*} \overset{\mathsf{val}}{<} \mathsf{r}_b$, and r_j is not close to r_b. In this case, by Step 1, we have that $\mathsf{r}_a \prec \mathsf{r}_j \prec \mathsf{r}_b$ for all compatible orderings. Consider the following queries:

  i. left endpoint in $[1, \mathsf{val}(\mathsf{r}_a)]$ (interval of length at least $N/4$), and right endpoint in $[\mathsf{val}(\mathsf{r}_j), \mathsf{val}(\mathsf{r}_j) + \lfloor \epsilon N/2 \rfloor - 1]$ (interval of length $\lfloor \epsilon N/2 \rfloor$). It matches $\{\mathsf{r}_a, \mathsf{r}_{i^*}, \mathsf{r}_j\}$, and does not match r_b or r_k.

  ii. left endpoint in $[\mathsf{val}(\mathsf{r}_j) - \lfloor \epsilon N/2 \rfloor + 1, \mathsf{val}(\mathsf{r}_j)]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_b), N]$ (interval of length at least $N/4$). It does not match either r_a or r_{i^*}, matches $\{\mathsf{r}_j, \mathsf{r}_b\}$, and maybe matches r_k.

  iii. left endpoint in $[\mathsf{val}(\mathsf{r}_a) + 1, \mathsf{val}(\mathsf{r}_b)]$ (interval of length at least $N/3$), and right endpoint in $[\mathsf{val}(\mathsf{r}_k), \mathsf{val}(\mathsf{r}_k) + \lfloor \epsilon N/2 \rfloor - 1]$ (interval of length $\lfloor \epsilon N/2 \rfloor$). It does not match r_a, maybe matches r_{i^*} or r_j, and matches $\{\mathsf{r}_b, \mathsf{r}_k\}$.

  iv. left endpoint in $[1, \mathsf{val}(\mathsf{r}_a)]$ (interval of length at least $N/4$), and right endpoint in $[\mathsf{val}(\mathsf{r}_{i^*}, \mathsf{val}(\mathsf{r}_{i^*}) + \lfloor \epsilon N/2 \rfloor - 1]$ (interval of length $\lfloor \epsilon N/2 \rfloor$). It matches $\{\mathsf{r}_a, \mathsf{r}_{i^*}\}$, and none of $\{\mathsf{r}_j, \mathsf{r}_b, \mathsf{r}_k\}$.

  If query ii does match r_k, so matches $\{\mathsf{r}_j, \mathsf{r}_b, \mathsf{r}_k\}$, then by Corollary 4.10 on queries i and ii, $\{\mathsf{r}_a, \mathsf{r}_{i^*}\} \prec \mathsf{r}_j \prec \{\mathsf{r}_b, \mathsf{r}_k\}$ for all compatible orderings $\prec$, so the records are correctly ordered.

  If query ii does not match r_k, so matches only $\{\mathsf{r}_j, \mathsf{r}_b\}$, then by Corollary 4.10 on queries i and ii, $\{\mathsf{r}_a, \mathsf{r}_{i^*}\} \prec \mathsf{r}_j \prec \mathsf{r}_b$ for all compatible orderings $\prec$. It therefore remains to prove that r_k is correctly ordered with respect to r_{i^*} and r_j. Consider now the three possibilities for query iii, based on which of the "maybe" elements it matched.

  * If query iii matches both r_{i^*} and r_j, so matches $\{\mathsf{r}_{i^*}, \mathsf{r}_j, \mathsf{r}_b, \mathsf{r}_k\}$, then by Corollary 4.10 on queries i and iii, $\mathsf{r}_a \prec \{\mathsf{r}_{i^*}, \mathsf{r}_j\} \prec \{\mathsf{r}_b, \mathsf{r}_k\}$ for all compatible orderings $\prec$. Thus, r_k is correctly ordered with respect to r_{i^*} and r_j, and the three records are correctly ordered.

* If query iii matches just r_j, so matches $\{\mathsf{r}_j, \mathsf{r}_b, \mathsf{r}_k\}$, then by Corollary 4.10 on queries i and iii, $\{\mathsf{r}_a, \mathsf{r}_{i^*}\} \prec \mathsf{r}_j \prec \{\mathsf{r}_b, \mathsf{r}_k\}$ for all compatible orderings $\prec$, so the records are correctly ordered.
* If query iii matches neither r_{i^*} nor r_j, so matches $\{\mathsf{r}_b, \mathsf{r}_k\}$, then we may apply Corollary 4.12 on queries iii and iv to "lift" $\mathsf{r}_a \prec \mathsf{r}_j \prec \mathsf{r}_b$ to $\{\mathsf{r}_a, \mathsf{r}_{i^*}\} \prec \mathsf{r}_j \prec \{\mathsf{r}_b, \mathsf{r}_k\}$ for all compatible orderings $\prec$. Thus, the three records are ordered correctly.

– $\mathsf{r}_a \overset{\mathsf{val}}{\leq} \mathsf{r}_{i^*} \overset{\mathsf{val}}{<} \mathsf{r}_b$, and r_k is not close to r_b. In this case, by Step 1, we have that $\mathsf{r}_a \prec \mathsf{r}_b \prec \mathsf{r}_k$ for all compatible orderings. Consider the following queries:

v. left endpoint in $[1, \mathsf{val}(\mathsf{r}_a)]$ (interval of length at least $N/4$), and right endpoint in $[\mathsf{val}(\mathsf{r}_b), \mathsf{val}(\mathsf{r}_b) + \lfloor \epsilon N/2 \rfloor - 1]$ (interval of length $\lfloor \epsilon N/2 \rfloor$). It matches $\{\mathsf{r}_a, \mathsf{r}_{i^*}, \mathsf{r}_j, \mathsf{r}_b\}$ and does not match r_k.

vi. left endpoint in $[\mathsf{val}(\mathsf{r}_{i^*}) - \lfloor \epsilon N/2 \rfloor + 1, \mathsf{val}(\mathsf{r}_{i^*})]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_b), N]$ (interval of length at least $N/4$). It does not match r_a nor r_{i^*}, does match $\{\mathsf{r}_j, \mathsf{r}_b\}$, and maybe matches r_k.

vii. left endpoint in $[1, \mathsf{val}(\mathsf{r}_a)]$ (interval of length at least $N/4$), and right endpoint in $[\mathsf{val}(\mathsf{r}_{i^*}, \mathsf{val}(\mathsf{r}_{i^*}) + \lfloor \epsilon N/2 \rfloor - 1]$ (interval of length $\lfloor \epsilon N/2 \rfloor$). It matches $\{\mathsf{r}_a, \mathsf{r}_{i^*}\}$, and none of $\{\mathsf{r}_j, \mathsf{r}_b, \mathsf{r}_k\}$.

If query vi matches r_k, so matches $\{\mathsf{r}_j, \mathsf{r}_b, \mathsf{r}_k\}$, then by Corollary 4.10 on queries v and vi, $\{\mathsf{r}_a, \mathsf{r}_{i^*}\} \prec \{\mathsf{r}_j, \mathsf{r}_b\} \prec \mathsf{r}_k$ for all compatible orderings, so the records are correctly ordered.

If query vi does not match r_k, so matches $\{\mathsf{r}_j, \mathsf{r}_b\}$, then by Corollary 4.12 on queries vi and vii, we may "lift" $\mathsf{r}_a \prec \mathsf{r}_b \prec \mathsf{r}_k$ to $\{\mathsf{r}_a, \mathsf{r}_{i^*}\} \prec \{\mathsf{r}_j, \mathsf{r}_b\} \prec \mathsf{r}_k$ for all compatible orderings, so the records are correctly ordered.

– $\mathsf{r}_{i^*} \overset{\mathsf{val}}{<} \mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b$, and r_j is not close to r_b. In this case, by Step 1, we have that $\mathsf{r}_a \prec \mathsf{r}_j \prec \mathsf{r}_b$ for all compatible orderings. Consider the following queries:

viii. left endpoint in $[\mathsf{val}(\mathsf{r}_a) + 1, \mathsf{val}(\mathsf{r}_b)]$ (interval of length at least $N/3$), and right endpoint in $[\mathsf{val}(\mathsf{r}_j), \mathsf{val}(\mathsf{r}_j) + \lfloor \epsilon N/2 \rfloor - 1]$ (interval of length $\lfloor \epsilon N/2 \rfloor$). It does not match r_{i^*} nor r_a, maybe matches r_j, and matches $\{\mathsf{r}_b, \mathsf{r}_k\}$.

ix. left endpoint in $[\mathsf{val}(\mathsf{r}_{i^*}) - \lfloor \epsilon N/2 \rfloor + 1, \mathsf{val}(\mathsf{r}_{i^*})]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_a), \mathsf{val}(\mathsf{r}_b) - 1]$ (interval of length at least $N/3$). It matches $\{\mathsf{r}_{i^*}, \mathsf{r}_a\}$, maybe matches r_j, and does not match r_b nor r_k.

x. left endpoint in $[\mathsf{val}(\mathsf{r}_j) - \lfloor \epsilon N/2 \rfloor + 1, \mathsf{val}(\mathsf{r}_j)]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_b), N]$ (interval of length at least $N/4$). It does not match $\{\mathsf{r}_{i^*}, \mathsf{r}_a\}$, matches $\{\mathsf{r}_j, \mathsf{r}_b\}$, and maybe matches r_k.

xi. left endpoint in $[1, \mathsf{val}(\mathsf{r}_a)]$ (interval of length at least $N/4$), and right endpoint in $[\mathsf{val}(\mathsf{r}_j), \mathsf{val}(\mathsf{r}_j) + \lfloor \epsilon N/2 \rfloor - 1]$ (interval of length $\lfloor \epsilon N/2 \rfloor$). It maybe matches r_{i^*}, matches $\{\mathsf{r}_a, \mathsf{r}_j\}$, and does not match $\{\mathsf{r}_b, \mathsf{r}_k\}$.

We again consider different cases based on which "maybe" elements the queries match.

* If query viii matches r_j, so matches $\{\mathsf{r}_j, \mathsf{r}_b, \mathsf{r}_k\}$, and query ix matches r_j, so matches $\{\mathsf{r}_{i^*}, \mathsf{r}_a, \mathsf{r}_j\}$, then by Corollary 4.10, $\{\mathsf{r}_{i^*}, \mathsf{r}_a\} \prec \mathsf{r}_j \prec \{\mathsf{r}_b, \mathsf{r}_k\}$ for all compatible orderings.
* If query viii does not match r_j, so matches $\{\mathsf{r}_b, \mathsf{r}_k\}$, and query ix does not match r_j, so matches $\{\mathsf{r}_{i^*}, \mathsf{r}_a\}$, then by Corollary 4.12, we may "lift" $\mathsf{r}_a \prec \mathsf{r}_j \prec \mathsf{r}_b$ to $\{\mathsf{r}_{i^*}, \mathsf{r}_a\} \prec \mathsf{r}_j \prec \{\mathsf{r}_b, \mathsf{r}_k\}$ for all compatible orderings.
* Suppose query viii does not match r_j, so matches $\{\mathsf{r}_b, \mathsf{r}_k\}$, and query ix matches r_j, so matches $\{\mathsf{r}_{i^*}, \mathsf{r}_a, \mathsf{r}_j\}$.

  If query x matches r_k, so matches $\{\mathsf{r}_j, \mathsf{r}_b, \mathsf{r}_k\}$, then by Corollary 4.10 on queries ix and x, $\{\mathsf{r}_{i^*}, \mathsf{r}_a\} \prec \mathsf{r}_j \prec \{\mathsf{r}_b, \mathsf{r}_k\}$ for all compatible orderings.

  If query x does not match r_k, so matches $\{\mathsf{r}_j, \mathsf{r}_b\}$, then by Corollary 4.10 on queries ix and x, $\{\mathsf{r}_{i^*}, \mathsf{r}_a\} \prec \mathsf{r}_j \prec \mathsf{r}_b$, so r_{i^*} and r_j are correctly ordered relative to r_b. By Lemma 4.9 on queries xiii and x, $\mathsf{r}_j \prec \mathsf{r}_b \prec \mathsf{r}_k$, so we conclude that the three records are correctly ordered for all compatible orderings.
* Suppose query xiii matches r_j, so matches $\{\mathsf{r}_j, \mathsf{r}_b, \mathsf{r}_k\}$, and query ix does not match r_j, so matches $\{\mathsf{r}_{i^*}, \mathsf{r}_a\}$.

  If query xi matches r_{i^*}, so matches $\{\mathsf{r}_{i^*}, \mathsf{r}_a, \mathsf{r}_j\}$, then by Corollary 4.10 on queries xi and xiii, $\{\mathsf{r}_{i^*}, \mathsf{r}_a\} \prec \mathsf{r}_j \prec \{\mathsf{r}_b, \mathsf{r}_k\}$ for all compatible orderings.

  If query xi does not match r_{i^*}, so matches $\{\mathsf{r}_a, \mathsf{r}_j\}$, then by Lemma 4.9 on queries ix and xi, $\mathsf{r}_{i^*} \prec \mathsf{r}_a \prec \mathsf{r}_j$, so r_{i^*} and r_j are correctly ordered relative to r_a. By Corollary 4.10 on queries xi and xiii, $\mathsf{r}_a \prec \mathsf{r}_j \prec \{\mathsf{r}_b, \mathsf{r}_k\}$, so we conclude that the three records are correctly ordered for all compatible orderings.

– $\mathsf{r}_{i^*} \overset{\mathsf{val}}{<} \mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b$, and r_k is not close to r_b. In this case, by Step 1, we have that $\mathsf{r}_a \prec \mathsf{r}_b \prec \mathsf{r}_k$ for all compatible orderings. Consider the following queries:

xii. left endpoint in $[\mathsf{val}(\mathsf{r}_j) - \lfloor \epsilon N/2 \rfloor + 1, \mathsf{val}(\mathsf{r}_j)]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_b), N]$ (interval of length at least $N/4$). It does not match r_{i^*} nor r_a, matches $\{\mathsf{r}_j, \mathsf{r}_b\}$, and maybe matches r_k.

xiii. left endpoint in $[\mathsf{val}(\mathsf{r}_{i^*}) - \lfloor \epsilon N/2 \rfloor + 1, \mathsf{val}(\mathsf{r}_{i^*}]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_a), \mathsf{val}(\mathsf{r}_b) - 1]$ (interval of length at least

$N/3$). It matches $\{\mathsf{r}_{i^*}, \mathsf{r}_a\}$, maybe matches r_j, and does not match $\{\mathsf{r}_b, \mathsf{r}_k\}$.

xiv. left endpoint in $[1, \mathsf{val}(\mathsf{r}_a)]$ (interval of length at least $N/4$), and right endpoint in $[\mathsf{val}(\mathsf{r}_b), \mathsf{val}(\mathsf{r}_b) + \lfloor \epsilon N/2 \rfloor - 1]$ (interval of length $\lfloor \epsilon N/2 \rfloor$). It maybe matches r_{i^*}, matches $\{\mathsf{r}_a, \mathsf{r}_j, \mathsf{r}_b\}$, and does not match r_k.

We consider four different cases based on which "maybe" elements queries xii and xiii match.

* If query xii matches r_k, so matches $\{\mathsf{r}_j, \mathsf{r}_b, \mathsf{r}_k\}$, and query xiii matches r_j, so matches $\{\mathsf{r}_{i^*}, \mathsf{r}_a, \mathsf{r}_j\}$, then by Corollary 4.10, $\{\mathsf{r}_{i^*}, \mathsf{r}_a\} \prec \mathsf{r}_j \prec \{\mathsf{r}_b, \mathsf{r}_k\}$ for all compatible orderings.
* If query xii does not match r_k, so matches $\{\mathsf{r}_j, \mathsf{r}_b\}$, and query xiii does not match r_j, so matches $\{\mathsf{r}_{i^*}, \mathsf{r}_a\}$, then by Corollary 4.12, we may "lift" $\mathsf{r}_a \prec \mathsf{r}_b \prec \mathsf{r}_k$ to $\{\mathsf{r}_{i^*}, \mathsf{r}_a\} \prec \{\mathsf{r}_j, \mathsf{r}_b\} \prec \mathsf{r}_k$ for all compatible orderings.
* If query xii does not match r_k, so matches $\{\mathsf{r}_j, \mathsf{r}_b\}$, and query xiii matches r_j, so matches $\{\mathsf{r}_{i^*}, \mathsf{r}_a, \mathsf{r}_j\}$, then by Corollary 4.10, $\{\mathsf{r}_{i^*}, \mathsf{r}_a\} \prec \mathsf{r}_j \prec \mathsf{r}_b$ for all compatible orderings. Since $\mathsf{r}_a \prec \mathsf{r}_b \prec \mathsf{r}_k$ for all compatible orderings from Step 1, the records are correctly ordered.
* Suppose query xii matches r_k, so matches $\{\mathsf{r}_j, \mathsf{r}_b, \mathsf{r}_k\}$, and query xiii does not match r_j, so matches $\{\mathsf{r}_{i^*}, \mathsf{r}_a\}$. If query xiv matches r_{i^*}, so matches $\{\mathsf{r}_{i^*}, \mathsf{r}_a, \mathsf{r}_j, \mathsf{r}_b\}$, then by Corollary 4.10 on queries xii and xiv, $\{\mathsf{r}_{i^*}, \mathsf{r}_a\} \prec \{\mathsf{r}_j, \mathsf{r}_b\} \prec \mathsf{r}_k$ for all compatible orderings. If query xiv does not match r_{i^*}, so matches $\{\mathsf{r}_a, \mathsf{r}_j, \mathsf{r}_b\}$, then by Corollary 4.10 on queries xiii and xiv, $\mathsf{r}_{i^*} \prec \mathsf{r}_a \prec \{\mathsf{r}_j, \mathsf{r}_b\}$ for all compatible orderings. Since $\mathsf{r}_a \prec \mathsf{r}_b \prec \mathsf{r}_k$ for all compatible orderings from Step 1, the records are correctly ordered.

This concludes the fifth of six subcases.

Subcase 6. Both r_j and r_k are greater than r_b. Without loss of generality, suppose $\mathsf{r}_j \overset{\mathsf{val}}{<} \mathsf{r}_k$, so the correct order of the records up to reflection is $\mathsf{r}_{i^*} \overset{\mathsf{val}}{<} \mathsf{r}_j \overset{\mathsf{val}}{<} \mathsf{r}_k$ (for $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_b$). In this case, r_j may be close to r_b, but r_k cannot. From Step 1, $\mathsf{r}_a \prec \mathsf{r}_b \prec \mathsf{r}_k$ for all compatible orderings $\prec$. We consider two cases based on which side of the anchor record r_a the record r_{i^*} is on.

First, suppose $\mathsf{r}_{i^*} \overset{\mathsf{val}}{<} \mathsf{r}_a$. Consider the following two queries:

i. left endpoint in $[\mathsf{val}(\mathsf{r}_{i^*}) - \lfloor \epsilon N/2 \rfloor + 1, \mathsf{val}(\mathsf{r}_{i^*})]$ (interval of length $\lfloor \epsilon N/2 \rfloor$), and right endpoint in $[\mathsf{val}(\mathsf{r}_a), \mathsf{val}(\mathsf{r}_b) - 1]$ (interval of length at least $N/3$). It matches $\{\mathsf{r}_{i^*}, \mathsf{r}_a\}$, and none of $\{\mathsf{r}_b, \mathsf{r}_j, \mathsf{r}_k\}$.

ii. left endpoint in $[\mathsf{val}(\mathsf{r}_a)+1, \mathsf{val}(\mathsf{r}_b)]$ (interval of length at least $N/3$), and right endpoint in $[\mathsf{val}(\mathsf{r}_j), \mathsf{val}(\mathsf{r}_j)+\lfloor \epsilon N/2 \rfloor]$ (interval of length $\lfloor \epsilon N/2 \rfloor$). It does not match $\{\mathsf{r}_{i^*}, \mathsf{r}_a\}$, matches $\{\mathsf{r}_b, \mathsf{r}_j\}$, and does not match r_k.

Using Corollary 4.12 with queries i and ii, we may then "lift" $\mathsf{r}_a \prec \mathsf{r}_b \prec \mathsf{r}_k$ to $\{\mathsf{r}_{i^*}, \mathsf{r}_a\} \prec \{\mathsf{r}_b, \mathsf{r}_j\} \prec \mathsf{r}_k$, thus the three records are ordered correctly by all compatible orderings.

Otherwise, $\mathsf{r}_a \overset{\mathsf{val}}{<} \mathsf{r}_{i^*}$. Consider the following three queries:

iii. left endpoint in $[1, \mathsf{val}(\mathsf{r}_a)]$ (interval of length at least $N/4$), and right endpoint in $[\mathsf{val}(\mathsf{r}_j), \mathsf{val}(\mathsf{r}_j)+\lfloor \epsilon N/2 \rfloor - 1]$ (interval of length $\lfloor \epsilon N/2 \rfloor$). It matches $\{\mathsf{r}_a, \mathsf{r}_{i^*}, \mathsf{r}_b, \mathsf{r}_j\}$ and does not match r_k.

iv. left endpoint in $[\mathsf{val}(\mathsf{r}_a)+1, \mathsf{val}(\mathsf{r}_b)]$ (interval of length at least $N/3$), and right endpoint in $[\mathsf{val}(\mathsf{r}_k), \mathsf{val}(\mathsf{r}_k)+\lfloor \epsilon N/2 \rfloor - 1]$ (interval of length $\lfloor \epsilon N/2 \rfloor$). It does not match r_a, maybe matches r_{i^*}, and matches $\{\mathsf{r}_b, \mathsf{r}_j, \mathsf{r}_k\}$.

v. left endpoint in $[1, \mathsf{val}(\mathsf{r}_a)]$ (interval of length at least $N/4$), and right endpoint in $[\mathsf{val}(\mathsf{r}_{i^*}), \mathsf{val}(\mathsf{r}_{i^*})+\lfloor \epsilon N/2 \rfloor - 1]$ (interval of length $\lfloor \epsilon N/2 \rfloor$). It matches $\{\mathsf{r}_a, \mathsf{r}_{i^*}\}$ and does not match $\{\mathsf{r}_b, \mathsf{r}_j, \mathsf{r}_k\}$.

If query iv matches r_{i^*}, so matches $\{\mathsf{r}_{i^*}, \mathsf{r}_b, \mathsf{r}_j, \mathsf{r}_k\}$, then by Corollary 4.10 on queries iii and iv, $\mathsf{r}_a \prec \{\mathsf{r}_{i^*}, \mathsf{r}_b, \mathsf{r}_j\} \prec \mathsf{r}_k$ for all compatible orderings. It therefore remains to prove that r_{i^*} and r_j are ordered correctly with respect to r_k. By Corollary 4.10 on queries v and iv, $\mathsf{r}_a \prec \mathsf{r}_{i^*} \prec \{\mathsf{r}_b, \mathsf{r}_j, \mathsf{r}_k\}$, as required, so the three records are ordered correctly.

If query iv does not match r_{i^*}, so matches $\{\mathsf{r}_b, \mathsf{r}_j, \mathsf{r}_k\}$, then by Corollary 4.10 on queries iii and iv, $\{\mathsf{r}_a, \mathsf{r}_{i^*}\} \prec \{\mathsf{r}_b, \mathsf{r}_j\} \prec \mathsf{r}_k$ for all compatible orderings, so the records are correctly ordered. This concludes the last of six subcases.

These six subcases assumed that r_a and r_{i^*} were close (i.e., their values were within $\lfloor \epsilon N/2 \rfloor$), and considered all possibilities for the positions of the other two records of interest, r_j and r_k. Due to the reflection symmetry that allows swapping r_a and r_b, these six subcases are sufficient to prove that the three records are ordered correctly whenever one of them is within $\lfloor \epsilon N/2 \rfloor$ of r_b. This concludes case 3 of Step 3: any three records at least ϵN away from each other, 1, and N are correctly ordered.

Step 4. At this point, we have proven that all records in $(\epsilon N, N+1-\epsilon N)$ that are at least ϵN away from each other are ordered correctly in all orderings $\prec$ compatible with the PQ tree, except with probability δ. Since, by Property 3, there are at least 3 such records, we may apply Lemma 4.7: the lowest common ancestor of any set of such records is a Q node, and it is also the lowest common ancestor of r_a and

r_b. Lemma 4.8 allows us to conclude that all of the ordered sets belong to different children of the lowest common ancestor. Property 3, which said that no majority of records is at the endpoints 1 or N, or in one range of length less than ϵN, ensures that Algorithm 4.3 will find this Q node, thus concluding the proof of the theorem. □

4.3.4 Experimental results for dense and sparse data

Assuming uniform queries, the ϵ-approximate ordered reconstruction attack succeeds within $\mathcal{O}\big(\epsilon^{-1}\log\epsilon^{-1}\big)$ queries for any given constant probability of success $\eta < 1$. We experimentally evaluate the tightness of this bound for a fixed number of records n_r, and various numbers of possible values, N, so that we generate both dense and sparse databases. In our experiments, we used a C++ implementation [25] of the PQ tree data structure and used the interface generator SWIG [77] to call it from Python.

Record values are sampled uniformly at random, so Properties 1, 2, and 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. We refer to the chosen Q node's children as *buckets*; these are the sets A_i output by the algorithm.

Figure 4.1 shows two properties of the resulting PQ tree (on the y-axis) for different numbers of queries (on the x-axis) and different values of N (corresponding to different lines). The first property, depicted in the bottom group of lines, is the maximum *symmetric value*, as a fraction of N, of any "sacrificed" record that was not in one of the Q node's children. The symmetric value of a record r is $\min\{\mathsf{val}(\mathsf{r}), N+1-\mathsf{val}(\mathsf{r})\}$. When Algorithm 4.3 succeeds, the only records that are not necessarily in buckets are those with values in $[1, \epsilon N)$ or $(N+1-\epsilon N, N]$, i.e., those with symmetric value less than ϵN. 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. 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 4.1 shows the second property: the maximum diameter, as a fraction of N, of the Q node's children (buckets). We compare this to the

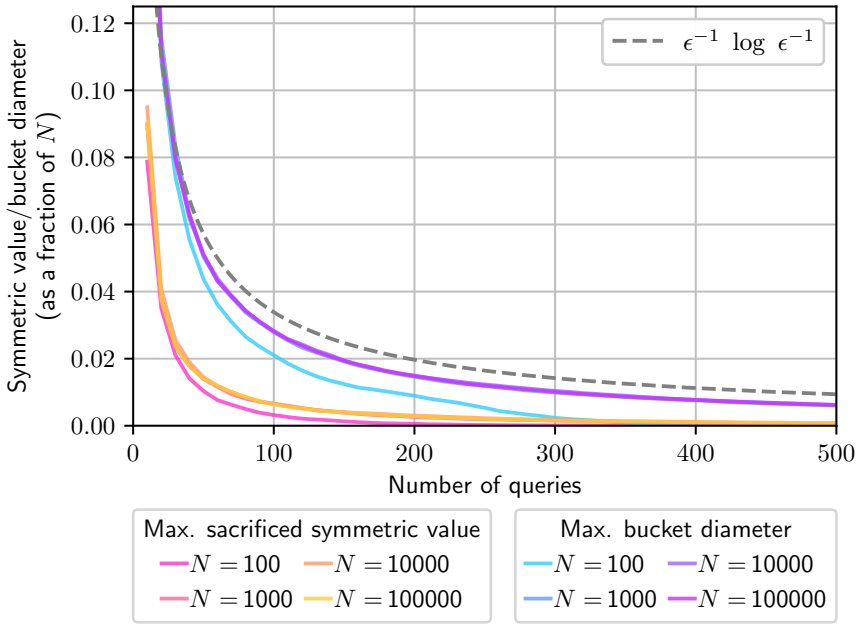
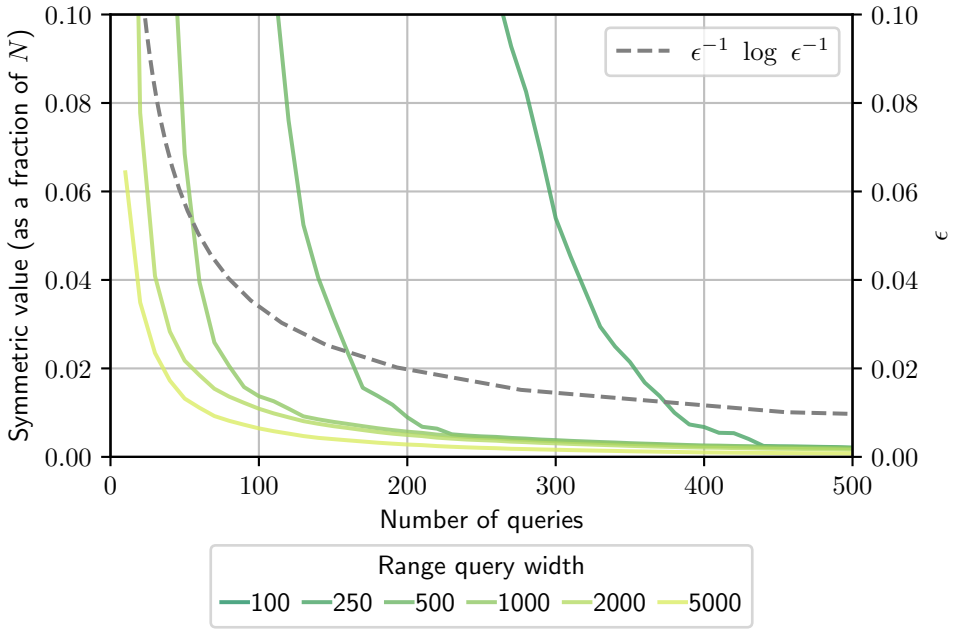


Figure 4.1: Maximum symmetric values of records not in buckets and maximum bucket diameters. Results averaged over 500 databases for each value of N, with $\mathsf{n_r} = 1000$ records.

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.

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 4.1 could be interpreted in this way.

Although our theoretical analysis for Algorithm 4.3 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 Algorithm 4.3 performs for a dataset of $\mathsf{n_r} = 1000$ records, $N = 10000$ possible values, and range queries of different widths. The results are in Figure 4.2. 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



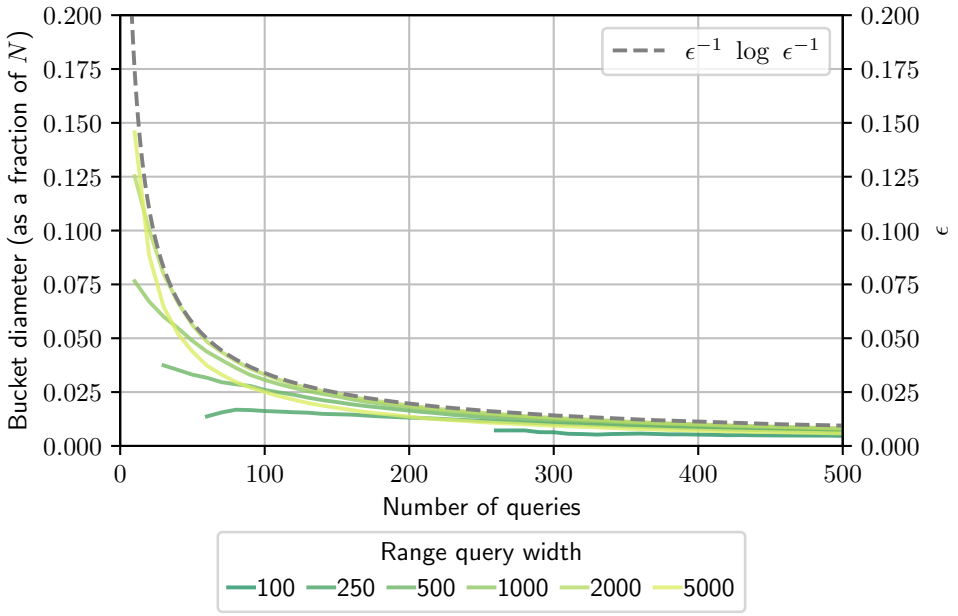


Figure 4.2: (Top) Maximum symmetric values of records not in buckets. (Bottom) Maximum bucket diameters. Results for fixed-width queries averaged over 500 databases for each value of range query width, with $\mathsf{n_r} = 1000$ records.

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 behavior of $\epsilon^{-1} \log \epsilon^{-1}$.

4.4 Conclusions

The attacks in this chapter target database reconstruction using only access pattern leakage from range queries. The exact and approximate algorithms in Sections 4.1 and 4.2 succeed with the same expected query complexity, $\mathcal{O}(N \log N)$, as the exact and approximate algorithms from the previous chapter, which additionally used rank leakage. Therefore, asymptotically, it seems that the only advantage of rank leakage is breaking the global symmetry of the reconstructed values. These two attacks are well suited to dense databases where the number of possible values a record can take, N, is not too high.

In Section 4.3, we designed an approximate *ordered* reconstruction (AOR) algorithm that performs well even on sparse databases. The algorithm uses a PQ tree to process query leakage in an efficient manner. Rather than assigning a value to each record, it groups together records with similar values, and orders these groups. The runtime of the algorithm and the number of queries required to group records into small-diameter "buckets" is independent of the number of possible values, N. The small cost to pay for this scale-free behavior is that records whose values are near the endpoints 1 and N are sacrificed—they are not necessarily included in any of the ordered buckets. In particular, an attacker may learn only that these records have very small or very large values.

Our analysis of the AOR algorithm used tools from statistical learning theory. Despite these tools giving only an asymptotic bound on the number of queries required for AOR to succeed, our experimental evaluations in Section 4.3.4 show the hidden constants are small, i.e., about 1, and suggest that our algorithm may be quite effective with few queries.

Chapter 5

Volume attacks

Background. Chapter 3 dealt with reconstruction attacks on access pattern and rank leakage from range queries, while the previous chapter, Chapter 4, dealt with reconstruction attacks using only access pattern leakage. In this chapter, we continue the trend of reconstruction attacks using even less leakage: we use only the number of records that were returned by a given range query $[a, b]$, which we call the *volume* of the query, $\mathsf{vol}([a, b])$.

This work was again inspired by the paper of Kellaris *et al.* [42] (KKNO), which also presented attacks on volume leakage. They were the first to show how to reconstruct element counts in a database given leakage from uniformly random range queries. However, their algorithm strongly depends on the queries being uniformly random, and it is not clear how to adapt it to other cases. KKNO proved that, in general, any algorithm successfully achieving exact reconstruction from volume leakage requires $\Omega(N^4)$ range queries.

With Paul Grubbs, Brice Minaud, and Kenny Paterson, we set out to design a more practical attack that would succeed with fewer queries on a more restricted set of databases. Our work was published in the paper "Pump up the Volume: Practical Database Reconstruction from Volume Leakage on Range Queries" at ACM CCS 2018 [27] and the full version appears on the IACR's Cryptology ePrint Archive [26]. My main contributions to the paper were designing the count reconstruction algorithm (in particular, the pre-processing step), correctness proofs, and designing and implementing an evaluation of the main attack.

Introduction. We target a different type of reconstruction than in previous chapters. Since no access pattern leakage is available, there are no record identifiers; there

is no way to distinguish a query matching records with identifiers 1 and 2 from a query matching records with identifiers 2 and 3. Our goal is instead *count reconstruction*: determining the exact number of elements in the database with each value from 1 to N. Like the access pattern attacks in the previous chapter, we can accomplish reconstruction only up to reflection since replacing each value $\mathsf{val}(\mathsf{r})$ in the database with $N+1-\mathsf{val}(\mathsf{r})$ and swapping query $[a, b]$ with $[N+1-b, N+1-a]$ yields exactly the same leakage.

Volume leakage. The volume of a query can leak in a number of ways. For instance, a network eavesdropper could simply measure packet sizes and infer how many records were returned—even if data is encrypted and authenticated, its length can leak. Another source of volume leakage could be diagnostic tables that record which queries match the most records [31]. Even the time the database server takes to process a query could indicate the number of matching records. Exploiting volume leakage is similar to traffic analysis attacks, such as identifying which video a user is streaming based on the burst rate [76].

Elementary ranges and volumes. Consider the ranges $[1, 1]$, $[1, 2]$, ..., $[1, N]$. We refer to these as *elementary ranges*. In a dense database, knowing the volumes of these N ranges is necessary and sufficient for count reconstruction: if we know the volumes of $[1, 1]$, $[1, 2]$, ..., $[1, N]$, then the number of records that have value k is the difference between the $(k-1)$st and kth element in the list (treating the 0th element as zero). The goal of our algorithm is to identify the volumes of these elementary ranges (which we call simply *elementary volumes*) among the set of all volumes.

Our approach is based on three properties of elementary ranges and volumes. First, every range is either an elementary range or a difference of elementary ranges: $[a, b]$ can be expressed either as $[1, b]$ if $a = 1$ or as $[1, b] \setminus [1, a-1]$ if $a > 1$. In terms of volumes, this property reflects that every volume is either an elementary volume or the difference of two elementary volumes. If all queries were observed, then the converse also holds: the difference of any two elementary ranges is a range, and the difference of any two elementary volumes is also an observed volume. Third, all elementary ranges other than $[1, N]$ are $\mathsf{n_r}$-*complemented*: each elementary range $[1, b]$ has a complement $[b+1, N]$ such that the sum of these two volumes is $\mathsf{n_r}$, the total number of records. If we consider the range $[1, N]$ as an elementary range and the volume $\mathsf{n_r}$ as an elementary volume, then the pairwise differences of all elementary volumes and the elementary volumes themselves generate the entire set of all query volumes. As we will explain, this last property translates well to a clique in a graph.

We will assume, as in other chapters, that the number of possible values, N, is known. When it is not, however, note that N nodes are not always necessary to generate all observed volumes V: two databases with different numbers of elements N can generate the same volume set. For example, consider $N = 5$ with all elements having counts of 2, so $V = \{2, 4, 6, 8, 10\}$. We could also generate this set of volumes with $N = 3$ and elements having counts 2, 4, and 4.

Practicality. Compared to the best previous known attack—KKNO's volume attack (q.v. Section 1.2)—ours does not require a particular query distribution, which makes it much more applicable to real-world settings. Recall that KKNO established a lower bound on the number of queries necessary to reconstruct counts using only volume leakage: $\Omega\big(N^4 \log N\big)$ uniformly random queries are required. Because our attack requires only that every possible range is queried at least once, which would take in expectation about $N^2 \log N$ queries if they were sampled uniformly at random, it must fail for some databases. However, we aim to have it succeed for realistic databases. In particular, our attack generally succeeds when the number of records in the database, $\mathsf{n_r}$, is at least $N^2/2$, and the complexity does not increase as the number of records increases.

Clique-finding. Our goal of practical attacks may seem at odds with our chosen technique of finding a clique in a graph to identify elementary volumes—after all, the clique decision problem is NP-complete. However, the graph we build has a lot of structure that we can use before resorting to generic clique-finding algorithms.

Density. As we saw in the last chapter, without any information about record values or the query distribution, access pattern leakage from range queries reveals only the *order* of records: our ϵ-approximate ordered reconstruction algorithm groups together records with similar values, and sorts these groups, but it does not assign values to these groups—it does not know where there are gaps between these groups. In the setting this chapter explores—having only volume leakage—an analogous property holds: we can reconstruct counts in order, but only non-zero counts. This still constitutes a considerable amount of information and some knowledge of the database distribution may enable reconstruction of all counts. This is discussed further in Section 5.2. Note that the database being sparse is equivalent to the volume 0 appearing in the set of all volumes.

Chapter overview. We cast the problem of finding the set of elementary volumes as a clique-finding problem in a graph constructed from the volume leakage. Our algorithm has three main steps: observing enough queries to obtain a complete set

of volumes, building the graph, and finding a clique in that graph. We present the details of our complete algorithm for count reconstruction in Section 5.1. Then, in Section 5.2, we present the results of an experimental evaluation of our algorithm on HCUP data (q.v. Section 1.3). We find that applying techniques based on the properties of elementary volumes is often enough to make clique-finding trivial, obviating the need for expensive generic clique-finding algorithms. In Section 5.3, we design a model of the volume graph for databases with uniformly random values, estimating the number of distinct volumes, complementary volumes (vertices), edges, and when there is expected to be only one clique. We find that when $\mathsf{n_r}$ is $\Omega\big(N^2\big)$, we expect our algorithm to perform well.

5.1 Reconstruction for dense data

As mentioned in this chapter's introduction, we target the ordered reconstruction of all non-zero counts in the database, up to reflection. We discuss how to extend results to sparse databases in Section 5.2. Our algorithm has three main steps:

1. observe sufficiently many queries to obtain a complete set of volumes,
2. build a graph using these volumes, and
3. find a clique in this graph.

Step 1. The number of queries needed to see all query volumes depends on the query distribution; we must assume something about it to analyze the query complexity.

If the queries are distributed uniformly at random, then each of the $N(N+1)/2$ ranges can be seen as a coupon that must be collected. Thus, the classic coupon collector problem's analysis applies: the number of necessary queries on expectation is $N(N+1)/2 \cdot \mathsf{H}_{N(N+1)/2}$. By Bound A.2, the expected number of queries to gather all volumes is thus at most

$$N(N+1)/2 \cdot (\log\left(N(N+1)/2\right) + 1) \approx N^2 \log N.$$

For a non-uniform distribution, if the least likely range has probability $\frac{\alpha}{N(N+1)/2}$, then an adaptation of coupon collection analysis shows that $\mathcal{O}\big(\alpha^{-1} N^2 \log N\big)$ queries suffice.

If the queries have a standard Zipfian distribution [80], where the k-th most likely element has probability proportional to $1/k$, then the expected number of queries until each range occurs at least once is $\mathcal{O}\big(N^2 \log^2 N\big)$ [23]. This implies that even if

the query distribution is Zipfian—a skewed distribution—the query complexity of our attack is not much higher than with a uniform distribution.

The complete set of volumes is V and its size is $\mathsf{n_v} \coloneqq |\mathsf{V}|$. Its elements are numbered in increasing order: $\mathsf{v}_1 < \mathsf{v}_2 < \ldots < \mathsf{v}_{\mathsf{n_v}}$. Since there are $N(N+1)/2$ queries, $\mathsf{n_v} \leq N(N+1)/2$. The largest volume, $\mathsf{v}_{\mathsf{n_v}}$, must be the number of records, $\mathsf{n_r} = \mathsf{vol}([1, N])$.

When the data is dense (i.e., each of the N values occurs at least once), $N \leq \mathsf{n_v} \leq N(N+1)/2$. If the data were sparse (i.e., iff the volume 0 appeared in V), then the number of values that would appear in the database (with non-zero counts) would be at least $N_{\min} \coloneqq -0.5 + 0.5 \cdot \sqrt{1 + 8 \cdot (\mathsf{n_v} - 1)}$. We explain this further in Section 5.2.

Step 2. Next, we form an undirected graph from the set of volumes V. Its nodes are the elements of V, and there is an edge between two nodes v and v' iff $|\mathsf{v} - \mathsf{v}'| \in \mathsf{V}$.

Step 3. To identify the set of elementary volumes, we must find a clique of N complemented nodes that generate all volumes: our goal is to identify a subset of $\mathsf{n_r}$-complemented nodes $\mathsf{v}_1^*, \mathsf{v}_2^*, \ldots, \mathsf{v}_N^*$ such that $\mathsf{V} = \{\mathsf{v}_i^*\}_{i=1}^N \cup \{|\mathsf{v}^* - \mathsf{v}^{*\prime}|\}_{(\mathsf{v}^*, \mathsf{v}^{*\prime}) \subseteq \mathsf{V}}$. Note that by reflection symmetry, the volumes $[N, N], [N-1, N], \ldots, [1, N]$ also form a clique in this graph.

Step 3 is carried out in two parts: first, using efficient heuristics to reduce the set of candidate elementary volumes and grow a set of necessary elementary volumes (pre-processing), and second, applying generic clique-finding techniques.

Reasoning behind pre-processing. We use the three properties of elementary ranges and volumes mentioned in the introduction of this chapter to design our efficient graph pre-processing. First, recall that all elementary volumes $\mathsf{vol}([1, b])$ with $b < N$ are $\mathsf{n_r}$-complemented because of the existence of the complementary ranges $[b+1, N]$. The elementary volume corresponding to the range $[1, N]$ is easy to identify—it is simply $\mathsf{n_r}$, the total number of records and the largest volume in V. Therefore, the sought-after elementary volumes must be in the subset $\{\mathsf{v} \in \mathsf{V} : \exists \mathsf{v}' \text{ with } \mathsf{v} + \mathsf{v}' = \mathsf{n_r}\}$ of $\mathsf{n_r}$-complemented volumes, and we may restrict the nodes of the graph to this set.

The remainder of the algorithm is spent alternating between identifying volumes that *must* be elementary volumes and removing nodes from the graph that cannot be elementary volumes. This procedure continues until the sets of nodes stabilize. In our experiments, we found that this pre-processing step often suffices to identify the clique of elementary volumes. When it does not, however, we run a traditional clique-

finding step that tries to identify the remaining nodes that should be in the set of necessary volumes. Pseudocode for pre-processing is in Algorithm 5.1.

Algorithm 5.1 Graph pre-processing: finding a smaller subgraph

Input: set of all volumes V.
Output: sets $\mathsf{V}_{\mathsf{nec}} \subseteq \mathsf{V}$, $\mathsf{V}_{\mathsf{cand}} \subseteq \mathsf{V}$ such that $\mathsf{V}_{\mathsf{nec}} \subseteq \mathsf{V}_{\mathsf{elem}} \subseteq \mathsf{V}_{\mathsf{cand}}$.

1: $\mathsf{n}_\mathsf{r} \leftarrow \max\{\mathsf{V}\}$
2: $\mathsf{V}_{\mathsf{comp}} \leftarrow \{\mathsf{v} \in \mathsf{V} : \mathsf{n}_\mathsf{r} - \mathsf{v} \in \mathsf{V}\} \cup \{\mathsf{n}_\mathsf{r}\}$
3: $\overline{\mathsf{V}_{\mathsf{comp}}} \leftarrow \mathsf{V} \setminus \mathsf{V}_{\mathsf{comp}}$
4: $\mathsf{V}_{\mathsf{cand}} \leftarrow \mathsf{V}_{\mathsf{comp}}$
5: $\mathsf{v}_{\mathsf{min}} \leftarrow \min\{\mathsf{V}_{\mathsf{comp}}\}$
6: $\mathsf{V}_{\mathsf{nec}} \leftarrow \{\mathsf{v}_{\mathsf{min}}, \mathsf{n}_\mathsf{r}\}$
7: allProcessed $\leftarrow$ False
8: **while not** allProcessed **do**
9: $\quad \mathsf{V}^*_{\mathsf{nec}} \leftarrow \textsc{AugmentNec}(\mathsf{V}_{\mathsf{cand}}, \mathsf{V}_{\mathsf{nec}}, \overline{\mathsf{V}_{\mathsf{comp}}}, N)$
10: $\quad \mathsf{V}^*_{\mathsf{cand}} \leftarrow \textsc{ReduceCand}(\mathsf{V}_{\mathsf{cand}}, \mathsf{V}^*_{\mathsf{nec}}, \mathsf{V})$
11: $\quad$ **if** $\mathsf{V}^*_{\mathsf{cand}} = \mathsf{V}_{\mathsf{cand}}$ **and** $\mathsf{V}^*_{\mathsf{nec}} = \mathsf{V}_{\mathsf{nec}}$ **then**
12: $\quad\quad$ allProcessed $\leftarrow$ True
13: $\quad \mathsf{V}_{\mathsf{nec}} \leftarrow \mathsf{V}^*_{\mathsf{nec}}$
14: $\quad \mathsf{V}_{\mathsf{cand}} \leftarrow \mathsf{V}^*_{\mathsf{cand}}$
15: **return** $\mathsf{V}_{\mathsf{cand}}, \mathsf{V}_{\mathsf{nec}}$

16: **procedure** $\textsc{AugmentNec}(\mathsf{V}_{\mathsf{cand}}, \mathsf{V}_{\mathsf{nec}}, \overline{\mathsf{V}_{\mathsf{comp}}}, N)$
17: $\quad$ **if** $|\mathsf{V}_{\mathsf{cand}}| = N$ **then**
18: $\quad\quad \mathsf{V}_{\mathsf{nec}} \leftarrow \mathsf{V}_{\mathsf{cand}}$
19: $\quad\quad$ **return** $\mathsf{V}_{\mathsf{nec}}$
20: $\quad$ **for all** $e \in \overline{\mathsf{V}_{\mathsf{comp}}}$ **do**
21: $\quad\quad$ **for all** $\mathsf{v} \in \mathsf{V}_{\mathsf{cand}} \setminus \mathsf{V}_{\mathsf{nec}}$ **do**
22: $\quad\quad\quad$ **if** $\nexists (w, w') \subseteq (\mathsf{V}_{\mathsf{cand}} \setminus \{\mathsf{v}\}) : |w - w'| = e$ **then**
23: $\quad\quad\quad\quad \mathsf{V}_{\mathsf{nec}} \leftarrow \mathsf{V}_{\mathsf{nec}} \cup \{\mathsf{v}\}$
24: $\quad$ **for all** $\mathsf{v} \in \mathsf{V}_{\mathsf{cand}} \setminus \mathsf{V}_{\mathsf{nec}}$ **do**
25: $\quad\quad$ **if** $\nexists (w, w') \subseteq (\mathsf{V}_{\mathsf{cand}} \setminus \{\mathsf{v}\}) : |w - w'| = \mathsf{v}$ **then**
26: $\quad\quad\quad \mathsf{V}_{\mathsf{nec}} \leftarrow \mathsf{V}_{\mathsf{nec}} \cup \{\mathsf{v}\}$
27: $\quad$ **return** $\mathsf{V}_{\mathsf{nec}}$

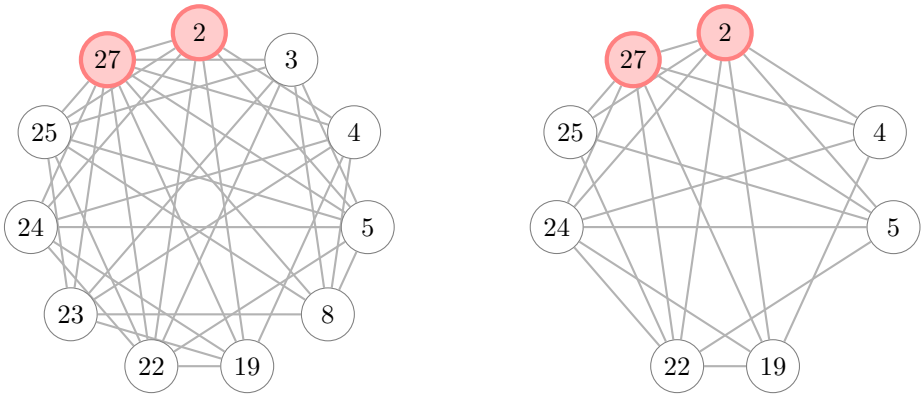
28: **procedure** $\textsc{ReduceCand}(\mathsf{V}_{\mathsf{cand}}, \mathsf{V}_{\mathsf{nec}}, \mathsf{V})$
29: $\quad$ **for all** $\mathsf{v} \in \mathsf{V}_{\mathsf{cand}} \setminus \mathsf{V}_{\mathsf{nec}}$ **do**
30: $\quad\quad$ **for all** $\mathsf{v}_{\mathsf{nec}} \in \mathsf{V}_{\mathsf{nec}}$ **do**
31: $\quad\quad\quad$ **if** $|\mathsf{v} - \mathsf{v}_{\mathsf{nec}}| \notin \mathsf{V}$ **then**
32: $\quad\quad\quad\quad \mathsf{V}_{\mathsf{cand}} \leftarrow \mathsf{V}_{\mathsf{cand}} \setminus \{\mathsf{v}\}$
33: $\quad$ **return** $\mathsf{V}_{\mathsf{cand}}$

Pre-processing algorithm details. We want to identify the N elementary volumes among the complete set of all possible range query volumes, V. In the pre-processing step, we will build up a set $\mathsf{V}_{\mathsf{nec}}$ of volumes that are *necessarily* elementary volumes (up to global reflection), and whittle down a set $\mathsf{V}_{\mathsf{cand}}$ of *candidate* elementary volumes.

The set of n_r-complemented volumes (line 2) contains n_r itself, pairs of volumes, and maybe the singleton volume $\mathsf{n}_\mathsf{r}/2$ if n_r is even and this volume was observed.

(a) The largest observed volume is $\mathsf{n_r}$=27, so we initialize $\mathsf{V_{cand}}$ to the set of $\mathsf{n_r}$-complemented volumes and $\mathsf{V_{nec}}$ to $\{\mathsf{v_{min}}, \mathsf{n_r}\}$=$\{2, 27\}$.

(b) Eliminate candidate volumes 3, 8, and 23 since they are not adjacent to both nodes in $\mathsf{V_{nec}}$.



(c) 4 and 19 are necessary since 15 arises only as their difference. 24 is necessary since it does not arise as a difference of candidate volumes, only as a candidate volume itself.

(d) The number of necessary volumes is N, so pre-processing succeeded. These elementary volumes correspond to element counts 2, 2, 15, 5, 3 (or 3, 5, 15, 2, 2).

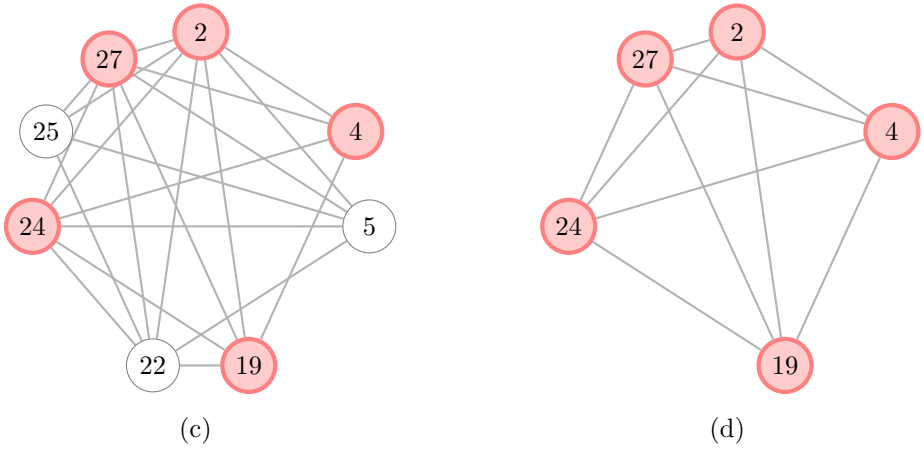


Figure 5.1: An example of pre-processing for a database with $N = 5$ distinct elements having counts 3, 5, 15, 2, and 2. The set of all possible range query volumes is $\{2, 3, 4, 5, 8, 15, 17, 19, 20, 22, 23, 24, 25, 27\}$. Nodes corresponding to necessary elementary volumes have thicker borders and red shading.

Since elementary volumes are $\mathsf{n_r}$-complemented, we set the initial set of candidate elementary volumes to be the set of $\mathsf{n_r}$-complemented volumes (line 4). This is the initial set of nodes. It must contain the volumes of the elementary ranges $[1, 1]$ through $[1, N-1]$ because of their complementary ranges $[2, N]$ through $[N, N]$. It must also contain the elementary volume $\mathsf{n_r}$ for range $[1, N]$.

We present an example of graph pre-processing in Figure 5.1. Subfigure (a) shows the initial graph. For the moment, ignore its distinguished nodes.

We seed the initial set of necessary elementary volumes with the smallest $\mathsf{n_r}$-complemented volume and $\mathsf{n_r}$ itself, which is the volume of $[1, N]$ (line 6). Let $\mathsf{v_{min}}$ be the smallest $\mathsf{n_r}$-complemented volume. Then, up to reflection, it must be an elementary volume: the largest volume strictly smaller than $\mathsf{n_r}$ must be $\mathsf{vol}([1, N-1])$ or $\mathsf{vol}([2, N])$ since every other range is included in one of those two ranges. Therefore, $\mathsf{v_{min}}$ must be either the volume of $[1, 1]$ or the volume of $[N, N]$. Since we are targeting count reconstruction only up to reflection, we choose to set $\mathsf{v_{min}} = \mathsf{vol}([1, 1])$ and thus it is a necessary elementary volume. These are the two nodes highlighted in subfigure (a) in the example in Figure 5.1.

Next, we repeatedly augment the set of necessary elementary nodes and reduce the set of candidate elementary nodes until these sets stabilize (lines 8–14). The augmentation procedure is defined on lines 16–27 and the reduction procedure on lines 28–33.

There are three ways to extend the set of necessary elementary volumes. First, if the set of elementary volume candidates is as small as it can be (i.e., of size N), then all candidate nodes must be necessary (line 17). Second, we look for a non-complemented volume ($e \in \overline{\mathsf{V_{comp}}}$, line 20) that arises only as an edge or edges incident to a single non-necessary candidate elementary volume v (line 22). The latter such non-necessary candidate elementary volume must then be a necessary elementary volume—otherwise e would not arise—so we add it to $\mathsf{V_{nec}}$. In the example in Figure 5.1, we see in subfigure (c) that nodes 4 and 19 have been added to the set of necessary nodes because non-complemented volume 15 arises only as an edge between them. Finally, if any non-necessary candidate elementary volume v (line 24) arises only as itself or as edges incident to itself in the graph (line 25), then it must be an elementary volume and is added to $\mathsf{V_{nec}}$. We see in subfigure (c) that node 24 was added to the set of necessary nodes for this reason. The example finishes in subfigure (d) when all remaining non-necessary candidate nodes are removed since they are not adjacent to all of the necessary nodes.

Reduction of the set of candidate elementary volumes is done by looking for a non-necessary candidate elementary volume v (line 29) that is not adjacent to all necessary elementary volumes. If such a node exists, then it cannot be an elementary volume—since they are adjacent to each other—so it is removed from the set of candidate elementary volumes $\mathsf{V}_{\mathsf{cand}}$ (line 32). In the example in Figure 5.1, we see in subfigure (b) that three nodes have been removed in this way.

The following lemma proves that the pre-processing procedure is correct: it does not eliminate any elementary volumes from the set of candidate elementary volumes, and all necessary elementary volumes actually correspond to elementary volumes.

Lemma 5.1 (Correctness of Algorithm 5.1)**.** *Consider any database* DB *with values in* $[1, N]$*. Let* V *be the set of all possible range query volumes, and let* $\mathsf{V}_{\mathsf{elem}}$ *be the set of elementary range volumes that contains the minimum complemented volume. Then, after running Algorithm 5.1 on* V *to obtain the sets* $\mathsf{V}_{\mathsf{nec}}$ *and* $\mathsf{V}_{\mathsf{cand}}$ *of necessary and candidate nodes, we have* $\mathsf{V}_{\mathsf{nec}} \subseteq \mathsf{V}_{\mathsf{elem}} \subseteq \mathsf{V}_{\mathsf{cand}}$*.*

Proof. We show that $\mathsf{V}_{\mathsf{nec}} \subseteq \mathsf{V}_{\mathsf{elem}} \subseteq \mathsf{V}_{\mathsf{cand}}$ holds throughout Algorithm 5.1. After line 4, we have $\mathsf{V}_{\mathsf{elem}} \subseteq \mathsf{V}_{\mathsf{cand}}$ since all elementary volumes are complemented. After line 6, we have $\mathsf{V}_{\mathsf{nec}} \subseteq \mathsf{V}_{\mathsf{elem}}$ since $\mathsf{n}_{\mathsf{r}} = \mathsf{vol}([1, N])$ is in $\mathsf{V}_{\mathsf{elem}}$, and $\mathsf{v}_{\mathsf{min}}$ is in $\mathsf{V}_{\mathsf{elem}}$ by design.

We now show that if $\mathsf{V}_{\mathsf{nec}} \subseteq \mathsf{V}_{\mathsf{elem}} \subseteq \mathsf{V}_{\mathsf{cand}}$, then (i) $\textsc{AugmentNec}(\mathsf{V}_{\mathsf{cand}}, \mathsf{V}_{\mathsf{nec}}, \overline{\mathsf{V}_{\mathsf{comp}}}, N)$ is contained in $\mathsf{V}_{\mathsf{elem}}$, and (ii) $\mathsf{V}_{\mathsf{elem}}$ is contained in $\textsc{ReduceCand}(\mathsf{V}_{\mathsf{cand}}, \mathsf{V}_{\mathsf{nec}})$. First, consider the three ways in which $\textsc{AugmentNec}$ can add elements to $\mathsf{V}_{\mathsf{nec}}$.

- (line 17) If $|\mathsf{V}_{\mathsf{cand}}| = N$ and $\mathsf{V}_{\mathsf{elem}} \subseteq \mathsf{V}_{\mathsf{cand}}$, then clearly $\mathsf{V}_{\mathsf{elem}} = \mathsf{V}_{\mathsf{cand}}$ since $|\mathsf{V}_{\mathsf{elem}}| = N$.

- (line 20) Let e be a non-complemented volume. Since $\mathsf{V}_{\mathsf{elem}} \subseteq \mathsf{V}_{\mathsf{cand}}$, we know that every volume, including e, arises as a node or an edge (or both) in the graph induced by $\mathsf{V}_{\mathsf{cand}}$. The volume e has no complement, so it must arise as an edge, i.e., as the absolute difference of two volumes in $\mathsf{V}_{\mathsf{cand}}$. If all such edges are incident to one node v in $\mathsf{V}_{\mathsf{cand}}$, then that node is necessarily in $\mathsf{V}_{\mathsf{elem}}$.

- (line 24) Let v be a non-necessary complemented volume in $\mathsf{V}_{\mathsf{cand}}$. Every volume, including v, arises as a node or an edge (or both) in the graph induced by $\mathsf{V}_{\mathsf{cand}}$. If the volume v arises only as itself and maybe edges incident to itself, then it must be in $\mathsf{V}_{\mathsf{elem}}$.

Next, consider ReduceCand. Let v be a non-necessary complemented volume in $\mathsf{V}_{\mathsf{cand}}$. Since $\mathsf{V}_{\mathsf{nec}} \subseteq \mathsf{V}_{\mathsf{elem}}$, and the volumes in $\mathsf{V}_{\mathsf{elem}}$ are all adjacent to each other, any node that is not adjacent to a subset of volumes in $\mathsf{V}_{\mathsf{elem}}$ cannot be in $\mathsf{V}_{\mathsf{elem}}$. □

Clique-finding details. After pre-processing, we have two sets of volumes, $\mathsf{V}_{\mathsf{nec}}$ and $\mathsf{V}_{\mathsf{cand}}$, satisfying $\mathsf{V}_{\mathsf{nec}} \subseteq \mathsf{V}_{\mathsf{elem}} \subseteq \mathsf{V}_{\mathsf{cand}}$. The next part of Step 3 is traditional clique-finding. Detailed pseudocode is given in Algorithm 5.2, along with some subroutines. The only subroutine we do not specify is FindMaximalCliques; we treat it as a black box for now. Algorithm 5.2 returns a set of sets of volumes that each (1) include $\mathsf{v}_{\mathsf{min}}$, (2) have size N, and (3) generate exactly the volumes in V and no others.

As we will see when we present our experimental results in Section 5.2, pre-processing was often enough to find a clique that generated all volumes in V—that is, the sets it found satisfied $\mathsf{V}_{\mathsf{nec}} = \mathsf{V}_{\mathsf{cand}}$ (line 1). This is the case in the example of Figure 5.1 as well. When this is not the case, however, we must find a clique of size N in the graph induced by $\mathsf{V}_{\mathsf{cand}}$ that generates exactly all volumes V. The graph may contain multiple cliques that generate all volumes of V. Each such subclique must be a subclique of a *maximal* clique—a set of nodes for which no node outside the set is adjacent to all nodes within the set. Although the clique of the elementary volumes $\mathsf{V}_{\mathsf{elem}}$ must be a subclique of a maximal clique, it is not necessarily a subclique of a *maximum* clique (the largest maximal clique).

Since the clique we want to find must include the nodes in $\mathsf{V}_{\mathsf{nec}}$, which already form a clique themselves, we can reduce our problem to finding the rest of the clique in the subgraph of non-necessary candidate elementary volumes—that is, the subgraph induced by $\mathsf{V}_{\mathsf{nn\text{-}cand}} \coloneqq \mathsf{V}_{\mathsf{cand}} \setminus \mathsf{V}_{\mathsf{nec}}$. If the number of elementary volumes is N, then this clique part must have size $N - |\mathsf{V}_{\mathsf{nec}}|$ (line 3). It must also generate all *missing* volumes—volumes in V that do not arise as nodes or edges in the subgraph induced by $\mathsf{V}_{\mathsf{nec}}$—and no other volumes outside of V. The missing volumes could arise either as edges among the nodes of this clique part, or as edges between its nodes and the nodes in $\mathsf{V}_{\mathsf{nec}}$. Given such a clique in the subgraph of non-necessary candidate elementary volumes, we recover the elementary volumes by combining it with $\mathsf{V}_{\mathsf{nec}}$.

The algorithm first uses a procedure that finds all maximal cliques in the subgraph induced by the set of non-necessary candidate elementary volumes (line 5). Again, we treat this procedure as a black box for now. For each of these maximal cliques, the algorithm checks whether its size is at least M, the number of missing elementary volumes (line 8). If so, it checks whether the clique, when combined with the known necessary elementary volumes $\mathsf{V}_{\mathsf{elem}}$, generates the entire set of volumes V and maybe

more (line 9). Since this clique could be too large, the next step is to find subcliques of the right size that generate exactly the volumes in V. This is done with the MinSubcliques procedure (lines 18–23). For each subclique $\mathsf{V}_{\mathsf{sk}} \subseteq \mathsf{V}_{\mathsf{k}}$ of size M, the algorithm checks if the previously identified necessary elementary volumes $\mathsf{V}_{\mathsf{nec}}$ combined with the subclique generate exactly the volumes V. If so, the union of these sets is added to the list of returned subcliques (line 22).

Algorithm 5.2 Recovering elementary volumes via clique-finding

Input: candidate elementary volumes $\mathsf{V}_{\mathsf{cand}}$, necessary elementary volumes $\mathsf{V}_{\mathsf{nec}}$, set of all volumes V.
Output: set solutions of sets of N volumes that generate all volumes V.

```
1:  if |V_cand| = |V_nec| then
2:      return {V_nec}
3:  M ← N − |V_nec|
4:  V_nn-cand ← V_cand \ V_nec
5:  cliques ← FindMaximalCliques(V_nn-cand)
6:  solutions ← {}
7:  for all V_k ∈ cliques do
8:      if |V_k| ≥ M then
9:          if GenAllVolumes(V_nec ∪ V_k, V) then
10:             solutions ← solutions ∪ MinSubcliques(V_k, V, M, V_nec)
11: return solutions

12: procedure GenAllVolumes(V_nodes, V)
13:     for all v ∈ V do
14:         if ∄ (v_1, v_2) ⊆ V_nodes : |v_2 − v_1| = v then
15:             if v ∉ V_nodes then
16:                 return False
17:     return True

18: procedure MinSubcliques(V_k, V, M, V_nec)
19:     subcliques ← {}
20:     for all V_sk ⊆ V_k : |V_sk| = M do
21:         if GenExactVolumes(V_nec ∪ V_sk, V) then
22:             subcliques ← subcliques ∪ {V_nec ∪ V_sk}
23:     return subcliques

24: procedure GenExactVolumes(V_nodes, V)
25:     if V_nodes ⊆ V and GenAllVolumes(V_nodes, V) then
26:         for all (v_1, v_2) ⊆ V_nodes do
27:             if |v_2 − v_1| ∉ V then
28:                 return False
29:         return True
30:     else
31:         return False
```

This set of lists of volumes must include $\mathsf{V}_{\mathsf{elem}}$, as the following lemma proves.

Lemma 5.2 (Correctness of Algorithm 5.2)**.** *Let* DB *be a database of elements with N possible different values, let* V *be the set of all range query volumes, and let* $\mathsf{V}_{\mathsf{elem}}$ *be the set of elementary range volumes that contains the minimum complemented*

volume. Suppose we are given two sets $\mathsf{V}_{\mathsf{nec}}$ *and* $\mathsf{V}_{\mathsf{cand}}$ *such that* $\mathsf{V}_{\mathsf{nec}} \subseteq \mathsf{V}_{\mathsf{elem}} \subseteq \mathsf{V}_{\mathsf{cand}}$. *Then, after running Algorithm 5.2 on* $(N, \mathsf{V}_{\mathsf{cand}}, \mathsf{V}_{\mathsf{nec}}, \mathsf{V})$ *to obtain the set* solutions, *we have* $\mathsf{V}_{\mathsf{elem}} \in \mathsf{solutions}$.

Proof. Consider the graph $G \coloneqq (\mathsf{V}_{\mathsf{cand}}, E)$ with edge set $E \coloneqq \{(\mathsf{v}_1, \mathsf{v}_2) \subseteq \mathsf{V}_{\mathsf{cand}} : |\mathsf{v}_2 - \mathsf{v}_1| \in \mathsf{V}\}$ and the subgraph $G_{\mathrm{nn}} \coloneqq G(\mathsf{V}_{\mathsf{nn\text{-}cand}})$ induced by the set of non-necessary candidate elementary volumes. Since the subgraph induced by $\mathsf{V}_{\mathsf{elem}}$ is a clique in G, the subgraph induced by $\mathsf{V}_{\mathsf{elem}} \setminus \mathsf{V}_{\mathsf{nec}}$ will also be a clique in G_{nn}. Since every clique is contained in (or simply is) a maximal clique, at least one of the maximal cliques in G_{nn} output by FindMaximalCliques (line 5), say $\mathsf{V}_{\mathsf{k}}^*$, will have $\mathsf{V}_{\mathsf{elem}} \setminus \mathsf{V}_{\mathsf{nec}}$ as a subclique. The size of $\mathsf{V}_{\mathsf{k}}^*$ must be at least $N - |\mathsf{V}_{\mathsf{nec}}|$, so the algorithm will proceed to line 9 in this iteration. Since $\mathsf{V}_{\mathsf{elem}} \subseteq \{\mathsf{V}_{\mathsf{nec}} \cup \mathsf{V}_{\mathsf{k}}^*\}$ generates all volumes in V (and maybe others), solutions will be updated to include the output of MinSubcliques (line 10).

Since $\mathsf{V}_{\mathsf{elem}} \setminus \mathsf{V}_{\mathsf{nec}}$ is a subset of $\mathsf{V}_{\mathsf{k}}^*$ of size $M = N - |\mathsf{V}_{\mathsf{nec}}|$, it will arise as a subclique on line 20 of MinSubcliques. $\mathsf{V}_{\mathsf{elem}}$ generates all volumes in V and no others, so the algorithm will proceed to add it to subcliques. Any element added to subcliques in MinSubcliques will form part of the solutions output by the algorithm, completing the proof. □

5.1.1 Efficiency analysis

The main loop of the pre-processing procedure (line 8 of Algorithm 5.1) augments $\mathsf{V}_{\mathsf{nec}}$ or reduces $\mathsf{V}_{\mathsf{cand}}$ at each iteration. Since there are at most distinct $N(N+1)/2$ volumes in $\mathsf{V}_{\mathsf{cand}}$ initially, this step iterates $\mathcal{O}(N^2)$ times.

The bulk of the time complexity comes if the clique-finding procedure (Algorithm 5.2) is run. Recall that we have not specified FindMaximalCliques. In general, a graph on n nodes can have an exponential (in n) number of maximal cliques [60]. This seems incompatible with our goal of *practical* reconstruction attacks, but when the number of nodes is small, it is still feasible to enumerate all of the maximal cliques with an algorithm such as Bron–Kerbosch [15], and for larger domains, there are logarithmic-time algorithms to sample one maximal clique at a time [52].

In the MinSubcliques procedure in Algorithm 5.2, the check on line 21 occurs for $\binom{|\mathsf{V}_{\mathsf{k}}|}{|\mathsf{V}_{\mathsf{sk}}|}$ subcliques. The maximal clique V_{k} has $\mathcal{O}(N^2)$ volumes, and the subclique's size M is $\mathcal{O}(N)$, so $\binom{|\mathsf{V}_{\mathsf{k}}|}{|\mathsf{V}_{\mathsf{sk}}|}$ is $\Omega(N^N)$, not practical at all for large values of N.

To address these potential impracticalities of finding all maximal cliques and finding minimal subcliques that generate all volumes, we design some more efficient procedures for our evaluation in the following section. The pseudocode for these practical variants is in Algorithm 5.3.

First, we design a probabilistic variant of FindMaximalCliques, which we call FindMaximalCliquesP, that replaces line 5 of Algorithm 5.2 with

5: cliques $\leftarrow$ FindMaximalCliquesP($\mathsf{V}_{\text{nn-cand}}, M, \mathsf{V}$).

For graphs with 20 or fewer nodes, we used the `find_cliques` routine from the `NetworkX` Python module (line 4) [33]. For graphs with more nodes, we sampled maximal cliques one at a time, 1000 times, (line 7) using Luby's efficient parallel algorithm for maximal independent sets, implemented as the `max_independent_vertex_set` routine from the `graph-tool` Python module [66].

We also modify Algorithm 5.2 to return all solutions (when possible), not only minimal ones. Specifically, we replace MinSubcliques with AllSubcliquesP as defined starting on line 15 in Algorithm 5.3. If we deem it impractical to enumerate the subcliques of the right size, then we do not return any subcliques—the final solutions will be incomplete.

The three points at which clique-finding may fail with these probabilistic variants are

(i) FindMaximalCliquesP fails to find any maximal cliques of size at least M (line 11),

(ii) it found such cliques, but none of them generated the set of missing volumes (line 13), or

(iii) there were such cliques that generated the set of missing volumes, but for all of them, it was impractical (line 17) to find all of their subclique solutions.

5.2 Experimental evaluation

In this section, we present an experimental evaluation of Steps 2 and 3 of the reconstruction attack from the previous section. We simulate an attacker who has observed enough queries to see all possible volumes of range queries. We implemented our algorithms in Python and used the `graph-tool` [66] and `NetworkX` [33] packages for finding cliques (or maximal independent vertex sets, the dual problem).

Datasets and methodology. We test our algorithm on various attributes from three different years of HCUP data medical records (q.v. Section 1.3), The attributes

Algorithm 5.3 Practical, probabilistic subroutines for Algorithm 5.2

procedure FINDMAXIMALCLIQUESP($\mathsf{V}_{\text{nn-cand}}, M, \mathsf{V}$)
  $\text{cliques} \leftarrow \{\}$
  if $|\mathsf{V}_{\text{nn-cand}}| \leq 20$ **then**
    $\text{cliques} \leftarrow$ FINDMAXIMALCLIQUES($\mathsf{V}_{\text{nn-cand}}$) ▷ NetworkX
  else
    for $i \in \{1, \ldots, 1000\}$ **do**
      $\mathsf{V}_k \xleftarrow{\$}$ FIND_A_MAXIMAL_CLIQUE($\mathsf{V}_{\text{nn-cand}}$) ▷ graph-tool
      if $|\mathsf{V}_k| \geq M$ **then**
        $\text{cliques} \leftarrow \text{cliques} \cup \{\mathsf{V}_k\}$
    if $\text{cliques} = \{\}$ **then**
      return $\perp$ ▷ All sampled cliques too small
    if $\nexists\, \mathsf{V}_k \in \text{cliques} :$ GENALLVOLUMES($\mathsf{V}_{\text{nec}} \cup \mathsf{V}_k, \mathsf{V}$) **then**
      return $\perp$ ▷ No sampled clique gen. all volumes
  return cliques

procedure ALLSUBCLIQUESP($\mathsf{V}_k, \mathsf{V}, m, \mathsf{V}_{\text{nec}}$)
  $\text{subcliques} \leftarrow \{\}$
  if $\binom{|\mathsf{V}_k|}{m} \leq 2000$ **then**
    for all $\mathsf{V}_{sk} \subseteq \mathsf{V}_k : |\mathsf{V}_{sk}| = m$ **do**
      if GENEXACTVOLUMES($\mathsf{V}_{\text{nec}} \cup \mathsf{V}_{sk}, \mathsf{V}$) **then**
        $\text{subcliques} \leftarrow \text{subcliques} \cup \{\mathsf{V}_{\text{nec}} \cup \mathsf{V}_{sk}\}$
  return subcliques

we chose to extract have domain sizes that range from $N = 4$ to $N = 366$ and they are all attributes on which range queries are meaningful. For more information about these datasets and how we extracted attributes, see Section 1.3. Each of the three years includes patient discharge records from about 1000 hospitals, giving us 3000 datasets for most attributes. (Some were not available in all years.)

We say the attack succeeds if there is a single solution output by Algorithm 5.2, and it is the *set* of elementary volumes (up to reflection). For dense datasets (where every value appears at least once and no range query has volume 0), this means that all element counts have been recovered exactly, up to reflection. For sparse datasets, this means that all non-zero element counts have been recovered in order (up to reflection), but it is not known which elements did not appear in the database—the attacker must make a decision about *which* values were not observed. In our evaluation of step 3, we discuss and evaluate one such strategy, which uses a small amount of auxiliary information, for assigning the recovered counts to a subset of elements in the domain.

Handling sparse datasets. Not all databases were dense and we adapted our algorithms to work with a lower bound, $N_{\min}$, of the number of elements with non-zero counts. A database is non-dense iff 0 was an observed query volume. Since the total number of ranges (and therefore the total number of distinct volumes, $|\mathsf{V}|$) is at

most $N(N+1)/2$ plus 1 for the empty range, we must have have

$$\begin{aligned} & |\mathsf{V}| - 1 \leq N(N+1)/2 \\ \Leftrightarrow \quad & 0 \leq N^2 + N - 2(|\mathsf{V}| - 1) \\ \Leftrightarrow \quad & 0 \leq N^2 + N - 2\,|\mathsf{V}| \\ \Leftrightarrow \quad & N \geq -0.5 + 0.5 \cdot \sqrt{1 + 8 \cdot (|\mathsf{V}| - 1)}. \end{aligned}$$

In graph pre-processing (Algorithm 5.1), we simply check after line 1 whether a query of volume 0 was observed. If so, we replace N by

$$N_{\min} \coloneqq \left\lceil -0.5 + 0.5 \cdot \sqrt{1 + 8 \cdot (|\mathsf{V}| - 1)} \right\rceil$$

and remove 0 from the set of volumes before continuing. In clique-finding (Algorithm 5.2), additional changes are necessary. In particular, we also use an upper bound on the number of non-zero counts. Since every distinct value appearing in the database has a corresponding elementary volume, and the elementary volumes must be a subset of the candidate elementary volumes after pre-processing, we deduce that an upper bound on the number of elements with non-zero counts is $N_{\max} \coloneqq |\mathsf{V}_{\mathsf{cand}}|$. Thus we obtain a range of possible values, $[N_{\min}, N_{\max}]$. These yield, in turn, a range of possible missing numbers of elementary volumes, $[M_{\min}, M_{\max}]$ instead of just M as in Algorithm 5.2. On line 8, we must check that the size of the clique is at least $M_{\min}$. In MinSubcliques, we must check not only subsets of size M of V_k, but all subsets of size in the range $[M_{\min}, \max\{|\mathsf{V}_\mathsf{k}|, M_{\max}\}]$. Our practical, probabilistic procedures in Algorithm 5.3 also required a few changes. On line 8 of FindMaximalCliquesP, we must check that the size of the clique is at least $M_{\min}$. On line 17, we check the sum $\sum_{m=M_{\min}}^{M_{\max}} \binom{|\mathsf{V}_\mathsf{k}|}{m}$, and on line 18 of AllSubcliquesP, we must check all subsets of V_k having size in the range $[M_{\min}, \max\{|\mathsf{V}_\mathsf{k}|, M_{\max}\}]$.

Pre-processing evaluation. For each dataset-attribute combination, we ran Algorithm 5.1 to obtain sets of necessary elementary volumes, $\mathsf{V}_{\mathsf{nec}}$, and candidate elementary volumes, $\mathsf{V}_{\mathsf{cand}}$. The plot in Figure 5.2 shows, for each attribute, the (average) number of datasets for which pre-processing was sufficient for the attack to succeed. The results are averaged for attributes that were available in more than one year, since results were similar.

For all attributes except `AGE` and `AGEDAY`, pre-processing correctly identified the non-zero element counts in order (up to reflection) for the vast majority of datasets. The difference in patterns on the bars indicates which datasets were dense. Attributes with smaller domain sizes, e.g., `AMONTH`, `MRISK`, `APRDRG_Severity` (`SEV`), and `ZIPINC`, were

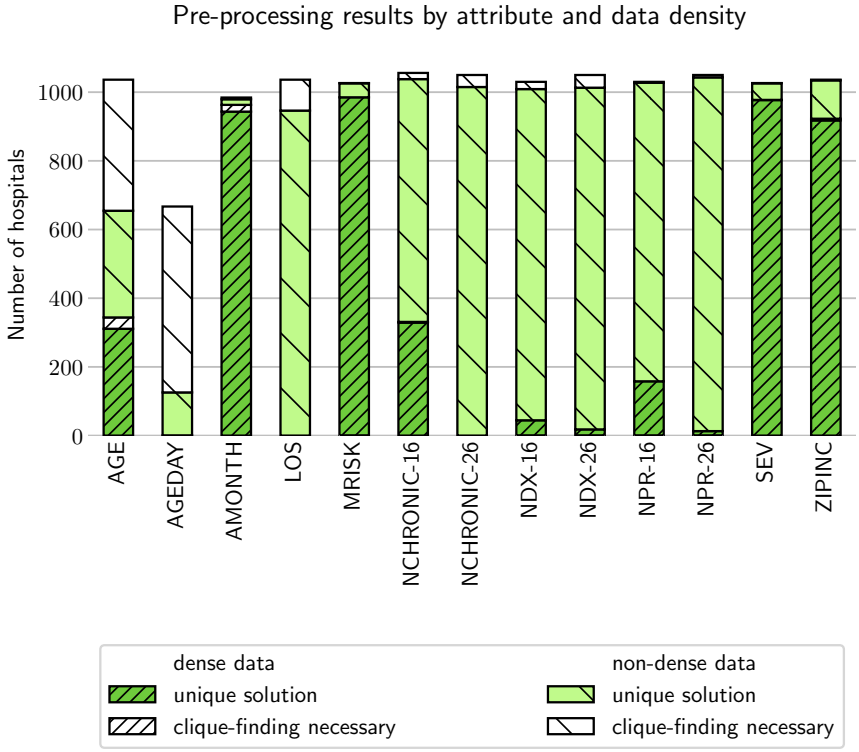


Figure 5.2: Pre-processing success and data density by attribute

dense most of the time. The attributes with the largest domain, `LOS` and `AGEDAY`, were dense in fewer than 0.01% of datasets. Pre-processing recovered the set of elementary volumes for at least 90% of all dense datasets for each attribute except `AGEDAY`. This attribute had a single dense dataset in each 2004 and 2008 that required clique-finding.

For sparse datasets, recovering the set of all non-zero counts provides a lot of information. Combining it with some rudimentary information about the database distribution can lead to recovering *all* element counts, just like in the dense case. For instance, one might guess that the length of stay (`LOS`), the number of chronic conditions (`NCHRONIC`), and the number of procedures (`NPR`), might be 0 most frequently, then decrease.

To illustrate just how valuable knowing the set of elementary volumes could be when combined with a tiny bit of knowledge about the domain, we evaluate the following strategy for assigning counts to elements: simply guess that they correspond to the first values in the domain. The results are displayed in Figure 5.3. We juxtapose the success of our simple strategy for `LOS`, `NCHRONIC`, and `NPR` with its mediocre results for the number of diagnoses, `NDX`, which is 1 more frequently than 0, and thus our strategy is not suitable. This strategy could be adapted to other distributions, e.g., by assigning the non-zero counts to elements in the middle or end of the domain.

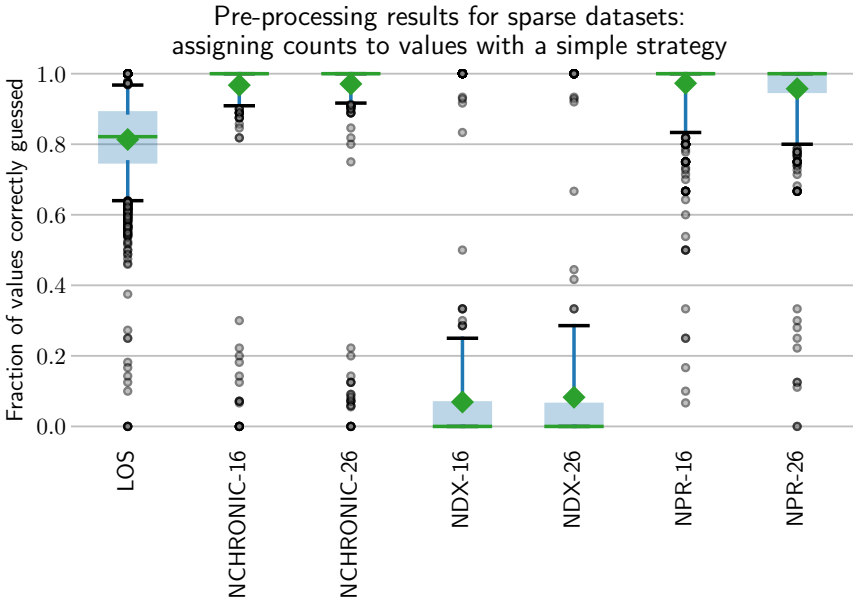


Figure 5.3: Extending pre-processing success for sparse datasets. The fraction of correct values is out of the actual number of values for each dataset. Whiskers indicate the 5th and 95th percentiles, while boxes indicate the 25th and 75th percentiles, with a line at the median and a diamond at the mean.

When pre-processing is not sufficient for sparse datasets, the attacker knows only that the number of values that appear is between N and $N_{\max}$. Although N may be much smaller than the number of candidate volumes $\mathsf{V}_{\mathsf{cand}}$ output by pre-processing, the size of the latter was exactly N in many of the `AGEDAY` datasets, suggesting another strategy for the attacker: if the set $\mathsf{V}_{\mathsf{cand}}$ has size at most $N_{\max}$ and it forms a clique, simply guess that these are the elementary volumes.

Clique-finding evaluation. Lastly, we ran Algorithm 5.2 (with the modifications described in Section 5.1.1) on the few dataset-attribute combinations for which pre-processing did not find a unique solution. Recall that the modifications include returning *all* solutions, not just minimal ones, due to replacing MinSubcliques in Algorithm 5.2 with AllSubcliquesP (described starting on line 15 in Algorithm 5.3). We also allowed the clique-finding algorithm to return an incomplete list of solutions, or to fail entirely. Figure 5.4 shows the overall attack results. Success, in green, occurs when pre-processing or clique-finding finds the solution and it is unique—there is a single clique whose size is in the right range that generates all observed volumes. Multiple cliques, in blue, arise when clique-finding has found all such solutions, but there is more than one, so that the correct solution cannot be precisely determined. Failure, in red, arises when Algorithm 5.2 returns FAILURE or {}, or when we sam-

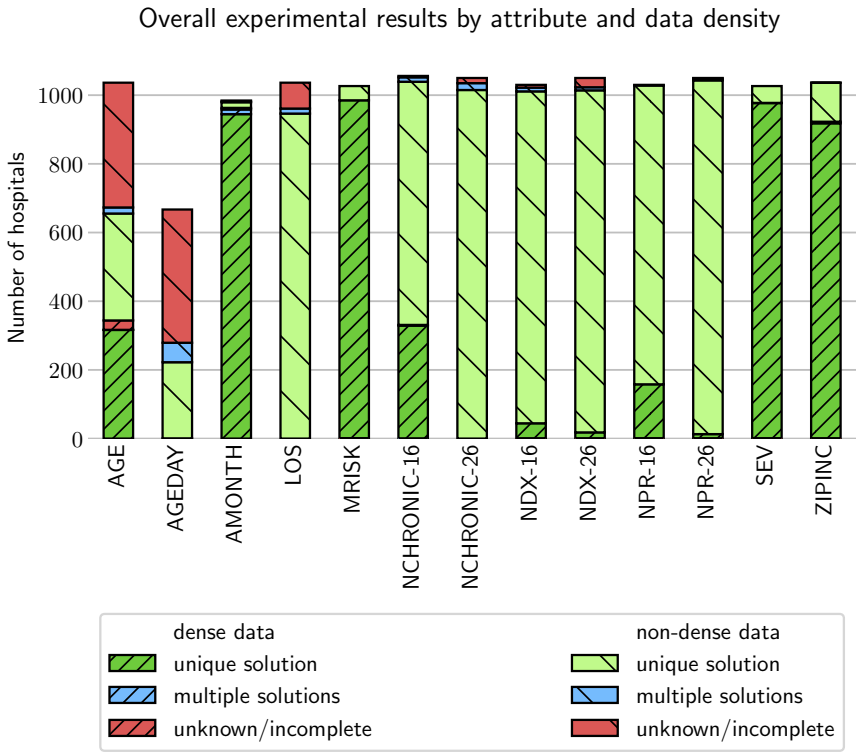


Figure 5.4: Overall results of the practical reconstruction attack

pled maximal cliques using Luby's algorithm (line 7) and may not have found all maximal cliques.

Recall from the end of Section 5.1.1 the three points where clique-finding may fail. In our experiments, the most common reason for failure overall was (iii): it was impractical to find all subcliques (about 60% of failures or incomplete cases). The second most common overall reason for failure was (ii), not finding any cliques that generated all missing volumes (about 36%). However, as one might expect because of the bound on line 17 in AllSubcliquesP, the attributes with fewer possible values (e.g., `AGE` with $N = 91$ compared to `AGEDAY` with $N = 365$) failed more often due to no cliques generating all volumes as opposed to too-big cliques.

Conclusions. Overall, our experiments indicate that our clique-finding approach yields overwhelming success in reconstructing counts of dense datasets—and that in most cases, no expensive clique-finding is even required (see the white bars corresponding to dense data in Figure 5.2).

For sparse data, the success of this approach mainly depends on what auxiliary information is available to the attacker. We showed how an attacker can leverage

rudimentary knowledge of a distribution (e.g., that the most frequent values are the smallest in the domain) to correctly assign exact counts to values (see Figure 5.3).

5.3 Analytical model of the graph

In this section, we develop an analytical model for the behavior of our main attack from Section 5.1. We estimate three quantities in terms of the number of possible values N and the number of records $\mathsf{n_r}$: the number of volume collisions (which determines the size of V), the number of complemented volumes (which determines the number of vertices in the initial graph), and the number of edges. The main takeaway is that the ratio $\mathsf{n_r}/N^2$ emerges as the critical value that determines whether our main attack succeeds.

For this model, we assume that the data is distributed uniformly at random in $[1, N]$, so the number of records matching any particular value follows a binomial distribution $\mathsf{Bin}\,(\mathsf{n_r}, 1/N)$. The number of records matching any particular range $[a, b]$ is also binomially distributed with success probability $(b - a + 1)/N$. The main idea behind our model is to use independent Poisson random variables to approximate the volume of each range. We retroactively validate our series of assumptions in Section 5.3.4.

5.3.1 Number of distinct volumes

Two volumes collide iff there are two distinct range queries that match the same number of records. Intuitively, since there are $\binom{N}{2} \approx N^2/2$ different ranges and each range matches between 0 and $\mathsf{n_r}$ queries, there should be few collisions when $\mathsf{n_r}/(N^2/2)$ is not too close to 1. We estimate the number of volume collisions by counting pairwise collisions, which provides a good estimate since three-way collisions are rare for our parameter choices. We also count only collisions between ranges of the same length—since we assume data is uniformly distributed, we expect nearly all collisions to be of this type. These assumptions will also help simplify the analysis.

Consider ranges of length d, i.e., $[a, b]$ with $d = b - a + 1$. The number of records each such range matches is binomially distributed and its expected value is $\mathsf{n_r} \cdot d/N$. We model this number as a Poisson random variable with rate $\lambda = \mathsf{n_r} \cdot d/N$. Then, to analyze the probability of a collision, we consider the difference $\mathsf{X} - \mathsf{X}'$ of two identically distributed random variables $\mathsf{X}, \mathsf{X}' \sim \mathsf{Pois}\,(\mathsf{n_r} \cdot d/N)$. Let $\Delta_d \coloneqq \mathsf{X} - \mathsf{X}'$ be the difference of these two Poisson random variables. By definition, the difference of two Poisson random variables has a Skellam distribution. In our case—for two identically distributed Poisson random variables—the Skellam pmf simplifies to

$\text{Prob}[\Delta_d = x] = e^{-2\mathsf{n_r}\cdot d/N} \cdot \mathsf{I_x}(2\mathsf{n_r} \cdot d/N)$ where $\mathsf{I}_\alpha(x)$ denotes the modified Bessel function of the first kind with order α and argument x. Therefore, the probability that the two volumes collide is $\text{Prob}[\Delta_d = 0] = e^{-2\mathsf{n_r}\cdot d/N} \cdot \mathsf{I}_0(2\mathsf{n_r} \cdot d/N)$. Using Approximation A.1 for the Bessel function $\mathsf{I}_\alpha(x)$, we have

$$\mathsf{I}_0(2\mathsf{n_r} \cdot d/N) \approx e^{2\mathsf{n_r}\cdot d/N}/\sqrt{2\pi(2\mathsf{n_r} \cdot d/N)},$$

so the probability of a collision of two length-d intervals is

$$\text{Prob}[\Delta_d = 0] \approx 1/2\sqrt{\pi\mathsf{n_r} \cdot d/N}.$$

Since there are $N + 1 - d$ ranges of length d, we can approximate the number of pairwise volume collisions among them by $\binom{N+1-d}{2} \cdot \text{Prob}[\Delta_d = 0]$. Next, considering all range lengths d from 1 to $N - 1$, we obtain the following approximation of the total number of pairwise volume collisions:

$$\begin{aligned}
\sum_{d=1}^{N-1} \binom{N+1-d}{2} \cdot \frac{1}{2\sqrt{\pi\mathsf{n_r} \cdot d/N}} &= \sum_{d=1}^{N-1} \frac{(N+1-d)(N-d)}{4\sqrt{\pi\mathsf{n_r} \cdot d/N}} \\
&\approx \frac{1}{4\sqrt{\pi\mathsf{n_r}}} \sum_{d=1}^{N-1} \frac{(N-d)^2}{\sqrt{d/N}} \\
&= \frac{N^2}{4\sqrt{\pi\mathsf{n_r}}} \sum_{d=1}^{N-1} \frac{(1-d/N)^2}{\sqrt{d/N}} \\
&= \frac{N^3}{4\sqrt{\pi\mathsf{n_r}}} \sum_{d=1}^{N-1} \frac{1}{N} \cdot \frac{(1-d/N)^2}{\sqrt{d/N}}.
\end{aligned}$$

The sum on the last line is the left Riemann sum for the integral $\int_{1/N}^{1} (1-x)^2/\sqrt{x}\, dx$ from $1/N$ to 1 with $N - 1$ steps of width $1/N$. We approximate it with the integral from 0 to 1:

$$\begin{aligned}
\int_0^1 (1-x)^2/\sqrt{x}\, dx &= \int_0^1 x^{-1/2} - 2x^{1/2} + x^{3/2}\, dx \\
&= \left[2x^{1/2} - \frac{4}{3}x^{3/2} + \frac{2}{5}x^{5/2}\right]_0^1 \\
&= 2 - \frac{4}{3} + \frac{2}{5} = \frac{16}{15} \approx 1.
\end{aligned}$$

Therefore, the expected total number of volume collision pairs is about $N^3/(4\sqrt{\pi\mathsf{n_r}})$. For there to be no collisions, the number of records would need to be $\Omega\big(N^6\big)$. The fraction of volumes that are not unique is about

$$\frac{N^3/(4\sqrt{\pi\mathsf{n_r}})}{N(N+1)/2} = \frac{N^3/\sqrt{\pi\mathsf{n_r}}}{N(N+1)} \approx N/\sqrt{\pi\mathsf{n_r}}.$$

Thus, as foreshadowed in the introduction of this section, the critical quantity for a constant fraction of volumes to be collision-free is $N/\sqrt{\mathsf{n_r}}$: if $\mathsf{n_r}$ is $\Omega(N^2)$, then the ratio of collisions among volumes is nearly 1.

5.3.2 Number of $\mathsf{n_r}$-complemented volumes

Recall that elementary volumes must be $\mathsf{n_r}$-complemented, so the vertices of the initial graph in Step 3 of our attack can be the set $\mathsf{V_{comp}} \coloneqq \{\mathsf{v} \in \mathsf{V} : (\mathsf{n_r} - \mathsf{v}) \in \mathsf{V}\}$. There are $2(N-1)$ ranges that are automatically complemented: the $N-1$ elementary ranges $[1,1],[1,2],\ldots,[1,N-1]$ and the $N-1$ complements $[2,N],\ldots,[N,N]$. In this section, we estimate how many volumes v are complemented "by accident"—not because the complement of their corresponding query is also a range, but simply because $\mathsf{n_r} - \mathsf{v}$ was the volume of another range.

Recall that there are $N+1-d$ ranges of length d, of which 2 are automatically complemented ($[1,d]$ and $[N+1-d,N]$). Let v be the volume of any of the $N-1-d$ other ranges.

Recall that we model the number of records in a range of length d as a Poisson random variable with rate $\mathsf{n_r} \cdot d/N$. Let $\mathsf{X}_d \sim \mathsf{Pois}\,(\mathsf{n_r} \cdot d/N)$ represent the volume of this range. If we consider only accidental collisions that occur from ranges of length $N-d$, a collisions corresponds to $\mathsf{X}_d = \mathsf{n_r} - \mathsf{X}_{N-d}$.

Let $[a, a+d-1]$ be any range of length d. The probability that its volume is $\mathsf{n_r} - k$ is exactly the probability that all k other records have values in $[1,a-1] \cup [a+d,N]$. Since we assume all values are distributed uniformly at random, the probability that k records have values in $[1,a-1] \cup [a+d,N]$ is exactly the same as the probability that k records have values in any *set* of $N-d$ values—they do not need to form an interval. Therefore, $\mathrm{Prob}[\mathsf{X}_d = \mathsf{n_r} - k] = \mathrm{Prob}[\mathsf{X}_{N-d} = k]$. On the other hand, the number of records matched by a range of length $N-d$ is $\mathsf{Pois}\,((N-d)\mathsf{n_r}/N)$. An accidental volume collision between a range of length d and a range of length $N-d$ therefore corresponds to a collision of two samples of X_{N-d}. Again using the fact that a difference of Poisson random variables has a Skellam distribution, we obtain

$$\begin{aligned}\mathrm{Prob}[\mathsf{X}_d = \mathsf{n_r} - \mathsf{X}_{N-d}] &= e^{-2\mathsf{n_r}(N-d)/N} \cdot \mathsf{I}_0(2\mathsf{n_r}(N-d)/N) \\ &\approx 1/2\sqrt{\pi(\mathsf{n_r}(N-d)/N)}.\end{aligned}$$

For each of the $N-1-d$ ranges of length d that is not automatically complemented, there are $d+1$ ranges of length $N-d$ that it could collide with. Considering all range lengths d from 1 to $N-1$ and using similar techniques as in the last section, we

obtain the following estimate of the number of accidentally complemented volumes:

$$\begin{aligned}
\sum_{d=1}^{N-1} \frac{(N-1-d)(d+1)}{2\sqrt{\pi \mathsf{n_r}(N-d)/N}} &\approx \sum_{d=1}^{N-1} \frac{(N-d)d}{2\sqrt{\pi \mathsf{n_r}(N-d)/N}} \\
&= \frac{N^2}{2\sqrt{\pi \mathsf{n_r}}} \sum_{d=1}^{N-1} \frac{(1-d/N)d/N}{\sqrt{1-d/N}} \\
&= \frac{N^3}{2\sqrt{\pi \mathsf{n_r}}} \sum_{d=1}^{N-1} \frac{1}{N} \cdot \sqrt{1-d/N} \cdot d/N \\
&\approx \frac{N^3}{2\sqrt{\pi \mathsf{n_r}}} \int_0^1 \sqrt{1-x} \cdot x \; dx \\
&= \frac{N^3}{2\sqrt{\pi \mathsf{n_r}}} \left[\frac{2(-3x-2)(1-x)^{3/2}}{15} \right]_0^1 \\
&= \frac{N^3}{2\sqrt{\pi \mathsf{n_r}}} \left(0 - \frac{-4}{15} \right) \\
&\approx \frac{N^3}{8\sqrt{\pi \mathsf{n_r}}}.
\end{aligned}$$

Taking into account the $2(N-1)$ volumes that are automatically complemented, we get that the total number of complemented volumes, and hence the initial number of vertices in the graph, is approximately $2N + \frac{N^3}{8\sqrt{\pi \mathsf{n_r}}}$.

5.3.3 Number of edges in the graph

By construction, two vertices are adjacent in the graph iff the absolute difference of their volumes is also a volume. We know that the graph contains two cliques of size N: one whose vertices corresponds to the elementary volumes $[1,1],[1,2],\ldots,[1,N]$ and the other, their complements $[N,N],[N-1,N],\ldots,[1,N]$. Since a clique of size N has $\binom{N}{2}$ edges, these two cliques account for $N(N-1)$ edges in the graph. In this section, we estimate how many edges occur "by accident"—not because the set difference of the two corresponding endpoints is a range, but simply because there exists a range whose volume is the difference of their volumes.

We consider only accidental edges that arise from the difference of volumes of two ranges of lengths d_1 and d_2 equaling the volume of another range of length $|d_2 - d_1|$.

Recall that the number of records having any value in a *set* of d values is distributed exactly like the number of records whose value is in a range of length d; these have a Poisson distribution, $\mathsf{Pois}\,(d\mathsf{n_r}/N)$. Suppose, without loss of generality, that $d_2 > d_1$ and consider $\mathsf{X}_{d_2} - \mathsf{X}_{d_1}$. We can think of it as representing the difference in volume of *any* two ranges of lengths d_2 and d_1. In particular, we could consider a range of

length d_2 that is a superset of the range of length d_1. The volume of the difference of these two ranges is distributed like the volume of a range of length $d_2 - d_1$.

Therefore, we can characterize an accidental edge as being a collision of two identically distributed Poisson random variables with rate $|d_2 - d_1|\,\mathsf{n_r}/N$. Following similar steps as in the previous section to derive the appropriate Skellam distribution, we find that the probability that two vertices corresponding to ranges of length d_1 and d_2 accidentally collide with the volume of a range of length $|d_2 - d_1|$ with probability approximately

$$1/2\sqrt{\pi \mathsf{n_r}\,|d_2 - d_1|\,/N}.$$

For any fixed $d_1 > d_2$, there are $N + 1 - d_2 + d_1$ ranges of length $d_2 - d_1$. Also, recall that the number of ranges of length d whose volumes are complemented (i.e., that are vertices in the graph) is about

$$\frac{(N-d)d}{2\sqrt{\pi \mathsf{n_r}(N-d)/N}} = \frac{d\sqrt{N-d}}{2\sqrt{\pi \mathsf{n_r}/N}}.$$

We use this for d_1 and d_2. Combining all of these pieces, we arrive at the following expression for an estimate of the number of accidental edges:

$$\begin{aligned}
&\sum_{d_1=1}^{N}\sum_{d_2=d_1+1}^{N} \frac{d_1\sqrt{N-d_1}}{2\sqrt{\pi \mathsf{n_r}/N}} \cdot \frac{d_2\sqrt{N-d_2}}{2\sqrt{\pi \mathsf{n_r}/N}} \cdot \frac{N+1-d_2+d_1}{2\sqrt{\pi \mathsf{n_r}(d_2-d_1)/N}} \\
&= \sum_{d_1=1}^{N}\sum_{d_2=d_1+1}^{N} \frac{N^3 \cdot d_1\sqrt{1-d_1/N}\cdot d_2\sqrt{1-d_2/N}}{4\pi \mathsf{n_r}} \cdot \frac{N(1+1/N-(d_2-d_1)/N)}{2\sqrt{\pi \mathsf{n_r}(d_2-d_1)/N}} \\
&= \frac{N^4}{8(\pi \mathsf{n_r})^{3/2}} \sum_{d_1=1}^{N}\sum_{d_2=d_1+1}^{N} d_1\sqrt{1-\frac{d_1}{N}} \cdot d_2\sqrt{1-\frac{d_2}{N}} \cdot \frac{1+1/N-(d_2-d_1)/N}{\sqrt{(d_2-d_1)/N}} \\
&= \frac{N^7}{8(\pi \mathsf{n_r})^{3/2}} \sum_{d_1=1}^{N}\sum_{d_2=d_1+1}^{N} \frac{d_1}{N}\sqrt{1-\frac{d_1}{N}} \cdot \frac{d_2}{N}\sqrt{1-\frac{d_2}{N}} \cdot \frac{1}{N} \cdot \frac{1+1/N-(d_2-d_1)/N}{\sqrt{(d_2-d_1)/N}} \\
&\approx \frac{N^7}{8(\pi \mathsf{n_r})^{3/2}} \sum_{d_1=1}^{N}\sum_{d_2=d_1+1}^{N} \frac{d_1}{N}\sqrt{1-\frac{d_1}{N}} \cdot \frac{d_2}{N}\sqrt{1-\frac{d_2}{N}} \cdot \frac{1}{N} \cdot \frac{1-(d_2-d_1)/N}{\sqrt{(d_2-d_1)/N}} \\
&\approx \frac{N^7}{8(\pi \mathsf{n_r})^{3/2}} \int_0^1\int_{x_1}^1 x_1\sqrt{1-x_1}\cdot x_2\sqrt{1-x_2} \cdot \frac{1-(x_2-x_1)}{\sqrt{x_2-x_1}}\,dx_1\,dx_2 \\
&= \frac{N^7}{8(\pi \mathsf{n_r})^{3/2}} \cdot \frac{43\pi}{1540} \\
&\approx \frac{N^7}{80(\pi \mathsf{n_r})^{3/2}}.
\end{aligned}$$

Hence, counting both the edges arising from $\mathsf{n_r}$-complemented ranges and accidental edges, we get approximately $N^2 + \frac{N^7}{80(\pi \mathsf{n_r})^{3/2}}$.

In the previous section, we saw that the number of vertices is approximately $\mathcal{O}\big(N^3/\sqrt{\mathsf{n_r}}\big)$, so the edge density of the graph is about

$$\frac{N^2 + \frac{N^7}{80(\pi \mathsf{n_r})^{3/2}}}{\binom{2N + \frac{N^3}{8\sqrt{\pi \mathsf{n_r}}}}{2}} \approx \frac{\frac{N^7}{\mathsf{n_r}^{3/2}}}{\left(\frac{N^3}{\sqrt{\mathsf{n_r}}}\right)^2} \approx \frac{N}{\sqrt{\mathsf{n_r}}}.$$

Edge density is relevant to our main algorithm: if it is too high, then there might be other or larger cliques than the ones corresponding to elementary volumes and their complements, generating multiple valid solutions.

Modeling as a random graph. Given the estimated number of vertices and the edge density, we can model the graph as a random graph, adding an edge between each pair of vertices with probability equal to the edge density. This modeling completely disregards the necessary structure and the existence of the two N-cliques, but we can apply results from random graph theory to estimate the maximal size of a clique in such a graph.

A result of Bollobás and Erdös from 1976 [12] states that the maximal clique size of a random graph with n vertices and edge density p is $\Theta\left(-\log n/\log p\right)$. Having a clique number greater than N would therefore require that the edge density p satisfy $\log p > -\log(n)/N$. Since $e^{-x} \approx 1 - x$ (q.v. Bound A.7), we get $p > 1 - \log(n)/N$. In our graph, the number of nodes n satisfies $\log n \approx 3\log(N) - \log\sqrt{\mathsf{n_r}}$ and the edge density p satisfies $\log p \approx \log(N) - \log\sqrt{\mathsf{n_r}}$. Therefore, we expect a clique of size larger than N when

$$\begin{aligned}(\log(N) - \log\sqrt{\mathsf{n_r}}) &> 1 - ((3\log(N) - \log\sqrt{\mathsf{n_r}})/N)\\ \log(N)(1 + 3/N) &> 1 + \log(\sqrt{\mathsf{n_r}})(1 + 1/N),\end{aligned}$$

or when, approximately, $\log(N) - 1 > \log(\sqrt{\mathsf{n_r}})$, i.e., $\mathsf{n_r}$ is $\mathcal{O}\big(N^2\big)$. Therefore, to ensure that no clique of size N exists by accident due to edge density, we would require $\mathsf{n_r}$ to be $\Omega\big(N^2\big)$.

5.3.4 Experimental validation

In the previous sections, we derived estimates for the number of distinct volumes, the number of complementary volumes (vertices in the initial graph), and the number of edges in this graph for uniformly distributed values in terms of $\mathsf{n_r}$, the total number of records, and N, the number of possible values. We now present the results of experiments that evaluated the accuracy of these estimates (Table 5.1).

	$\lvert\mathsf{V}\rvert$		$\lvert\mathsf{V}_{\mathsf{comp}}\rvert$		Number of edges	
$\mathsf{n_r}$	Observed	Computed	Observed	Computed	Observed	Computed
			$N = 50$			
1250	710	776	375	349	52381	42183
2500	907	922	313	276	26787	16530
5000	1034	1025	230	224	10949	7460
10000	1098	1098	179	188	5966	4253
			$N = 100$			
5000	2803	3055	1406	1197	730625	644936
10000	3553	3639	1116	905	323943	234483
20000	3979	4052	784	698	120779	89367
40000	4291	4344	607	552	60511	38060
			$N = 200$			
20000	11061	12121	5344	4389	10448021	10198981
40000	13885	14458	4144	3220	4465672	3631742
80000	15927	16110	2793	2394	1376980	1309872
160000	17158	17279	1836	1810	458622	488967

Table 5.1: Experimental evaluation of the Poisson model. $|\mathsf{V}|$ is the number of distinct volumes, estimated to be $N(N+1)/2 - N^3/(4\sqrt{\pi \mathsf{n_r}})$. $|\mathsf{V}_{\mathsf{comp}}|$ is the number of volumes with $\mathsf{n_r}$-complements, estimated to be $2N + N^3/(8\sqrt{\pi \mathsf{n_r}})$. The number of edges was estimated to be $N^2 + N^7/(80(\pi \mathsf{n_r})^{3/2})$.

For various combinations of N and $\mathsf{n_r}$, we present the average (over 30 different databases) of the number of distinct volumes, number of complementary volumes, and number of edges. We compare these values to the computed estimates from the previous sections.

We chose three different values of N—50, 100, and 200. For each, we tested $\mathsf{n_r} = N^2/2, N^2, 2N^2$, and $4N^2$. We found that our estimates are reasonably accurate, especially for the number of distinct volumes. The computed number of edges had the highest relative error, being sometimes 40% lower than the experimental value. Part of the reason could be that our estimate for the number of edges relied on estimates for the number of complemented volumes of particular lengths, and these were often too low in our experiments as well.

5.4 Conclusions

Volume leakage is hard to hide from passive network adversaries. In the context of encrypted databases, packet sizes could leak how many records matched a query. In this chapter, we showed how an adversary can exploit this leakage with fewer queries than the best previous attack [42].

We developed a practical attack that uses the complete set of query volumes to identify elementary volumes and then reconstruct all counts. Our attack works better when there are many records relative to the number of possible values N. The query distribution does not matter as long as all queries are observed eventually. This is probably the main limitation to using this attack in practice: the volumes of *all* queries must be observed. In some settings, it might be impractical to observe all volumes before the values change or new records are inserted.

Some specific questions remain surrounding our volume leakage attack, which is the most heuristic attack in this thesis. Is it possible (and if so, how) to characterize the volume graphs for which there is not a unique clique of elementary volumes? Is it possible to apply other techniques from graph theory to delay generic clique-finding, or avoid it entirely? Future research could investigate whether it is possible to make this attack robust to a few missing volumes, or whether it could detect inconsistencies that indicate changes were made to the database.

Chapter 6

Conclusions

Encrypted search schemes are an important component of a defense-in-depth approach to database security: they allow storage providers to process queries on data without decrypting it. The main contributions of this thesis are discovering new trade-offs between security, functionality, and efficiency in encrypted search schemes. In this chapter, I reflect on a few salient themes and discuss future work in the area of encrypted search schemes.

6.1 Reflections

One of the themes of this thesis is that an adversary can learn a lot from a little bit of leakage. It was initially surprising, at least to me, that an adversary can reconstruct element counts in a database by observing only how many records match each query. Another surprise was how soon ϵ-approximate ordered reconstruction succeeded with relatively few queries in our experiments.

Second, the implications of leakage of new encrypted search schemes must be thoroughly explored before they are deployed. Although many schemes supporting range queries leak rank, our work was the first to consider attacks on rank leakage and we were able to reconstruct records' values.

Third, experimenting with real datasets yields meaningful insights. For instance, despite our frequency-smoothing encryption schemes from Chapter 2 requiring high encoding lengths for typical cryptographic levels of security, experiments showed that even small encoding lengths were able to limit an optimal adversary's success to the success of random guessing.

6.2 Future directions

There are many problems left to solve in the area of encrypted search schemes. The work in this thesis considered only two types of queries on databases—equality queries and range queries—but real-world databases support many more. For each type of query, researchers could create new schemes, evaluate countermeasures to existing attacks, or design new attacks.

Recent research has begun to explore attacks on different query types. For example, Kornaropoulos *et al.* exploit leakage from k-nearest neighbor queries [44], and Akshima *et al.* reconstruct data from multi-dimensional range queries [3]. I hope to see future work on leakage from pattern-matching queries on text data. It would be particularly gratifying to see work adapt our query analysis techniques based on statistical learning theory to establish bounds for different query classes.

One of the major limitations of work in this area is the lack of publicly available query distributions. For the analysis of our attacks, we had to assume a particular query distribution. We chose the uniform, but not because this distribution is thought to occur in practice—because previous work used it and we wanted our work to be comparable. If real query information were available, analyses would be much more relevant to practice (and the information might inspire new encrypted search constructions). Relatedly, more research could be done on the design and evaluation of countermeasures to attacks that exploit properties of particular query distributions. Query information could include, for example, the proportion of database interactions that are read-only, as opposed to inserting new records or modifying existing records. I believe stronger interaction with industry will be required if secure encrypted search schemes are ever to be widely deployed.

Appendix A

Formulae and Bounds

Approximation A.1 (Limiting form of the modified Bessel function of the first kind)**.** *[20, Eq. 10.30.4] When the order α is fixed, and as the argument x tends towards infinity, the modified Bessel function of the first kind $\mathsf{I}_\alpha(x)$ tends to $e^x/\sqrt{2\pi x}$.*

Bound A.2. *For any positive integer n, the nth harmonic number H_n satisfies*

$$\log n + 1/n \leq \mathsf{H}_n \leq \log n + 1.$$

Bound A.3. *For any positive integers m and n with $m > n$, the difference of H_m and H_n satisfies*

$$\mathsf{H}_m - \mathsf{H}_n \leq \log(m/n).$$

Proof. The difference $\mathsf{H}_m - \mathsf{H}_n$ equals the right Riemann sum for the area under $1/x$ from n to m with $m - n$ steps:

$$\sum_{j=1}^{m-n} 1 \cdot \frac{1}{n+j} = \mathsf{H}_m - \mathsf{H}_n.$$

Since $1/x$ is decreasing in this interval, the right Riemann sum is an underestimation of the area, thus

$$\mathsf{H}_m - \mathsf{H}_n \leq \int_n^m 1/x \, dx = \log(m) - \log(n) = \log(m/n). \qquad \square$$

Formula A.4. *For any discrete random variable* X *whose support is the non-negative integers, its expected value* $\mathbb{E}[\mathsf{X}]$ *equals* $\sum_{i=1}^{\infty} \text{Prob}[\mathsf{X} \geq i]$.

Proof. Starting with the definition of expected value for a discrete random variable, we have

$$\mathbb{E}[\mathsf{X}] \coloneqq \sum_{y=1}^{\infty} y \cdot \text{Prob}[\mathsf{X} = y] = \sum_{y=1}^{\infty} \sum_{x=1}^{y} \text{Prob}[\mathsf{X} = y] = \sum_{x=1}^{\infty} \sum_{y=x}^{\infty} \text{Prob}[\mathsf{X} = y]$$
$$= \sum_{x=1}^{\infty} \text{Prob}[\mathsf{X} \geq x]\,. \qquad \square$$

Formula A.5. *For any constant* $c \in \mathbb{R}$ *and ratio* $r \in \mathbb{R}$ *with* $|r| < 1$, $\sum_{x=0}^{\infty} c \cdot r^x = \frac{c}{1-r}$.

Formula A.6. *For any constant* $c \in \mathbb{R}$, *ratio* $r \in \mathbb{R}$ *with* $|r| < 1$, *and integer* $a \geq 0$,

$$\sum_{x=a}^{\infty} c \cdot r^x = \frac{c \cdot r^a}{1-r}.$$

Bound A.7. *For any* $x \in \mathbb{R}$, $e^x \geq x + 1$.

Proof. By definition, $\frac{d}{dx} e^x = e^x > 0$, so e^x is a convex function (concave upwards). It is therefore lower-bounded by any of its tangent lines, i.e., any line with slope $e^{x'}$ that goes through the point $(x', e^{x'})$. In particular, the tangent line at $x = 0$ has slope 1 and goes through point (0,1), thus we conclude that the line $x + 1$ is below e^x across its entire domain. $\square$

Bound A.8. *For any real* $x > 0$, $\log x \leq x - 1$.

Proof. Substitute $\log x$ for x in Bound A.7 to obtain $x \geq \log x + 1$. $\square$

Bound A.9. *For any non-zero* $x \in \mathbb{R}$, $x \leq \frac{1}{1-e^{-1/x}}$.

Proof. Substitute $-1/x$ for x in Bound A.7 to obtain $e^{-1/x} \geq -1/x + 1$. Negating both sides and adding 1 yields $1 - e^{-1/x} \leq 1/x$. The sides of this inequality are either both positive or both negative, so taking reciprocals yields $\frac{1}{1-e^{-1/x}} \geq x$. $\square$

Bound A.10. *For any non-zero* $x \in \mathbb{R}$, $\frac{1}{1-e^{-1/x}} \leq x + 1$.

Proof. Substitute $1/x$ for x in Bound A.7 to obtain $e^{1/x} \geq 1/x + 1$. Since $e^{1/x}$ is always positive and $1/x + 1$ is negative for $x \in (-1, 0)$, taking reciprocals yields

$$\begin{cases} e^{-1/x} \leq \frac{1}{1/x+1} & \text{for } x \in (-\infty, -1] \cup (0, \infty), \\ e^{-1/x} \geq \frac{1}{1/x+1} & \text{for } x \in (-1, 0). \end{cases}$$

In the first case, negating both sides and adding 1 yields $1 - e^{-1/x} \geq \frac{1}{x+1}$. When $x < -1$, both sides are negative, and when $x > 0$, both are positive. Thus, $\frac{1}{1-e^{-1/x}} \leq x + 1$ for $x \in (-\infty, -1] \cup (0, \infty)$. In the second case, negating both sides and adding 1 yields $1 - e^{-1/x} \leq \frac{1}{x+1}$, where the left-hand side is negative and the right-hand side is positive for $x \in (-1, 0)$. Thus, $\frac{1}{1-e^{-1/x}} \leq x + 1$ for $x \in (-1, 0)$ as well. □

Index of Notation

$x \xleftarrow{\mathsf{D}} \mathcal{S}$ The element x is assigned an element from the set $\mathcal{S}$ according to the distribution D.

$x \xleftarrow{\$} \mathcal{S}$ The element x is assigned a uniformly random element from the set $\mathcal{S}$; equivalent to $x \xleftarrow{\mathsf{U}} \mathcal{S}$.

$\lfloor x \rceil$ The integer nearest to x. When the fractional part of x is 0.5, it is always rounded up.

$2^{\mathcal{S}}$ The power set of the elements $\mathcal{S}$; the set of all possible subsets of $\mathcal{S}$.

$[a, b]$ A query with left endpoint a and right endpoint b. Equal to the range $\{a, \ldots, b\}$.

A An adversary in a security game.

α A bound on an adversary's advantage.

$\mathsf{I}_\alpha(x)$ Modified bessel function of the first kind. One of the solutions to the differential equation $0 = x^2 \frac{d^2y}{dx^2} + x\frac{dy}{dx} - x^2y - \alpha^2 y$. When the order α is 0, equal to $\sum_{k=0}^{\infty} \frac{(x^2/4)^k}{k!\ k!}$.

$\mathsf{Bin}\,(n, p)$ Binomial distribution with success probability p. Number of successes in n trials. $\text{Prob}[X = x] = \binom{n}{x} p^x (1-p)^{n-x}$, $\mathbb{E}[X] = np$.

$x \parallel y$ The concatenation of two strings.

Δ A distribution adaptation parameter indicating how different an estimated distribution $\tilde{\mathsf{D}}$ may be from the actual, unknown distribution D; input to an adaptive FSE scheme's Setup algorithm.

$\mathcal{D}_S$ The space of all possible probability distributions over a set S; the subscript may be removed if the set is clear from context.

D A distinguisher in a security game.

DB A set of records and their values; a table with one attribute.

D A statistical distribution, defined by a pmf f or cdf F, sometimes written with its domain as a subscript, e.g, $\mathsf{D}_{\mathcal{M}}$.

$\mathcal{E}$ The encoding space of an HE scheme; all members are fixed-length in this thesis.

F Cumulative distribution function; $\mathsf{F}(x) \coloneqq \text{Prob}[\mathsf{X} \leq x]$.

f Probability mass function; $\mathsf{f}(x) \coloneqq \text{Prob}[\mathsf{X} = x]$.

$\mathsf{Geo}\,(p)$ Geometric distribution with success probability p. Number of trials required to get 1 success.

$H(\mathsf{D})$ The Shannon entropy of the distribution D with pmf f, equal to $H(\mathsf{D}) = -\sum_{m \in \mathsf{supp}(\mathsf{D})} \mathsf{f}(m) \cdot \log \mathsf{f}(m)$, measured in nats.

$\mathcal{H}^{\mathsf{FSE}}_{\mathsf{sk},\mathsf{s}}$ The set of FSE homophones of all possible messages under key sk for any state s attainable from an initial state s_0 output by the FSE scheme's Setup algorithm.

$\mathcal{H}^{\mathsf{FSE}}_{\mathsf{sk},\mathsf{s}}(m)$ The set of FSE homophones of one message m under key sk for any state s attainable from an initial state s_0 output by the FSE scheme's Setup algorithm.

$\mathcal{H}^{\mathsf{HE}}_{\mathsf{s}}$ The set of HE homophones of all possible messages for any state s attainable from an initial state s_0 output by the HE scheme's Setup algorithm.

$\mathcal{H}^{\mathsf{HE}}_{\mathsf{s}}(m)$ The set of HE homophones of one message m for any state s attainable from an initial state s_0 output by the HE scheme's Setup algorithm.

H_n The nth harmonic number; $\mathsf{H}_n \coloneqq \sum_{i=1}^{n} 1/i$.

$\mathrm{KL}\,(\mathsf{D}_0, \mathsf{D}_1)$ The Kullback–Leibler divergence of distribution D_0 with respect to D_1, where both distributions have the same support. Equal to $\sum_{x \in \mathsf{supp}(\mathsf{D}_0)} \mathsf{f}_{\mathsf{D}_0}(x) \cdot \log \frac{\mathsf{f}_{\mathsf{D}_0}(x)}{\mathsf{f}_{\mathsf{D}_1}(x)}$.

log The natural logarithm.

$\mathcal{M}$ The message space of an encryption or encoding scheme.

$\mathsf{NegBin}\,(k, p)$ Negative binomial distribution with success probability p. Number of trials required to get k successes. $\text{Prob}[X = x] = \binom{x-1}{k-1} p^k (1-p)^{x-k}$, $\mathbb{E}[X] = k/p$.

N The number of distinct values an attribute can take. Values are assumed to be $\{1, \ldots, N\}$.

$\mathsf{n_q}$ The total number of queries issued, counting repetition.

$\mathsf{n_r}$ The number of records in a database.

$\mathsf{Pois}(\lambda)$ Poisson distribution with rate λ. Number of times an event occurs in an interval. $\text{Prob}[X = x] = \frac{\lambda^x}{x!e^x}$, $\mathbb{E}[X] = \lambda$.

Φ The cdf of the standard normal distribution.

$\mathcal{R}$ The set of all records in a database or just their identifiers.

R A subset of records or record identifiers, e.g., those matching a range query.

r A record or record identifier.

rank (of a record or value) The number of records with value less than or equal to this (record's) value, $\mathsf{rank}(0) \coloneqq 0$ by definition.

$\mathsf{Skellam}(\lambda, x)$ Skellam distribution with rates λ_1 and λ_2. Difference between two Poisson random variables with rates λ_1 and λ_2. $\text{Prob}[X = x] = e^{-(\lambda_1+\lambda_2)}\left(\frac{\lambda_1}{\lambda_2}\right)^{x/2} \cdot \mathsf{I}_{\mathrm{x}}\left(2\sqrt{\lambda_1\lambda_2}\right)$,.

λ A security parameter.

$\mathsf{supp}(\mathsf{D})$ (of a distribution) The set of elements in the distribution's domain for which the pmf f is strictly greater than 0.

$\mathsf{Trunc}(x, n)$ Truncating the string x to a length of n bits, removing the bits from the right.

val The value of a record, in the range $\{1, \ldots, N\}$.

(x, y) The two rank values corresponding to a query $[a, b]$. $x \coloneqq \mathsf{rank}(a - 1)$ and $y \coloneqq \mathsf{rank}(b)$.

References

[1] Agency for Healthcare Research and Quality, Rockville, MD. HCUP Nationwide Inpatient Sample (NIS), Healthcare Cost and Utilization Project (HCUP), 2009–2019. http://www.hcup-us.ahrq.gov/nisoverview.jsp.

[2] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 563–574, New York, NY, USA, 2004. ACM.

[3] Akshima, D. Cash, F. Falzon, A. Rivkin, and J. Stern. Multidimensional database reconstruction from range query access patterns. Cryptology ePrint Archive, Report 2020/296, 2020. https://eprint.iacr.org/2020/296.

[4] 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*, pages 435–446, April 2015.

[5] T. Baignères, P. Junod, and S. Vaudenay. How far can we go beyond linear cryptanalysis? In P. J. Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 432–450. Springer, Heidelberg, Dec. 2004.

[6] M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers. Format-preserving encryption. In M. J. Jacobson Jr., V. Rijmen, and R. Safavi-Naini, editors, *SAC 2009*, volume 5867 of *LNCS*, pages 295–312. Springer, Heidelberg, Aug. 2009.

[7] V. Bindschaedler, P. Grubbs, D. Cash, T. Ristenpart, and V. Shmatikov. The tao of inference in privacy-protected databases. *Proc. VLDB Endow.*, 11(11):1715–1728, July 2018.

[8] J. Black and P. Rogaway. Ciphers with arbitrary finite domains. In B. Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 114–130. Springer, Heidelberg, Feb. 2002.

[9] T. Boelter, R. Poddar, and R. A. Popa. A secure one-roundtrip index for range queries. Cryptology ePrint Archive, Report 2016/568, 2016. http://eprint.iacr.org/2016/568.

[10] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving symmetric encryption. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 224–241. Springer, Heidelberg, Apr. 2009.

[11] A. Boldyreva, N. Chenette, and A. O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 578–595. Springer, Heidelberg, Aug. 2011.

[12] B. Bollobás. Random graphs. In *Modern graph theory*, volume 184 of *Graduate Texts in Mathematics*, pages 215–252. Springer-Verlag New York, 1998.

[13] 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 E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 563–594. Springer, Heidelberg, Apr. 2015.

[14] K. S. Booth and G. 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.

[15] C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Communications of the ACM*, 16(9), September 1973.

[16] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. Leakage-abuse attacks against searchable encryption. In I. Ray, N. Li, and C. Kruegel, editors, *ACM CCS 2015*, pages 668–679. ACM Press, Oct. 2015.

[17] N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu. Practical order-revealing encryption with limited leakage. In T. Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 474–493. Springer, Heidelberg, Mar. 2016.

[18] CipherCloud. CipherCloud Information Protection Overview v2.5, June 2013. http://pages.ciphercloud.com/rs/ciphercloud/images/CipherCloud%20Technical%20Whitepaper%20v2.5.pdf.

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

[20] NIST Digital Library of Mathematical Functions. Release 1.0.24 of 2019-09-15. F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain, eds. http://dlmf.nist.gov.

[21] F. B. Durak, T. M. DuBuisson, and D. Cash. What else is revealed by order-revealing encryption? In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 2016*, pages 1155–1166. ACM Press, Oct. 2016.

[22] A. Ehrenfeucht, D. Haussler, M. Kearns, and L. Valiant. A general lower bound on the number of examples needed for learning. *Inf. Comput.*, 82(3):247–261, Sept. 1989.

[23] P. Flajolet, D. Gardy, and L. Thimonier. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Applied Mathematics*, 39:207–229, 1992.

[24] 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*, pages 172–191. IEEE Computer Society Press, May 2017.

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

[26] P. Grubbs, M.-S. Lacharité, B. Minaud, and K. Paterson. Pump up the volume: Practical database reconstruction from volume leakage on range queries. Cryptology ePrint Archive, Report 2018/965, 2018. https://eprint.iacr.org/2018/965.

[27] P. Grubbs, M.-S. Lacharité, B. Minaud, and K. G. Paterson. Pump up the volume: Practical database reconstruction from volume leakage on range queries. In D. Lie, M. Mannan, M. Backes, and X. Wang, editors, *ACM CCS 2018*, pages 315–331. ACM Press, Oct. 2018.

[28] P. Grubbs, M.-S. Lacharité, B. Minaud, and K. G. Paterson. Learning to reconstruct: Statistical learning theory and encrypted database attacks. In *2019 IEEE Symposium on Security and Privacy*, pages 1067–1083. IEEE Computer Society Press, May 2019.

[29] P. Grubbs, M.-S. Lacharité, B. Minaud, and K. G. Paterson. Learning to reconstruct: Statistical learning theory and encrypted database attacks. Cryptology ePrint Archive, Report 2019/011, 2019. https://eprint.iacr.org/2019/011.

[30] P. Grubbs, R. McPherson, M. Naveed, T. Ristenpart, and V. Shmatikov. Breaking web applications built on top of encrypted data. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 2016*, pages 1353–1364. ACM Press, Oct. 2016.

[31] P. Grubbs, T. Ristenpart, and V. Shmatikov. Why your encrypted database is not secure. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*, HotOS '17, pages 162–168, New York, NY, USA, 2017. ACM.

[32] 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*, pages 655–672. IEEE Computer Society Press, May 2017.

[33] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, 2008.

[34] D. Haussler and E. 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.

[35] V. T. Hoang, T. Krovetz, and P. Rogaway. Robust authenticated-encryption AEZ and the problem that it solves. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 15–44. Springer, Heidelberg, Apr. 2015.

[36] T. Hunt. Have i been pwned (hibp), 2019. https://haveibeenpwned.com.

[37] IBM Knowledge Center. Db2 native encryption. Db2 11.5, 2019. https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.5.0/com.ibm.db2.luw.admin.sec.doc/doc/c0061758.html.

[38] IETF TLS Working Group. Tls, 2019. https://tlswg.org/.

[39] M. S. Islam, M. Kuzu, and M. Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *NDSS 2012*. The Internet Society, Feb. 2012.

[40] A. Juels and T. Ristenpart. Honey encryption: Security beyond the brute-force bound. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 293–310. Springer, Heidelberg, May 2014.

[41] S. Kamara and T. Moataz. SQL on structurally-encrypted databases. In T. Peyrin and S. Galbraith, editors, *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*, pages 149–180. Springer, Heidelberg, Dec. 2018.

[42] G. Kellaris, G. Kollios, K. Nissim, and A. O'Neill. Generic attacks on secure outsourced databases. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 2016*, pages 1329–1340. ACM Press, Oct. 2016.

[43] F. Kerschbaum. Frequency-hiding order-preserving encryption. In I. Ray, N. Li, and C. Kruegel, editors, *ACM CCS 2015*, pages 656–667. ACM Press, Oct. 2015.

[44] E. M. Kornaropoulos, C. Papamanthou, and R. Tamassia. Data recovery on encrypted databases with k-nearest neighbor query leakage. In *2019 IEEE Symposium on Security and Privacy*, pages 1033–1050. IEEE Computer Society Press, May 2019.

[45] M. R. Kosorok. *Introduction to empirical processes and semiparametric inference*. Springer, New York; London, 2008.

[46] M.-S. Lacharité. Breaking encrypted databases: Generic attacks on range queries. Technical report, Black Hat USA, 2019. http://i.blackhat.com/USA-19/Thursday/us-19-Lacharite-Breaking-Encrypted-Databases-Generic-Attacks-On-Range-Queries-wp.pdf.

[47] M.-S. Lacharité, B. Minaud, and K. G. Paterson. Improved reconstruction attacks on encrypted data using range query leakage. Cryptology ePrint Archive, Report 2017/701, 2017. http://eprint.iacr.org/2017/701.

[48] M.-S. Lacharité, B. Minaud, and K. G. Paterson. Improved reconstruction attacks on encrypted data using range query leakage. In *2018 IEEE Symposium on Security and Privacy*, pages 297–314. IEEE Computer Society Press, May 2018.

[49] M.-S. Lacharité and K. G. Paterson. Frequency-smoothing encryption: preventing snapshot attacks on deterministically encrypted data. Cryptology ePrint Archive, Report 2017/1068, 2017. https://eprint.iacr.org/2017/1068.

[50] M.-S. Lacharité and K. G. Paterson. Frequency-smoothing encryption: preventing snapshot attacks on deterministically encrypted data. *IACR Trans. Symm. Cryptol.*, 2018(1):277–313, 2018.

[51] K. Lewi and D. J. Wu. Order-revealing encryption: New constructions, applications, and lower bounds. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C.

Myers, and S. Halevi, editors, *ACM CCS 2016*, pages 1167–1178. ACM Press, Oct. 2016.

[52] M. Luby. A simple parallel algorithm for the maximal independent set problem. In *17th ACM STOC*, pages 1–10. ACM Press, May 1985.

[53] M. Maffei, M. Reinert, and D. Schröder. On the security of frequency-hiding order-preserving encryption. In S. Capkun and S. S. M. Chow, editors, *CANS 17*, volume 11261 of *LNCS*, pages 51–70. Springer, Heidelberg, Nov. / Dec. 2017.

[54] Microsoft SQL Docs. Always Encrypted (database engine), 2017. https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/always-encrypted-database-engine?view=sql-server-2017.

[55] Microsoft SQL Docs. Transparent Data Encryption (TDE), 2019. https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/transparent-data-encryption.

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

[57] S. S. Moghadam, G. Gavint, and J. Darmonti. A secure order-preserving indexing scheme for outsourced data. In *2016 IEEE International Carnahan Conference on Security Technology (ICCST)*, pages 1–7, Oct 2016.

[58] MongoDB. Client-Side Field Level Encryption. MongoDB Manual 4.2, June 2019. https://docs.mongodb.com/manual/core/security-client-side-encryption/.

[59] MongoDB. Encryption at Rest. MongoDB Manual 4.2, 06 2019. https://docs.mongodb.com/manual/core/security-encryption-at-rest/.

[60] J. W. Moon and L. Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3(1):23–28, 1965.

[61] B. Morris, P. Rogaway, and T. Stegers. How to encipher messages on a small domain. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 286–302. Springer, Heidelberg, Aug. 2009.

[62] M. Naveed, S. Kamara, and C. V. Wright. Inference attacks on property-preserving encrypted databases. In I. Ray, N. Li, and C. Kruegel, editors, *ACM CCS 2015*, pages 644–655. ACM Press, Oct. 2015.

[63] Oracle. MySQL 5.7 reference manual, 2017. https://dev.mysql.com/doc/refman/5.7/en/storage-requirements.html.

[64] Oracle. Using Transparent Data Encryption. In *Oracle Database Online Documentation Library, 12c Release 1 (12.1.0.2)*. 2019. https://docs.oracle.com/database/121/ASOAG/asopart1.htm.

[65] A. Papadimitriou, R. Bhagwan, N. Chandran, R. Ramjee, A. Haeberlen, H. Singh, A. Modi, and S. Badrinarayanan. Big data analytics over encrypted datasets with Seabed. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 587–602, GA, 2016. USENIX Association.

[66] T. P. Peixoto. The graph-tool python library. 2014. http://figshare.com/articles/graph_tool/1164194.

[67] 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.

[68] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. Cryptdb: Protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 85–100, New York, NY, USA, 2011. ACM.

[69] D. Pouliot, S. Griffy, and C. V. Wright. The strength of weak randomization: Easily deployable, efficiently searchable encryption with minimal leakage. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 517–529, June 2019.

[70] D. Pouliot and C. V. Wright. The shadow nemesis: Inference attacks on efficiently deployable, efficiently searchable encryption. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 2016*, pages 1341–1352. ACM Press, Oct. 2016.

[71] T. Ristenpart and S. Yilek. The mix-and-cut shuffle: Small-domain encryption secure against N queries. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 392–409. Springer, Heidelberg, Aug. 2013.

[72] D. S. Roche, D. Apon, S. G. Choi, and A. Yerukhimovich. POPE: Partial order preserving encoding. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 2016*, pages 1131–1142. ACM Press, Oct. 2016.

[73] P. Rogaway. Nonce-based symmetric encryption. In B. K. Roy and W. Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 348–359. Springer, Heidelberg, Feb. 2004.

[74] P. Rogaway and T. Shrimpton. A provable-security treatment of the key-wrap problem. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, Heidelberg, May / June 2006.

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

[76] R. Schuster, V. Shmatikov, and E. Tromer. Beauty and the burst: Remote identification of encrypted video streams. In E. Kirda and T. Ristenpart, editors, *USENIX Security 2017*, pages 1357–1374. USENIX Association, Aug. 2017.

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

[78] Wikipedia contributors. List of data breaches — Wikipedia, the free encyclopedia, 2019. https://en.wikipedia.org/w/index.php?title=List_of_data_breaches&oldid=910064898.

[79] Y. Zhang, J. Katz, and C. Papamanthou. All your queries are belong to us: The power of file-injection attacks on searchable encryption. In T. Holz and S. Savage, editors, *USENIX Security 2016*, pages 707–720. USENIX Association, Aug. 2016.

[80] G. K. Zipf. *The psycho-biology of language: an introduction to dynamic philology.* Houghton Mifflin, 1935.