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

Seny Kamara[1], Abdelkarim Kati[2], Tarik Moataz[3], Thomas Schneider[4], Amos Treiber[4] and Michael Yonli[4]

[1]Brown University, seny@brown.edu
[2]Mohammed-VI Polytechnic University, abdelkarim.kati@um6p.ma
[3]Aroki Systems, tarik@aroki.com
[4]Technical University of Darmstadt,
{schneider, treiber}@encrypto.cs.tu-darmstadt.de, michael.yonli@stud.tu-darmstadt.de

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 by Islam et al. (NDSS'12), many attacks have been proposed that exploit different leakage profiles under various assumptions. While they aim to improve our common understanding of leakage, it is difficult to draw definite conclusions about their practical risk. This uncertainty stems from many limitations including a lack of reproducibility due to closed-source implementations, empirical evaluations conducted on small and/or unrealistic data, and reliance on very strong assumptions that can significantly affect accuracy. Particularly, assumptions made about the query distribution do not have any empirical basis because datasets containing users' queries are hard to find.

In this work, we address the main limitations of leakage cryptanalysis. First, we design and implement an open-source framework called LEAKER that can evaluate the major leakage attacks against a given dataset and can serve as a common leakage analysis reference for the community. 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 evaluate known attacks on our datasets to assess their practical risks and gain insights about the properties that increase or diminish their accuracy.

Keywords: Encrypted Search, Cryptanalysis, Leakage Attacks

1 Introduction

Encrypted search algorithms (ESAs) allow a user to encrypt its data while preserving the ability to search over it. ESAs have received much attention due to applications to cloud storage and database security, see the survey by Fuller et al. [23] for an ESA scheme overview and examples of commercial products. 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 a variety of cryptographic techniques: property-preserving encryption (PPE) [3, 7], searchable/structured symmetric encryption (SSE/STE) [69,

18, 14], fully-homomorphic encryption (FHE) [24], functional encryption [10], or oblivious RAM (ORAM) [27]. These approaches 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 data so they are usually considered impractical. For practical purposes, one needs sublinear ESAs: Oblivious ESAs based on ORAM require $\mathsf{opt} \cdot \log^{O(1)} n$ time, where opt is the optimal time to search and n is the amount of data blocks of the data collection. On the other hand, structured and property-preserving ESAs, based on STE or PPE respectively, can achieve optimal search time.

Adversarial models and leakage. ESAs can be analyzed in different adversarial models. The most common are the *snapshot* and the *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 [4], whereas in the persistent model zero-leakage ESAs can be designed using FHE at the cost of linear-time queries. In the persistent model, however, oblivious, structured, and property-preserving ESAs all leak some information; though recent work has shown how to suppress some of this leakage for structured ESAs [43, 42, 25]. For instance, prominent leakage profiles include the *response identity* (or access) pattern, revealing identities of data entries matching the query, or the *query equality* (or search) pattern, revealing 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. [34], leakage attacks were first designed against structured ESAs in the persistent model. Later, Naveed et al. [55] designed attacks against PPE in the snapshot model and Kellaris et al. [44] showed attacks against oblivious ESAs in the persistent model.

These works have been improved by a series of papers [11, 49, 30, 31, 32, 8, 47, 59, 48, 66]. While they present interesting results and aim to improve our common understanding of leakage, it can sometimes be difficult to draw definite conclusions about their practical relevance. This is due to several limitations: (1) With the exception of [66], no implementations are available to replicate results or reproduce them outside the original setting; (2) prior evaluations have been limited to just a few, small datasets that might not be reproducible in other settings; (3) attacks or their evaluations made assumptions that might not be realistic, most prominently *sampling queries without any empirical foundation.* A major open problem to justify these assumptions is to evaluate the attacks on real-world query distributions, as identified in, e.g., [32, 66]. This has been a long-standing challenge because real-world query data is very hard to find. Therefore, it is inconsistently modeled in evaluations, e.g., with keyword attack evaluations using the most frequent [34, 11] or infrequent [8, 66] keywords in the data collection, which significantly affects results.

Standardization. While the implications of leakage attacks remain ambiguous, efforts to deploy ESAs are continuing to move forward. For example, NIST's recent standardization intentions of privacy-preserving technologies include, among others, ESAs and call for reference material and security analyses [58]. Thus, a common attack evaluation framework and more realistic evaluations are paramount to enable a widespread, secure use of ESAs and their standardization.

Our contributions. With our work, we address the main limitations of ESA leakage cryptanalysis by making 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 of leakage attacks or mitigation, and with it we enable and invite the community to contribute further constructions and evaluations to continually advance our common 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 been a long-standing and 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 variety of datasets we consider covers significantly more settings than those used in previous work and, in most cases, our data is at least one order of magnitude larger.

3. We use LEAKER to evaluate oblivious and structured ESA attacks on our broad real-world query logs and data. Our analysis is an important step towards a practical understanding of leakage by using real-world and as-realistic-as-possible queries across settings. It shows that the impact of existing keyword attacks is more nuanced than previously thought. In particular, two major attacks, the IKK [34] and the Count attacks [11], do not work well on our real-world datasets. However, our analysis surprisingly shows that the BKM attacks [8] can have significant recovery rates in many realistic cases, contradicting previous intuitions [8, 66]. Similarly, for range ESAs, our analysis shows that none of the established attacks work against our real-world query logs and data collections. However, when using synthetic query distributions rooted in observations about our query logs, the results are different as we identify specific settings in which the attacks can be practical.

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 LEAKER. Our novel real-world datasets are presented in §6 and our evaluation of attacks on them is shown in §7. We conclude our work in §8.

2 Related Work

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

Searchable/structured encryption. Searching on encrypted data was first explicitly considered by Song, Wagner and Perrig [69], though previous work by Goldreich and Ostrovsky on ORAM [27] could also be used. Security definitions for searchable symmetric encryption (SSE) were proposed by Goh [26] and Chang and Mitzenmacher [12]. Curtmola et al. [18] 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. Index-based SSE constructions were later generalized as structured encryption (STE) by Chase and Kamara [14]. Boneh et al. [9] first considered the problem of public-key searchable encryption.

Fuller et al. [23] provide a survey of encrypted search and Yao et al. [77] provide a survey of leakage attacks describing known attacks. Our work focuses on evaluating attacks in realistic environments.

Query log analysis. Researchers across fields have tried to better understand users' search behavior. This is known as *query log analysis* or *search log analysis*, surveyed in [36, 37, 38].

Many works analyze query logs gathered by proprietary systems that only the authors had access to. Unfortunately, almost all of these works on restricted-access logs [40, 37, 54, 73, 41] do not present results we deem related to leakage attacks, given that we barely found selectivity information—the main success parameter for keyword attacks—in keyword log or actual range queries in numeric log analyses. 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 corresponding 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 *selectivity* of a keyword query w is the number of entries in $\mathbf{D}$ 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 [8]:

- 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: $\mathsf{vol}\,(\mathbf{C}, q_1, \ldots, q_t) = \Big((|e|_b)_{e \in \mathbf{C}(q_1)}, \ldots, (|e|_b)_{e \in \mathbf{C}(q_t)}\Big)$.

Table 1: Major leakage attacks and our estimated general 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 successful real-world instances, ◑ 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 evaluated instances.

Attack	Target	Leakage Profile (§3)	Auxiliary Data		Assumptions		Risk (§7)
			Queries	Data	Queries	Data	
IKK [34]	K	co	Partial	Partial	Non-rep.	–	○
DetIKK [66]	K	co	–	Partial	Non-rep.	–	◑ K
Count v.2 [11]	K	co	–	Partial	Non-rep.	–	◑ K
Subgraph-ID [8]	K	rid	–	Partial	Non-rep.	–	●
Subgraph-VL [8]	K	vol	–	Partial	Non-rep.	–	●
VolAn [8]	K	tvol	–	Partial	–	–	◑ K
SelVolAn [8]	K	tvol, rlen	–	Partial	–	–	◑ K
LMP-RK [49]	RV	rid, rank	–	–	–	Dense	◑ D
LMP-ID [49]	RV	rid	–	–	–	Dense	◑ D
LMP-App [49]	RV	rid	–	–	–	Dense	◑ D
GenKKNO [31]	RV	rid	–	–	Uniform	–	◑ W∨¬E
ApproxValue [31]	RV	rid	–	–	Uniform	Specific	◑ S∧(W∨¬E)
ARR [47]	RV	rid, qeq	–	–	–	–	◑ W
ARR-OR [47]	RV	rid, qeq, order	–	–	–	–	●
GLMP [30]	RC	rlen	–	–	$\mathcal{A}$	–	○
GJW-Basic [32]	RC	rlen	B	–	$\mathcal{A}_B$	–	○
GJW-Missing [32]	RC	rlen	B, k	–	$\mathcal{A}_{B,k}$	–	○
APA [48]	RC	rlen, qeq	–	–	–	–	◑ E

- The *total volume* pattern tvol leaks the total bit length of matching entries: $\mathsf{tvol}(\mathbf{C}, q_1, \ldots, q_t) = \left(\sum_{e \in \mathbf{C}(q_1)} |e|_b, \ldots, \sum_{e \in \mathbf{C}(q_t)} |e|_b\right)$.

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

4 Existing Leakage Attacks

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

Adversarial model. The two main adversarial models considered against ESAs are *snapshot* adversaries 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 which corrupts a server and observes all of its interactions.

Target. A leakage attack can target different information. 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 issued by users. 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.

Risk factors. In our work, we consider attacks uncovering more than 15% of queries or having a reconstruction error less than 15% in realistic conditions to pose noteworthy risk. This depends on a plethora of factors, which we identify in §7. They stem from observations of our concrete evaluation instances and can serve as *general intuitions* for other cases that share these aspects and, hence, might be at risk as well. Practically notable factors to keep in mind concern the adversary's knowledge (if the attack only works when knowing a significant fraction of the data), the query width, and the data (density and whether specific values appear). In addition, we observed that a skew in the data towards endpoints (1 or N) can result in significant effects and, hence, denote data with such a skew as unevenly and data without this property as evenly distributed. More details on these skews are given in App. C.2.1.

Our work focuses on attacks in *strong* security models, i.e., passive attacks against structured or oblivious ESAs by persistent adversaries that only observe leakage patterns of a fraction of possible queries, which have been issued by the user. Additionally, we limit ourselves to *known-data* and *known-query* attacks and attacks that require no auxiliary data. We thus leave a realistic modeling of sampled-data or sampled-query attacks as future work, as they require additional, related real-world data from distinct sources. We summarize the properties of major leakage attacks we evaluated and our determined general risk (cf. §7) in Tab. 1. Unless noted otherwise, an attack acronym stems from the author names.

4.1 Attacks Against Keyword ESAs

For keyword search, the major class of attacks are *known-data attacks* that require some partial knowledge of the data collection.

The IKK attacks. The first leakage attack was described by Islam et al. [34], 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* [45] to probabilistically find an approximation. Originally introduced as a sampled-data attack, we evaluate it as a known-data attack as in [11]. Recently, Roessink et al. [66] showed how it can be improved with deterministic techniques from the Count attack [11] (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. [11]. Like IKK, Count is a query reconstruction attack that exploits the co-occurrence pattern co. There are two versions of this attack. 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. [8] 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.

Prior evaluations. The above attacks have solely been evaluated on the Enron e-mail dataset [16], with [8, 66] also using public e-mail sets [17]. Since only the implementation of [66] is available, replicating results or comparing the attacks encompasses major efforts. It is therefore cumbersome to evaluate new data sources to, e.g., show an overfitting of the attacks to e-mail data. Furthermore, no evaluation has considered real-world queries but rather just sampled keywords from the data collection in an inconsistent manner (e.g., highest-selectivity [34, 11] vs. lowest-selectivity [8, 66]). Given that this enables or breaks the attacks [8, 66], evaluation on real-world queries remains a major open problem that we tackle.

Other attacks not covered. An attack that leverages qeq with auxiliary query frequency samples was given by Liu et al. [51], which was recently improved with an attack of Oya and Kerschbaum [59] also using rid. Active file-injection attacks are considered by Zhang et al. [79], Blackstone et al. [8], as well as Poddar et al. [64]. Additional specialized attacks against *specific* ESA instantiations were considered in [65, 70, 1]. We focus on general passive attacks that do not rely on query samples close to the actual distribution.

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. [44]. 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. [49] 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 data. 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. [30] 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. [32] 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 recent works have attempted to weaken the assumptions needed by the KKNO attacks and their extensions. Grubbs et al. [31] 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. [47] 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 [47] 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. [48] also applied support size estimation to the setting where only the response length and query equality patterns (rlen and qeq) are available,

resulting in an approximate count reconstruction attack[1]. 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 [21, 19]. As a result, the attack is *parameterized* by the underlying ESA, and denoted by APA (agnostic parameterized attack). Crucially, APA requires no assumptions about the query distribution and also deals with non-dense data, for which the GLMP and GJW attacks require auxiliary data information. Another new method within the attack shows how to deal with multiple possible reconstructions.

Prior evaluations. No implementations of the previously mentioned attacks are publicly available and, thus, comparative or replicative research is severely hindered, as for the keyword case (§4.1). For their evaluations, [44, 30, 32, 48] all use relatively small subsets of the HCUP [2] medical dataset and the remainder of the above attacks were never evaluated on real-world data. To the best of our knowledge, no prior evaluation considered real-world queries, which was identified as a major challenge in range ESA research [32]. The real-world impact of range attacks is therefore left open.

Other attacks not covered. The above works also introduced sampled-data variants: A version of LMP [49] uses auxiliary data information to require fewer queries, while Grubbs et al. [30] also incorporate frequency information. ApproxData [31] demonstrates how to use ApproxOrder to uncover exact values if density and an auxiliary data distribution are given, and [32] show how GJW-Basic can be improved with side information about the data. [30] also recover values of update operations in case the frequencies have already been determined. k-NN queries are considered in [46, 47] and Falzon et al. [22] recently moved range data reconstruction for a one-dimensional collection to one for two columns. We focus on passive attacks in the established single-column range query setting that do not rely on data samples, which we described above. Also note that theoretical attacks were given in [53], which we did not consider as they assume that all possible queries are issued.

5 The LEAKER Framework

Here, we discuss the design and architecture of our framework for evaluating leakage attacks on real-world data.

Motivation & overview. Looking at the prior evaluations summarized in §4, it clearly is currently very hard to evaluate the effectiveness of leakage attacks, mainly because of the closed-source implementations. This hinders research and practice in encrypted search as the community is not able to verify claims (replicate results) or evaluate attacks in environments other than the ones chosen by the original authors (reproduce results). Another challenge is that real-world data and query logs are very difficult to obtain. Because of this, leakage attacks are evaluated on datasets without query logs, i.e., no real-world queries, leading to questionable conclusions. Thus, for continuously developing a better understanding of the risk of ESA leakage and contributing to standardization efforts [58], there is urgent need of a *common reference toolkit* for the community to easily integrate and compare new attacks and evaluate attacks on new data sources or countermeasures.

To satisfy these demands, we designed and implemented our framework with the corresponding 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.

[1] It uncovers the ordered values, which is equivalent information for integer entries.

- *comparisons:* LEAKER includes implementations of the main leakage attacks for both point 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 and further advancements of the understanding of risk (cf. §7) by contributing evaluation results.
- *usability:* LEAKER is interoperable with many platforms and does not require domain-specific knowledge – one only needs to specify the data sources and evaluation criteria.
- *open-source:* LEAKER is freely available as open source 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 (of which 1 148 are tests). LEAKER evaluates a leakage attack on a *data collection* and 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 an 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 behind data collections and query logs is important because, unlike previous work, one of our main goals is to evaluate attacks on large and realistic datasets. Most previous work used small datasets with, e.g., 500 keywords [11]. With LEAKER, we can now evaluate that attack on datasets with more than 250 000 keywords (cf. §6). To achieve this, we use NumPy [33] arrays to store and process numerical data and Whoosh [13] 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 [6]. 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. Examples of attack and pattern implementations are given in Listings 1 and 2 in App. B.

[2]For these two attacks we used synthetic data because we could not access 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 [60]	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

All attacks in LEAKER's library are purely in Python except for ApproxOrder [31], where we use a C++ implementation of PQ-trees [29]. Since we could not find any public Python-compatible implementations of the Jackknife or Valiant-Valiant estimators used in the ARR and APA attacks [47, 48] we implemented them ourselves in Python.

Evaluator. The evaluation module evaluates attacks. Given an attack and a leakage pattern from the library and a data collection and query log from the datastore, an evaluator proceeds as follows. It first creates the *observed leakage* by evaluating the leakage patterns on the data collection and query log. Since the observed leakage 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 evaluate attacks that require known data and/or queries, it first executes an attack-specific sampling algorithm to choose 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 are (sometimes combined) LEAKER outputs.

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

In many instances, attack evaluations were limited due to small and/or non-representative data collections without any query logs, which are hard to find. To address this, we identified completely new datasets including query logs that we describe here and believe to capture more realistic scenarios than previously used data. We show alternative sources and pre-processing details in App. C.1 and more involved statistics in App. C.2. All pre-processing is already integrated into LEAKER and allows our (public) data sources to easily serve as more realistic benchmarks for future work.

6.1 Keyword Data

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

Search engines. A major application of ESAs are encrypted search engines, e.g., for desktop search applications or to add search capabilities to email clients or file managers.

Table 3: Summary of our *scientific data* range query logs on the PhotoObjAll.dec collection [67] ($n = 5\,242\,134$ 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

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 private data of 7 voluntary participants by providing scripts to locally extract their GMail and Google Drive query logs and data collections as LEAKER inputs. 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 for 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.

Given that the number of private users we had access to was so small and because we cannot release their data, we also evaluated *public* search engine data. Specifically, we used Wikipedia [74] as data collection and the AOL dataset [60] as query log. 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. Nevertheless, we believe that Wikipedia and the AOL queries provide a good model of English language data and search queries.

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 [50] as data collection as well as its publicly released query log [20] to obtain queries. 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 to 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, truncating, and mapping to the integers which allows us to evaluate 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 an even or uneven data distribution (cf. §4).

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 [67]. In addition to astronomy, the SDSS has also been used to investigate user behavior [78, 56]. We used the SDSS to create one data collection and 3 query logs (cf. Tab. 3) with

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

Case	Data collection	Scale	n	N	δ (%)	Distr.
Medical	MIMIC-T4 [39]	×10	8 058	73	80.8	¬E
	MIMIC-PC [39]	×10	7 709	2 684	8.6	¬E
	MIMIC-CEA [39]	×1	2 844	9 978	3.3	¬E
HR	Salaries [28]	×0.01	536	395	2.3	E
Sales	Sales [72]	×1	143	6 288	2.3	E
Insurance	Insurance [15]	×1	886	25 425	1.2	¬E

scale factor 100.

Medical. Due to high sensitivity, medical data has long been proposed as an ESA application and many works used medical datasets like HCUP [2] to evaluate attacks [55, 44, 49, 30, 32, 48]. While HCUP is a real-world dataset with millions of patients, the attributes considered usually have a small domain, e.g., the patient's age. A broader dataset for *medical test data* is the *Medical Information Mart for Intensive Care* (MIMIC) dataset [39] which includes lab 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 [28].

Sales. Sales data can also be sensitive as it contains trade secrets. We use data released by Walmart for a prediction competition [72], 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 [15] released by the New York Department of Transportation. They were filed with Allstate in September 2018.

6.3 Privacy Considerations

None of the experiments described in this work required 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 and the MIMIC data. We obtained consent from the participants to publish the attack evaluations and some statistics of their search engine data. Access to PhysioNet's MIMIC [39] data is restricted and was only handled by approved authors who completed the required training courses and strictly adhered to PhysioNet's data use agreement.

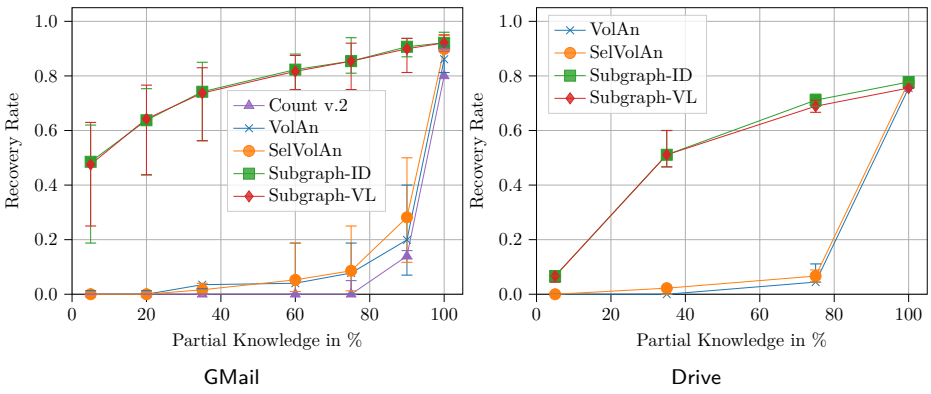


Figure 1: 3×3 private data evaluations (full query sampling, no repetitions, and single-user setting). Only one user evaluated COUNT v.2 for GMail.

7 Real-World Attack Evaluation

In this section, we use LEAKER to evaluate all attacks described in §4 on the datasets from §6 and identify major parameters that impact each attack's accuracy. As prior evaluations (cf. §4, App. C.2) were restricted to one respective data setting and sampled queries without any empirical basis, we believe our independent evaluation on a range of novel datasets *using queries from real-world sources* provides a more accurate understanding of attack practicality. We again stress that understanding attacks under real-world queries has been identified as a major open challenge in the area [32, 66].

7.1 Keyword Attacks

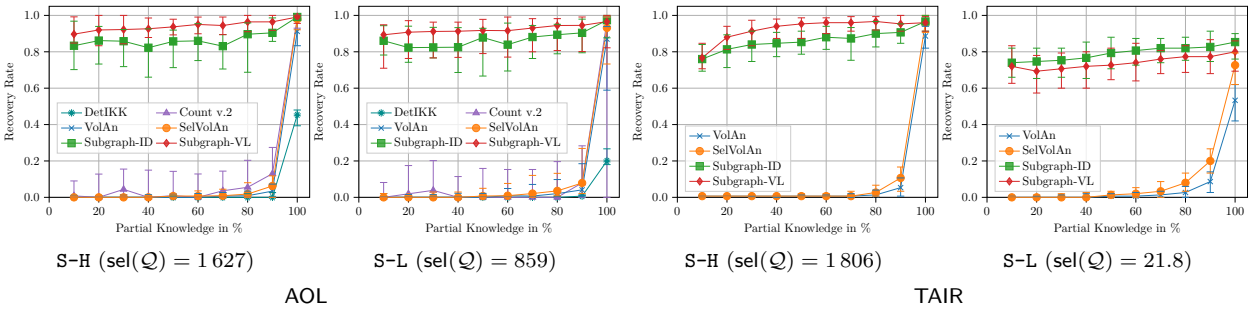
We present results for the private GMail and Drive logs in Fig. 1 and results for the public AOL and TAIR query logs in Figs. 2 and 3. Further plots are given in App. D. Prior to describing our results, we introduce relevant parameters and our experimental setting.

Selectivity. Each query in the query log matches a number of entries in the dataset, its *selectivity*. We investigated two settings: *high* selectivity and *low* selectivity. The former includes an average selectivity ranging from 1 806 to 5 707, whereas for the latter, it ranges from 1 to 859 depending on the queries found in the query log.

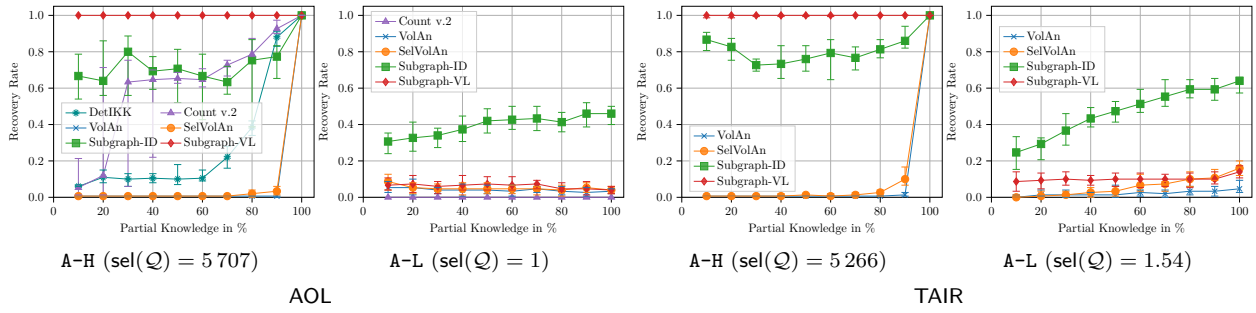
Number of users. When it comes to the number of users, there are two main settings in which structured and oblivious ESAs can be deployed: *single-user* or *multi-user*. The former characterizes a setting in which a single user queries its own dataset, while the latter depicts settings in which multiple users query the same dataset. For the single-user setting, our evaluation consists of taking the average over 5 distinct users with the additional requirement of having each query log composed of at least 2 000 queries, while the multi-user setting is simply composed of all users in the dataset, which accounts for 656 038 users for AOL and 1 263 for TAIR. For the private Google data, we only evaluated the single-user setting, given that each user queried their own data collection.

User activity. We noticed that users have different query activities, with some issuing a lot of queries while others are less active. In our evaluation of public data sources, the most active user issued 6 389 queries while the least active had 2 000 queries. As we did not find any noticeable difference, only the least active case is given in Fig. 2, with the case of most active users being given in Fig. 7 in App. D.

Query sampling. Dealing with real-world datasets, query logs can be extremely large,



(a) Evaluation for *single* users (S)



(b) Evaluation for *all* users (A)

Figure 2: 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 (a) each of the least active users (single user setting; S) or (b) all users (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 selectivity is given by $\mathsf{sel}(\mathcal{Q})$.

and given the non-trivial computational complexity of the evaluated attacks, we were constrained to limit the number of queries we sample from the query log. There were two main approaches to our sampling: *full* and *partial* sampling. The full sampling outputs a new, smaller query log composed of keywords independently of whether they exist in the adversarys' partial knowledge or not, i.e., it is possible that a keyword appears in the query log but is unknown to the adversary. In contrast, the partial sampling generates a new, smaller query log that is instead only composed of keywords known to the adversary. The partial sampling, while seeming unrealistic at first, captures a worst case in which a user only queries for keywords that appear at least in one of the records known by the adversary. We varied the maximum number of keywords of any public query log to be equal to 500, 2500, and 5000. Because we did not observe major deviations in other sizes, we show only the results for size 500. For the private query logs in Fig. 1, the size was equal to the entire length, i.e., the attacks targeted the entire query logs.

Query repetition. We observed that some of the queries in the query log are repeated; previous works assumed however that all queries are distinct. This assumption makes sense when the adversary can distinguish between queries based on the query equality, which is disclosed by many structured ESAs; but the assumption no longer holds when dealing with newer structured ESAs [43, 25] or oblivious ESAs. Given that some of the attacks apply to both structured and oblivious ESAs and may be affected by this, it was then 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 dataset, and a specific combination of the parameters highlighted above, we sample a new query log and dataset. For a fixed rate of the adversarial knowledge, we first sample a subset of the entire dataset ℓ times, and for each dataset sample, we sample a subset of the query log λ times, which we denote by a $\ell \times \lambda$ evaluation and accounts for $\ell \cdot \lambda$ evaluations. For most experiments we picked $\ell = \lambda = 5$ and display the median, maximum and minimum recovery rate.[3] In the public data setting, we attack 150 queries drawn from the query log according to their frequencies therein. We denote by `X-Y` a setting in which the type of user and the selectivity are set to be equal to `X` and `Y`, respectively.[4] Recall that `X` can be set to *all* (`A`; multi-user setting) users or the single-user setting (`S`). `Y` can be determined as *low* (`L`) or *high* (`H`) selectivity. 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.

7.1.1 The IKK Attacks [34, 66]

Given that IKK [34] has costs quadratic in the number of keywords, running it on TAIR was infeasible so we ran this evaluation solely on the AOL log.

We saw that IKK does not work in any of our settings. This stands in contrast to previous results for full knowledge [11, 66], where much smaller data was considered. The best recovery we achieved is less than 15% even with full knowledge in the `A-H` setting (all users, high selectivity) - which is the *least* realistic setting. We stopped the attack's annealing process after it ran for 48 hours. We attribute these results to a large search space, as DetIKK [66] with a reduced number of possible states does not suffer from this in the high selectivity cases (cf. Fig. 2). However, similarly to Count v.2 (see next), we consider DetIKK as successful only in high partial knowledge settings.

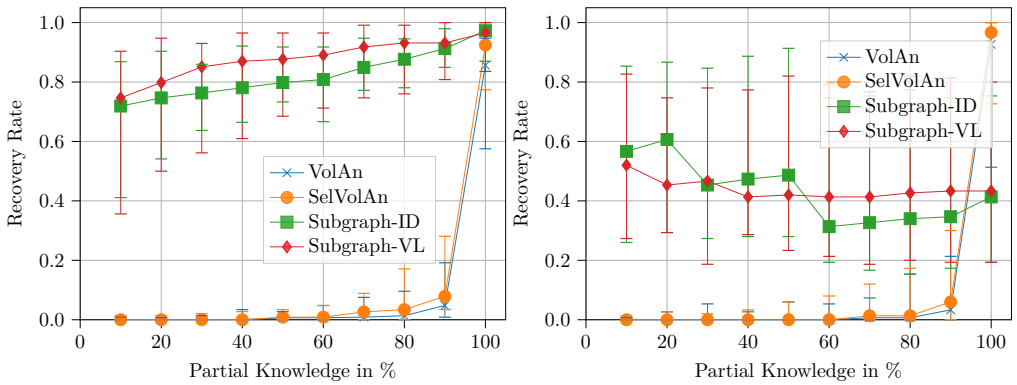
7.1.2 The Count Attack [11]

Similar to the IKK attack, Count v.2 [11] also has costs quadratic in the amount of keywords and we only ran it on the AOL query log.

In our evaluations, it only worked without perfect partial knowledge in the `A-H` setting with a recovery of 63% when the adversary knows 30% of the dataset. Note that in this setting the selectivity is highest with a mean equal to 5 707. This aligns with the results stated in [8]. However, we observed that for all other settings, the attack failed to obtain an adequate recovery rate. E.g., in the `S-L` setting, it has a recovery rate equal to 8% even when the adversary knows 90% of the dataset (cf. Fig. 2a for more data points). We thus consider it to only work with high partial knowledge in practice.

[3] Note that we limited the number of re-evaluations to 25 *per user* due to the non-trivial computational overhead incurred by some of the attacks. As an instance, the Count v.2 attack took one day to complete a single iteration for 5 users.

[4] 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.



(a) No restrictions (b) With replacement

Figure 3: 5×5 evaluation of [8] on lowest-selectivity queries from the 5 least frequent AOL users for different query sampling/repetition scenarios (samples are not restricted to the partial knowledge or are sampled with replacement).

7.1.3 The BKM Attacks [8]

The BKM attacks [8] had efficient runtimes and we were able to evaluate them using all public and private query logs, which for the latter evaluations involved less powerful commodity machines.

The Subgraph framework. Against private data (Fig. 1), both attacks did surprisingly well. For the email case we see that even at 5% knowledge, more than 18.75% of the real-world queries are reconstructed.[5] Despite the limited number of users, the relatively low variablitly for GMail shows consistent recovery always greater than 18.75%. For Drive, they have a much lower recovery for a low partial knowledge, though both attacks already uncover around half of the queries with 35% partial knowledge.

In the public data setting (Fig. 2), both SUBGRAPH attacks also achieve non-trivial recovery rates. SUBGRAPHVL achieves very high rates almost independently of the adversarial knowledge for high selectivities or whether the query log is for single or aggregated users, i.e., for all high-selectivity settings (`.-H`); the SUBGRAPHID attack however has a slightly lower recovery rate but still recovers significant information, e.g., 86% in the `S-H` setting for AOL with partial adversarial knowledge as small as 10%. For low selectivities (`.-L`), both of these attacks still perform well for a selectivity average as low as 4.85 (cf. Fig. 7 in App. D). For extremely low selectivities equal to 1, however, we found that SUBGRAPHVL does not work at all, while SUBGRAPHID can still correctly recover queries at a low yet significant rate (cf. Fig. 2b). This stands in contrast to the original evaluation [8], where queries with a selectivity of 1 sampled from the data collection could not be attacked successfully, again underlining the importance of evaluating against real queries.

In Fig. 3, we allowed for queries to be fully sampled (Fig. 3a) as opposed to the previous partial sampling–refer to the *query sampling* parameter detailed above–or when there is some query repetition involved (Fig. 3b) to simulate a more realistic setting. We noticed: (1) that there is more variability involved in the full setting, and (2) that the recovery rate of the SUBGRAPH attacks is reduced by about 40% with query repetitions. In particular, both attacks have lower minimum rates around 40% for 10% partial knowledge, whereas their median rate drops to this level for all knowledge rates if queries repeat. We still

[5] Given that email data often contains public information such as updates and spam, we consider 5% knowledge realistic.

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

Instance	GKKNO	AVALUE	ARR	ARR-OR	APA-OR$^{\text{BT}}$	APA-OR$^{\text{ABT}}$
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

consider this significant information. Accuracy is thus affected by full sampling or repeated queries but remains significant in all cases.

Total volume attacks. Compared to SUBGRAPH, the VOLAN and the SELVOLAN attacks achieve significantly lower recovery rates against private data, with only a recovery greater than 20% for GMail when the adversary knows at least 75% of the data, see Fig. 1 for more data points.

In the public setting of Fig. 2, we similarly noticed that the attacks did poorly. For high selectivities, they require almost perfect adversarial knowledge for an adequate recovery. Concretely, the adversary needs to know strictly more than 70% of the data to be able to recover more than 7% of the queries in S-H. For very low selectivities, none of the attacks worked except for 100% knowledge. Thus, we only consider them to pose risk for a high partial knowledge.

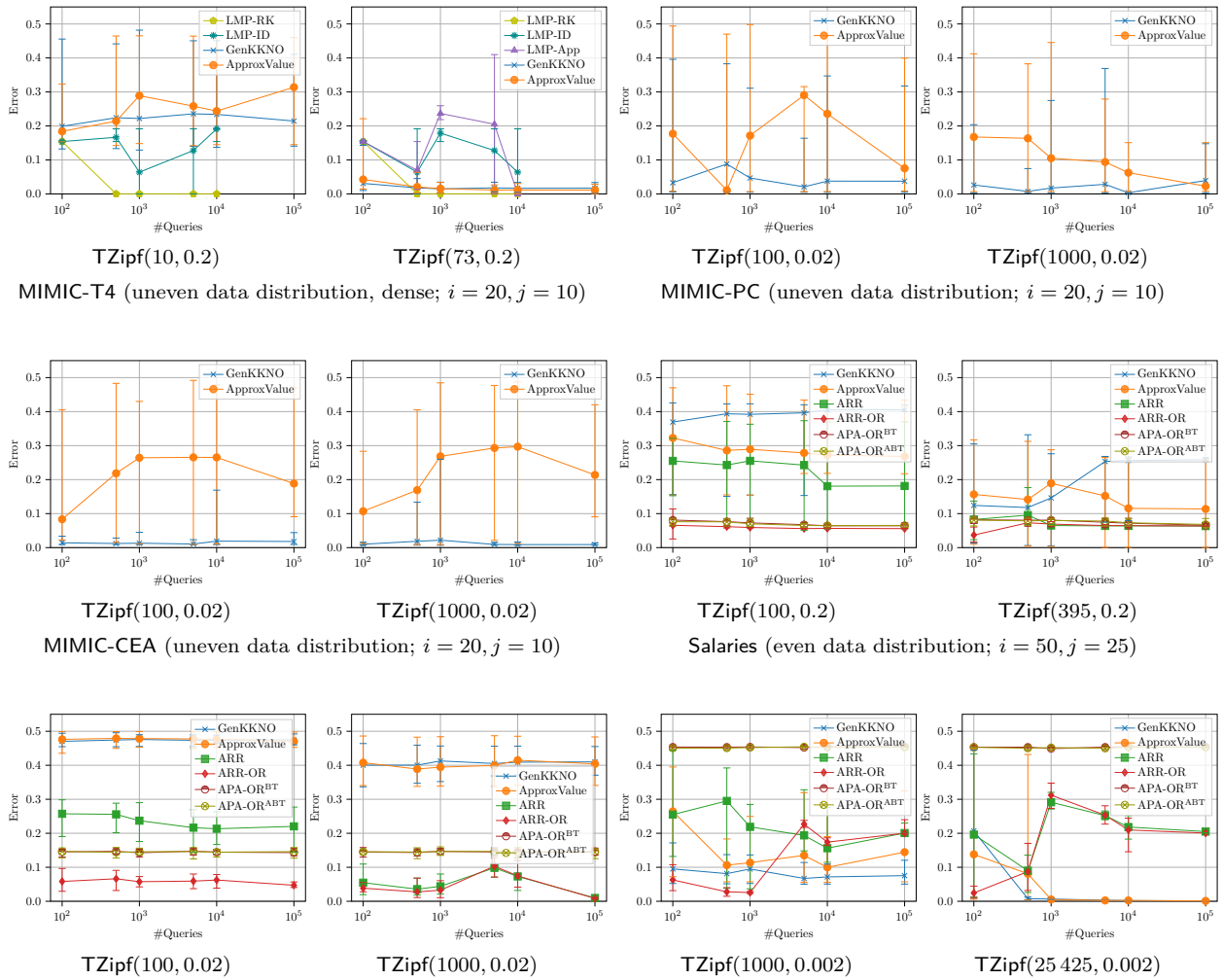
7.2 Range Attacks

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 parameter 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. That is, there are no required additional assumptions when it comes to the user's query distribution. For datasets with no available query log, we had to consider synthetic query distributions. Prior query distributions include the *uniform* distribution which picks ranges uniformly at random and variants of the *beta* distribution [47, 48]. Given that none of these distributions are supported by real-world query logs, we introduce a more realistic distribution called *truncated Zipf* that retains 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) where we fix a to be equal to 5. There are two additional parameters: the *maximum width* B, and the *fraction* f. The former removes any range that has a width larger than B, whereas the latter ensures that only a fraction f of possible ranges occur. This new variant is dubbed TZipf(B, f). We varied both B and f and found that B can have a significant impact on reconstruction accuracy. Thus, in our evaluation we present results for a relatively small and large B, and set small, realistic fractions of f that still give us a meaningful query space.

Amount of queries. For the SDSS dataset, we used the entire query logs and therefore do not control the amount of queries in this setting, see Tab. 3. However, for the other numerical datasets, we sample with replacement an amount of queries from 10^2 to 10^5.

Data distribution. Given the different datasets introduced in §6, the corresponding data distributions have different shapes and properties, which allows us to assess the attacks in different scenarios. Though we cannot provide all exact distributions due to access restrictions, we recall some basic properties of §6.2 (details in App. C.2.1): the medical



TZipf$(10, 0.2)$ TZipf$(73, 0.2)$ TZipf$(100, 0.02)$ TZipf$(1000, 0.02)$

MIMIC-T4 (uneven data distribution, dense; $i = 20, j = 10$) MIMIC-PC (uneven data distribution; $i = 20, j = 10$)

TZipf$(100, 0.02)$ TZipf$(1000, 0.02)$ TZipf$(100, 0.2)$ TZipf$(395, 0.2)$

MIMIC-CEA (uneven data distribution; $i = 20, j = 10$) Salaries (even data distribution; $i = 50, j = 25$)

TZipf$(100, 0.02)$ TZipf$(1000, 0.02)$ TZipf$(1000, 0.002)$ TZipf$(25\,425, 0.002)$

Sales (even data distribution; $i = 50, j = 25$) Insurance (uneven data distribution; $i = 10, j = 5$)

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.

dataset, MIMIC, and the insurance dataset, Insurance, are skewed towards low values with just a few high-value outliers (which we call an *uneven* distribution as a risk factor, cf. §4), whereas entries in the human resources dataset, Salaries, and the sales dataset, Sales, are spread more evenly across the domain range (which we call an *even* distribution).

Data density and size. Another basic property is data density and data size n, where the former captures the percentage of unique values in the interval $[N]$ with domain size N; and the latter is simply the number of entries in the collection. Note that n can be larger than N (cf. Tab. 3 and Tab. 4) and recall that the medical dataset MIMIC-T4 is much more dense than the other datasets.

Experimental setting. We use the same machine as in the keyword case. Every range attack is evaluated against all datasets and all query distributions. Contrary to the keyword

attacks, where the goal of the attack is to recover the queries, the goal for range attacks is to recover the data itself.[6] 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 [49, 30]

All LMP [49] attacks failed completely in all instances, even under the unrealistic uniform query distribution. The only exception is MIMIC-T4, the only dataset with density. In particular, the attack was able to recover all contiguous values of the dense subset of the dataset even for the truncated Zipf distribution. The error can be very small (0.0003) because the non-dense outliers represent a tiny fraction of the data (around 0.37% of all values). If the attacks output the reflection, the error is significantly larger, increasing variability. We consider LMP to only be successful if the dataset is very dense.

The count reconstruction attack GLMP [30] similarly failed in all instances. The only exception is MIMIC-T4, where we observed perfect reconstructions *if we disregard density*[7]. However, this was only achieved with 10^5 queries with a truncated Zipf distribution with $\mathtt{B} = 73$ and $\mathtt{f} = 0.2$. As such, we believe that GLMP can only have some recovery if and only if the collection is completely dense and the query distribution is uniform which is very unlikely.

7.2.2 The GJW Attacks [32]

With the exception of the Salaries dataset, both GJW-Basic and GJW-Missing aborted due to an infeasible search space. More precisely, for Salaries, the recovered counts were always completely incorrect when queries are sampled from truncated Zipf, 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. Similarly to GLMP, we do not consider them to work under real-world conditions.

7.2.3 Approximate Reconstruction Attacks [31]

GenKKNO and ApproxVal of [31] failed to recover any meaningful data in the SDSS case which, as detailed in §6.2, depicts the most realistic setting in our experiments. As an instance, both attacks have an error equal to at least 0.41 across all instances of SDSS (cf. Tab. 5). However, both attacks can uncover significant information for the truncated Zipf distribution (cf. Fig. 4), despite them assuming the uniform query distribution.

We identified two major parameter regimes where GenKKNO succeeds: (1) If B is relatively large, it generally works well with the only exception being Sales, or (2), if the data distribution is skewed towards lower values such as the MIMIC and Insurance datasets, where it even succeeds for a small B. However, it does not work for the relatively uniformly distributed collection Sales. We believe (1) holds because queries with large width are more

[6] This is an over-simplification where all range attacks assume that there is one encrypted range structure for every attribute in the dataset (where the dataset should be viewed here as a relational table with a single attribute). Note that attacking a structure used to encrypt more than just one attribute was only considered very recently [22].

[7] Comparing the counts on a modified collection that solely contains contiguous values, i.e., we here ignore domain values that do not occur in the original collection.

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 $\mathtt{B} = 100$.

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 [47, 48]

Contrary to the previous attacks, these attacks obtained non-trivial errors on the real-world SDSS queries but are computationally demanding. As they have to solve non-convex or nonlinear optimization problems with solution size $n+1$, evaluating them on our MIMIC instances was infeasible. We also had to rely on SciPy [71] instead of the original MATLAB optimization to meet our open-source goals of LEAKER (cf. §5). In the following, we give more details on attack performance.

ARR. Recall again that ARR-OR requires the order as an input which is not the case for ARR. ARR-OR achieves significantly low errors less than 0.15 in all evaluated settings. This includes SDSS, where ARR-OR is the only attack to succeed in attacking a real-world query log. While the error is still high (0.249) for SDSS-S containing only 215 unique queries, the error drops to 0.128 for SDSS-M containing 5.6k unique queries (cf. Tabs. 3 and 5). This stands in contrast to ARR, where no significant SDSS information is uncovered. For the other datasets, ARR-OR achieves low errors even for just 100 queries and low maximum query widths B, see Fig. 4. ARR, on the other hand, only comes close to ARR-OR performance for the evaluations allowing for large B. Thus, as order is usually not leaked, the ARR attack is sensitive to the order reconstruction of ApproxOrder [31], which diminishes performance for low B.

We also note a peculiarity against Insurance, where the error for more than 10^3 queries is significantly larger than for as little as 100 queries. It stems from slightly overestimating distances between entries with the same, low value making up almost half of the collection (cf. Fig. 5 in App. C.2.1), therefore cumulatively increasing the error. We believe this to occur for more queries as they yield more possible result sets outside the set of this specific, low value, thereby decreasing the weight for its result set, which consequently enables a low yet cumulatively significant error.

APA. Since APA [48] is parameterized by the underlying range scheme, we show results for the state-of-the-art schemes BT [21] and ABT [19]. Also, APA recovers ordered values, whereas no attack uncovering the order based on this leakage profile is known.

In general, APA is the only range attack not utilizing rid that can uncover significant information outside the case of dense data and a uniform query distribution. This was not the case for GLMP and GJW that also do not rely on rid (but do not need qeq). Although APA only achieves an error of about 0.24 on the real-world SDSS queries, it performs significantly well on other data sources. We observe that it is very robust towards differences in query amount and width but is sensitive to the data distribution. In particular, it achieves low errors of about 0.06 for Salaries and about 0.15 for Sales, whereas it does not perform well on Insurance with errors of about 0.45. For the latter case, the

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

found solutions greatly underestimate large value distances while they are more accurate in the former case on an even data distribution containing smaller distances, which we thus see as a risk factor for APA. The impact of using different range schemes seems to be insignificant.

8 Conclusions

Due to closed-source implementations and restricted evaluations without real-world query data, the implications of leakage attacks against encrypted search have long remained questionable. With our open-source LEAKER framework, we enable the community to easily implement, evaluate, independently verify, and compare current and future attacks. By using it with novel real-world queries and datasets, we determined settings where leakage attacks pose credible risks (cf. Tab. 1 on Page 5). The implications of our findings for ESA security are as follows.

Keyword search. In this setting, both the IKK [34] and Count [11] attacks do not work well on our real-world queries and data whereas the Subgraph attacks of [8] consistently perform well even for a low partial knowledge and low selectivity queries on small private databases. This contradicts prior assumptions that in such instances, users' queries would be so specific that the attacks fail to work [8]. We therefore recommend hiding the identity and volume patterns in such settings, e.g., by using schemes from [43, 42, 61, 8, 25, 5]. Our results also show that leaking the total volume and response length patterns only poses risk in high partial knowledge settings, since VolAn and SelVolAn [8] consistently fail for known-data rates $< 75\%$.

Range search. In contrast to keyword search, our evaluations uncover many subtleties in the setting of range search. The only successful attack on our real-world query logs is ARR [47] if the order is leaked. Therefore, designing more accurate *order reconstruction attacks* remains important.

We found that leaking the response identity in range search is risky if query widths are large (ARR [47]) or if the values are skewed towards endpoints (GenKKNO [31]). Moreover, leaking the response length and the query equality on evenly distributed data is risky in light of the APA [48] attack.

For this, the recent emergence of techniques for *leakage suppression* [43, 42, 61, 25, 5], which eliminate various leakage patterns through either black-box compilation or data structure transformations, could potentially be extended and applied to the range setting—at some computational and/or storage cost. In fact, suppression techniques already exist to address the response length [42, 61, 5] and the query equality [43, 25] so it remains open to find ways to integrate them into existing encrypted range schemes.

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] Mohamed Ahmed Abdelraheem, Tobias Andersson, Christian Gehrmann, and Cornelius Glackin. Practical attacks on relational databases protected via searchable encryption. In *International Conference on Information Security (ISC)*, 2018.

[2] 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.

[3] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *International Conference on Management of Data (SIGMOD)*, 2004.

[4] Ghous Amjad, Seny Kamara, and Tarik Moataz. Breach-resistant structured encryption. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2019.

[5] Ghous Amjad, Sarvar Patel, Giuseppe Persiano, Kevin Yeo, and Moti Yung. Dynamic volume-hiding encrypted multi-maps with applications to searchable encryption. *IACR ePrint*, 765, 2021.

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

[7] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In *Annual International Cryptology Conference (CRYPTO)*, 2007.

[8] Laura Blackstone, Seny Kamara, and Tarik Moataz. Revisiting leakage abuse attacks. In *Network and Distributed System Security Symposium (NDSS)*, 2020.

[9] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2004.

[10] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference (TCC)*, 2011.

[11] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2015.

[12] Yan-Cheng Chang and Michael Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *International Conference on Applied Cryptography and Network security (ACNS)*, 2005.

[13] Matt Chaput. Whoosh. https://whoosh.readthedocs.io/en/latest/, 2012. Accessed 2020-10-16.

[14] Melissa Chase and Seny Kamara. Structured encryption and controlled disclosure. In *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, 2010.

[15] 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.

[16] William W Cohen. Enron dataset. https://www.cs.cmu.edu/~./enron/, 2015. Accessed 2021-04-15.

[17] Gordon V Cormack and Thomas R Lynam. TREC public spam corpus. https://plg.uwaterloo.ca/~gvcormac/treccorpus07/, 2007. Accessed 2021-03-24.

[18] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2006.

[19] Ioannis Demertzis, Stavros Papadopoulos, Odysseas Papapetrou, Antonios Deligiannakis, and Minos Garofalakis. Practical private range search revisited. In *International Conference on Management of Data (SIGMOD)*, 2016.

[20] Maria Esch, Jinbo Chen, Stephan Weise, Keywan Hassani-Pak, Uwe Scholz, and Matthias Lange. A query suggestion workflow for life science IR-systems. *Journal of Integrative Bioinformatics*, 11(2), 2014.

[21] Sky Faber, Stanislaw Jarecki, Hugo Krawczyk, Quan Nguyen, Marcel Rosu, and Michael Steiner. Rich queries on encrypted data: Beyond exact matches. In *European Symposium on Research in Computer Security (ESORICS)*, 2015.

[22] Francesca Falzon, Evangelia Anna Markatou, Akshima, David Cash, Adam Rivkin, Jesse Stern, and Roberto Tamassia. Full database reconstruction in two dimensions. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2020.

[23] Benjamin Fuller, Mayank Varia, Arkady Yerukhimovich, Emily Shen, Ariel Hamlin, Vijay Gadepally, Richard Shay, John Darby Mitchell, and Robert K Cunningham. SoK: Cryptographically protected database search. In *IEEE Symposium on Security and Privacy (S&P)*, 2017.

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

[25] Marilyn George, Seny Kamara, and Tarik Moataz. Structured encryption and dynamic leakage suppression. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2021.

[26] Eu-Jin Goh. Secure indexes. *IACR ePrint*, 216, 2003.

[27] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. *Journal of the ACM (JACM)*, 43(3), 1996.

[28] 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.

[29] Greg Grothaus. General implementation of the PQ-tree algorithm. https://github.com/Gregable/pq-trees, 2008. Accessed 2020-10-16.

[30] Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth 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.

[31] Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G Paterson. Learning to reconstruct: Statistical learning theory and encrypted database attacks. In *IEEE Symposium on Security and Privacy (S&P)*, 2019.

[32] Zichen Gui, Oliver Johnson, and Bogdan Warinschi. Encrypted databases: New volume attacks against range queries. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019.

[33] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E Oliphant. Array programming with NumPy. *Nature*, 585(7825), 2020.

[34] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *Network and Distributed System Security Symposium (NDSS)*, 2012.

[35] Shrainik Jain, Dominik Moritz, Daniel Halperin, Bill Howe, and Ed Lazowska. SQLShare: Results from a multi-year SQL-as-a-service experiment. In *International Conference on Management of Data (SIGMOD)*, 2016.

[36] Bernard J Jansen. Search log analysis: What it is, what's been done, how to do it. *Library & Information Science Research*, 28(3), 2006.

[37] Bernard J Jansen and Amanda Spink. How are we searching the World Wide Web? A comparison of nine search engine transaction logs. *Information Processing & Management (IP&M)*, 42(1), 2006.

[38] Daxin Jiang, Jian Pei, and Hang Li. Mining search and browse logs for web search: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(4), 2013.

[39] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3(1), 2016.

[40] Steve Jones, Sally Jo Cunningham, Rodger McNab, and Stefan Boddie. A transaction log analysis of a digital library. *International Journal on Digital Libraries*, 3(2), 2000.

[41] Emilia Kacprzak, Laura M Koesten, Luis-Daniel Ibáñez, Elena Simperl, and Jeni Tennison. A query log analysis of dataset search. In *International Conference on Web Engineering (ICWE)*, 2017.

[42] Seny Kamara and Tarik Moataz. Computationally volume-hiding structured encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2019.

[43] Seny Kamara, Tarik Moataz, and Olya Ohrimenko. Structured encryption and leakage suppression. In *Annual International Cryptology Conference (CRYPTO)*, 2018.

[44] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. Generic attacks on secure outsourced databases. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

[45] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *Science*, 220(4598), 1983.

[46] Evgenios M Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. Data recovery on encrypted databases with k-nearest neighbor query leakage. In *IEEE Symposium on Security and Privacy (S&P)*, 2019.

[47] Evgenios M Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. The state of the uniform: Attacks on encrypted databases beyond the uniform query distribution. In *IEEE Symposium on Security and Privacy (S&P)*, 2020.

[48] Evgenios M Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. Response-hiding encrypted ranges: Revisiting security via parametrized leakage-abuse attacks. In *IEEE Symposium on Security and Privacy (S&P)*, 2021.

[49] Marie-Sarah Lacharité, Brice Minaud, and Kenneth G Paterson. Improved reconstruction attacks on encrypted data using range query leakage. In *IEEE Symposium on Security and Privacy (S&P)*, 2018.

[50] Philippe Lamesch, Kate Dreher, David Swarbreck, Rajkumar Sasidharan, Leonore Reiser, and Eva Huala. Using the Arabidopsis information resource (TAIR) to find information about Arabidopsis genes. *Current Protocols in Bioinformatics*, 30(1), 2010.

[51] Chang Liu, Liehuang Zhu, Mingzhong Wang, and Yu-An Tan. Search pattern leakage in searchable encryption: Attacks and new construction. *Information Sciences*, 265, 2014.

[52] Yiqun Liu, Junwei Miao, Min Zhang, Shaoping Ma, and Liyun Ru. How do users describe their information need: Query recommendation based on snippet click model. *Expert Systems with Applications*, 38(11), 2011.

[53] Evangelia Anna Markatou and Roberto Tamassia. Full database reconstruction with access and search pattern leakage. In *International Conference on Information Security (ISC)*, 2019.

[54] Gilad Mishne and Maarten De Rijke. A study of blog search. In *European Conference on Information Retrieval (ECIR)*, 2006.

[55] Muhammad Naveed, Seny Kamara, and Charles V Wright. Inference attacks on property-preserving encrypted databases. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2015.

[56] Hoang Vu Nguyen, Klemens Böhm, Florian Becker, Bertrand Goldman, Georg Hinkel, and Emmanuel Müller. Identifying user interests within the data space-A case study with SkyServer. In *International Conference on Extending Database Technology (EDBT)*, 2015.

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

[58] NIST. Toward a PEC use-case suite (preliminary draft). NIST White Paper (Draft), 2021.

[59] Simon Oya and Florian Kerschbaum. Hiding the access pattern is not enough: Exploiting search pattern leakage in searchable encryption. In *USENIX Security Symposium (USENIX Security)*, 2021.

[60] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. A picture of search. In *International Conference on Scalable Information Systems (INFOSCALE)*, 2006.

[61] Sarvar Patel, Giuseppe Persiano, Kevin Yeo, and Moti 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.

[62] 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.

[63] Phoenix Bioinformatics. 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.

[64] Rishabh Poddar, Stephanie Wang, Jianan Lu, and Raluca Ada Popa. Practical volume-based attacks on encrypted databases. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2020.

[65] David Pouliot and Charles V Wright. The shadow nemesis: Inference attacks on efficiently deployable, efficiently searchable encryption. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

[66] Ruben Groot Roessink, Andreas Peter, and Florian 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.

[67] Vik Singh, Jim Gray, Ani Thakar, Alexander S Szalay, Jordan Raddick, Bill Boroski, Svetlana Lebedeva, and Brian Yanny. SkyServer traffic report-The first five years. *arXiv preprint cs/0701173*, 2007.

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

[69] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy (S&P)*, 2000.

[70] Cédric Van Rompay, Refik Molva, and Melek Önen. A leakage-abuse attack against multi-user searchable encryption. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2017(3), 2017.

[71] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J van der Walt, Matthew Brett, Joshua Wilson, K Jarrod Millman, Nikolay Mayorov, Andrew R J Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E A Quintero, Charles R Harris, Anne M Archibald, Antônio H Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17, 2020.

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

[73] Wouter Weerkamp, Richard Berendsen, Bogomil Kovachev, Edgar Meij, Krisztian Balog, and Maarten 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.

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

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

[76] 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.

[77] Jing Yao, Yifeng Zheng, Yu Guo, and Cong 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.

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

[79] Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. All your queries are belong to us: The power of file-injection attacks on searchable encryption. In *USENIX Security Symposium (USENIX Security)*, 2016.

A ARR with Repeating Values

Agnostic Reconstruction Range (ARR) [47] can be slightly modified in order to also cover the case of repeating values, i.e., a non-injective mapping from records to values [47]. This can be achieved by allowing the distance between values to be 0 and requires a deviation from the original pseudocode of [47] 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 [47], 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).

B LEAKER Code

We provide example LEAKER implementations in Listings 1 and 2.

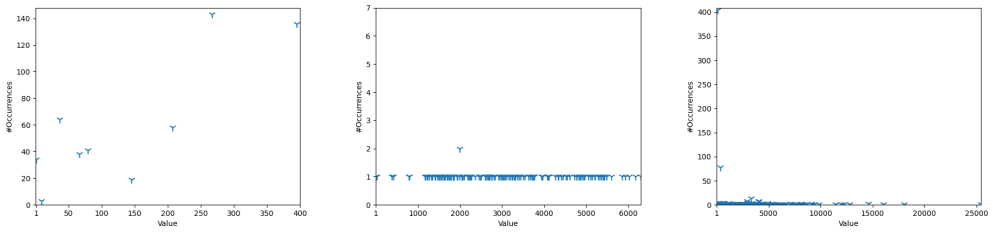


Figure 5: Frequencies of numerical dataset values.

C Further Data Information

C.1 Alternate Sources and Pre-Processing

Search engines. There are some other query logs we could have chosen from, including from the Excite [36], Yandex [76], and Sogou [68, 52] search engines or from Yahoo! Answers [75] or the TREC session track [57]. 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 because from the data, 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 [62] together with the TAIR 2013 update data [63]. 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 [35] 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 have been added.

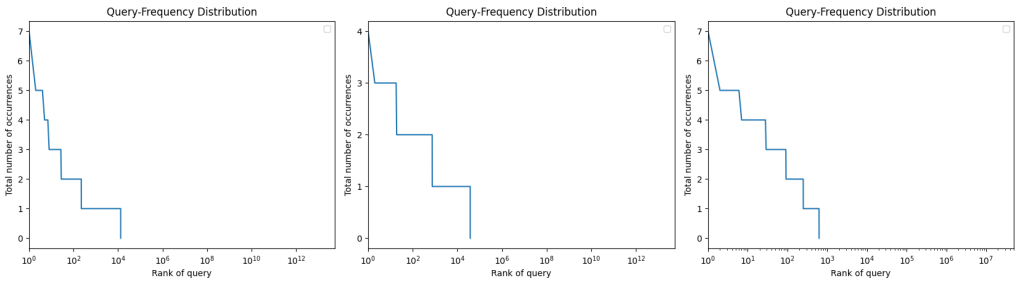
C.2 Statistical Analysis

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

C.2.1 Data Distributions

Fig. 5 shows the data distribution (frequencies of values) of various datasets that we can display because they do not have access restrictions. Notice that Insurance has a significant skew towards low values, with values greater than 10 000 of a maximum $N = 25\,425$ being very rare outliers (8 out of 886). We note that his is also similarly the case in all MIMIC data, in that most entries have a low value (for MIMIC-T4, only 301 out of 8 058 entries have a value greater than 20 of a maximum $N = 73$; for MIMIC-PC it is 9 out of 7 709

[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.



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 $N = 10^4$.

with a value greater than 500 of a maximum $N = 2\,684$) and for MIMIC-CEA it is 25 out of 2 844 with a value greater than 2 500 of a maximum $N = 9\,978$.

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), 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 – Selectivity distribution

The selectivity of the queries has been identified as the main attack performance metric [8, 66] and, therefore, we analyzed the selectivity distribution of queries issued in real-world systems. In particular, [8] 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 [8] to argue that, because most keywords appear in the tail with a low occurrence, most queries might have a very low selectivity as well. A similar argument for low-selectivity keywords in real-world queries can be found in [66]. 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 selectivity (a mean of 1 804 for AOL, 2 023 for TAIR and 326 for GMail). Only Drive has a mean query selectivity of 11.2, which was considered as *pseudo-low* by [8]. We conclude that, in our evaluations, *users are not interested in querying keywords of a low selectivity* and conjecture that they 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' selectivities, but found no correlation between number of queries and mean selectivity (Pearson correlation of about 0.1 for TAIR and 0.014 for AOL).

C.2.3 Range data – Query distribution

The central part 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 [47, 48], 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 this, we considered a more realistic query distribution in our experiments for data without query logs, where queries are distributed according to Zipf, but, in contrast to [47, 48], 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 large such upper bound (close to N) has been identified as a risk factor (cf. §7).

D Further 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. 2a in §7.1. This confirms our statistical analysis that activity does not influence selectivity 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 selectivity.

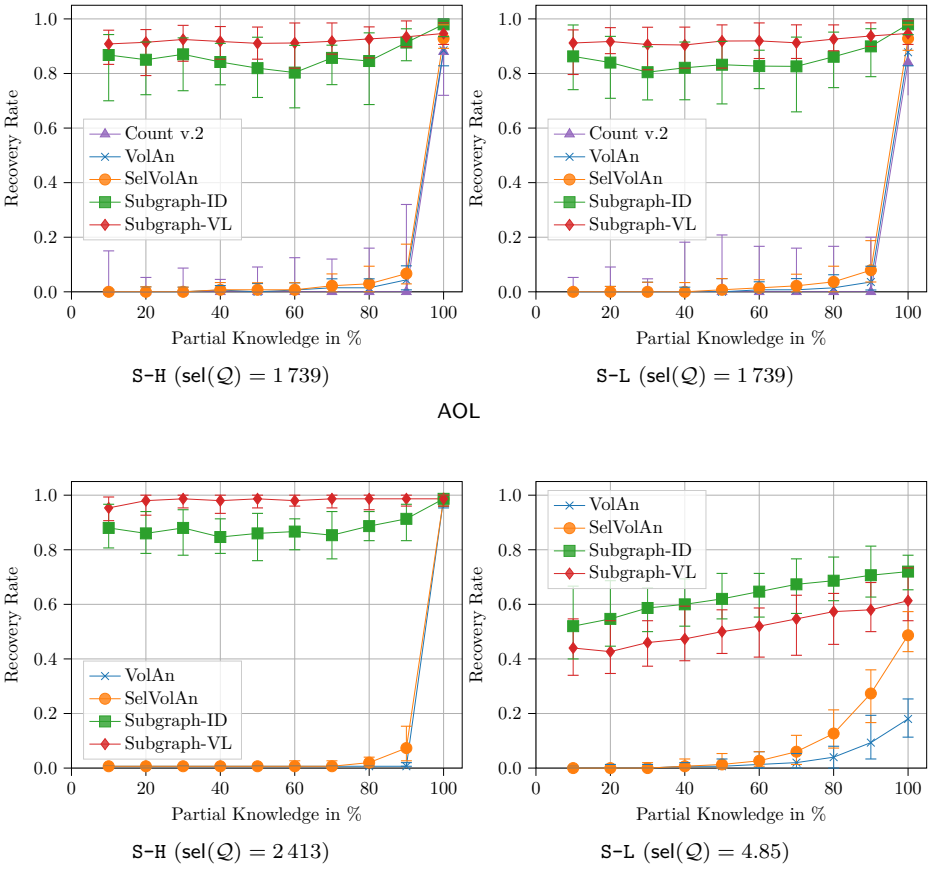


Figure 7: `X-Y` attack evaluations against AOL and TAIR. All evaluations are 5×5, except for 3×3 evaluations done for Count v.2. 150 queries are drawn for each of the most active users (single user setting; `S`) without replacement according to their query frequency from the 500 most (`H`) or least (`L`)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

Listing 1: Slightly simplified example of LEAKER code for implementing the basic count attack (Algorithm 1 of [11]) 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
```

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]
```