Evaluating Leakage Attacks Against Relational Encrypted Search

Patrick Ehrler
Technical University of Darmstadt
Darmstadt, Germany
patrickehrler@gmail.com

Abdelkarim Kati
University of Waterloo
Waterloo, ON, Canada
akati@uwaterloo.ca

Thomas Schneider
Technical University of Darmstadt
Darmstadt, Germany
schneider@encrypto.cs.tu-darmstadt.de

Amos Treiber
Technical University of Darmstadt
Darmstadt, Germany
treiber@encrypto.cs.tu-darmstadt.de

Abstract

Encrypted Search Algorithms (ESAs) are a technique to encrypt data while the user can still search over it. ESAs can protect privacy and ensure security of sensitive data stored on a remote storage. Originally, ESAs were used in the context of documents that consist of keywords. The user encrypts the documents, sends them to a remote server and is still able to search for keywords, without exposing information about the plaintext. The idea of ESAs has also been applied to relational databases, where queries (similar to SQL statements) can be privately executed on an encrypted database. But just as traditional schemes for Keyword-ESAs, also Relational-ESAs have the drawback of exposing some information, called leakage. Leakage attacks have been proposed in the literature that use this information together with auxiliary information to learn details about the plaintext. However, these leakage attacks have overwhelmingly been designed for and applied to Keyword-ESAs and not Relational-ESAs.

In this work, we review the suitability of major leakage attacks against ESAs in the relational setting by adapting them accordingly. We perform extensive re-evaluations of the attacks on various relational datasets with different properties.

Our evaluations show that major attacks can work against Relational-ESAs in the known-data setting. However, the attack performance differs between datasets, exploited patterns, and attacks.

CCS Concepts

• **Security and privacy → Cryptanalysis and other attacks**; **Management and querying of encrypted data**; Privacy-preserving protocols.

Keywords

Encrypted Search, Cryptanalysis, Leakage Attacks

ACM Reference Format:
Patrick Ehrler, Abdelkarim Kati, Thomas Schneider, and Amos Treiber. 2024. Evaluating Leakage Attacks Against Relational Encrypted Search. In *Proceedings of the 2024 Cloud Computing Security Workshop (CCSW '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3689938.3694776

1 Introduction

Data is often not stored locally anymore, but on remote storage. Therefore, it is more important than ever to keep data secure against breaches by keeping it private and inaccessible even to cloud providers.

Encrypted Search Algorithms (ESAs). The above observation is a natural motivation for ESAs. They encrypt a data collection such that the data can remain encrypted on the remote storage whilst the user is still able to search over it. The concept was originally designed for documents that consist of keywords [31]. We refer to the survey of Fuller et al. [11] for more information on ESAs.

Leakage and Leakage Attacks. All efficient schemes of ESAs have the drawback of exposing some information (so called *leakage*) [9]. For instance, this may entail the number of documents that match a search query.

The leakage can be used to uncover plaintext information, e.g., uncovering what the user was searching for. The process of using leakage to uncover plaintext information is called a leakage attack [17]. There are many attacks published in the literature, which have been surveyed by Kamara et al. [19]. Attack success depends on a plethora of factors, which include the type of leakage pattern they exploit, the data they are evaluated on, the auxiliary information the attacker has access to, and the adversary model.

In this work, we focus on *known-data* attacks exploiting leakage patterns associated with *passive, persistent* adversaries, which is the most common adversary model in the literature. A known-data attack assumes that the attacker has access to a subset of the plaintext data collection. A passive adversary can observe query operations, but does not interact with the system. A persistent adversary has access to the encrypted data collection and to the transcript of all interactions between the user and the server.

Relational-ESAs. The idea of ESAs was extended to relational databases [6, 20, 21]. Instead of searching for keywords in encrypted documents, queries are executed on remote encrypted databases. In this work, we focus on selection-queries with a single equality-based condition. Such a query is equivalent to this SQL statement:

CCSW '24, October 14–18, 2024, Salt Lake City, UT, USA

SELECT $*$ FROM T WHERE *attribute* = *value*.

If executed on a database, this query returns all rows of the table T where the value of the field *attribute* equals *value*. Just as in the non-relational case, Relational-ESAs also leak some information.

Leakage Attacks Against Relational-ESAs. Passive, persistent attacks against Relational-ESAs based on the Keyword-ESA attack of Cash et al. [5] have so far been proposed by Abdelraheem et al. [1, 2]. These attacks entail the assumption that the attacker has *full data knowledge* and they were evaluated on only three datasets (one banking and two census datasets).

Recently, Gui et al. [13] proposed two passive relational attacks without full data knowledge. They target a preview version of the ESA product MongoDB queryable encryption (QE) [23, 24] released for customer feedback. The considered adversary is not persistent but a *snapshot attacker* that receives a copy of the server state. Additionally, the attacker has knowledge of an auxiliary database. The authors use census data to evaluate their attacks.

While these specific attacks have been leveraged for certain instances of Relational-ESAs, it remains open how to utilize the existing plethora of leakage attacks against Keyword-ESAs [4, 5, 10, 28, 29] for attacking Relational-ESAs and to investigate which evaluation factors influence their attack performance.

Therefore, we believe a systematic study of attack performance in the relational case across multiple databases is necessary to form a more complete picture for the relational case.

Our Contributions. In this work, we provide the systematic study mentioned above in the common passive, persistent, known-data attacker model. We investigate how major, appropriate leakage attacks against encrypted keyword search perform when applied to the relational case by making the following contributions:

(1) We look at major attacks against Keyword-ESAs [4, 5, 10, 28, 29], review their suitability as attacks against Relational-ESAs, and propose adaptions to make them work in the relational setting.
(2) We extend the open-source LEAKER framework [19], that so far was restricted to evaluating leakage attacks against encrypted keyword and range search on arbitrary data, to also support Relational-ESAs, including our attack adaptions. We will make our extensions publicly available as open-source for future research.
(3) We identify new relational datasets for our evaluations that entail various dataset properties and domains, including census, medical, and voting data. These data sources are publicly available and may function as common means to evaluate future relational leakage attacks.
(4) We conduct a systematic re-evaluation of our adapted attacks on our uncovered relational datasets to identify dataset properties where known-data attacks against Relational-ESAs perform well. This includes an in-depth look into the uncovered queries to determine how sensitive the attack results are, which, so far, has not been performed yet. Our results show that these attacks also apply to the relational setting and we identify attack and database properties that influence attack efficacy.

Concurrent Work. Concurrently and independently to our work, Hoover et al. [16] propose Relational-ESA leakage attacks in the sampled-data setting. They evaluate their attacks on one dataset. In our work, we focus on *evaluating and comparing* the suitability and performance of attacks on Keyword-ESAs used against Relational-ESAs on a wide range of datasets in the known-data setting.

2 Related Work

(Relational) Encrypted Search Algorithms (ESAs). ESAs can be built on different techniques [19]. They include oblivious RAM (ORAM) [12], searchable symmetric encryption (SSE) [9], and structured encryption (STE) [7]. Usually STE is a building-block of SSE. In this work, when we talk of ESAs we refer to ESAs built on STE (whereby the SAP attack [28] which we consider could also be applied to ORAM-based techniques).

Searching on encrypted data was already considered more than 20 years ago: Song, Wagner, and Perrig [31] proposed the first explicit cryptographic scheme that allows searching on data in a provably secure way. In [9], the now standard security notation of adaptive security (where the adversary can choose queries to the server adaptively) was introduced and leakage of SSE was first identified and formalized.

When we talk of ESAs, we refer to *static ESAs*, which only support search operations. In contrast, *dynamic ESAs* also include updates. Recently, Xu et al. [34] have shown that dynamic ESAs may even be more susceptible to leakage attacks.

The problem of executing SQL queries over an encrypted database was first considered in [14]. A more recent approach is called SPX, proposed in [20]. SPX is an encrypted relational database scheme that does not reveal the query response to the server (response-hiding). An extension to SPX is called OPX [21] which improves the efficiency and supports conjunctive queries with optimal time and space complexity. Both schemes are based on STE and are asymptotically optimal in terms of query complexity (time and space). Another scheme for Relational-ESAs was introduced in [6] and is also based on STE. Compared to OPX it reduces the bandwidth needed and improves the leakage for joins.

MongoDB queryable encryption (QE) [23, 24] is a commercial application based on STE that is able to run encrypted queries on fully randomized encrypted data. The encrypted data is never decrypted on the server side.

Leakage Attacks. The first proposed leakage attack by Islam et al. [17] utilizes simulated annealing to exploit leakage. Since then, many attacks have been published that aim for different targets. Most common are keyword-recovery attacks as in [4, 5, 10, 28, 29] which aim to retrieve information about the underlying plaintext of an observed encrypted keyword query. These are the attacks we will adapt for the relational setting. We will go into more details about these attacks in Section 4. We refer to [19] for an overview on leakage attacks.

Leakage Attacks against Relational-ESAs. Attacks against Relational-ESAs were already proposed and evaluated, but in different scenarios than ours.

In [1], an adapted Count attack [5] (called Relational-Count) is proposed which requires full knowledge of the query frequency distribution. In [2], two passive, persistent attacks are proposed: The first attack focuses on uncovering only the attributes of observed queries with minimal auxiliary knowledge (only meta-data information). The second approach combines the first attack with the Relational-Count attack of [1] to improve its performance.

Concurrently to our work, Hoover et al. [16] propose a partial query recovery attack against SELECT operations of encrypted SQL databases. The SELECT attack exploits equality group leakages from co-occurrence information in the passive persistent model, where the adversary has access to sampled-data as auxiliary information. The adversary is also able to differentiate which columns/rows are repeated between different queries and when a SELECT query is repeated. Their attacks are evaluated using one real-world dataset [8].

In our work, we evaluate *all major* attacks [4, 5, 10, 28, 29] with assumptions on attacker knowledge which does not entail full knowledge of the query frequency but a subset of the data (known-data scenario) and use existing leakage patterns. Additionally, we include a wide range of datasets in our evaluations.

In [13], two attacks against a *preview release*[1] of MongoDB QE are proposed. They are sampled-data attacks (utilizing an auxiliary dataset) of a snapshot attacker. In one attack, the attacker has access to a copy of the encrypted documents and a query log. In the other attack, it has access to an operation log. The attacks use a simulated annealing approach to uncover field values in the encrypted dataset. In contrast, our work is in a different setting as it evaluates known-data leakage attacks on general leakage patterns of relational STE-based constructions by passive, persistent adversaries.

In [19], an open-source framework for leakage attack evaluation on real-world data called LEAKER is presented. The Python framework is designed to allow easy integration, evaluation, and comparison of new attacks against ESAs, but also to work with different data sources without requiring domain-specific knowledge. So far, LEAKER [19] only supported attacks against single keyword searches or range searches. We extend the framework to the relational setting and use it for our evaluations.

3 Preliminaries

Notation. A set of integers $\{1, \ldots, n\}$ is denoted as $[n]$, with the corresponding power set as $2^{[n]}$. Given a sequence s of n elements, its i-th element is referred to as s_i. The cardinality of a set S is denoted as $|S|$.

Let $\mathbf{D}$ be defined as a collection of tables T_i, with $i \in [1, \ldots, n]$. A table T_i is a collection of rows r_j (with $j \in [1, \ldots, m_i]$), whereby each row takes k values: $r_j = [(r_j)_1, \ldots, (r_j)_k)]$ (for all $j \in [1, \ldots, m_i]$). The a-th attribute (or column) is defined as a sequence of the a-th element over all m_i rows of table T_i: $[(r_1)_a, \ldots, (r_m)_a]$, whereby all values of one attribute have the same data type. A query consists of a combination of a table T_i, an attribute identifier a, and a value v: $q = (T_i, a, v)$. The query space Q is a set of all possible unique queries $q \in Q$. A query q can also be written as an SQL statement:

$$\text{SELECT} * \text{FROM } T_i \text{ WHERE } a = v.$$

A query $q = (T_i, a, v)$ can be executed on a database $\mathbf{D}$, and returns all rows of the table that apply to the condition: $\mathbf{D}(q) = \{r \in T_i : r_a = v\}$. $|\mathbf{D}(q)|$ refers to the cardinality or frequency of a query, which defines the number of rows that suit the query q. $|T|$ defines the table cardinality of T (total number of rows). $|\mathbf{D}(q)|/|T_i|$ is referred to as the selectivity of the query q. $\text{ids}_i : Q \rightarrow 2^{[m_i]}, \text{ids}_i(q) \mapsto j \in [m_i] : d_j \in D(q)$ refers to the identifier function, which maps a query q to all row-identifiers of table T_i that suit the query. We overload notation and refer to ids_q as ids_i such that query q selects from table T_i. Additionally, for a row r of table T_i, we denote its identifier within T_i as $\text{ids}(r)$. The bit length of a row r is defined as $|r|_b$.

Encrypted Search. In its initially considered setting, encrypted search is based on documents that consist of words. The user is able to search for a keyword on an encrypted collection of documents and receives all documents where the keyword occurs. We generalize this setting to relational databases. A document corresponds in our scenario to a row of a table and is specified by an identifier of the table and the row. We provide more details of this in Section 5.

As in [21], we call an encrypted query a *token*. The token is sent to the encrypted database. The ESA computes and returns the response rows. In our work both terms queries and tokens are used equivalently, as we do not actually consider encryption, but the leakage of ESAs.

Additionally, for simplicity we consider the queried table to be public information, i.e., for a query $q = (T_i, a, v)$, T_i is known by the attacker. This is realistic because the table may be identified by the length of the returned rows (although measures like padding could, of course, somewhat hide this).

Leakage Patterns. Each operation of an ESA leaks information, which composes of so called leakage patterns. In this paper, we only consider leakage based on the query operation (so called query-leakage) with passive, persistent adversaries. Here, we refer to the query leakage patterns as defined in LEAKER [19] and will apply them to the relational setting in Section 5:

- The *response identity* (rid) pattern leaks for each query the identifiers of the matching rows: $\text{rid}(\mathbf{D}, q_1, \ldots, q_t) = (\text{ids}_{q_1}(q_1), \ldots, \text{ids}_{q_t}(q_t))$.
- The *response length* (rlen) pattern leaks the number of rows that match a certain query: $\text{rlen}(\mathbf{D}, q_1, \ldots, q_t) = (|\text{ids}_{q_1}(q_1)|, \ldots, |\text{ids}_{q_t}(q_t)|)$. This leakage is implied by the response identity pattern and can also be derived from the co-occurrence pattern (diagonal values; see next).
- The *co-occurrence* (co) pattern leaks for each pair of queries the number of rows they match together: $\text{co}(\mathbf{D}, q_1, \ldots, q_t) = M \in \{0, \ldots, n\}^{t \times t}$, where $M[i, j] = |\text{ids}_{q_i}(q_i) \cap \text{ids}_{q_j}(q_j)|$.
- The *volume* (vol) pattern leaks for each query the bit length of the resulting rows: $\text{vol}(\mathbf{D}, q_1, \ldots, q_t) = ((|r|_b)_{r \in \mathbf{D}(q_1)}, \ldots, (|r|_b)_{r \in \mathbf{D}(q_t)})$.
- The *query equality* (qeq) pattern leaks if a pair of queries is equal: $\text{qeq}(\mathbf{D}, q_1, \ldots, q_t) = M \in \{0, 1\}^{t \times t}$, where $M[i, j] = \begin{cases} 1, \text{iff } q_i = q_j \\ 0, \text{else} \end{cases}.$

4 Leakage Attacks

We refer to [19] for an in-depth overview of leakage attacks and provide the basics necessary for our work in the following.

[1] This release was used to collect (customer) feedback for the feature [23].

Attacks on encrypted search exploit the leakage of a scheme using some auxiliary data to recover plaintext queries (*query recovery attacks*) and/or recover the plaintext of documents (*plaintext reconstruction attacks*) [5]. Attacks can be further categorized based on the query type, which can be either a *keyword query* or a *range query*. We consider a setting similar to the keyword-attack setting, as our queries are always equality based. Since this setting has mainly been considered with query recovery attacks, we are also concerned with query recovery in our work. The success of an attack is measured as the fraction of successfully recovered queries (called *recovery rate*).

Leakage attacks can be separated into *active* and *passive* attacks [19]: An active attacker is able to interact with the user. We consider the passive attack scenario where the attacker has knowledge of a fixed set of observed queries and an auxiliary dataset.

Auxiliary Data. In addition to the leakage itself, leakage attacks are given access to some auxiliary data (motivated by, e.g., a data breach scenario). So-called *sampled-data* attacks require an auxiliary dataset that is statistically close to the targeted dataset. *Known-data attacks* require a subset of the targeted dataset. In this work, we are concerned with known-data attacks.

Even though some of the attacks we consider (cf. Section 4.1) are built as sampled-data attacks, they can also be run as known-data attacks. This means the attacker is more powerful and does not have access to a similar dataset, but instead utilizes a subset of the actual dataset (which is also statistically close to the full dataset).

4.1 Considered Attacks

Various query recovery attacks have been published in the literature, but not all might be suitable for our relational setting. This includes attacks which are based on volume leakage, which however are not suited well for our considered relational setting (cf. Section 5.3), because the ciphertexts of every row have the same bit length. We will thus focus on attacks that use the response length, response identity, or co-occurrence leakage patterns, which are suitable in our relational setting.

We give an overview of all used attacks in Table 1. All of these attacks were previously implemented for the keyword-only setting in the LEAKER [19] attack evaluation framework. In the following, we describe the relevant details of each attack. We will describe how to adapt them in the relational setting in Section 5.

Table 1: Comparison of applicable attacks for the relational setting. The Scoring and Refined (Ref.) Scoring attacks [10] require additional query knowledge. Some of the attacks can also consider frequency information (based on query equality leakage).

Attack	Original Setting	Leakage
Count v2 [5]	known-data	rlen, co
Subgraph ID [4]	known-data	rid, rlen
SAP [28]	sampled-data + sampled-query	rlen, qeq
Scoring [10]	sampled-data + known queries	co
Ref. Scoring [10]	sampled-data + known queries	co
IHOP [29]	sampled-data + sampled-query	co, qeq

Count v2. The so-called Count v2 attack [5] uses response length and co-occurrence leakage and is built as a known-data attack. The general approach of the Count v2 attack is to reduce the set of possible mappings between tokens and queries by estimating cardinality bounds.

Subgraph ID. The so-called Subgraph attack [4] works with any *atomic leakage*. An atomic leakage is a pattern that reveals a function for each resulting row (e.g., vol and rid) [4]. We only use the Subgraph attack with response identity leakage (called Subgraph ID attack). In the considered relational setting (cf. Section 5.3), the ciphertexts of every row have the same bit length, which rules out volume attacks. In this attack, leakage and known data are modeled as bipartite graphs that are used to eliminate candidate keywords by uncovering inconsistencies.

SAP. The Search and Access Pattern-Based Attack (SAP) attack [28] is based on a Maximum Likelihood Estimation (MLE) approach. It is built as a sampled-data and sampled-query attack, utilizing auxiliary document and query data. It uses the number of queries in a time interval identified via the query equality pattern to obtain the frequency of a query (how often a query is executed). This information and the response length leakage is exploited to optimize a distance between observed query frequency and response length and the corresponding metric obtained from the auxiliary data.

Scoring. The foundation of the Scoring attack [10] is a confidence scoring function that should be maximized in the case when a token and a query are correctly paired. It was also originally designed as a sampled-data attack. The attack additionally requires some tokens to be already uncovered. An improved version of the Scoring attack is called the Refined Scoring attack. It uses an iterative refinement strategy: The strategy is able to add previously uncovered queries to the set of known queries, in order to recover even more queries. This can reduce the amount of known queries needed.

IHOP. IHOP [29] formulates query recovery as a quadratic optimization problem. The attack is based on the SAP attack [28] but uses co-occurrence leakage instead of response length leakage. It is also built as a sampled-data attack with additional frequency knowledge (sampled-query). IHOP [29] uses linear solvers to iteratively solve the overall quadratic optimization problem.

5 Attacks on Relational Encrypted Search

We describe and review Relational-ESA schemes, their leakage, and how attacks against Keyword-ESAs can be used to attack them.

5.1 Relational ESAs

One relational ESA scheme is OPX by Kamara et al. [21]. It is based on three algorithms. The setup algorithm initializes the encrypted database. It takes the plaintext database **D** as input, then samples a secret key K, and computes the encrypted data structure based on K and the database **D**. The encrypted data structure **ED** can then be stored on an untrusted remote server.

To perform a query operation on OPX [21], the client first encrypts its query. A query can usually be represented by a query-tree in the relational setting (e.g., each SQL query can be represented as a tree). The token algorithm takes K and the query-tree Q, then returns an encrypted representation of the query-tree TK (called token-tree). The user sends the token-tree to the remote server, where the query algorithm is invoked. It computes and returns, based on the encrypted data structure **ED** and the token tree TK, the encrypted result. With the key K the user is able to obtain the plaintext result of the query (suitable rows of the table).

Another Relational-ESA scheme was introduced by Cash et al. [6]. It includes several improvements to the basic structured encryption approach, like pre-computing joins on the server. The authors introduce three techniques that are called FpSj, PpSj, and HybStl. FpSj is similar to OPX [21] and works by fully computing joins on the server. PpSj is a technique that works by partially pre-computing joins. HybStI is a hybrid scheme that merges the indexes of both techniques in order to be able to answer both kinds of tokens.

Leakage. Relational-ESA schemes like OPX [21] leak data during the setup, token, and query computation. For query-recovery attacks especially the token and query leakage is relevant. Leakage in OPX [21] is represented as a tree (just as the actual query). Each node in the query tree corresponds to a node in the leakage tree, whereby the leakage differs depending on the functionality of the node. The selection-query leakage of FpSj, PpSj, and HybStI [6] is comparable to the leakage of OPX [21]: Depending on the underlying encrypted structure the query equality pattern and the response length/co-occurrence patterns are leaked (as in OPX [21]). With joins, PpSj [6] can only have leakage based on the individual tables. E.g., the attacker can obtain the number of relevant rows from each table but does not get knowledge of the response-length of the fully-computed join. HybStI [6] is able to answer fully computed or partially-precomputed joins, and therefore has an intermediate join query leakage level between FpSj and PpSj.

5.2 Our Simplified Leakage

The leakage profiles of Relational-ESAs are very convoluted, which makes it hard to adapt and leverage existing attacks.

Thus, in our attack scenario, we consider a simplified ESA scheme that may leak the response identity, response length, or the co-occurrence pattern. We apply attacks using one or more of these patterns $(\mathsf{patt}_1, ..., \mathsf{patt}_\mathsf{p})$. When we evaluate an attack in the relational setting, the results thus apply only to Relational-ESAs whose leakage profile contains the patterns $(\mathsf{patt}_1, ..., \mathsf{patt}_\mathsf{p})$.

Though simplified, some of these attacks also apply to the existing Relational-ESAs, which we clarify below. Other attacks (e.g., those using the response identity pattern) do not apply, but we still evaluate their effectiveness to gain knowledge of attack performance that is useful when designing a potential Relational-ESA that leaks identities.

OPX. The OPX [21] construction uses standard data structure encryption schemes. This means that the underlying scheme only leaks response-identity and query equality patterns.

Our attacks are targeted towards selection queries, so we are only concerned with the leakage for selection predicates based on a condition (WHERE attr = value). Therefore, we only consider the selection node leakage of the leakage tree.

For a simple query that only consists of one input table and one selection predicate, the selection node is a leaf in the leakage tree. This is the basic setting we consider and where our attack evaluation applies.

When an observed query consists of multiple selection predicates (and input tables), each predicate would correspond to one selection node in the leakage tree that we could attack individually. In this case, usually join nodes are present in the leakage tree (because multiple inputs are joined). We do not consider the leakage of the join nodes (merge tables based on conditions), as this would result in a much more complex leakage construction. But as described above, we can attack the individual selection leakage of the query and consider it as a simple query which corresponds to the basic setting. In this case, however, we have to assume that the selection nodes are leaves. Otherwise, the leakage would not be based directly on the input table but on the input of the node, which would again make the leakage construction more complex. Therefore, all our evaluations assume that attacked selection nodes are leafs in the leakage tree. We consider modifications of our simplified scheme to allow for attacks on intermediate selection nodes and joined tables to be valuable future work.

Let q be a basic query in the set of observed queries Q with only one selection and the condition att $= a$ on the table T. In the basic setting we consider, OPX [21] leaks for query q the following pattern [21]:

$$L^{\mathrm{OPX}}(q) = \{|r|, \mathsf{AccP}(r)\}_{r \in T_{\mathrm{att=a}}}.$$

From this, the attacker can obtain the number of rows that suit the predicate (response length pattern). Additionally, the access pattern AccP in [21] is defined as if and when row r was accessed. Thus, from the above pattern over all observed queries Q, the co-occurrence $\{\mathrm{co}(q, q')\}_{q \in Q}$ of a query $q \in Q$ can be derived. Hence, for the considered basic queries, attacks that utilize the response length and/or co-occurrence patterns apply.

Partially Computed Joins and Hybrid Indexing Schemes. FpSj, PpSj, and HybStl [6] also leak the response length and co-occurrence patterns for selection-queries. Compared to FpSj [6] and OPX [21], where the response length and co-occurrence pattern of the final join output is leaked, PpSj [6] (and HybStI [6], depending on its mode) only leaks information based on the individual tables. However, this lower join leakage of PpSj and HybStl [6] does not matter for us, because we do not attack join leakage. It is also not relevant for the attacker to differentiate between selections and relation retrievals. We only consider selection leakage in our setting. Therefore, our simplified Relational-ESA scheme also applies to the constructions in [6].

5.3 Attack Adjustments for Relational-ESAs

We consider all attacks in a known-data setting, even if the attacks are built as sampled-data attacks (as it is also done in [5, 19]). This is possible because a uniformly sampled known-dataset can also be considered as a dataset that is statistically close to the full dataset. The corresponding adaption of an attack is simple, only the normalization of possibly different sizes of the known and full database need to be adjusted. We leave the evaluation of sampled-data attacks for future work.

Utilized Leakage Patterns. The SAP [28] and IHOP [29] attacks are built to use additional frequency information. This is data that is very hard to obtain [19]. For the datasets we use, we do not have access to frequency information and therefore we adapt these attacks to run without frequency information. This also has the advantage to have comparable results to the other attacks. For the SAP attack [28] we only consider response length leakage, i.e., only the cost matrix based on response length is considered for the optimization problem. In the case of IHOP [29], only co-occurrence leakage is used. We evaluate the remaining attacks without modifications in data knowledge and leakage. This includes the Subgraph ID [4], Scoring [10], and Refined Scoring [10] attacks.

Mapping Keyword Attacks to the Relational Setting. Most attacks are built for a setting with documents which consist of keywords (and not a relational database). As mentioned in Section 3, instead of having documents that consist of keywords, we define rows and queries in a manner that allows to employ keyword leakage attacks:

Instead of keywords, we consider column values (keyword = $(\text{attr}_{id}, \text{value})$). Instead of data collections (a set of documents), we consider a table, i.e., a set of rows (document = row = $((\text{attr}_1, \text{value}), ..., (\text{attr}_k, \text{value}))$). We apply this setting to all the attacks we consider. All attacks we use are listed in Table 1 on Page 4.

Restrictions. One thing to consider is that every row (document) has the same number of queries (keywords). Furthermore, every row (document) has exactly one keyword for each attribute from the attribute's space of possible values. This usually does not apply in the case of documents and keywords. Leakage attacks that rely on documents having a unique number of keywords, like the LEAP [26] attack, are not suitable in our relational setting. Additionally, we have the restriction that the ciphertexts of every row have the same bit length, which rules out volume attacks.

6 Evaluation

Our row-query setting is very similar to the already existing implementation [19] of the document-keyword setting, which makes the implementation and comparison of attack results easier.

6.1 Implementation

We extend the existing open-source LEAKER framework [19] to fully support our relational attack setting. Our additions mainly include a MySQL backend that holds the original tables of the dataset as well as pre-computed leakage information. We provide a new interface that maps the existing keyword-document infrastructure onto our relational setting (cf. Section 5.3). With that, modeling leakage patterns, pre-processing and using datasets, launching and evaluating the existing leakage attacks, and plotting the results can be used for relational data exactly as for the already existing keyword data.

We opened a pull request[2] in the LEAKER [19] repository to make our extensions to the framework available as open source.

6.2 Datasets

For our evaluation, we especially consider datasets that have sensitive content and would be suitable to be stored in an encrypted database. However, the datasets we use are not actually sensitive, as we do not have access to real private data. We therefore have to rely on datasets that are anonymized or are public information but could be considered sensitive in a different context (e.g., a different country or regarding different subjects). The attacks we employ aim to recover basic equality queries, therefore the dataset should mostly consist of discrete values.

Our goal is to have a wide variety of datasets with different properties. Nevertheless we have the restriction to not be able to include very large datasets (more than one million rows or much more than 600 000 queries), because of limitations in available computing power, memory, and evaluation time.

An overview of all datasets we consider with their properties is given in Table 2.

The UCI Datasets. The mimic_adult [3], uci_bank [25], and uci_census [22] datasets from the University of California Irvine (UCI) machine learning repository have already been used for evaluations of attacks against Relational-ESAs in [1, 2]. The uci_adult and uci_census tables [3, 22] are extracted from the 1994 US Census bureau database and consist of personal demographic data, like age, education, and relationship status. The uci_bank table consists of personal customer data from a Portuguese banking institution, including details about existing loans.

The DMV Dataset. The Department of Motor Vehicle (DMV) dataset [32] consists of real vehicle, snowmobile, and boat registrations in the State of New York from 2022 to 2023 and has not been used to evaluate attacks against Relational-ESAs before. We consider two cases, one where we use all columns of the dataset and another with a limited number of columns. We limit the number of columns to 11 in order to reduce the number of possible queries, which is done frequently in usages of this data [35]. We also restrict it by the number of rows depending on the evaluation (100 000 or 1 000 000 rows selected uniformly at random). This yields the respective tables we use in our evaluations.

The MIMIC IV Dataset. MIMIC IV [18] is a medical dataset and was already used in the evaluations of LEAKER [19]. It contains data from electronic health records, including procedures and diagnoses. We choose three tables for our evaluations: The mimic_icustays table contains data about entry and exit of patients in the intensive care unit. The mimic_hcpcsevents table contains mappings between patients and the service they took in the hospital. The mimic_admissions table consists of data about the entry of patients into the hospital and where they have been discharged to. Additionally information about the insurance, race, and marital status of the patient are included.

The NC Voters Dataset. The North Carolina (NC) Voter Registration Dataset [27] consists of current data about individuals registered to vote in North Carolina and has not been used to evaluate attacks against Relational-ESAs before. This dataset is especially interesting as it contains data that has not been anonymized. We utilize three different counties of varying sizes as tables.

6.3 Evaluation Setup

We assume that the attacker has access to a subset of the full dataset, which could be extracted from a data breach or obtained by hackers (known-data attack). For simplicity, we assume a known dataset that is uniformly sampled in the range from 1% to 100% of the full dataset, as is common in the literature [19].

[2] https://github.com/encryptogroup/LEAKER/pull/4

Table 2: Comparison of the main datasets and tables we use in the evaluations and their properties.

Dataset	Table/Names	Number of Rows	Number of Attributes	Number of Queries
Adult [3]	uci_adult	32 561	15	22 146
Bank [25]	uci_bank	45 211	17	9 543
Census [22]	uci_census	199 523	42	103 419
DMV [32]	dmv_100k_11cols	100 000	11	1 989
	dmv_100k	100 000	20	119 202
	dmv_1M_11cols	1 000 000	11	3 053
MIMIC IV [18]	mimic_icustays	73 181	5	190 358
	mimic_hcpcsevents	150 771	6	228 643
	mimic_admissions	431 231	11	613 769
NC Voters [27]	ncvoters_tyrrell	2 593	67	14 048
	ncvoters_richmond	32 156	67	142 549
	ncvoters_caldwell	59 138	67	243 450

The attacker observes several queries (tokens) and their results. The leakage of the queries is computed and given to the attacker. We assume that users mostly execute high-cardinality queries and therefore it is more likely for the attacker to observe these. This has been observed in real-world query logs [19]. If not specified otherwise, we uniformly select 150 out of the 500 highest cardinality queries (150/500) of the known dataset as observed queries.

The Scoring [10] and Refined Scoring [10] attacks require initially known queries. We give the corresponding instances access to 15% of the observed queries, i.e., 15% of the target queries are already revealed to the attacker. When looking at the evaluations one has to keep in mind that these attacks start with this higher prior knowledge and can never fall below a 15% recovery rate.

We repeat each evaluation three times with a freshly sampled known dataset. Each time, the evaluation is again repeated three times with freshly sampled queries (called a 3x3 evaluation).

We run all experiments on an Ubuntu 22.04 virtual machine with 8 cores and 128 GB of main memory.

6.4 Overall Evaluation Results

We evaluate all attacks of Table 1 on all the datasets of Table 2. For feasibility, we skip the resource-heavy IHOP attack [29] on datasets with more than 100 000 queries and the Refined Scoring attack [10] on datasets with more than 500 000 queries. Our results are shown in Figure 1 on Pages 9-10. All plots show the percentage of queries of the observed query set that have been uncovered (y-axis) for different sizes of known data the attacker has knowledge of (x-axis).

General Result Pattern. Broadly speaking, the results we observe for the relational setting are mostly similar in performance to the attacks being evaluated in their original keyword setting. The Refined Scoring attack [10] is the best performing attack, with Scoring [10] and IHOP [29] also reaching significant query recovery ($\geq$ 20% for 20% known-data). Of these attacks, IHOP is the only one that does not require initially known queries. Subgraph ID [4] stays mostly around 50% recovery and Count v2 [5] and SAP [28] only reach significant recovery with full knowledge of the dataset.

However, we uncover some subtle and unique differences depending on the data and attack. We discuss these differences in detail below.

Results on the UCI Datasets (Figure 1 a-c). The Subgraph ID attack [4] on the uci_bank and uci_adult datasets outperforms all other instances with $\leq$ 30% known data. The attack has an interesting behavior also seen to a lesser extent on the other data sources: Its recovery rate decreases with more known data before rising again at 80% known data. This could be explained by the fact that both datasets only consist of relatively few queries (around 9 000 and 22 000 respectively), compared to uci_census (with around 103 000 queries). Therefore, low known data rates result in very few queries known to the attacker (much less than 500 queries). Less known queries again result in fewer possibilities of mappings and easier finding of inconsistencies.

Apart from that, the attacks largely follow the general result pattern, although we can observe an increased performance for low known-data rates in the case of uci_census.

Results on the MIMIC IV Dataset (Figure 1 d-f). These results have notable differences compared to the general result pattern: For the mimic_hcpcsevents and mimic_icustays tables [18], the Scoring attack [10] does not uncover new queries (only 15% initial queries). The Refined Scoring attack [10] works better, but requires high known-data rates ($>$ 80% on mimic_hcpcsevents, 100% on mimic_icustays) to uncover a substantial number of queries ($\sim$ 50%). This could be explained by the observation that both datasets do not consist of a substantial number of queries with a unique cardinality in their top 500 query set (mimic_hcpcsevents: $\sim$ 20%, mimic_icustays: 4%).

In comparison, all attacks (except the Subgraph ID attack [4]) on the mimic_admissions table work better than on mimic_hcpcsevents and mimic_icustays, possibly because of the substantially higher fraction of queries with a unique cardinality in its top 500 query set (70%).

Results on the DMV Dataset (Figure 1 g-i). Here, attack performance largely follows the general result pattern and is very similar across all DMV datasets [32], even though the table properties are actually very different (cf. Table 2). However, all three datasets are extracted from the same source, which leads to a similar query cardinality distribution in their top 500 query set and may explain the similarities across instances.

Results on the NC Voters Dataset (Figure 1 j-l). Compared to the other datasets and the general result pattern, all NC Voters evaluations yield worse results, especially in cases with a known data rate of 100% (full data knowledge).

One reason for that could be the distribution of query cardinalities in the NC Voters dataset. If we look at the ncvoters_tyrrell table, we only observe a few queries with a high cardinality (only 6% have a cardinality of ≥ 1 000), which leads to a low number of queries with a unique cardinality (~ 4%). Therefore, the attacks based on response length or co-occurrence leakage have difficulties of mapping queries correctly.

6.5 A Deeper Look Into the Uncovered Queries

We evaluated the attacks on datasets which contain sensitive content. The results of our evaluations so far (cf. Section 6.4) are the fraction of observed queries that can be uncovered. However, though the data may contain sensitive content, our previous evaluations (and all previous evaluations in the literature, which have been performed in a similar manner) do not actually show whether sensitive content is among the uncovered plaintexts. This is interesting as it is especially crucial if an attacker is able to uncover queries that have direct co-occurrence with a personal identity (e.g., last name or patient id co-occurs with a specific disease). Consequently, if only values that cannot be linked to persons are uncovered (e.g., diseases are uncovered but not personal identities), one may argue that no significant sensitive information is uncovered.

Hence, in this section we evaluate how many of the uncovered queries contain personal identifiers. To the best of our knowledge, this approach has not been considered before.

Setup. More concretely, we look at two datasets where the attacks worked well and where personal identifiers exist: The mimic_hcpcsevents table and the ncvoters_tyrrell table. We only consider the Refined Scoring attack [10], as it is the best performing attack in most of our evaluations, and evaluate what fraction of uncovered queries concern personal identifiers.

Table 3: We look at the ncvoters_tyrrell table (last_name attribute) and the mimic_hcpcsevents table (subject_id attribute). We run the Refined Scoring attack [10] on both datasets and show the fraction of uncovered queries with sensitive attributes which have been uncovered. The total number of queries with the corresponding attributes in the observed query set is shown in brackets. The values are averages over 3x3 runs.

Known-data Rate	mimic_hcpcsevents	ncvoters_tyrrell
1%	0.17 (40)	0.15 (11)
5%	0.17 (47)	0.11 (20)
10%	0.14 (50)	0.15 (20)
20%	0.18 (49)	0.27 (21)
40%	0.17 (49)	0.31 (15)
60%	0.26 (48)	0.59 (20)
80%	0.41 (47)	0.87 (18)
100%	0.63 (48)	1.00 (19)

Results. We present the results in Table 3. The mimic_hcpcsevents table includes the attribute subject_id, which can be mapped (with additional knowledge) to a personal identity. If we look at the 500 queries with the highest cardinality, then 161 queries with the subject_id attribute appear. Our results show that the recovery rate computed over all queries is similar to the fraction of recovered subject_id-attribute queries.

The second dataset, we look at is the ncvoters_tyrrell table. In this dataset we look at the sensitive attribute last_name. Overall, the attribute appears 27 times in the set of 500 queries with the highest cardinality. We observe that for high known-data rates (≥ 40%), the recovery rate of last_name-attribute queries is higher than the overall recovery rate. With full data knowledge, the attack is able to recover all queries with the last_name attribute which it was not able for the other attributes.

Overall, this shows that queries with sensitive attributes are vulnerable and are part of the uncovered queries of the attacks. We give details of additional evaluation results in Appendix A.

7 Conclusion

We conclude that existing leakage attacks against Keyword-ESAs can be leveraged against Relational-ESAs and lead to similar evaluation performance. *However, the same difficulties as for the non-relational case apply for estimating the practicality of the attacks:* Successful instances require auxiliary data knowledge, which is hard to model realistically. Furthermore, we note that these attacks take part in the persistent adversary model that requires access to the entire transcript between client and server.

In Table 4, we present an overview of the attack results. The Subgraph ID [4], Scoring, and Refined Scoring attacks [10] require a relatively low fraction of known data to uncover significant information. However, the Scoring attacks require known queries and the Subgraph ID attack does not apply to any existing Relational-ESA scheme. The Count v2 [5] and SAP attack [28] only pose a major risk when the attacker has knowledge of (nearly) all the data. The IHOP [29] attack requires significantly less knowledge than all other attacks to reach significant recovery.

Table 4: Comparison of the main attacks and the amount of data they need to uncover different fractions of queries. The table refers to the worst-case (highest recovery rate) of the attacks with 150/500 queries of high cardinality. Cases with the lowest known-data rate (δ) are highlighted in bold (highest vulnerability).
***The (Refined) Scoring attack [10] has a prior query knowledge of 15%.**

Attack	δ to uncover 20%	δ to uncover 40%	δ to uncover 80%
Count v2 [5]	≥ 30%	≥ 80%	~ 100%
SubgraphID [4]	≥ **1%**	≥ 80%	~ 100%
SAP [28]	≥ 30%	~ 100%	~ 100%
Scoring* [10]	≥ **1%**	≥ 15%	≥ 35%
Refined Scoring* [10]	≥ **1%**	≥ **5%**	≥ **20%**
IHOP [29]	≥ 15%	≥ 55%	≥ 75%

Future Work. For simplicity, we only consider single-table attacks. For future work, this could be extended to attack multiple tables. This could include to consider leakage of joins, which might be much more complex than just considering selection leakage. It represents a more realistic scenario, as data is often split between multiple tables and has to be joined.

Most of our considered attacks use a very simple selectivity estimation. However, there are more advanced estimators [33] which could help to improve the attacks against Relational-ESAs. We give more details on our considerations for using estimator techniques, in Appendix B.

Acknowledgments

We would like to thank Zheguang Zhao for pointers regarding selectivity estimators.

This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 850990 PSOTI).
It was co-funded by the Deutsche Forschungsgemeinschaft (DFG) within SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/251805230.

References

[1] Abdelraheem, M.A., Andersson, T., Gehrmann, C.: Searchable encrypted relational databases: Risks and countermeasures. In: Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2017 International Workshops. pp. 70–85. Springer (2017)

[2] Abdelraheem, M.A., Andersson, T., Gehrmann, C., Glackin, C.: Practical attacks on relational databases protected via searchable encryption. In: International Conference on Information Security (ISC) (2018)

[3] Becker, B., Kohavi, R.: Adult. UCI Machine Learning Repository (1996), DOI: https://doi.org/10.24432/C5XW20

[4] Blackstone, L., Kamara, S., Moataz, T.: Revisiting leakage abuse attacks. In: 27th Annual Network and Distributed System Security Symposium. The Internet Society (2020)

[5] Cash, D., Grubbs, P., Perry, J., Ristenpart, T.: Leakage-abuse attacks against searchable encryption. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. pp. 668–679. ACM (2015)

[6] Cash, D., Ng, R., Rivkin, A.: Improved structured encryption for sql databases via hybrid indexing. In: Applied Cryptography and Network Security: 19th International Conference. pp. 480–510. Springer (2021)

[7] Chase, M., Kamara, S.: Structured encryption and controlled disclosure. In: International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT) (2010)

[8] City of Chicago: crime, taxi rides, rideshare services, and car crashes (2019)

[9] Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: Improved definitions and efficient constructions. In: ACM SIGSAC Conference on Computer and Communications Security (CCS) (2006)

[10] Damie, M., Hahn, F., Peter, A.: A highly accurate query-recovery attack against searchable encryption using non-indexed documents. In: USENIX Security Symposium (USENIX Security) (2021)

[11] Fuller, B., Varia, M., Yerukhimovich, A., Shen, E., Hamlin, A., Gadepally, V., Shay, R., Mitchell, J.D., Cunningham, R.K.: SoK: Cryptographically protected database search. In: IEEE Symposium on Security and Privacy (S&P) (2017)

[12] Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. Journal of the ACM (JACM) **43**(3) (1996)

[13] Gui, Z., Paterson, K.G., Tang, T.: Security analysis of MongoDB queryable encryption. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 7445–7462 (2023)

[14] Hacigümüş, H., Iyer, B., Li, C., Mehrotra, S.: Executing sql over encrypted data in the database-service-provider model. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. pp. 216–227. ACM (2002)

[15] Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association **58**(301), 13–30 (1963)

[16] Hoover, A., Ng, R., Khu, D., Lim, J., Ng, D., Lim, J., Song, Y., et al.: Leakage-abuse attacks against structured encryption for sql. In: 30th USENIX Security Symposium (2024)

[17] Islam, M., Kuzu, M., Kantarcioglu, M.: Access pattern disclosure on searchable encryption: ramification, attack and mitigation. In: NDSS (2012)

[18] Johnson, Alistair, Bulgarelli, Lucas, Pollard, Tom, Horng, Steven, Celi, Leo Anthony, Mark, Roger: MIMIC-IV. https://doi.org/10.13026/6MM1-EK67, https://physionet.org/content/mimiciv/2.2/

[19] Kamara, S., Kati, A., Moataz, T., Schneider, T., Treiber, A., Yonli, M.: Sok: Cryptanalysis of encrypted search with leaker–a framework for leakage attack evaluation on real-world data. In: 2022 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 90–108. IEEE (2022)

[20] Kamara, S., Moataz, T.: Sql on structurally-encrypted databases. In: Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security. pp. 149–180. Springer (2018)

[21] Kamara, S., Moataz, T., Zdonik, S., Zhao, Z.: An optimal relational database encryption scheme. Cryptology ePrint Archive (2020)

[22] Lane, T., Kohavi, R.: Census-Income (KDD). UCI Machine Learning Repository (2000), DOI: https://doi.org/10.24432/C5N30T

[23] MongoDB: Mongodb announces queryable encryption with equality query type support (2023), https://www.mongodb.com/blog/post/mongodb-announces-queryable-encryption

[24] MongoDB: Run expressive queries on fully randomized encrypted data (2023), https://www.mongodb.com/products/queryable-encryption

[25] Moro S., R.P., P., C.: Bank Marketing. UCI Machine Learning Repository (2012), DOI: https://doi.org/10.24432/C5K306

[26] Ning, J., Huang, X., Poh, G.S., Yuan, J., Li, Y., Weng, J., Deng, R.H.: LEAP: Leakage-abuse attack on efficiently deployable, efficiently searchable encryption with partially known dataset. In: ACM SIGSAC Conference on Computer and Communications Security (CCS) (2021)

[27] North Carolina State Board Of Elections: Voter registration data (2023), https://www.ncsbe.gov/results-data/voter-registration-data

[28] Oya, S., Kerschbaum, F.: Hiding the access pattern is not enough: Exploiting search pattern leakage in searchable encryption. In: USENIX Security Symposium (USENIX Security) (2021)

[29] Oya, S., Kerschbaum, F.: Ihop: Improved statistical query recovery against searchable symmetric encryption through quadratic optimization. In: USENIX Security Symposium (USENIX Security) (2022)

[30] Riondato, M., Akdere, M., Çetintemel, U., Zdonik, S.B., Upfal, E.: The vc-dimension of sql queries and selectivity estimation through sampling. In: Machine Learning and Knowledge Discovery in Databases. pp. 661–676. Springer (2011)

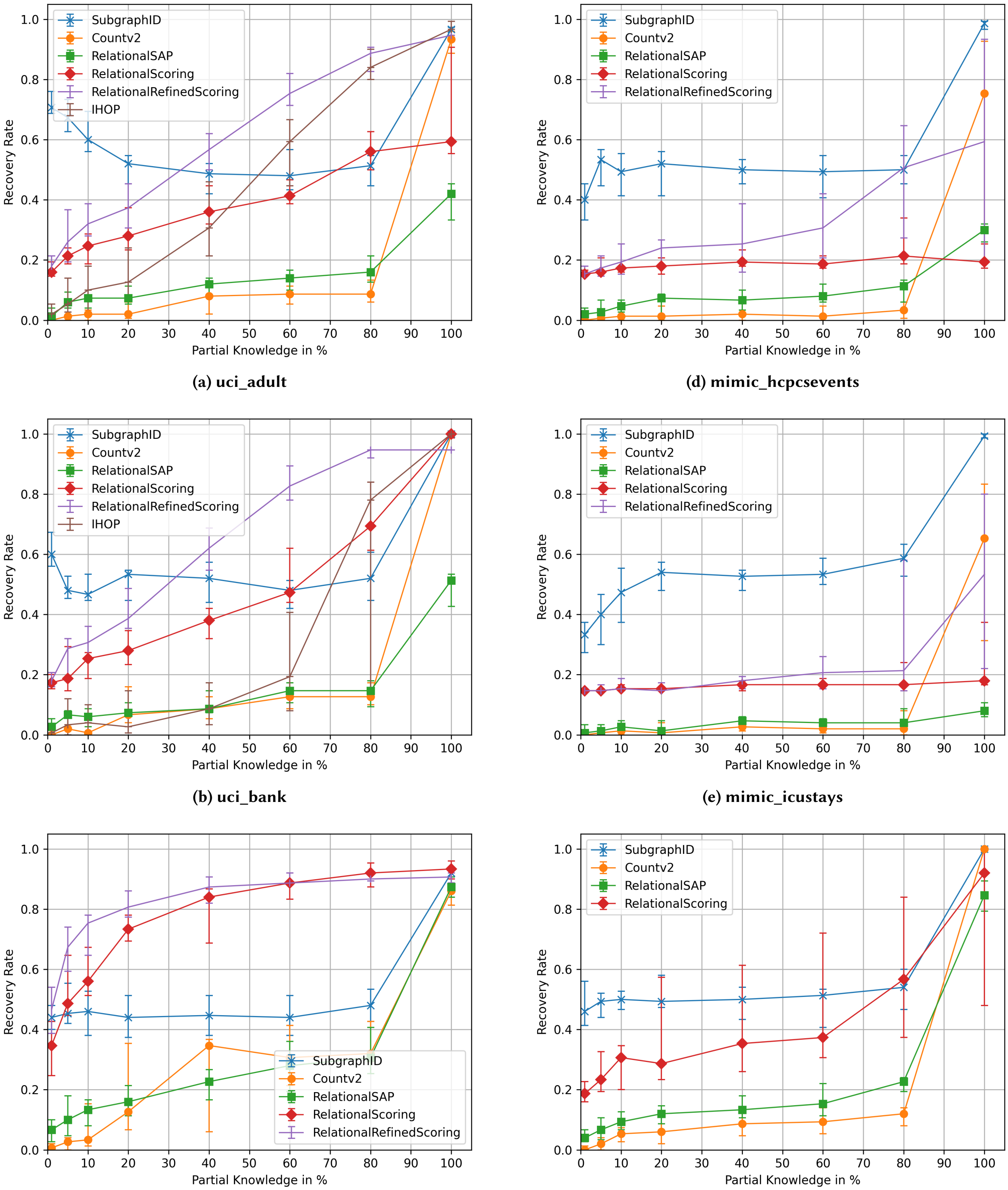
[31] Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: IEEE Symposium on Security and Privacy (S&P) (2000)

[32] State of New York: Vehicle, snowmobile, and boat registrations (2020), https://catalog.data.gov/dataset/vehicle-snowmobile-and-boat-registrations

[33] Wang, X., Qu, C., Wu, W., Wang, J., Zhou, Q.: Are we ready for learned cardinality estimation? Proceedings of the VLDB Endowment **14**(9), 1640–1654 (2021)

[34] Xu, L., Zheng, L., Xu, C., Yuan, X., Wang, C.: Leakage-abuse attacks against forward and backward private searchable symmetric encryption. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. p. 3003–3017. ACM (2023)

[35] Yang, Z., Liang, E., Kamsetty, A., Wu, C., Duan, Y., Chen, X., Abbeel, P., Hellerstein, J.M., Krishnan, S., Stoica, I.: Deep unsupervised cardinality estimation. Proceedings of the VLDB Endowment **13**(3), 279–292 (2019)



(a) uci_adult

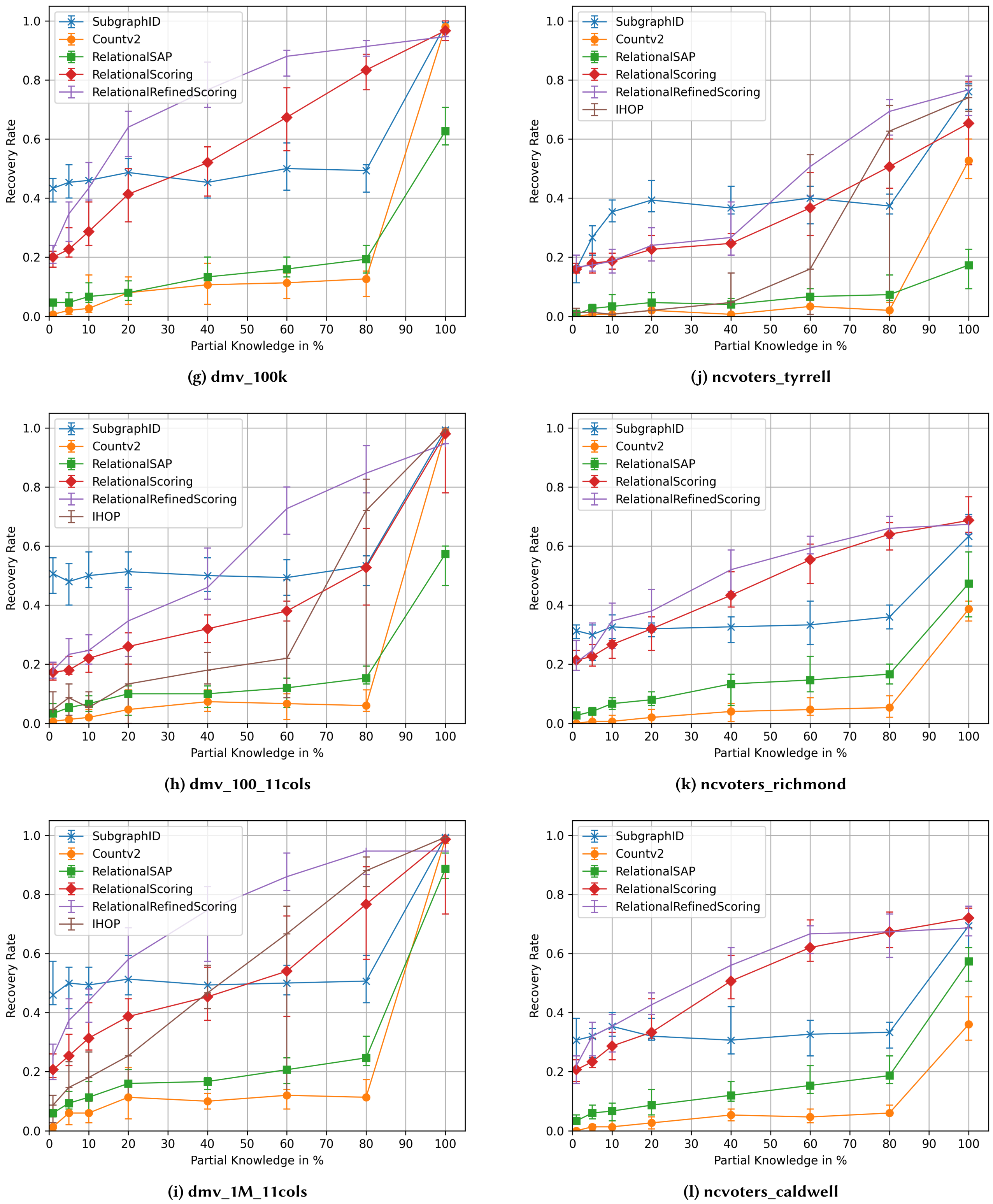
(d) mimic_hcpcsevents

(b) uci_bank

(e) mimic_icustays

(c) uci_census

(f) mimic_admissions



(g) dmv_100k

(j) ncvoters_tyrrell

(h) dmv_100_11cols

(k) ncvoters_richmond

(i) dmv_1M_11cols

(l) ncvoters_caldwell

Figure 1: Our evaluation results of all attacks from Table 1 on all datasets from Table 2. The results on tables of the same data source mostly have similar recovery rates.

A Additional Evaluation Results

Larger Set of Observed Queries In previous evaluations, we assume the attacker to observe 150 queries (from the 500 highest cardinality queries) and their results (and compute the leakage from it). In this section, we re-run some evaluations with 300 observed queries which are selected independently from the 1 000 highest cardinality queries. This might be a more realistic use-case, because users might also execute queries which are not in the top 500.

Overall, the results in Figure 2 are very similar to the general experiments with 150 observed queries drawn from the 500 queries with the highest cardinality. However, we observe a slight decrease in the recovery rate. This could be explained by the observation of how many queries have the same cardinality: Usually we observe that only few high-cardinality queries, but many low cardinality queries exist in a dataset, therefore also many low-cardinality queries have the same cardinality. If we consider a larger query-space (top-1 000), then also more low-cardinality queries are part of the observed query set.

Combined Scoring and Count v2 Attack The Scoring attacks [10] require prior knowledge of uncovered queries. The authors [10] argue that these (few) queries could be obtained by other attacks. The Count v2 attack [5] does not require initial uncovered queries. On the other hand it did not perform as well as the Scoring attacks in our evaluations in Section 6.4. We combine the Scoring and Refined Scoring attacks with the Count v2 attack. This means, we first run the Count v2 attack and use its uncovered queries as input for the (Refined) Scoring attack. This way, we replace the additional knowledge (in our case 15% initially uncovered queries) the Scoring attacks require and replace it with the output of the Count v2 attack. We select the Count v2 attack [5] because it requires the same leakage (co-occurrence) as the Scoring attacks [10]. We run the experiment on the uci_bank and ncvoters_tyrrell datasets. Our results in Figure 3 show that both the combined Scoring and combined Refined Scoring attacks uncover at least as many new queries than in the case of the basic (Refined) Scoring attack with access to 15% of uncovered queries (our baseline). On the uci_bank dataset the combined attack even performs better than the baseline in the cases with ≥ 40% partial knowledge. This could be explained by the observation that the Count v2 attack [5] has a recovery rate of around 10% on the uci_bank dataset compared to around 2% on the ncvoters_tyrrell dataset. However, this is still less than our baseline which has access to 15% initially uncovered queries. It seems to be the case, that the uncovered queries of the Count v2 attack [5] are a better starting point than randomly selected queries.

Overall, this is an interesting observation, because Refined Scoring [10] is one of the best performing attacks in our evaluations. Considering that we can omit the additional prior knowledge it can be a suitable alternative to the IHOP attack [29] (which is resource heavy).

Therefore, the Scoring and Refined Scoring attacks [10] also face a threat even when an attacker does not have access to initially uncovered queries.

Queries with Pseudo-Low Cardinality We always assumed that the attacker is able to observe queries that have a very high cardinality. We repeated some evaluations with the assumption that the attacker is only able to select queries that have a low cardinality, but at least a cardinality of 10 (we call it pseudo-low). We use this lower bound, because most datasets tend to have a lot of queries with a very low cardinality (especially 1).

We evaluate the attacks in this setting on the uci_census and ncvoters_caldwell, and dmv_1M_11cols datasets. The result are shown in Figure 4. Most attacks do not work at all. Only the Subgraph ID attack [4] is able to uncover queries. The result on the uci_census and ncvoters_caldwell datasets can be explained by the observation that for both datasets the query space (500 queries with the lowest cardinality, but at least 10), only consists of queries with a cardinality of 10 or 11. The attacks based on response length and co-occurrence depend on differences in cardinality. Therefore they are not able to uncover queries. On the dmv_1M_11cols datasets also only the Subgraph ID attack [4] is able to uncover queries. This is surprising, because the pseudo-low query space has a wider range in this dataset (from cardinality 10 to 727). We would have expected to be able to at least uncover some queries.

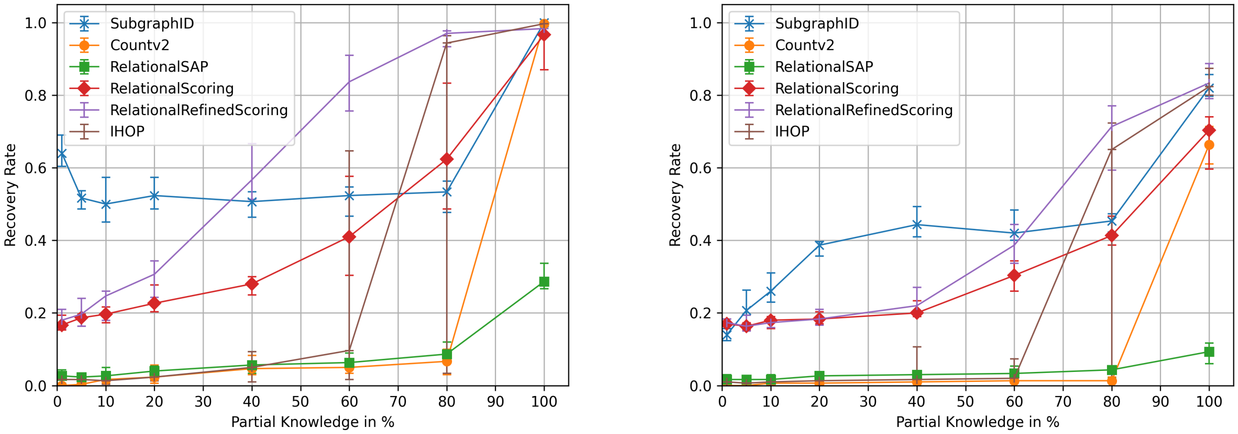
Overall, we see that most attacks do not work well with observed queries of pseudo-low selectivity. This means, that the vulnerability of Relational-ESAs is low when an attacker is not able to observe queries with a high selectivity (but pseudo-low). Only attacks based on response identity leakage really work in this setting. Response identity leakage is a stronger assumption than response length or co-occurrence leakage and therefore less realistic for an attacker to obtain.

B Selectivity Estimation for Leakage Attacks

We introduce the parameter δ that defines the fraction of rows of a table that is known to the attacker (known data rate). We define N as the full size (number of rows) of the dataset.

Response length/co-occurrence (rlen/co) values of observed queries are based on the full dataset, rlen/co values of the known queries only include information based on the known dataset (which is usually smaller than the full dataset). Therefore, all the attacks use some kind of selectivity estimation. The attacks in Section 4.1 use two simple approaches which are very similar: The estimation v of rlen/co can be computed by scaling the rlen/co values c of the known queries to the full size of the dataset ($v = \frac{c}{\delta}$). This is equivalent to computing the selectivity of the queries based on the known dataset and use it to estimate the cardinality of the full dataset by multiplying the selectivity with the total rows of the full dataset. This is usually referred to as *Sampling*. Another approach used by the SAP [28] and IHOP [29] attacks is to normalize both, the values of the known queries and the values of the observed queries, by dividing them with the table cardinality N of the known or full dataset respectively ($v = \frac{c}{\delta N}$). Except the Count v2 [5] attack, all the attacks we consider need to perform this operation in the known-data setting. The Count v2 attack [5] uses a different approach involving a bound-estimate to limit the number of possible queries.

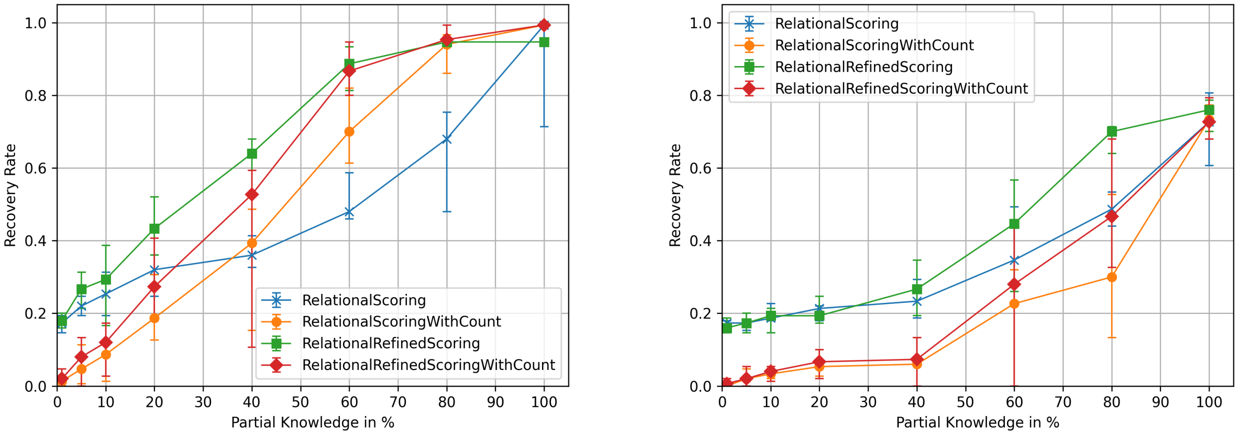
To improve the attacks, we would need an estimator that is trained on the partially known dataset that the attacker has access to. Depending on the attack it would need to estimate the selectivity of single queries (response-length attacks) or pairs of queries (co-occurrence attacks). In all cases, the estimator needs to be better than Sampling. For most of the attacks, instead of using a response



(a) uci_bank

(b) ncvoters_tyrrell

Figure 2: Results of the attacks with 300 observed queries selected independently from the 1000 highest cardinality queries. The attack results are similar to the 150/500 case.

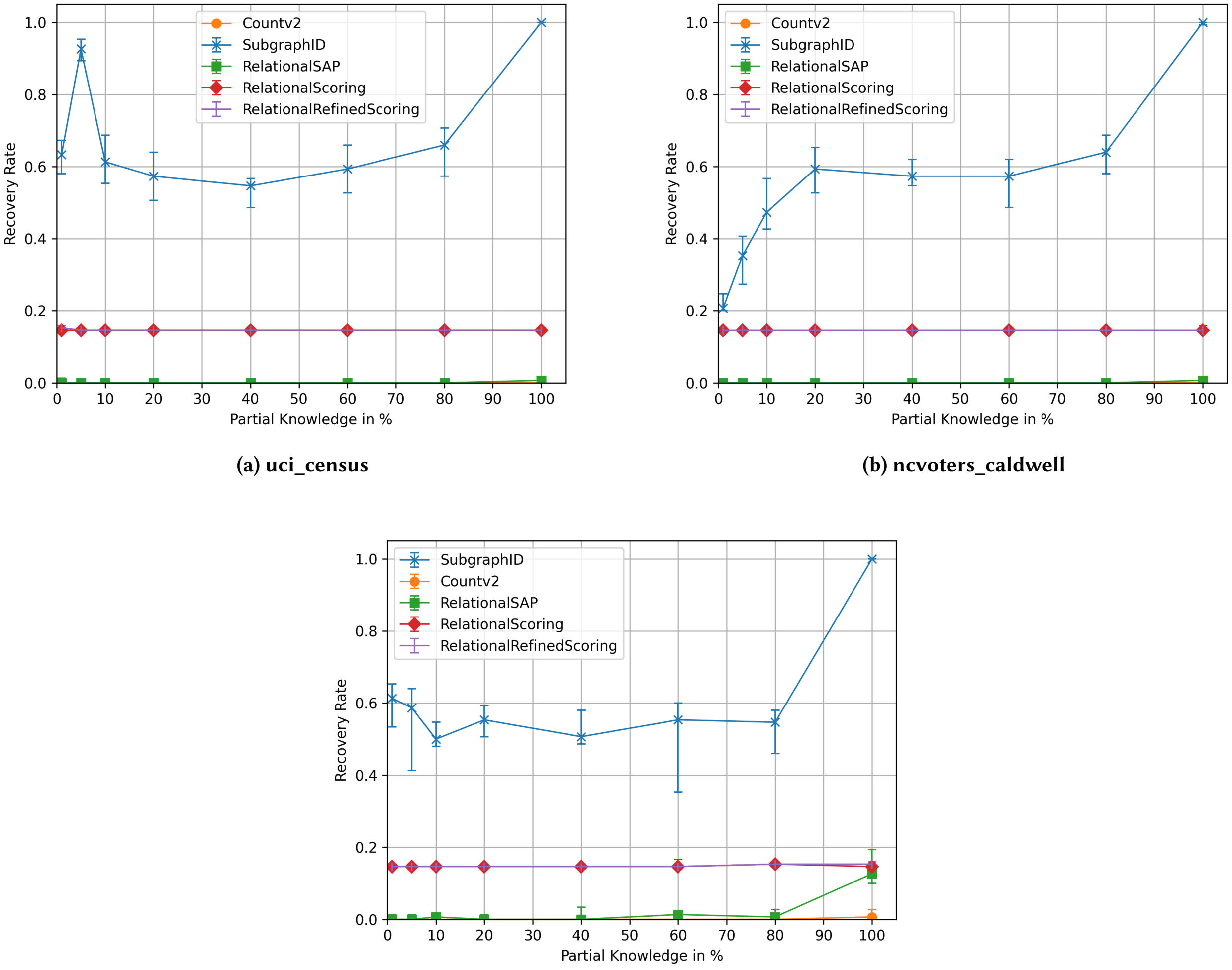


(a) uci_bank

(b) ncvoters_tyrrell

Figure 3: Results of the Scoring and Refined Scoring attacks [10], where the Count v2 attack [5] is used to uncover the required initial queries. We plot the attacks using 15% initially uncovered queries as a baseline. The combined attacks have around 15% less recovery than their baseline. If we deduct the 15% initial recovery rate from the baseline the combined attacks always perform at least as good as the baseline. With a higher known data rate the difference reduces. On the uci_bank dataset the combined Scoring attack (RelationalScoringWithCount) performs even much better than its baseline with ≥ 40% partial knowledge.

length vector/co-occurrence matrix based on the known dataset, the estimated values would be used as inputs. We did some experiments with the Naru [35] estimator in estimating point-estimates for the Scoring attack [10]. However, this did not improve the attack results, because Naru does not consider the uncertainty of only having access to a subset of the full dataset. Future work could adapt Naru to work in this setting. The Count v2 [5] attack already uses the Hoeffding's bounds estimation [15]. To improve the attack, a suitable estimator would need to be better than Hoeffding's bounds [15]. We evaluated the Count v2 attack with the Riondato estimator [30]. However, this also did not improve the attack.



(a) uci_census

(b) ncvoters_caldwell

(c) dmv_1M_11cols

Figure 4: Results of the attacks with a search space of low cardinality (but at least 10) queries. Except the Subgraph ID attack [4], all attacks do not work on all three datasets.