SoK: Cryptanalysis of Encrypted Search with LEAKER – A framework for LEakage AttacK Evaluation on Real-world data

Seny Kamara*, Abdelkarim Kati†, Tarik Moataz‡, Thomas Schneider§, Amos Treiber§ and Michael Yonli§
*Brown University, seny@brown.edu
†Mohammed-VI Polytechnic University, abdelkarim.kati@um6p.ma
‡Aroki Systems, tarik@aroki.com
§TU Darmstadt, {schneider, treiber}@encrypto.cs.tu-darmstadt.de, michael.yonli@stud.tu-darmstadt.de

2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P) | 978-1-6654-1614-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/EUROSP53844.2022.00014

***Abstract*—An encrypted search algorithm (ESA) allows a user to encrypt its data while preserving the ability to search over it. As all practical solutions leak some information, cryptanalysis plays an important role in the area of encrypted search. Starting with the work of Islam et al. (NDSS'12), many attacks have been proposed that exploit different leakage profiles under various assumptions. While these attacks improve our understanding of leakage, it can sometimes be difficult to draw definite conclusions about their practical performance. This is due to several reasons, including a lack of open-source implementations (which are needed to reproduce results), empirical evaluations that are conducted on restricted datasets, and in some cases reliance on relatively strong assumptions that can significantly affect accuracy.**

In this work, we address these limitations. First, we design and implement LEAKER, an open-source framework that evaluates the major leakage attacks against any dataset and that we hope will serve the community as a common way to evaluate leakage attacks. We identify new real-world datasets that capture different use cases for ESAs and, for the first time, include real-world user queries. Finally, we use LEAKER to systematically evaluate known attacks on our datasets, uncovering sometimes unexpected properties that increase or diminish accuracy. Our evaluation shows that some attacks work better on real-world data than previously thought and that others perform worse.

1. Introduction

Encrypted search algorithms (ESAs) allow a user to encrypt its data while preserving the ability to search over it. ESAs have received a lot of attention due to applications to cloud storage and database security (see the survey by Fuller et al. [23] for an overview). At a high level, ESAs consist of two algorithms: a setup algorithm that encrypts a data collection (or database) and a search/query algorithm that is used to query it. Dynamic ESAs also have an update algorithm to add or remove data.

ESAs are designed via various, sometimes intersecting cryptographic techniques: property-preserving encryption (PPE) [4], [8], fully-homomorphic encryption (FHE) [24], searchable/structured symmetric encryption (SSE/STE) [15], [17], [76], functional encryption [11], or ORAM [28]. They have varying tradeoffs between efficiency, expressiveness of the search, and security.

Efficiency. When evaluating the efficiency of an ESA, we usually focus on the query or search time; that is the time needed to search over the encrypted data. ESAs based on FHE require linear time in the size of the data so they are usually considered impractical. For practical purposes, one needs sub-linear ESAs which can be achieved with ORAM in $\mathsf{opt} \cdot \log^{O(1)} n$ time, STE in opt time or PPE also in opt time, where opt is the optimal time to search and n is the number of items in the data collection.

Adversarial models and leakage. ESAs can be analyzed in different adversarial models. The most common are the *snapshot* and *persistent* models. A snapshot adversary receives a copy of the encrypted data at various times and tries to recover information about the data collection. A persistent adversary receives the encrypted data and a transcript of the query operations and tries to recover information about the data collection and the queries. The information an adversary can recover about the data or queries is referred to as *leakage* and, ideally, one would prefer a zero-leakage solution, which can be achieved in several ways. In the snapshot model, it is possible to design very efficient zero-leakage ESAs using structured encryption [5], whereas in the persistent model zero-leakage ESAs can be designed using FHE at the cost of linear-time queries. In the persistent model oblivious, structured, and property-preserving ESAs all leak some information; though recent work has shown how to suppress some of this leakage for certain encrypted data structures [6], [25], [46], [47], [68]. For instance, prominent leakage profiles include the *response identity* (or access) pattern, which reveals the response to a query, or the *query equality* (or search) pattern, which reveals whether and when a query repeats.

Leakage attacks. Because sub-linear solutions leak information, cryptanalysis plays an important role in the area of encrypted search. By designing *leakage attacks* one can try to ascertain whether a leakage profile is exploitable. Starting with the work of Islam et al. [38], leakage attacks were first designed against structured ESAs in the persistent model. Later, Naveed et al. [63] designed attacks against PPE in the snapshot model and Kellaris et al. [50] showed attacks against oblivious ESAs in the persistent model.

These works were improved by a series of papers [9], [12], [32]–[34], [53]–[55], [66], [73]. While the attacks improve our understanding of leakage, it can sometimes be difficult to draw definite conclusions about their performance in practical settings. This is due to several limitations: (1)

DOI 10.1109/EuroSP53844.2022.00014

none of the implementations are open-source (with the exception of [73]) which makes it burdensome to reproduce results, especially in new empirical settings; (2) most of the prior evaluations are based on a few and often small datasets without real query logs; (3) some attacks and/or their evaluations are based on assumptions which may or may not be realistic, depending on the application scenario. We stress that some of these limitations are due to the fact that obtaining real-world query logs is very challenging. In fact, this has been recognized as an important impediment to the evaluation of attacks for some time [34], [73]. A consequence of this is that prior evaluations vary greatly in what assumptions they make about queries, e.g., some evaluations of keyword attacks use the most frequent keywords [12], [38] while others use the least frequent keywords [9], [73], a choice that can significantly affect results.

Our contributions. We broaden the scope of ESA cryptanalysis with the following contributions:

1) We design and implement an open-source Python framework called LEAKER that evaluates the major leakage attacks against keyword and range (oblivious or structured) ESAs on arbitrary datasets. It is intended as an easy-to-use reference tool for research on leakage attacks and mitigations. With LEAKER we enable and invite the community to contribute additional attacks and evaluations to continually advance our understanding of leakage.
2) We uncover a wide set of real-world datasets that capture different realistic use cases for ESAs and, *for the first time, include real user queries* (query logs), which has long been a well-known challenge in the field. For keyword search, this includes search engine and genetic data. For range search, this includes scientific, medical, human resources, sales, and insurance data. The datasets we consider cover significantly more settings than those used in previous work and can serve the community for future benchmarks.
3) We use LEAKER to systematically evaluate attacks on oblivious and structured ESAs on real-world data. Our analysis is an important step towards understanding the practicality of many well-known leakage attacks and provides some new insights which were not obtained by previous evaluations. For example, we find that the BKM attacks[1] of Blackstone et al. [9] can achieve higher recovery rates than previously reported when evaluated with real query logs. On the other hand, the recovery rates of the IKK attack of Islam et al. [38] and of the COUNT attacks of Cash et al. [12] were lower on our datasets. Similarly, the recovery rates of many well-known range attacks were (expectedly) lower on our real-world query logs and data collections. However, when using synthetic query distributions based on statistics from our query logs, the recovery rates improved enough to be considered practical.

Guidance on how to interpret our results. We stress that the goal of our work is to better understand the state-of-the-art leakage attacks and not to provide a security analysis of various leakage profiles. In other words, the fact that an attack has low recovery rates on a given leakage profile under real-world queries and data says nothing about the security of that leakage profile. Understanding whether an attack improves on real-world data or does worse, however, is important for several reasons. First, since the recovery rates of leakage attacks often depend (strongly) on the query and data distributions, it is natural to ask how they perform on a variety of distributions and especially on distributions that capture real-world scenarios. Second, understanding how attacks perform on real-world data allows designers to improve their intuition about their designs and whether there is enough of a "security margin" for practical deployment or whether they should switch to a scheme with a different leakage profile. What constitutes the right "security margin" here is, of course, subjective and different opinions are possible but making such evaluations available to the community is important. A third reason that evaluating state-of-the-art attacks on a variety of datasets is important is that it can help us to uncover new data and query characteristics that affect recovery rates.

Outline. We give related work in §2, preliminaries in §3, and a general overview of leakage attacks in §4. In §5, we detail the design of our LEAKER framework for evaluating leakage attacks. Our real-world datasets are presented in §6 and our evaluation of attacks is shown in §7. We conclude in §8.

2. Related Work

Our work is on the evaluation of leakage attacks. Since these are described in §4, we focus here on ESAs and query log analysis.

Encrypted search algorithms. We use the term *encrypted search algorithm* to refer to any cryptographic primitive/protocol that allows one to execute search algorithms on encrypted data. ESAs can be built using a variety of techniques including fully-homomorphic encryption (FHE) [24], property-preserving encryption (PPE) [4], [8], functional encryption [11], oblivious RAMs (ORAM) [28], secure multi-party computation (MPC) [27], [84], and SSE/STE (see below). In this work we refer to ESAs built from ORAM as oblivious ESAs and to ESAs built from SSE/STE as structured ESAs. Note, however, that the line between these notions is blurry as one can also view ORAM as a (low-leakage) structured encryption scheme for arrays [47].

Searchable/structured encryption. Searching on encrypted data was first explicitly considered by Song, Wagner and Perrig [76], though previous work by Goldreich and Ostrovsky on ORAM [28] could also be used. Boneh et al. [10] then considered the problem of public-key searchable encryption. Security definitions for searchable symmetric encryption (SSE) were proposed by Goh [26] and Chang and Mitzenmacher [13]. Curtmola et al. [17] introduced the now standard notion of adaptive security for SSE, proposed the first optimal-time solution and were the first to identify and formalize leakage. The index-based SSE constructions of [17] were later generalized as structured encryption (STE) by Chase and Kamara [15]. The purpose of STE (referred to as structure-only STE in [15]) is to encrypt data structures in such a way that they can be privately queried. This is different than the purpose of SSE which is to encrypt data collections so that they

1. Throughout, we denote attacks by first letter of author names.

can support private keyword search. Of course, STE and, specifically the special case of encrypted multi-maps, are a building block for optimal SSE schemes. STE, however, has other applications beyond SSE including various kinds of encrypted databases [2], [47]–[49], [88].

Other SoKs. Fuller et al. [23] provide a survey of encrypted search and Yao et al. [85] provide a survey of leakage attacks, largely focusing on property-preserving encryption. Our work focuses on leakage attacks against structured and oblivious ESAs and provides a new software framework and datasets to evaluate these attacks.

Query log analysis. Researchers across different fields have tried to better understand users' search behavior. This is known as *query log analysis*, and is surveyed in [40]–[42].

Many works analyze query logs gathered by proprietary systems that only the authors had access to. Unfortunately, almost none of these works [41], [44], [45], [61], [80] had data we felt was appropriate to evaluate leakage attacks. We also reached out to multiple services for relevant statistics, but none were willing to provide us with even basic information. As no information relevant to leakage attacks was available, we thus identified novel, real-world query data sources (cf. §6) and performed our own analyses of leakage attacks on them (cf. §7).

3. Preliminaries

With $[n]$ we denote the set $\{1, \ldots, n\}$, with $2^{[n]}$ its power set, and with $|s|_b$ the bit length of a string s.

Document collections. A document collection $\mathbf{D}$ over a keyword space $\mathbb{W}$ is a sequence of documents $(D_1, \ldots, D_n)$, each of which is a set $D_i \subseteq \mathbb{W}$. Given a keyword $w \in \mathbb{Q}$ from a query space $\mathbb{Q} \subseteq \mathbb{W}$, a keyword search returns the documents that contain w which we denote as $\mathbf{D}(w) = \{D \in \mathbf{D} : w \in D\}$. The function $\mathsf{ids} : \mathbb{W} \to 2^{[n]}$ takes a keyword as input and returns the identifiers of the documents in $\mathbf{D}(w)$. The frequency of a keyword w is the number of entries that contain w.

Numerical collections. A numerical collection $\mathbf{N}$ over the universe $[N]$ is a (multi) sequence of positive integers $(e_1, \ldots, e_n)$, each of which is an integer $e_i \in [N]$. The *density* of a numerical collection $\mathbf{N}$ is defined as $\delta(\mathbf{N}) = \#\{e \in \mathbf{N}\}/N$; i.e., the number of unique values in $\mathbf{N}$ over the universe size.

Given a range $r = (a, b)$, where $a, b \in [N]$ and $a \leq b$, a range query returns $\mathbf{N}(r) = \{e \in \mathbf{N} : a \leq e \leq b\}$. We overload the function $\mathsf{ids} : [N] \times [N] \to 2^{[n]}$ which takes a range as input and returns the identifiers of the elements that are within the range $r = (a, b)$. The *width* of a range query $r = (a, b)$ is the value $b - a + 1$.

Leakage. The *leakage profile* of an ESA is a set of *leakage patterns* which are functions that map the collection and queries to some target space. We denote a (document or numerical) collection of data entries as $\mathbf{C}$, and a query which can be either a keyword or a range as q. We recall common leakage patterns as defined in [9]:

- The *response identity* pattern rid (or access pattern) reveals the identifiers: $\mathsf{rid}(\mathbf{C}, q_1, \ldots, q_t) = \big(\mathsf{ids}(q_1), \ldots, \mathsf{ids}(q_t)\big)$.
- The *query equality* pattern qeq (or search pattern) reveals if and when queries are equal: $\mathsf{qeq}(\mathbf{C}, q_1, \ldots, q_t) = M \in \{0,1\}^{t \times t}$, where $M[i,j] = 1$ iff $q_i = q_j$.
- The *response length* pattern rlen leaks the number of matching entries: $\mathsf{rlen}(\mathbf{C}, q_1, \ldots, q_t) = (|\mathsf{ids}(q_1)|, \ldots, |\mathsf{ids}(q_t)|)$.
- The *co-occurrence* pattern co leaks how often keywords co-occur within the same document: $\mathsf{co}(\mathbf{D}, w_1, \ldots, w_t) = M \in [n]^{t \times t}$, where $M[i,j] = |\mathsf{ids}(w_i) \cap \mathsf{ids}(w_j)|$. This information is implied by rid and implies rlen.
- The *volume* pattern vol leaks the bit length of matching entries: $\mathsf{vol}(\mathbf{C}, q_1, \ldots, q_t) = \Big((|e|_b)_{e \in \mathbf{C}(q_1)}, \ldots, (|e|_b)_{e \in \mathbf{C}(q_t)}\Big)$.
- The *total volume* pattern tvol leaks the total bit length of matching entries: $\mathsf{tvol}(\mathbf{C}, q_1, \ldots, q_t) = \Big(\sum_{e \in \mathbf{C}(q_1)} |e|_b, \ldots, \sum_{e \in \mathbf{C}(q_t)} |e|_b\Big)$.

Additional patterns include $\mathsf{order}(\mathbf{N})$, the order of the elements, and $\mathsf{rank}_v(\mathbf{N})$, the number of entries that are less than or equal to a given value v.

4. Existing Leakage Attacks

Here, we provide an overview of leakage attacks against oblivious and structured keyword (*point*) and range ESAs and classify them according to the following properties.

Adversarial model. The two main adversarial models considered against ESAs are *snapshot* and *persistent* adversaries. A snapshot adversary has only access to the encrypted structures and any associated ciphertexts. This captures attackers that, e.g., corrupt a server and read its memory. A persistent adversary has access to the encrypted structures, ciphertexts, and to the transcripts of query and update operations. This captures an attacker that corrupts a server and observes all interactions.

Target. Different information can be targeted. In a *data* reconstruction attack, the adversary tries to recover information about the data, whereas in a *query* reconstruction attack, it tries to recover information about the queries.

Auxiliary data. Many leakage attacks require some auxiliary data or knowledge. A *sampled-data* attack requires a sample from a distribution that is close to the data's distribution and a *sampled-query* attack requires one from a distribution close to the queries. On the other hand, a *known-data* attack requires explicit knowledge of a subset of the data (partial knowledge). A *known-query* attack requires knowledge of a subset of the queries.

Passive or active. In a passive attack, the adversary does not choose any data or queries. In an active attack, it is able to interact with the user, e.g., by injecting data.

This work. We focus on the evaluation of persistent query- and data-recovery attacks that require either no auxiliary data, known data or known queries. More precisely, in the case of keyword search we evaluate query-recovery attacks with known-data and in the case of ranges we evaluate data-recovery attacks. We leave the evaluation of sampled-data and sampled-query attacks under real-world data as future work. Properly modeling real-world auxiliary data is challenging since the auxiliary data needs to come from a real-world distribution that is "close" to real-world

Table 1: Major leakage attacks and our perceived risk. The target is either Keyword query (K) or Range data (Value/Count, RV/RC) reconstruction. B is the maximum width and k the amount of missing queries per width. $\mathcal{A}$ denotes that all possible response lengths occur (only within all widths $\leq B$ for $\mathcal{A}_B$, or k missing therein for $\mathcal{A}_{B,k}$). ○ shows no success on our real-world datasets, ◑ denotes some success (K for high partial Knowledge $\geq 75\%$, D for Dense data, W for large Widths close to N, S for Specific data values, E for Evenly and ¬E for unevenly distributed data collections), ● is severe risk across all of our evaluated instances.

Attack	Target	Leakage Profile (§3)	Auxiliary Data		Assumptions		Risk (§7)
			Queries	Data	Queries	Data	
IKK [38]	K	co	Partial	Partial	Non-rep.	–	○
DetIKK [73]	K	co	–	Partial	Non-rep.	–	◑ K
Count v.2 [12]	K	co	–	Partial	Non-rep.	–	◑ K
Subgraph-ID [9]	K	rid	–	Partial	Non-rep.	–	●
Subgraph-VL [9]	K	vol	–	Partial	Non-rep.	–	●
VolAn [9]	K	tvol	–	Partial	–	–	◑ K
SelVolAn [9]	K	tvol, rlen	–	Partial	–	–	◑ K
LMP-RK [55]	RV	rid, rank	–	–	–	Dense	◑ D
LMP-ID [55]	RV	rid	–	–	–	Dense	◑ D
LMP-App [55]	RV	rid	–	–	–	Dense	◑ D
GenKKNO [33]	RV	rid	–	–	Uniform	–	◑ W∨¬E
ApproxValue [33]	RV	rid	–	–	Uniform	Specific	◑ S∧(W∨¬E)
ARR [53]	RV	rid, qeq	–	–	–	–	◑ W
ARR-OR	RV	rid, qeq, order	–	–	–	–	●
GLMP [32]	RC	rlen	–	–	$\mathcal{A}$	–	○
GJW-Basic [34]	RC	rlen	B	–	$\mathcal{A}_B$	–	○
GJW-Missing [34]	RC	rlen	B, k	–	$\mathcal{A}_{B,k}$	–	○
APA [54]	RC	rlen, qeq	–	–	–	–	◑ E

query or data distribution. Collections of datasets with these properties are hard to find.

Risk factors. We summarize our findings in Tab. 1, where we consider query-recovery attacks that recover more than 15% of the queries and data-recovery attacks with reconstruction error less than 15% on our datasets as successful. This threshold is, of course, subjective so the reader can use our detailed empirical results from §7 to formulate their own interpretation and conclusions. Note that the recovery rate of an attack depends on a wide range of factors which we identify in §7. For example, in the case of ranges, these factors include the adversary's knowledge of the data, the query width, and characteristics of the data such as density and whether specific values appear. In addition, we also observed that a skew in the data towards endpoints (1 or N) can result in very different recovery rates (see details in App. C.2.1).

4.1. Attacks Against Keyword ESAs

For keyword search, most attacks are *known-data attacks* which require some partial knowledge of the data collection.

The IKK attacks. The first leakage attack was described by Islam et al. [38], who proposed a query reconstruction attack in the persistent model using the co-occurrence pattern co and a known fraction of the queries. Known as IKK it, roughly speaking, solves an optimization problem minimizing a distance between candidate and observed co-occurrence. Since finding the optimal solution is NP-complete, IKK uses *simulated annealing* [51] to probabilistically find an approximation. Originally introduced as a sampled-data attack, we evaluate it as a known-data attack as in [12]. Recently, Roessink et al. [73] showed how IKK can be improved with deterministic techniques from the Count attack [12] (described next). This modified attack, which we refer to as DetIKK, reduces the search space for the annealing process and, compared to IKK, requires no query knowledge.

The Count attacks. A simpler attack called the Count attack was proposed by Cash et al. [12]. Like IKK, Count is a query reconstruction attack that exploits the co-occurrence pattern co. There are two versions of it. The first, which we refer to as Count v.1, requires knowledge of some fraction of the data and queries. Its original description contained a bug which was addressed in an updated version of the paper and lead to an improved variant of the attack, Count v.2, which only requires knowledge of some fraction of the data. Count v.2 constructs a co-occurrence matrix and compares it to the observed co-occurrences from co. Candidate matches are identified via confidence intervals, and are iteratively eliminated if they are inconsistent with previously confirmed matches.

The BKM attacks. Recently, Blackstone et al. [9] introduced three new passive query reconstruction attacks. The first, VolAn, exploits the total volume pattern tvol and matches a query to the keyword with the closest expected total volume from the adversary's known data. The second attack, SelVolAn, extends this by further identifying candidates with a total volume in the known data falling within a window of the expected volume based on tvol. It then selects the best candidate using the response length pattern rlen.

The third attack, Subgraph, is a framework used to design several attacks using any *atomic* leakage pattern, i.e., any pattern leaking information about individual documents. It constructs two bipartite graphs: one from the observed leakage and the other from the known data. It then filters candidates via inconsistencies between the graphs, confidence intervals (the leakage roughly has to match the expected value), and an optional cross-filtering that tries to

invert the leakage and checks if the candidate appears in all resulting entries. Two concrete attacks that result from the framework are SUBGRAPH-ID and SUBGRAPH-VL, which exploit the response identity pattern rid and the volume pattern vol, respectively.

Other attacks not covered. An attack that exploits qeq with auxiliary data was given by Liu et al. [57], and was recently improved by Oya and Kerschbaum [66] also using rid. Recently, Damie et al. [18] and Gui et al. [35] proposed sampled-data attacks. Active file-injection attacks are considered by Zhang et al. [87], Blackstone et al. [9], as well as Poddar et al. [71]. Additional specialized attacks against *specific* ESA instantiations were considered in [1], [72], [77]. We focus on general passive attacks that do not rely on auxiliary data.

4.2. Attacks Against Range ESAs

We now turn to attacks against oblivious or structured range ESAs. In this setting, there are three variants of attacks: *reconstruction* attacks, *approximate reconstruction* attacks and *count reconstruction* attacks. More precisely, a reconstruction attack recovers the exact values in a numerical collection whereas an approximate reconstruction attack only recovers an approximation of the values. A count reconstruction attack recovers the (approximate) number of times the values occur. Range attacks tend to work "up to reflection", meaning that the attack recovers either the original numerical collection $(e_1, \ldots, e_n)$ or its reflection $(N - e_1 + 1, \ldots, N - e_n + 1)$. This can be viewed as a loss of 1 bit of information. In our experiments in §7 we always report the minimum error rate over either the original collection or its reflection.

The KKNO attacks. The first attacks against encrypted range schemes were proposed by Kellaris et al. [50]. Two attacks were described, both of which are data reconstruction attacks in the persistent model. The first, KKNO-1, exploits the response identity pattern rid and assumes that queries are chosen uniformly at random. At a high level, it determines an entry as having minimal (or maximal) value if it is not contained in the largest proper subset of all identifiers, and determines other entries based on co-occurrence with that entry. The second attack, KKNO-2, only needs the response length pattern rlen but still assumes uniform queries. Intuitively, the attack solves a system of quadratic equations of distances between values and the amount of observed queries with the corresponding response length to uncover the distances between entries. Because both attacks were directly improved upon (described next), we did not evaluate them in our work.

The (G)LMP attacks. Lacharité et al. [55] improved KKNO-1, proposing three attacks which we refer to as LMP-RK, LMP-ID, and LMP-APP. These are data recovery attacks in the persistent model exploiting the response identity pattern rid, with LMP-RK also using the rank pattern. The third attack only recovers an approximation of the values based on reconstructed intervals, but all attacks assume dense numerical collections. In general, the attacks identify the left endpoints of the queries (e.g., via rank) and assign these values to entries by excluding differing entries seen in the response.

Grubbs et al. [32] also improved on the KKNO-2 attack with a new attack we refer to as GLMP that only requires the response length rlen. GLMP, however, only recovers the frequency (or *value counts*) of values as opposed to the values themselves. The attack relies on the assumption that the queries are made in such a way that all response lengths are observed. At a high level, it reduces the observed response lengths to "elementary" queries (in the sense that they have the form $(1, y)$) and uses graph theory to reconstruct counts based on basic properties of elementary queries.

The GJW attacks. Gui et al. [34] proposed attacks in the persistent model that only exploit the response length rlen for count reconstruction. The main attack, called GJW-BASIC, works when the query width is bounded. It builds an initial solution similar to GLMP and uses a breadth-first search that incrementally extends solutions consistent with the leakage. Modifications were introduced for missing queries (GJW-MISSING) and other countermeasures, which we consider outside the scope of this work.

Approximate reconstruction attacks. Several works attempted to weaken the assumptions needed by the KKNO attacks and their extensions. Grubbs et al. [33] describe three attacks that do not require density but still assume uniform queries. The first is GENKKNO, which extends KKNO-1 to an approximate data reconstruction attack by assigning values to entries based on a comparison of the observed and expected amount of occurrences in the leakage. Due to these estimations being individually symmetric with regard to the endpoints (1 and N), queries close to one endpoint are used to establish a global reflection. The second attack is APPROXVAL, which assumes the existence of at least one entry with values occurring in a specific data range, and uses a single favorably-located entry as an anchor that can be identified more easily than other entries. The values around the anchor are then estimated similarly to the GENKKNO attack. The third attack is called APPROXORDER and uses the PQ-tree data structure to approximate the order of the data collection based on the response identity pattern rid, assuming that the data is not heavily concentrated over a few values.

The KPT attacks. Kornaropoulos et al. [53] describe an approximate value reconstruction attack in the persistent model that is agnostic to the query distribution, denoted as ARR (agnostic reconstruction range). It reduces to the problem of support size estimation, in which the number of outcomes not observed is estimated from the frequency of observed outcomes. By estimating support sizes of identifier subsets using the response identity and query equality patterns (rid and qeq), the corresponding distances and, therefore, the values can be uncovered and are assigned according to an order, which can be uncovered via APPROXORDER. While ARR does not require density or uniform queries, the instantiation of [53] assumes that the values are all unique. They suggest alternatives for dealing with the more general case of repeating values, which we use in our work. We provide more details on this in App. A. In our evaluation, ARR uses APPROXORDER to uncover the order, but we also investigate the case where it is directly leaked, which we denote by ARR-OR.

Very recently, Kornaropoulos et al. [54] also applied support size estimation to the setting where only the re-

sponse length and query equality patterns (rlen and qeq) are available, resulting in an approximate count reconstruction attack. It generalizes previous rlen attacks to estimating the number of queries for specific response lengths and solving their relation to value distances, including observations about state-of-the-art encrypted range search schemes [19], [21]. As a result, the attack is *parameterized* by the ESA, and denoted by APA (agnostic parameterized attack). Crucially, APA requires no assumptions about the query distribution and also deals with non-dense data.

Other attacks not covered. The above works also introduced sampled-data variants: A version of LMP [55] uses auxiliary data information to require fewer queries, while Grubbs et al. [32] also incorporate frequency information. APPROXDATA [33] demonstrates how to use APPROXORDER to uncover exact values if density and an auxiliary data distribution are given, and [34] show how GJWBASIC can be improved with auxiliary information about the data. [32] also recover values of update operations in case the frequencies have already been determined. k-NN queries are considered in [52], [53] and Falzon et al. [22] and Markatou et al. [59] recently proposed attacks on two-dimensional ranges. Also we did not consider the attack of Markatou et al. [60] which targets one-dimensional range queries but assumes that all possible queries are issued.

5. The LEAKER Framework

We designed and implemented LEAKER with the following goals in mind:

- *integration:* LEAKER makes the integration and evaluation of new attacks effortless. Researchers can focus on the design and leave the evaluation process to LEAKER. This ranges from interfacing with various data sources to plotting and visualizing the results.
- *comparisons:* LEAKER includes implementations of the main leakage attacks for both point/keyword and range queries. By having attacks implemented in the same framework, they are easier to compare.
- *data sources:* practitioners with proprietary data can use LEAKER to evaluate attacks on their specific data, leading to a tailored evaluation.
- *usability:* LEAKER runs on many platforms and does not require domain-specific knowledge—one only needs to specify the data sources and evaluation criteria.
- *open-source:* LEAKER (including its evaluation scripts) are freely available as open source software at https://encrypto.de/code/LEAKER.

5.1. Architecture

For interoperability, we implemented LEAKER in Python 3.8. It has 8 149 lines of code (1 148 are tests). LEAKER evaluates a leakage attack on a *data collection* using queries from a *query log* and outputs a visualization of the results. It consists of several modules described next: a pre-processor, a datastore, an attack and pattern library, an evaluator, a visualizer, and a statistical analyzer.

Pre-processor. The *pre-processor* parses and prepares data collections and query logs for use by the other LEAKER modules. It includes a set of parsers that can be arbitrarily combined to build a data collection or query log from a directory of files. Currently, LEAKER includes file parsers for `.csv`, `.json`, `.xml`, `.txt`, `.mbox`, `.pdf`, `.docx` and `.pptx` files. In the query logs that we identified, range queries were often part of a larger SQL query so we implemented a SQL parser that identifies and extracts range queries contained in complex SQL queries.

Once parsed, LEAKER uses standard information retrieval techniques to tokenize strings into keywords, extract stems, remove stop words, and identify numerical values. Due to its modular design, the pre-processor can be easily extended for new file types.

Datastore and cache. After a file has been processed, it is passed to an indexer which stores the data in a set of internal data structures for later use. Numerical data is additionally discretized before being stored. Our choice of data structures to store and manage data collections and query logs is important because one of our main goals is to evaluate attacks on large datasets. For example, compared to previous evaluations of co-occurrence attacks, which used datasets with 500 [12], 1 500 [73], or up to 2 500 keywords [38], LEAKER evaluates the same attacks on datasets with more than 250 000 keywords (cf. §6). To achieve this, we use NumPy [37] arrays to store and process numerical data and Whoosh [14] to store and process keyword data. We chose Whoosh because of its Python compatibilitiy and ability to store additional metadata such as document volume in the index.

To get significant results, a single LEAKER analysis can require a large number of repeated evaluations. We speed up NumPy operations by integrating the optimized just-in-time compiler Numba [7]. To address the costly repeated querying of Whoosh structures on large datasets, we use memoization and store the results of Whoosh queries in a cache so we can reuse them across evaluations.

Attack & pattern library. LEAKER comes with a library of attacks and leakage patterns that can be called on any data collection and query log. We implemented the main attacks from Tab. 1 and—with the exception of GLMP and GJW[2]—verified their correctness by replicating the results from the original papers. The attack library is easily extendable to new attacks. One just has to realize an interface performing recovery given the observed required leakage. Examples are given in Listings 1 and 2 in App. B.

All attacks in LEAKER's library are purely in Python except for APPROXORDER [33], where we use a C++ implementation of PQ-trees [30]. Since we could not find any public Python-compatible implementations of the Jackknife or Valiant-Valiant estimators used in the ARR and APA attacks [53], [54] we implemented them ourselves in Python.

Evaluator. This module evaluates attacks. Given an attack and a leakage pattern from the library and a data collection and query log from the datastore, LEAKER proceeds as follows. It creates the *observed leakage* of the leakage patterns on the data collection and query log. Since this can be expensive (e.g., the co-occurrence pattern requires $O(\#\mathbb{W} \cdot (\#\mathbb{W} + n))$ storage and time) it is also cached. Then, it executes the attack on the observed leakage a number of times (in parallel) and stores the results. To

2. We used synthetic data to verify the correctness of these two attacks because we did not have the original data that was used.

Table 2: Overview of our keyword search use cases and dataset properties. $\#\mathbb{Q}_D$ is the size of the entire log and $\#\mathbb{Q}$ the amount of unique queries. n is the amount of documents and $\#\mathbb{W}$ the amount of unique keywords.

Case	Data	Query log			Data collection	
		#users	$\#\mathbb{Q}_D$	$\#\mathbb{Q}$	n	$\#\mathbb{W}$
Web	AOL [67]	656k	52M	2.9M	151k	268k
Genetic	TAIR [20]	1.3k	650k	54k	115k	690k
Email	GMail (ours)	6	–	16-100	6k-47k	60k-895k
Cloud	Drive (ours)	1	–	45	200	19k

Table 3: Summary of our *scientific data* range query logs on the PhotoObjAll.dec collection [74] ($n = 5\,242\,134$ entries with domain $N = 10\,456$, density 95.82%, and an even data distribution). $\#\mathbb{Q}_D$ is the size of the entire log and $\#\mathbb{Q}$ the amount of unique queries.

Data	#users	$\#\mathbb{Q}_D$	$\#\mathbb{Q}$
SDSS-S	1	1.4k	215
SDSS-M	1	13.4k	5 562
SDSS-L	1	38.2k	8 220

evaluate attacks that require known data and/or queries, it first executes an attack-specific sampling algorithm for the known data.

Visualizer. Once all results have been collected, they are used to compute the desired accuracy results (e.g., depending on a choice of available errors) and passed to a visualizer. The visualizer then translates them into graphical `.png` and TikZ plots, which can easily be extended to different error or visualization types. All plots in this work were generated with LEAKER.

Statistical analyzer. LEAKER also includes a statistical analyzer module which computes and plots statistics over its data sources. This is useful to get a better understanding of a data collection's and a query log's characteristics.

6. Data Collections & Query Logs

We describe new datasets including query logs that we believe capture a broad range of realistic scenarios. The datasets include both publicly-available and private data collections and query logs. We provide in App. C.1 a list of other possible data sources that could be used with LEAKER. In App. C.1 we also describe how we pre-processed our data and in App. C.2 we provide some statistics of our datasets.

Data availability. All data is publicly available, except for the private datasets we collected from our volunteers. All our data pre-processing is integrated into LEAKER.

6.1. Keyword Data

We present an overview of all of our keyword data in Tab. 2 and give more background in the following.

Search engines. A major proposed application for ESAs are encrypted search engines, e.g., for desktop search applications or to add search capabilities to email clients or file managers. Due to privacy concerns, we were not able to find public datasets matching these settings so we proceeded as follows.

First, we evaluated attacks on the private data of 7 volunteers by providing scripts to locally extract their GMail and Google Drive query logs and data collections. Out of these participants, 6 evaluated the attacks on their GMail accounts and 1 on their Google Drive account. They returned to us basic statistics of their data, the accuracy of the attacks, and their consent to use and publish the results. We will note the average results of all evaluations as well as the worst and best cases. No personal information is included in this work or in LEAKER.

Because the number of private users we had access to was small and because we cannot release their data, we also used *public* search engine data. Specifically, we used Wikipedia [81] as a data collection and the AOL query log [67]. We recognize, of course, that Wikipedia is *public* so it is not completely representative of the scenarios mentioned above in which one often queries *private* data. Furthermore, it is clear that using AOL queries on a Wikipedia data collection is not ideal but, given the scarcity of real-world data with matching query logs, this is the best one can do at the current time.

Genetic. We were also interested in domain-specific instances that might be queried in a totally different manner. As a prominent case, consider a lab querying large-scale human genetic data for health research, e.g., looking for expressions of specific proteins in gene annotations. Because this query data is very sensitive and not publicly available, we used the following approach.

We performed evaluations on publicly available data that can be seen as related to the above case. Concretely, we use *The Arabidopsis Information Resource* (TAIR) database [56] as a data collection as well as its publicly released query log [20]. The data contains genetic annotations and expression information of the *Arabidopsis Thaliana* plant, which itself is not sensitive. However, we believe it provides a close model for querying in genomic research, as similar information might be queried in the sensitive case of human genomic data.

6.2. Numerical Data

We found five datasets that capture scientific, medical, human resources, sales, and insurance scenarios. The dataset characteristics are summarized in Tab. 3 and Tab. 4. The data is discretized by scaling[3], rounding, and mapping to integers which allows us to evaluate attacks without losing too much precision. We display selected data distributions in Fig. 5 in App. C.2.1, from which we derive the general risk factors of different distributions.

Scientific data. Data from scientific research can often be sensitive. This is the case, e.g., with data generated from satellites, drug and medical studies, or nuclear experiments. For this, we use the Sloan Digital Sky Survey (SDSS), which contains a variety of astronomical data [74]. In addition to astronomy, the SDSS has also been used to investigate user behavior [64], [86]. We used the SDSS to create one data collection and 3 query logs (cf. Tab. 3) using a scale factor of 100.

Medical. Due to high sensitivity, medical data has long been proposed as an ESA application and many works used health data like HCUP [3] to evaluate attacks [32],

3. To scale a value we multiply it by a scale factor 10^a, where $a \in \mathbb{Z}$.

Table 4: Summary of our range use cases and data with n entries, domain size N, and density δ. E and ¬E denote even and uneven data distributions, respectively (cf. §4).

Case	Data collection	Scale	n	N	δ (%)	Distr.
Medical	MIMIC-T4 [43]	×10	8 058	73	80.8	¬E
	MIMIC-PC [43]	×10	7 709	2 684	8.6	¬E
	MIMIC-CEA [43]	×1	2 844	9 978	3.3	¬E
HR	Salaries [29]	×0.01	536	395	2.3	E
Sales	Sales [79]	×1	143	6 288	2.3	E
Insurance	Insurance [16]	×1	886	25 425	1.2	¬E

[34], [50], [54], [55], [63]. While HCUP is a real-world dataset with millions of patients, its attributes usually have small domain, e.g., a patient's age. While evaluation on small domains is important, we also wanted to know how various attacks performed on varying and large domains so we considered the *Medical Information Mart for Intensive Care* (MIMIC) dataset [43] which includes records of medical blood and urine tests performed on ICU patients of the Beth Israel Deaconess Medical Center between 2001 and 2012. We used MIMIC to create three data collections: (1) MIMIC-T4 for patients' free thyroxine, used to evaluate thryroid function; (2) MIMIC-PC for patients' protein/creatine ratio to detect kidney damage or pregnancy; and (3) MIMIC-CEA for patients' carcinoembyronic antigen to detect cancer.

Human resources. Human resource databases contain personally identifiable information like salaries and demographic information. To capture this, we used a March 2018 snapshot of minimum salaries of the UK Attorney General's Office junior civil servants [29].

Sales. Sales data can also be sensitive as it contains trade secrets. We use data released by Walmart for a prediction competition [79], containing weekly sales of department 1 of store 36 from 2010 to 2012.

Insurance. Insurance data may be sensitive since it can reveal financial or health challenges. We use a dataset of property damage insurance claims [16] released by the New York Department of Transportation. They were filed with Allstate in September 2018.

6.3. Privacy Considerations

The experiments described in this work were exempt of IRB approval from our institutions. None of the datasets used were de-anonymized and we stress that LEAKER cannot be used to de-anonymize data; its only use is to evaluate the efficacy of leakage attacks. All the datasets we use are public with the exception of the private data for the search engine scenario. We obtained consent from all involved parties to publish the attack evaluations and some statistics. Access to PhysioNet's MIMIC [43] data is constrained and was only handled by approved authors who completed the required training courses and strictly adhered to PhysioNet's data use agreement.

7. Empirical Evaluation

In this section, we use LEAKER to evaluate all the attacks described in §4 on the datasets from §6 and identify the main characteristics that impact each attack's recovery rate.

7.1. Keyword Attacks

We present results for the private GMail and Drive logs in Fig. 1 and results for the public AOL and TAIR query logs in Figs. 2 and 3. Further plots are given in App. D. Prior to describing our results, we introduce relevant parameters and our experimental setting.

Frequency. Each query in the query log matches a number of entries in the data collection which is its *frequency*. We investigated two settings: *high* frequency and *low* frequency. The former includes an average frequency ranging from 1 806 to 5 707, whereas the latter ranges from 1 to 859.

Number of users. When it comes to the number of users, there are two main settings in which structured and oblivious ESAs can be deployed: *single-user* or *multi-user*. The former characterizes a setting in which a single user queries its dataset, while in the latter multiple users query the same dataset. In the single-user case, we evaluated the attacks by taking the average recovery rate over the queries of 5 users each of which made at least 2 000 queries. In the multi-user setting, we evaluate attacks on a sample of the query logs which, in the case of AOL consists of 656 038 users and in the case of TAIR consists of 1 263 users. For the private Google data, we only evaluated the attacks in the single-user setting since each user queried their own data collection. While the number of users in our private Google dataset is low, the experiments on AOL and TAIR can give some indications as to how the attacks would perform on datasets with a large number of users.

User activity. We noticed that users have different query activities, with some issuing a lot of queries while others are less active. Over all of our datasets, with the exception of the Google data, the most active user issued 6 389 queries while the least active user issued 2 000 queries. Since we did not find any noticeable difference in recovery rates (for all attacks) against the most active and the least active users, we only report the least active case in Fig. 3. The most active case is provided in Fig. 7 in App. D.

Query sampling. Real-world query logs can be extremely large, and given the computational complexity of some attacks, we had to limit the number of queries we used. To do this we sampled the query logs using two approaches which we refer to as *full* sampling and *partial* sampling. Full sampling outputs a new, smaller query log composed of keywords sampled independently of whether they exist in the adversary's known-data set or not; in other words, it is possible that a keyword appears in the query log but is unknown to the adversary. In contrast, partial sampling generates a new and smaller query log that is only composed of keywords in the adversary's known-data set. This captures a worst-case scenario where the user only queries for keywords in documents known to the adversary.

For each of the public query logs, we sampled logs of size 500, 2500 and 5000. Since we did not observe any major deviations in the recovery rates of the attacks based on size we only show the results for (sampled) query logs of size 500. Since the private query logs were already small, we did not sample them.

Query repetition. We observed that some of the queries

in the log are repeated whereas some previous works assumed distinct queries. This assumption makes sense if the adversary can distinguish between queries based on the query equality (which is disclosed by many structured ESAs) and ignore repeated queries but the assumption no longer holds for oblivious ESAs or structured ESAs that do not leak the query equality [25], [47]. Given that some attacks apply to both structured and oblivious ESAs and may be affected by this, it was important to evaluate both settings: *with* and *without* repetition.

Experimental setting. For public datasets, all our evaluations were run on an `Ubuntu 20.04` machine with 390GB memory and 1TB disk space. Unless explicitly specified, every attack is evaluated as follows: given a query log, a data collection, and a specific combination of the parameters highlighted above, we sample a new query log and data collection. For a fixed rate of adversarial knowledge (i.e., known-data rate), we first sample a subset of the entire data collection ℓ times, and for each sample, we sample a subset of the query log λ times. We denote this as a $\ell \times \lambda$ evaluation and it accounts for $\ell \cdot \lambda$ evaluations. For most experiments we picked $\ell = \lambda = 5$ and display the median, maximum and minimum recovery rate.[4] In the public setting, we attack 150 queries drawn from the query log according to their frequencies. We denote by `X-Y` a setting in which the type of user and the frequency are set to `X` and `Y`, respectively.[5] Recall that `X` can be set to *all* users (`A`) or to the single-user setting (`S`). `Y` can be set as *low* (`L`) or *high* (`H`) frequency. For private data, each participant ran the evaluation on their own machine using their entire query log, which did not require any query sampling process.

Discussion. There are obvious limitations to some of our experiments. One example is in how we obtain the adversary's known-data. Since we do not know how an adversary would choose/obtain this data in practice, we have to make some assumptions. For our experiments, we chose to run experiments with multiple known-data samples chosen uniformly at random but other approaches could be considered and it would be interesting to know how they affect the recovery rates. Another aspect is in how we sample the public query logs to generate smaller more tractable logs. We considered different strategies and opted for uniform sampling but, again, one could consider other ways to sample.

7.1.1. The IKK Attacks [38], [73]. Given the attacks' quadratic costs in the number of keywords, evaluating them on TAIR was infeasible so we only considered AOL.

IKK did not work in any of our settings. This stands in contrast to previous results which showed high recovery rates but assuming almost full knowledge of the data [12], [73] on small datasets. The best recovery rate we observed was less than 15%; even with full knowledge of the data and in the `A-H` setting (all users, high frequency), which is the least realistic setting. We stopped the attack's annealing process after it ran for 48 hours. We attribute this to a large search space, since the DETIKK [73] attack—which has a reduced number of states—does not suffer from this in the high-frequency setting (cf. Fig. 3). This evaluation suggests that DETIKK may only work in practical settings with high known-data rates.

7.1.2. The Count Attack [12]. Like IKK, COUNT V.2 is also quadratic in the number of keywords so we only ran it on the AOL query log.

In our evaluations, COUNT V.2 succeeded in all of our settings but using full knowledge of the data. Without full knowledge, it only succeeded in the `A-H` setting with a recovery rate of 63% based on a 30% known-data rate. Here, the query frequency had a mean of 5 707. This aligns with the results stated in [9]. However, we observed that without full knowledge of the data, COUNT V.2 failed to achieve adequate recovery rates. For example, in the `S-L` setting, it achieved 8% recovery rate even with knowledge of 90% of the data (cf. Fig. 3 for more data points). Similarly to IKK, this suggests that COUNT V.2 may require very high known-data rates in more practical settings.

7.1.3. The BKM Attacks [9]. The BKM attacks were efficient enough to evaluate on all of our query logs.

The SUBGRAPH framework. Against our private dataset (Fig. 1), both attacks did surprisingly well. For the email case, more than 18.75% of the real-world queries are recovered with knowledge of only 5% of the data.[6] Though the private dataset had a very small number of users, we did observe that the recovery rates of the Subgraph attacks on the Gmail data were very consistent across users and always greater than 18.75%. On the Drive dataset, the attacks achieved a much lower recovery rate with low known-data rates; however when 35% of the data is known, both attacks recovered half the queries.

In the public data setting (Fig. 3), both SUBGRAPH attacks also achieve non-trivial recovery rates. For high-frequency queries, SUBGRAPHVL achieves very high recovery rates almost independently of the known-data rate and of whether the queries come from a single user or aggregated users. SUBGRAPHID, however, has a slightly lower but still significant recovery rate: 86% in the `S-H` setting for AOL with known-data rate of 10%. For low-frequency queries (`.-L`), both attacks still perform well for an average frequency as low as 4.85 (cf. Fig. 7 in App. D). For frequency 1, however, we found that SUBGRAPHVL does not work at all, while SUBGRAPHID can still correctly recover queries at a low but significant rate (cf. Fig. 3). This is in contrast to the original evaluation of the attack [9], where queries with frequency 1 (sampled from the data collection) could not be recovered. This again illustrates the value of using real-world query logs for evaluations as it can uncover new settings in which existing attacks work.

In Fig. 2, the queries are fully sampled from the query logs and we allow for repeated queries. We noticed that: (1) the variance of the recovery rate increases; and (2) the recovery rate of SUBGRAPH is reduced by about 40% if

4. We limited evaluations to 25 *per user* due to the computational overhead incurred by some attacks. As an instance, the COUNT V.2 attack took one day to complete a single iteration for 5 users.

5. For ease of exposition, we mainly focus on these two parameters, but we also varied the type of the query sampling, the query repetition, or the length of the query log.

6. Given that email data often contains public information such as updates and spam, we consider a 5% known-data rate to be realistic.

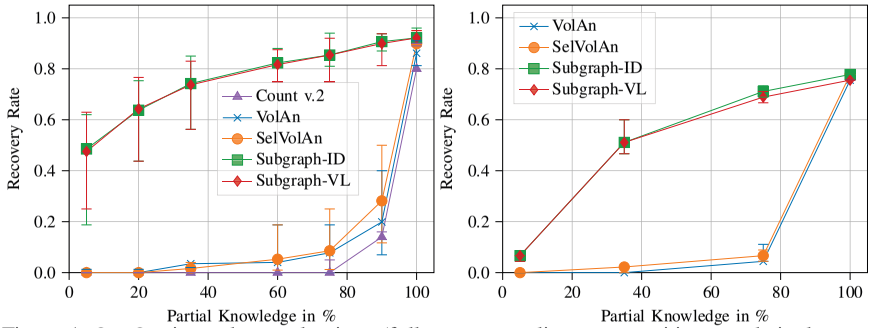


Figure 1: 3×3 private data evaluations (full query sampling, no repetitions, and single-user setting) on GMail (left) and Drive (right). Only one user evaluated COUNT v.2 for GMail.

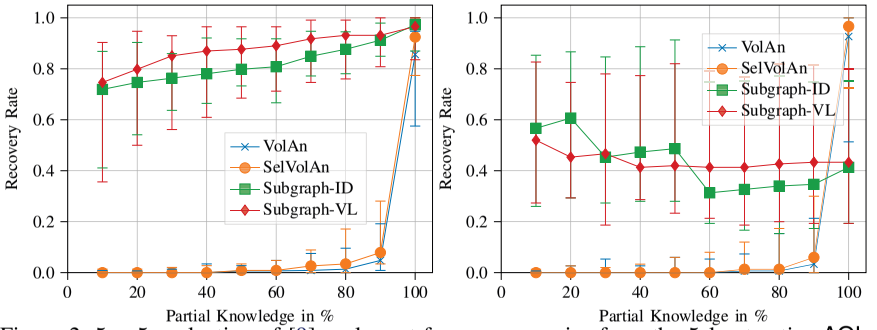


Figure 2: 5×5 evaluation of [9] on lowest-frequency queries from the 5 least active AOL users for full sampling without restrictions (left) and sampling with replacement (right).

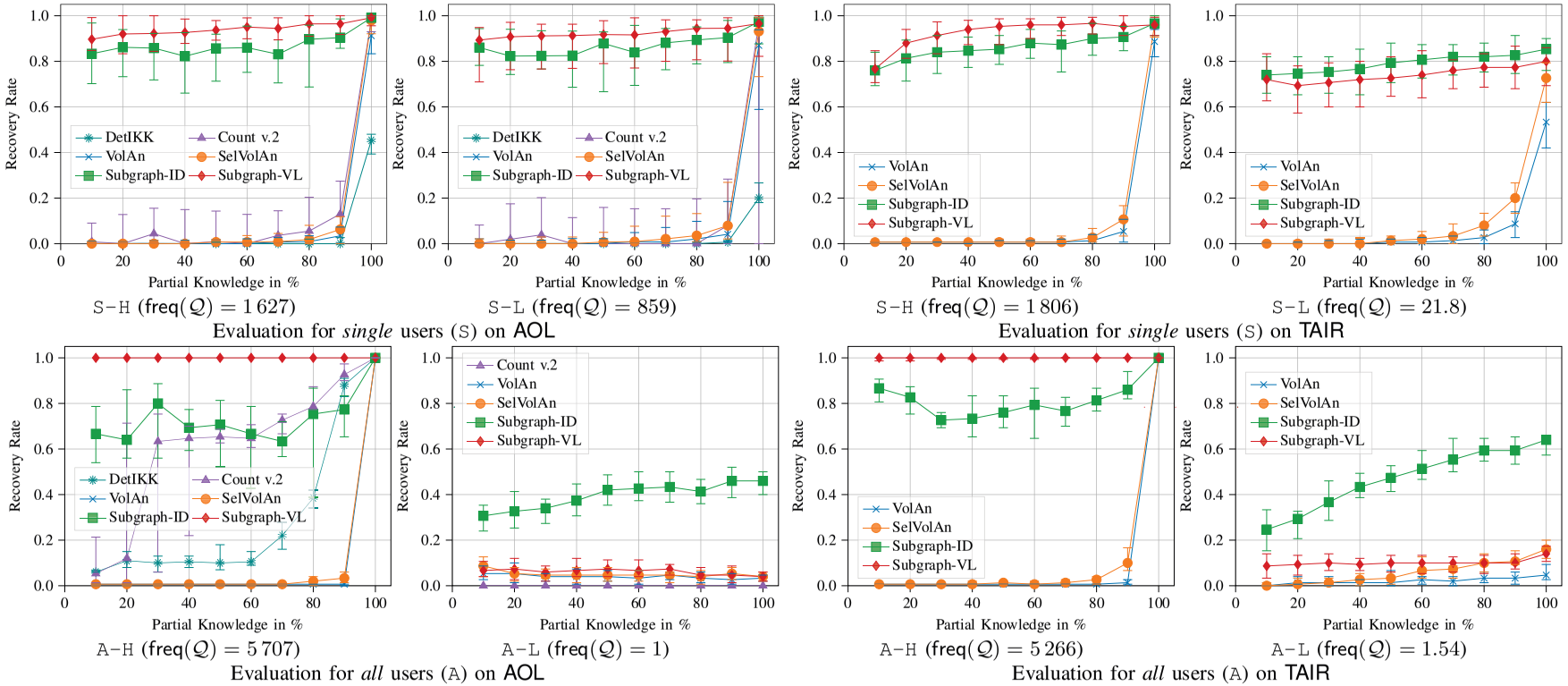


Figure 3: Attack evaluations against AOL and TAIR. All evaluations are 5×5, except for 3×3 evaluations done for COUNT v.2 and DETIKK. 150 queries are drawn for either each of the least active users (top; single user setting S) or all users (bottom; A) without replacement according to their query frequency from the 500 most (H) or least (L) selective queries in the query log that are also contained in the partial knowledge, respectively. The resulting mean frequency is given by $\mathsf{freq}(\mathcal{Q})$.

Table 5: Normalized mean errors on the entire SDSS query logs. The collection is sampled $25\times$ uniformly at random with size $n = 10^4$ ($n = 10^3$ for APA and ARR).

Instance	GKKNO	AVALUE	ARR	ARR-OR	APA-OR^{BT}	APA-OR^{ABT}
SDSS-S	0.413	0.432	0.473	0.249	0.242	0.239
SDSS-M	0.408	0.435	0.287	0.128	0.242	0.240
SDSS-L	0.417	0.456	0.286	0.141	0.241	0.242

we allow for repeated queries. Both attacks achieve lower minimum recovery rates—around 40% for 10% known-data rate—whereas their median recovery rate drops to similar levels for all known-data rates if queries repeat. We can see that the recovery rate is affected by full sampling and repeated queries but remains significant in all cases. The attacks also work slightly better even with lower known-data rates on AOL compared to our private datasets (Fig. 1). A possible intuition why these attacks may work better than others is that they rely on atomic leakage concerning individual documents.

Total volume attacks. Compared to the SUBGRAPH attacks, the VOLAN and the SELVOLAN attacks achieved significantly lower recovery rates in the private data setting. More precisely, they only achieved 20% recovery rate on the GMail dataset even with 75% known-data rate (see Fig. 1).

We also noticed that the attacks did poorly in the public setting considered in Fig. 3. For high-frequency queries, they needed almost perfect knowledge of the data to achieve an adequate recovery rate. Specifically, at least 70% of the data needs to be known in order to recover more than 7% of the queries in the S−H setting. For very low frequencies, none of the attacks worked with less than 100% known-data rate. Therefore, based on our datasets, we only consider them to pose a risk for very high known-data rates.

7.2. Range Attacks

We ran our evaluation against the numerical datasets described in §6.2. Our results are summarized in Tab. 5 and Fig. 4. Similar to our keyword attack evaluation (§7.1), we first describe the main parameters that impact accuracy as well as our experimental setting, and then proceed to a high-level description of our results.

Query distribution. This captures how a user queries its own dataset. For the SDSS dataset, we were fortunate to have access to both the query log and its corresponding data collection. For datasets with no available query log, we had to consider synthetic query distributions. Previously considered query distributions include the uniform distribution and instances of the beta distribution [53], [54]. Given that none of these distributions are supported by real-world query logs, we introduce a new distribution we call the *truncated Zipf* distribution which has some of the basic properties of the distributions we observed in SDSS (cf. App. C.2.3). To summarize, this distribution is a variant of the standard Zipf(a, N) distribution where we fix a to be 5. It also has two additional parameters: the *maximum width* B, and the *fraction* f. The former removes any range that has width larger than B, and the latter ensures that only a fraction f of possible ranges occur. We denote this new variant of the Zipf distribution TZipf(B, f). We varied both B and f and found that B can have a significant impact on recovery rates. Thus, in our evaluation we present results for both (relatively) small and large B, and set f to be small but large enough to give us a meaningful query space.

Amount of queries. For the SDSS dataset, we did not sample the query logs and, instead, used them in their entirety, see Tab. 3. For the other numerical datasets, however, we sampled 10^2 to 10^5 queries with replacement.

Characteristics of the data. The datasets we use and that are described in §6, have different characteristics and properties which allows us to assess attacks in different scenarios. We recall some of their basic properties here (more details are provided in App. C.2.1): the medical dataset, MIMIC, and the insurance dataset, Insurance, are skewed towards low values with just a few high-value outliers so we refer to them as *uneven* distributions, whereas entries in the human resources dataset, Salaries, and the sales dataset, Sales, are spread more evenly across the domain range so we refer to them as *even* distributions.

Data density and size. Other basic properties are the size of a dataset and the density of an attribute/column of dataset. The size refers to the number of records in the collection and the density of an attribute/column is the ratio of unique values in the attribute/column over its domain size. We recall that the medical dataset MIMIC-T4 is much more dense than the other datasets.

Experimental setting. Contrary to the keyword attacks we evaluated which were all query recovery attacks, the range attacks we consider are all data recovery attacks. To measure success, we use a *normalized mean absolute* error. This error is a distance measure equal to the mean of the normalized differences between the true and recovered values. For count reconstruction attacks, we compute the error between the *sorted* values, i.e., independent of the order. To make this difference clear and comparable to all attacks (disregarding order information), we add the suffix -OR in the plots for these cases. As a reference, an error close to 0.5 means that the attack does not work, while one close to 0 is synonymous to a practical attack. Every attack is evaluated several times and we report the mean, maximum, and minimum error.

7.2.1. The (G)LMP Attacks [32], [55]. As expected, the LMP [55] attacks were successful on MIMIC-T4, which is dense, under uniform queries. The attacks were also successful on MIMIC-T4 with queries sampled from a truncated Zipf and they achieved very small error rates (e.g., LMP-RK achieves a 0.0003 error rate). If the attacks output the reflection, the error is significantly larger. Based on our experiments, as predicted, we consider LMP as risky if the data is dense.

The GLMP [32] attack, which is a count-reconstruction attack, achieved perfect recovery on MIMIC-T4 if we disregard density.[7] This occurred with 10^5 queries from a truncated Zipf distribution with B = 73 and f = 0.2.

7.2.2. The GJW Attacks [34]. With the exception of the Salaries case, both GJW-BASIC and GJW-MISSING aborted due to infeasible search spaces. More precisely,

7. Running the attack on a collection that has been made completely dense by removing values that do not occur from the collection's universe.

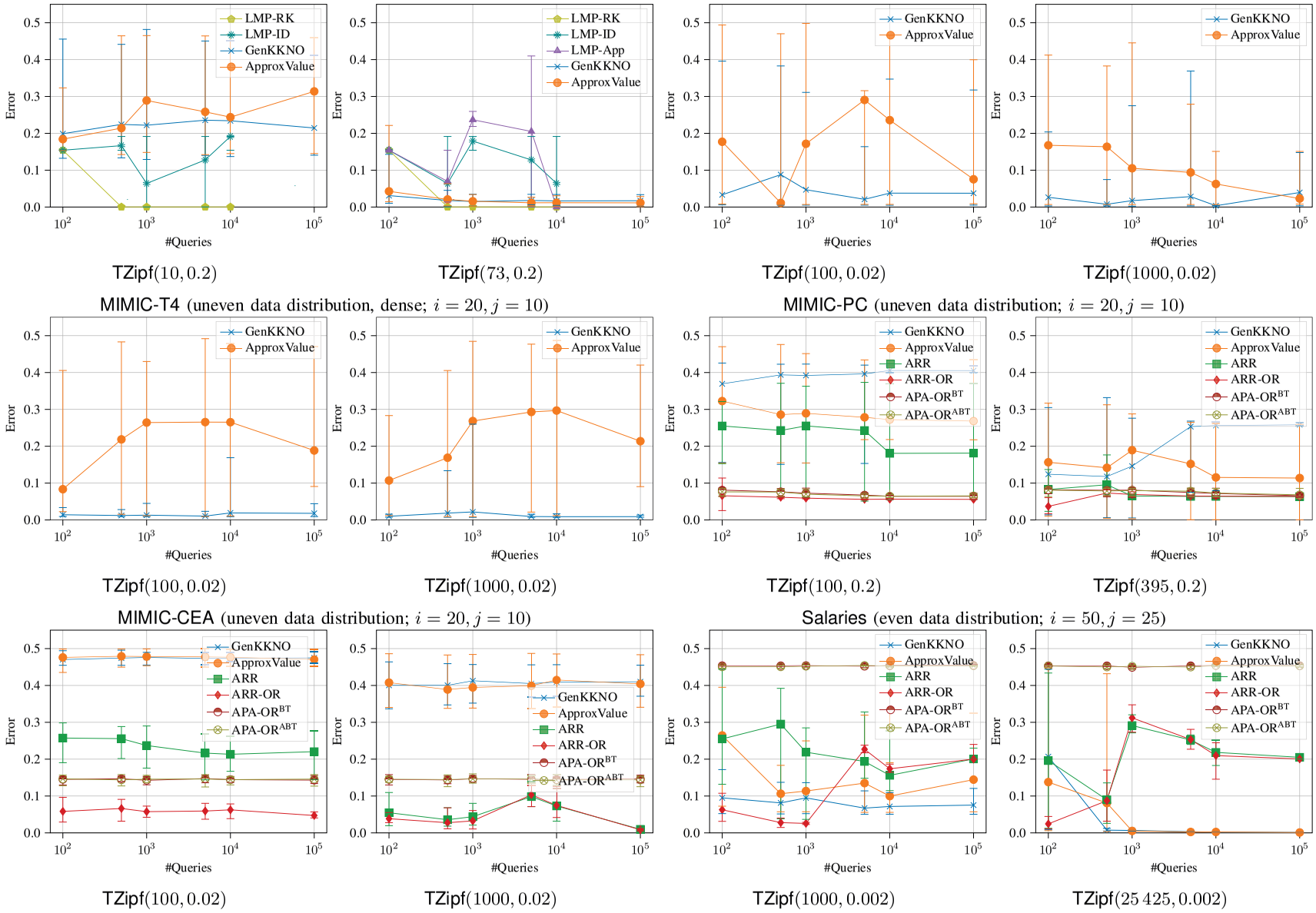


Figure 4: Normalized mean absolute error of value reconstruction attacks on different datasets using a truncated Zipf distribution. Captions include the respective dataset's general data distribution (cf. §4, App. C.2.1) identified as risk factors. i resp. j iterations were performed for GENKKNO and APPROXVAL resp. LMP, ARR, and APA.

for Salaries, the recovered counts were always completely incorrect when queries are sampled from TZipf, and while the correct counts were uncovered for a uniform distribution for 10^5 queries, they were assigned to the wrong values due to the low density.

7.2.3. Approximate Reconstruction Attacks [33]. Though the GENKKNO and APPROXVAL attacks of [33] were designed to work with uniform queries, we still evaluated them using our SDSS data collections and query logs. As expected, they failed to recover any meaningful data, achieving an error of at least 0.41 (cf. Tab. 5). However, and perhaps surprisingly, we did find that both attacks achieved significant recovery rates when queries were sampled from the truncated Zipf distribution (cf. Fig. 4).

We identified two settings where GENKKNO succeeds with a truncated Zipf: (1) when it has relatively large B; and (2) when the data is skewed towards lower values as is the case in MIMIC and Insurance, where it even succeeds for a small B. Note that there was an exception to (1) which was when we evaluated it on the Sales dataset. Generally, we believe (1) holds because queries with large width are more likely to cover values close to one of the endpoints (1 or N), which is required to determine the global reflection (i.e., whether the value belongs to the first or second $N/2$-half of the domain).[8] For (2), we believe that the skewness of the values in a dataset helps to easily determine one of the endpoints, and therefore the global reflection. This is not the case for Sales, where the probability of hitting any value is almost uniform, as seen by an error larger than 0.4 for $\mathrm{B} = 100$.

The same holds for APPROXVAL, under the additional condition that specific values have to be present in the collection. Namely, the attack assumes that at least one value in the dataset is in the range $[0.2N, 0.3N]$ or its reflection to find its anchor. Though this is true for all data collections, the fraction of such records is much lower for MIMIC-PC (0.03%) and MIMIC-CEA (0.4%) than the others ($\geq 2.9\%$), which are exactly the cases where it has much worse and unpredictable performance compared to GENKKNO with a maximum error equal to 0.49 for $5\,000$ queries, see Fig. 4.

7.2.4. The KPT Attacks [53], [54]. Contrary to the previous attacks, these attacks achieved low error rates on the SDSS data and queries but were computationally demanding since they have to solve non-convex or nonlinear optimization problems with solution size $n + 1$. This computational overhead also meant we could not evaluate them on our MIMIC datasets. We also had to rely on SciPy [78] instead of the original MATLAB optimization to meet our open-source goals (cf. §5). In the following, we give more details on attack performance.

ARR and ARR-OR. ARR-OR reconstructs with error rate 0.15 in almost all of our settings but standard encrypted ranges schemes do not leak the order [19], [21]. In our experiments, ARR only came close to ARR-OR's performance when queries were sampled from truncated Zipf distributions with large B.

APA. Since APA [54] is parameterized by the underlying range scheme, we show results for the state-of-the-art schemes [21] and [19]. Although APA only achieves an error rate of about 0.24 on the SDSS data collection and queries, it performs well on other datasets and with various query amounts and query widths. It is sensitive, however, to the data distribution. In particular, while it achieves an error rate of 0.06 on Salaries and of 0.15 on Sales, it had an error rate of 0.45 on Insurance.

8. This is not the case for the SDSS logs, also diminishing accuracy.

8. Conclusions

In this work, we described LEAKER, a new framework to implement and evaluate leakage attacks on real-world data collections and query logs. We hope that LEAKER will enable the community to easily implement, independently evaluate, and compare current and future attacks and countermeasures. Additional evaluations on more data would also be valuable in our opinion. We then used our framework to re-evaluate the major leakage attacks for keyword and range queries. The implications of our evaluation are as follows.

Keyword search. The IKK [38] and COUNT [12] attacks did not work as well on our datasets whereas the SUBGRAPH attacks of [9] performed surprisingly well even in settings with low known-data rates, and on low-frequency queries on small private datasets. This contradicts previous intuitions that SUBGRAPH attacks might not work as well on real-world data [9]. As a consequence, we view SUBGRAPH attacks as practical even for low-frequency keywords and therefore recommend schemes that hide the response identity and response length patterns in such settings (for example, building on [6], [9], [25], [31], [36], [46], [47], [68] for various structures). Compared to the other attacks, SUBGRAPH relies on *atomic* leakage of individual documents, and our evaluations strengthen the intuition that such leakage might be more risky. Our results also confirm the conclusions of [9] that the VOLAN and SELVOLAN attacks (which exploit the total volume and response length patterns) could pose some risk for known-data rates $\geq 75\%$.

Range search. In contrast to keyword search, our evaluations uncovered many subtleties in the case of range search. As expected—since many range attacks are designed to work for specific query distributions—the only attack to succeed on all of our real-world datasets was ARR-OR which requires leakage of the response identity, the query equality and the order. Standard encrypted range schemes, however, do not leak the order [19], [21].

If range queries have large width or if they are skewed towards the end points, we found that leaking the response identity is risky because the GENKKNO attack was successful. If, in addition, the query equality is also leaked then we found that the ARR attack was also successful. We found that leaking the response length and the query equality on evenly distributed data could be risky in light of the APA [54] attack. However, our experiments showed that attacks that solely rely on the response length rarely worked on our datasets.

Acknowledgment

The authors would like to thank Johannes Leupold and Tobias Stöckert for implementation work. 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) — SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/251805230, and by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within ATHENE.

References

[1] M. A. Abdelraheem, T. Andersson, C. Gehrmann, and C. Glackin, "Practical attacks on relational databases protected via searchable encryption," in *International Conference on Information Security (ISC)*, 2018.

[2] D. Adkins, A. Agarwal, S. Kamara, and T. Moataz, "Encrypted blockchain databases," in *ACM Conference on Advances in Financial Technologies (AFT)*, 2020.

[3] Agency for Healthcare Research and Quality, "The National (Nationwide) Inpatient Sample (NIS) of the Healthcare Cost and Utilization Project (HCUP)," https://www.hcup-us.ahrq.gov/nisoverview.jsp, 1988, accessed 2020-11-17.

[4] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *International Conference on Management of Data (SIGMOD)*, 2004.

[5] G. Amjad, S. Kamara, and T. Moataz, "Breach-resistant structured encryption," in *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2019.

[6] G. Amjad, S. Patel, G. Persiano, K. Yeo, and M. Yung, "Dynamic volume-hiding encrypted multi-maps with applications to searchable encryption," *IACR ePrint*, vol. 765, 2021.

[7] Anaconda, Inc., "Numba – A just-in-time compiler for numerical functions in Python," http://numba.pydata.org, 2018, accessed 2021-03-25.

[8] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in *Annual International Cryptology Conference (CRYPTO)*, 2007.

[9] L. Blackstone, S. Kamara, and T. Moataz, "Revisiting leakage abuse attacks," in *Network and Distributed System Security Symposium (NDSS)*, 2020.

[10] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2004.

[11] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *Theory of Cryptography Conference (TCC)*, 2011.

[12] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2015.

[13] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *International Conference on Applied Cryptography and Network security (ACNS)*, 2005.

[14] M. Chaput, "Whoosh," https://whoosh.readthedocs.io/en/latest/, 2012, accessed 2020-10-16.

[15] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, 2010.

[16] City of New York, "Recoupment for damaged city-owned property," https://data.cityofnewyork.us/Transportation/Recoupment-for-Damaged-City-owned-Property/68k5-hdzw, 2018, accessed 2020-10-20.

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

[18] M. Damie, F. Hahn, and A. Peter, "A highly accurate query-recovery attack against searchable encryption using non-indexed documents," in *USENIX Security Symposium (USENIX Security)*, 2021.

[19] I. Demertzis, S. Papadopoulos, O. Papapetrou, A. Deligiannakis, and M. Garofalakis, "Practical private range search revisited," in *International Conference on Management of Data (SIGMOD)*, 2016.

[20] M. Esch, J. Chen, S. Weise, K. Hassani-Pak, U. Scholz, and M. Lange, "A query suggestion workflow for life science IR-systems," *Journal of Integrative Bioinformatics*, vol. 11, no. 2, 2014.

[21] S. Faber, S. Jarecki, H. Krawczyk, Q. Nguyen, M. Rosu, and M. Steiner, "Rich queries on encrypted data: Beyond exact matches," in *European Symposium on Research in Computer Security (ESORICS)*, 2015.

[22] F. Falzon, E. A. Markatou, Akshima, D. Cash, A. Rivkin, J. Stern, and R. Tamassia, "Full database reconstruction in two dimensions," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2020.

[23] 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 *IEEE Symposium on Security and Privacy (S&P)*, 2017.

[24] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *ACM Symposium on Theory of Computing (STOC)*, 2009.

[25] M. George, S. Kamara, and T. Moataz, "Structured encryption and dynamic leakage suppression," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2021.

[26] E.-J. Goh, "Secure indexes," *IACR ePrint*, vol. 216, 2003.

[27] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *ACM Symposium on Theory of Computing (STOC)*, 1987.

[28] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious RAMs," *Journal of the ACM (JACM)*, vol. 43, no. 3, 1996.

[29] Government Digital Service, "Organogram of staff roles & salaries of Government Legal Department," https://data.gov.uk/dataset/34d08a53-6b96-4fb6-b043-627e2b25840d/organogram-of-staff-roles-salaries, 2018, accessed 2020-10-20.

[30] G. Grothaus, "General implementation of the PQ-tree algorithm," https://github.com/Gregable/pq-trees, 2008, accessed 2020-10-16.

[31] P. Grubbs, A. Khandelwal, M.-S. Lacharité, L. Brown, L. Li, R. Agarwal, and T. Ristenpart, "Pancake: Frequency smoothing for encrypted data stores," in *USENIX Security Symposium (USENIX Security)*, 2020.

[32] 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 *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2018.

[33] ——, "Learning to reconstruct: Statistical learning theory and encrypted database attacks," in *IEEE Symposium on Security and Privacy (S&P)*, 2019.

[34] Z. Gui, O. Johnson, and B. Warinschi, "Encrypted databases: New volume attacks against range queries," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019.

[35] Z. Gui, K. G. Paterson, and S. Patranabis, "Rethinking searchable symmetric encryption," *IACR ePrint*, vol. 879, 2021.

[36] Z. Gui, K. G. Paterson, S. Patranabis, and B. Warinschi, "Swissse: System-wide security for searchable symmetric encryption," *IACR ePrint*, vol. 1328, 2020.

[37] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, 2020.

[38] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation," in *Network and Distributed System Security Symposium (NDSS)*, 2012.

[39] S. Jain, D. Moritz, D. Halperin, B. Howe, and E. Lazowska, "SQLShare: Results from a multi-year SQL-as-a-service experiment," in *International Conference on Management of Data (SIGMOD)*, 2016.

[40] B. J. Jansen, "Search log analysis: What it is, what's been done, how to do it," *Library & Information Science Research*, vol. 28, no. 3, 2006.

[41] B. J. Jansen and A. Spink, "How are we searching the World Wide Web? A comparison of nine search engine transaction logs," *Information Processing & Management (IP&M)*, vol. 42, no. 1, 2006.

[42] D. Jiang, J. Pei, and H. Li, "Mining search and browse logs for web search: A survey," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 4, no. 4, 2013.

[43] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-Wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, "MIMIC-III, a freely accessible critical care database," *Scientific Data*, vol. 3, no. 1, 2016.

[44] S. Jones, S. J. Cunningham, R. McNab, and S. Boddie, "A transaction log analysis of a digital library," *International Journal on Digital Libraries*, vol. 3, no. 2, 2000.

[45] E. Kacprzak, L. M. Koesten, L.-D. Ibáñez, E. Simperl, and J. Tennison, "A query log analysis of dataset search," in *International Conference on Web Engineering (ICWE)*, 2017.

[46] S. Kamara and T. Moataz, "Computationally volume-hiding structured encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2019.

[47] S. Kamara, T. Moataz, and O. Ohrimenko, "Structured encryption and leakage suppression," in *Annual International Cryptology Conference (CRYPTO)*, 2018.

[48] S. Kamara, T. Moataz, A. Park, and L. Qin, "A decentralized and encrypted national gun registry," in *IEEE Symposium on Security and Privacy (S&P)*, 2021.

[49] S. Kamara, T. Moataz, S. Zdonik, and Z. Zhao, "An optimal relational database encryption scheme," *IACR ePrint*, vol. 274, 2020.

[50] G. Kellaris, G. Kollios, K. Nissim, and A. O'Neill, "Generic attacks on secure outsourced databases," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

[51] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, 1983.

[52] E. M. Kornaropoulos, C. Papamanthou, and R. Tamassia, "Data recovery on encrypted databases with k-nearest neighbor query leakage," in *IEEE Symposium on Security and Privacy (S&P)*, 2019.

[53] ——, "The state of the uniform: Attacks on encrypted databases beyond the uniform query distribution," in *IEEE Symposium on Security and Privacy (S&P)*, 2020.

[54] ——, "Response-hiding encrypted ranges: Revisiting security via parametrized leakage-abuse attacks," in *IEEE Symposium on Security and Privacy (S&P)*, 2021.

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

[56] P. Lamesch, K. Dreher, D. Swarbreck, R. Sasidharan, L. Reiser, and E. Huala, "Using the Arabidopsis information resource (TAIR) to find information about Arabidopsis genes," *Current Protocols in Bioinformatics*, vol. 30, no. 1, 2010.

[57] C. Liu, L. Zhu, M. Wang, and Y.-A. Tan, "Search pattern leakage in searchable encryption: Attacks and new construction," *Information Sciences*, vol. 265, 2014.

[58] Y. Liu, J. Miao, M. Zhang, S. Ma, and L. Ru, "How do users describe their information need: Query recommendation based on snippet click model," *Expert Systems with Applications*, vol. 38, no. 11, 2011.

[59] E. A. Markatou, F. Falzon, W. Schor, and R. Tamassia, "Reconstructing with less: Leakage abuse attacks in two-dimensions," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2021.

[60] E. A. Markatou and R. Tamassia, "Full database reconstruction with access and search pattern leakage," in *International Conference on Information Security (ISC)*, 2019.

[61] G. Mishne and M. De Rijke, "A study of blog search," in *European Conference on Information Retrieval (ECIR)*, 2006.

[62] MIT Lincoln Laboratory, "SPARTA framework," https://github.com/mit-ll/SPARTA, 2015, accessed 2022-02-08.

[63] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2015.

[64] H. V. Nguyen, K. Böhm, F. Becker, B. Goldman, G. Hinkel, and E. Müller, "Identifying user interests within the data space-A case study with SkyServer," in *International Conference on Extending Database Technology (EDBT)*, 2015.

[65] NIST, "TREC 2014 session track," http://ir.cis.udel.edu/sessions/guidelines14.html, 2014, accessed 2020-11-17.

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

[67] G. Pass, A. Chowdhury, and C. Torgeson, "A picture of search," in *International Conference on Scalable Information Systems (INFOSCALE)*, 2006.

[68] S. Patel, G. Persiano, K. Yeo, and M. Yung, "Mitigating leakage in secure cloud-hosted data structures: Volume-hiding for multi-maps via hashing," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019.

[69] Phoenix Bioinformatics, "Araport11 genome release," https://www.arabidopsis.org/download/index-auto.jsp?dir=%2Fdownload_files%2FGenes%2FAraport11_genome_release, 2011, accessed 2020-11-17.

[70] ——, "TAIR data release," https://www.arabidopsis.org/download/index-auto.jsp?dir=%2Fdownload_files%2FPublic_Data_Releases%2FTAIR_Data_20131231, 2013, accessed 2020-11-17.

[71] R. Poddar, S. Wang, J. Lu, and R. A. Popa, "Practical volume-based attacks on encrypted databases," in *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2020.

[72] D. Pouliot and C. V. Wright, "The shadow nemesis: Inference attacks on efficiently deployable, efficiently searchable encryption," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

[73] R. G. Roessink, A. Peter, and F. Hahn, "Experimental review of the IKK query recovery attack: Assumptions, recovery rate and improvements," in *International Conference on Applied Cryptography and Network Security (ACNS)*, 2021.

[74] V. Singh, J. Gray, A. Thakar, A. S. Szalay, J. Raddick, B. Boroski, S. Lebedeva, and B. Yanny, "SkyServer traffic report-The first five years," *arXiv preprint cs/0701173*, 2007.

[75] Sogou, Inc., "SogouQ," http://www.sogou.com/labs/resource/q.php, 2008, accessed 2020-11-17.

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

[77] C. Van Rompay, R. Molva, and M. Önen, "A leakage-abuse attack against multi-user searchable encryption," *Proceedings on Privacy Enhancing Technologies (PoPETs)*, vol. 2017, no. 3, 2017.

[78] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, 2020.

[79] Walmart Inc., "Walmart recruiting - Store sales forecasting," https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting/overview, 2014, accessed 2020-10-20.

[80] W. Weerkamp, R. Berendsen, B. Kovachev, E. Meij, K. Balog, and M. De Rijke, "People searching for people: Analysis of a people search engine log," in *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2011.

[81] Wikimedia Foundation, "Simple English Wikipedia," https://simple.wikipedia.org/, 2014, accessed 2020-11-03.

[82] Yahoo Labs, "Yahoo Labs Webscope language data," https://webscope.sandbox.yahoo.com/catalog.php?datatype=l&guccounter=1, 2016, accessed 2020-11-17.

[83] Yandex N.V., "Yandex personalized web search challenge 2014," https://www.kaggle.com/c/yandex-personalized-web-search-challenge/data, 2014, accessed 2020-11-17.

[84] A. C. Yao, "Protocols for secure computations," in *Annual Symposium on Foundations of Computer Science (FOCS)*, 1982.

[85] J. Yao, Y. Zheng, Y. Guo, and C. Wang, "SoK: A systematic study of attacks in efficient encrypted cloud data search," in *International Workshop on Security in Blockchain and Cloud Computing (SBC)*, 2020.

[86] J. Zhang, *Data use and access behavior in eScience: Exploring data practices in the new data-intensive science paradigm*. Drexel University Philadelphia, PA, 2011.

[87] Y. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," in *USENIX Security Symposium (USENIX Security)*, 2016.

[88] Z. Zhao, S. Kamara, T. Moataz, and S. Zdonik, "Encrypted databases: From theory to systems," in *Conference on Innovative Data Systems Research (CIDR)*, 2021.

Appendix A. ARR with Repeating Values

Agnostic Reconstruction Range (ARR) [53] can be slightly modified in order to also cover the case of repeating values, i.e., a non-injective mapping from records to values [53]. This can be achieved by allowing the distance between values to be 0 and requires a deviation from the original pseudocode of [53] since the employed error function would be undefined if a distance of 0 was to be used. Concretely, for finding the distance $L_i = e_i - e_{i-1}$ between ordered data collection entries e_i and e_{i-1}, an error function E between pairs L_i, L_j, $j > i$ and the support size $\hat{L}_{i,j}$ estimating $L_{i,j} = L_i \cdot L_j$ is used for finding the minimum solutions L_i, L_j, thereby reconstructing the (ordered) data collection. The original error function $E_2(L_i, L_j) = \log(L_i) + \log(L_j) - \log(\hat{L}_{i,j})$ stems from a logarithmic transform of products into sums, allowing for an efficient representation of the optimization problem with a convex, linear function. However, this prevents a solution L_i to be 0, thus assuming no repeated values. We therefore use $E_1(L_i, L_j) = (L_i \cdot L_j - \hat{L}_{i,j})$, which was introduced in [53], as the error function to cover the more general and realistic case of repeated values occurring in the data collection. Additionally, the default value for lengths needs to be set to 0 rather than 1. Changing the error function also results in a new optimization problem, which is not convex in general. As a result of the more complex optimization problem, we noticed increased runtimes and attacking our MIMIC instances became infeasible (cf. §7.2.4).

Appendix B. LEAKER Code

We provide example LEAKER implementations in Listings 1 and 2.

Appendix C. Additional Data Information

C.1. Alternate Sources and Pre-Processing

Search engines. There are other query logs we could have used, including from the Excite [40], Yandex [83], and Sogou [58], [75] search engines or from Yahoo! Answers [82] or the TREC session track [65]. We preferred the AOL dataset, however, due to its large size and the fact that it comes divided by user.

The AOL query log contains queries issued between March 1st and May 31st, 2006. We discarded the 1 000 most active users from the data because they appeared to be bots. Also, 67 383 of the 2.9M keywords of the AOL query log can be found in the data collection which provides us with a large dataset.

Genetic. The TAIR query log contains all queries issued between January 1st, 2012 and April 30th, 2013 and is associated with user *sessions*. To obtain the TAIR data collection state at the time of the queries, we use the *Araport11* release [69] together with the TAIR 2013 update data [70]. Out of the 650k queries, 5 272 can be found in the collection, giving us a sufficiently large dataset[9].

Scientific data. We also identified SQLShare [39] as a potential dataset. It contains a range of scientific measurements by physicists, biologists and social scientists. However, after integrating it and analyzing the query logs we found very few range queries; a total of 12. We therefore discarded this dataset, though it could prove useful in the future if more relevant queries are added.

Census data. The SPARTA project [62] includes data and SQL query generation based on census data. We did not use this in our work, as the queries are generated randomly to fit desired response lengths, but we believe this could potentially be useful in other evaluation scenarios that require the generation of relational queries.

C.2. Statistical Analysis

We used LEAKER to analyze the datasets of §6 and describe relevant statistical insights here.

C.2.1. Data Distributions. Fig. 5 shows the frequencies of values for the datasets that we can display without violating access restrictions. Notice that Insurance has a significant skew towards low values, with values greater than 10 000 out of a maximum of $N = 25\,425$ being very rare outliers (8 out of 886). We note that this is also the case in all MIMIC data, in that most entries have a low

9. We attribute the low size of the intersection between query and data to the missing publications and people datasets, which have not been released for download.

Listing 1: Slightly simplified example of LEAKER code for implementing the basic count attack (Algorithm 1 of [12]) using co leakage.

```
class BasicCount(KeywordAttack):
    def __init__(self, known_data_collection):
    # Set up self._known_keywords the set of known keywords, self._known_coocc the known
        ↪ co-occurence matrix, and _known_unique_rlens mapping unique rlens to known keywords.

    @classmethod
    def required_leakage(cls):
        return [CoOccurrence()]

    def _known_response_length(self, keyword):
        #rlen is the diagonal of co matrix
        return self._known_coocc.co_occurrence(keyword, keyword)

    def __initialize_known_queries(self, queries, rlens):
        return {i: self._known_unique_rlens[rlens[i]] for i, _ in enumerate(queries) if rlens[i]
            ↪ in self._known_unique_rlens}

    def recover(self, data_collection, queries):
        coocc = self.required_leakage()[0](data_collection, queries)
        rlens = [coocc[i][i] for i, _ in enumerate(queries)]

        known_queries = self.__initialize_known_queries(queries, rlens)

        while True:
            unknown_queries = [i for i, _ in enumerate(queries) if i not in known_queries]
            old_size = len(known_queries)
            for i in unknown_queries:
                candidate_keywords = [k for k in self._known_keywords if k not in
                    ↪ known_queries.values() and rlens[i] == self._known_response_length(k)]
                    for s in candidate_keywords[:]:
                        for j, k in known_queries.items():
                            if coocc[i][j] != self._known_coocc.co_occurrence(s, k):
                                candidate_keywords.remove(s)
                                break
                if len(candidate_keywords) == 1:
                    known_queries[i] = candidate_keywords[0]
                    if old_size >= len(known_queries):
                        break

        uncovered = []
        for i, _ in enumerate(queries):
            if i in known_queries:
                uncovered.append(known_queries[i])
            else:
                uncovered.append("")

        return uncovered
```

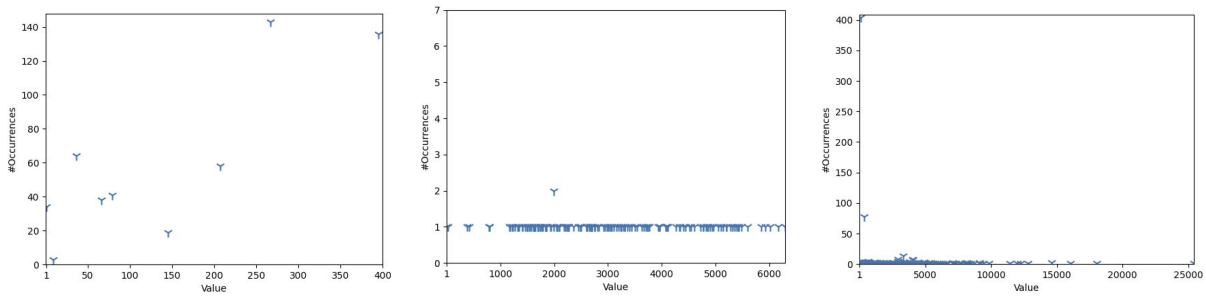


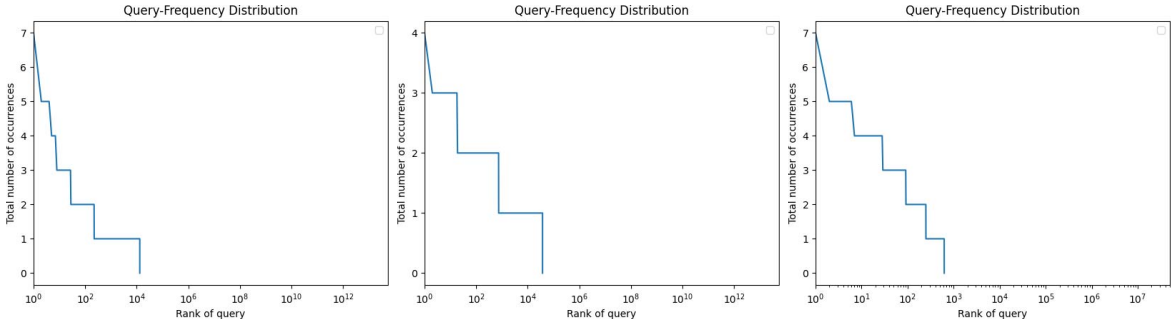
Figure 5: Frequencies of numerical dataset values.

value: for MIMIC-T4, only 301 out of 8 058 entries have a value greater than 20 out of a maximum of $N = 73$; for MIMIC-PC only 9 out of 7 709 entries have value greater than 500 out of a maximum of $N = 2\,684$; and for MIMIC-CEA only 25 out of 2 844 entries have value greater than 2 500 out of a maximum of $N = 9\,978$.

In contrast, this is clearly not the case for Salaries and Sales, where no such skew is noticeable in Fig. 5. Since we notice that different attacks behave very differently according to whether this skew exists (cf. §7),

Listing 2: Simple example of LEAKER code for implementing the co pattern (without pre-computation and caching).

```
class CoOccurrence(LeakagePattern):
    def leak(self, data_collection, queries):
        doc_ids = {q: map(lambda doc: doc.id(), data_collection(q)) for q in queries}
        return [[len([i for i in doc_ids[qp] if i in doc_ids[q]]) for qp in queries] for q in
            ↪ queries]
```



SDSS-M | SDSS-L | Zipf distribution ($a = 5$) restricted to 0.002% of all queries

Figure 6: Query frequency distribution of SDSS query logs on the PhotoObjAll.dec collection (here scaled by $\times 10^5$; $N = 10\,455\,488$) as well as an artificial Zipf distribution on a random collection with $N = 10^4$.

we call the former case with the skew an *uneven* data distribution in our potential general risk factors (cf. §4), and consequently we denote the latter case as an *even* data distribution.

C.2.2. Keyword data – Frequency distribution. The frequency of queries has been identified as the main attack performance metric [9], [73] and, therefore, we analyzed the frequency distribution of queries issued in real-world systems. In particular, [9] already noted that keyword data usually follows the Pareto principle, i.e., the probability mass function is *heavy-tailed* with the bulk of the keywords appearing in a few documents. This was used by [9] to argue that, because most keywords appear in the tail with a low occurrence, most queries might have a very low frequency as well. A similar argument for low-frequency keywords in real-world queries can be found in [73]. Being able to see which keywords are queried in real systems for the first time, we noted that this is not the case for any of our data sources: The queries are usually not from the tail of the data distribution and have a high frequency (a mean of 1 804 for AOL, 2 023 for TAIR and 326 for GMail). Only Drive has a mean query frequency of 11.2, which was considered as *pseudo-low* by [9]. We conclude that, in our evaluations, *users are not interested in querying keywords of a low frequency* and conjecture that they may rely on the system's ranking to obtain the desired results.

Additionally, we investigated if the activity of a user has an effect on their queries' frequencies, but found no correlation between number of queries and mean frequency (Pearson correlation of about 0.1 for TAIR and 0.014 for AOL).

C.2.3. Range data – Query distribution. The core of range attack analysis has been the query distribution. The heavily-used uniform distribution is an unlikely case in the real world, and while specific parametrizations of the beta (family) distribution were considered [53], [54] and already provide important insights, these do not have any empirical basis. Using real query logs (cf. §6.2), we investigated two major factors of query distributions: query frequencies and their widths.

We plot frequencies of *all possible* queries for SDSS-M and SDSS-L in Fig. 6 as well as a comparable Zipf distribution. The case of SDSS-S does not need to be plotted, as queries only appear once or twice. The Zipf distribution has a probability mass function of

$$p(k) = \frac{k^{-a}}{\zeta(a)},$$

where ζ is the Riemann Zeta function and a is the shape parameter. This means that an element's frequency is inversely proportional to its rank among all elements according to decreasing frequency. From Fig. 6, we deduced that queries are roughly sampled according to a Zipf distribution, but this only holds for a *tiny fraction of queries*.

We also looked at the query widths and found that they are fixed: SDSS-S either has a width of 113 or 112, while sizes range between 161 and 173 for SDSS-M and 51 and 61 for SDSS-L.

Based on these empirical observations, we considered a new kind of query distribution in our experiments for data without query logs, where queries are distributed according to Zipf, but, in contrast to [53], [54], they are restricted to specific widths and a fraction of possible queries before the probabilities are assigned. However, since this analysis was confined to one specific case, it might not be representative. While we consider a large fraction of possible queries missing as intuitive, the fixed-sized widths might be unique to SDSS and we expect more variable widths in other cases. We thus varied the *upper bound* of widths in our experiments. For some attacks, a

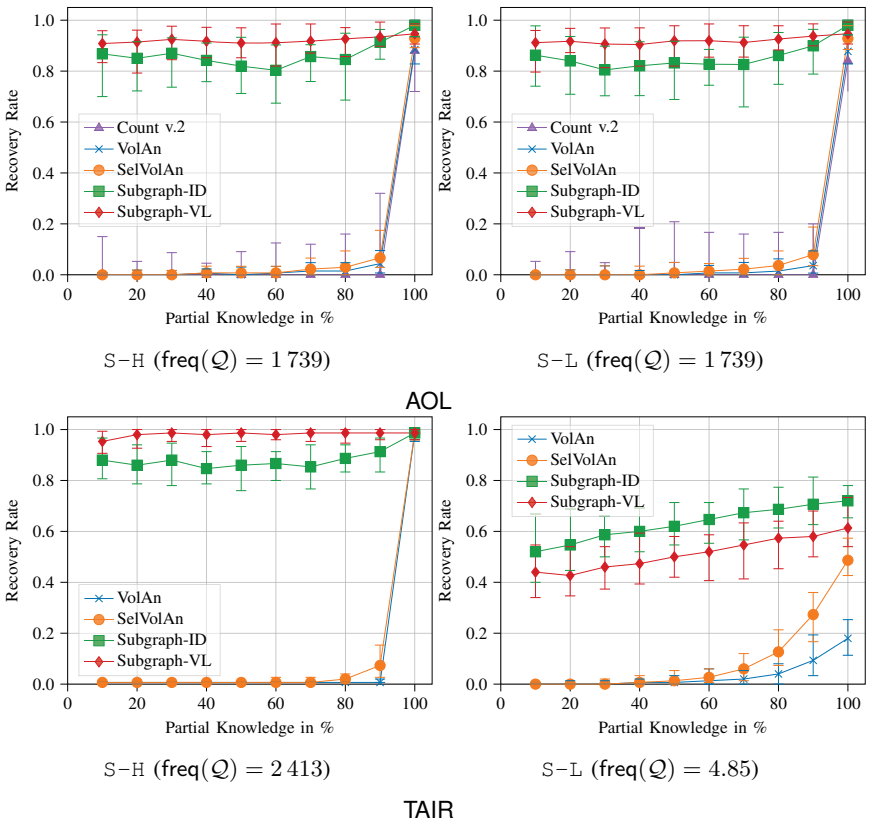


Figure 7: X−Y attack evaluations against AOL and TAIR. All evaluations are 5×5, except for 3×3 evaluations done for COUNT V.2. 150 queries are drawn for each of the most active users (single user setting; S) without replacement according to their query frequency from the 500 most (H) or least (L) frequent queries in the query log that are also contained in the partial knowledge, respectively. The resulting mean frequency is given by $\text{freq}(\mathcal{Q})$.

large upper bound (close to N) has been identified as a risk factor (cf. §7).

Appendix D. Additional Evaluations for Most Active Users

We show further evaluations of the AOL and TAIR datasets in Fig. 7 for the single user setting using five users with the highest activity. Note that the results are not significantly different to the least active users case in Fig. 3 in §7.1. This confirms our statistical analysis that activity does not influence the frequency of queries (cf. App. C.2). Further, we note an anomaly of a low amount of unique queries for AOL's most active users, resulting in equal query spaces for highest and lowest frequency.