Too Close for Comfort? Measuring Success of Sampled-Data Leakage Attacks Against Encrypted Search

Dominique Dittert
dominique.dittert@stud.tu-darmstadt.de
Technical University of Darmstadt
Germany

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

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

ABSTRACT

The well-defined information leakage of Encrypted Search Algorithms (ESAs) is predominantly analyzed by crafting so-called leakage attacks. These attacks utilize adversarially known auxiliary data and the observed leakage to attack an ESA instance built on a user's data. *Known-data* attacks require the auxiliary data to be a subset of the user's data. In contrast, *sampled-data* attacks merely rely on auxiliary data that is, in some sense, statistically close to the user's data and hence reflect a much more realistic attack scenario where the auxiliary data stems from a publicly available data source instead of the private user's data.

Unfortunately, it is unclear what "statistically close" means in the context of sampled-data attacks. This leaves open how to measure whether data is close enough for attacks to become a considerable threat. Furthermore, sampled-data attacks have so far not been evaluated in the more realistic attack scenario where the auxiliary data stems from a source different to the one emulating the user's data. Instead, auxiliary and user data have been emulated with data from one source being split into distinct training and testing sets. This leaves open whether and how well attacks work in the mentioned attack scenario with data from different sources.

In this work, we address these open questions by providing a measurable metric for statistical closeness in encrypted keyword search. Using real-world data, we show a clear exponential relation between our metric and attack performance. We uncover new data that are intuitively similar yet stem from different sources. We discover that said data are not "close enough" for sampled-data attacks to perform well. Furthermore, we provide a re-evaluation of sampled-data keyword attacks with varying evaluation parameters and uncover that some evaluation choices can significantly affect evaluation results.

CCS CONCEPTS

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

CCSW '23, November 26, 2023, Copenhagen, Denmark

KEYWORDS

Encrypted Search; Leakage Attacks; Privacy Metric

ACM Reference Format:
Dominique Dittert, Thomas Schneider, and Amos Treiber. 2023. Too Close for Comfort? Measuring Success of Sampled-Data Leakage Attacks Against Encrypted Search. In *Proceedings of the 2023 Cloud Computing Security Workshop (CCSW '23), November 26, 2023, Copenhagen, Denmark*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3605763.3625243

1 INTRODUCTION

Encrypted Search Algorithms (ESAs) allow to privately search encrypted data without requiring to decrypt them. This reduces the severity of data breaches, since not even a server storing the data can access plaintext information. As a result, ESAs are highly relevant for cloud services, e.g., to outsource private email data or genetic databases onto cloud storage, and have become real-world products. For instance, MongoDB's *Queryable Encryption* capability [32] allows to run expressive queries on encrypted data. ESAs are based on various cryptographic notions such as Searchable Symmetric Encryption (SSE) [7, 10, 16, 38] / Structured Encryption (STE) [8], Fully Homomorphic Encryption (FHE) [15], and Oblivious RAM (ORAM) [18]. We refer to the survey in [14] for an overview. In this work, we are concerned with encrypted keyword search, i.e., ESAs that allow to privately search a collection of encrypted documents with a keyword query.

Leakage. All efficient ESA constructions, i.e., constructions with sub-linear search time, disclose some information upon setup and execution. For example, the server learns at least a bound on the size of the encrypted data structure when the client uploads it. Furthermore, upon execution of queries in the case of keyword search, the server may learn the size of the corresponding responses and/or additional details like the number of returned documents or their identifiers. This revealed information is referred to as *leakage*. ESAs represent a trade-off between leakage and efficiency.

Leakage Attacks. To establish the security of an ESA, several steps are necessary. First, the leakage profile, that is comprised of all types of information exposed by the scheme, needs to be specified. In a second step, it has to be proven that the scheme does not leak any further information [10]. Finally, cryptanalysis is used to examine how this leakage could be exploited by a potential adversary (we refer to the survey in [26] for a state-of-the-art overview of leakage cryptanalysis). For this purpose, researchers design so-called *leakage attacks* on ESAs that exploit specific leakage patterns together

with some adversarially known *auxiliary information*. Attack performance on encrypted keyword search is usually measured as the percentage of plaintext queries that could be correctly restored. The information of attack performance can then be used to strike a sensible balance between leakage and efficiency in potential ESA deployments. However, several aspects of the way attacks are (and, due to scarcity of relevant data, have to be) evaluated make it hard to properly determine and quantify the practical threat resulting from leakage [4, 26]. One central aspect is what adversarial information can be seen as realistic and whether attacks still work on such information.

Adversarial Information. When a subset of the user's data collection is known, the attacks are called *known-data* attacks [4, 6]. These have recently been re-evaluated by Kamara et al. [26] using various data and evaluation scenarios. If the auxiliary information comprises reference data that is, in a way, close to the user's data but distinct from it, the attacks are called *sampled-data* attacks [11, 22, 33, 35]. Sampled-data attacks may be regarded as more realistic since the attacker may more easily acquire a publicly available, yet related, auxiliary dataset for mounting its attack. For instance, publicly available email lists may be used to attack private ESA instances on email clients that include semantically similar correspondence.

Attack Evaluations. So far, sampled-data attacks have not been re-evaluated on new data and across scenarios. This actually includes a complete lack of evaluating the underlying motivation of sampled-data attacks by using two different but statistically close datasets. Instead, the state-of-the-art evaluation of these attacks [11, 33, 35] relies on *splitting one evaluated dataset into distinct training and testing sets*[1] which, as a result, are deemed statistically close. The training set comprises the auxiliary data and the testing set is used for evaluations. While this technically applies to sampled-data attacks, the underlying attacker scenarios differ: Since the training and testing data are *from the same source/dataset*, the conclusions one can empirically draw from attack performance most closely apply to the scenario where a part of the dataset was breached and then deleted. Thus, it is hard to draw conclusions for the more realistic scenario where the training data stems from a *source other than the source of the testing data*.

Measuring Susceptibility. Moreover, there is neither a consistent definition of statistical closeness nor a standardized metric to measure it. Therefore, it is not possible to see *how* close data needs to be for instances to be susceptible to attacks.

Recent works on the theoretical quantification of privacy in ESAs [27, 28] obtain bounds on how much an attacker can learn in specific instances. This *quantifies how private* an ESA instance is irrespective of currently available attacks. While these theoretical metrics are very valuable, we believe an important complimentary approach to be *measuring how susceptible* an instance is based on the current best empirical knowledge on attack performance.

Ideally, a measurable *metric* defining statistical closeness of data collections in relation to ESA cryptanalysis would be able to indicate whether collections are close enough such that attacks are likely to be successful. Using this metric with a re-evaluation of sampled-data attacks on a pair of datasets *from different sources* that is intuitively similar can then establish a more practical picture of the relation of dataset closeness and attack success.

[1]For instance, data of two different years or distinct subsets of the dataset may make up separate training/testing sets.

Table 1: Values of our statistical closeness metric necessary for correctly recovering 10 %, 15 %, and 20 % of queries for different sampled-data attacks. The values are calculated using the exponential fit functions that describe the results of our evaluations. Note that our intuitively similar data pair from different sources yields a closeness value of 0.65 whereas higher values are only reached with data sampled from the same source dataset.

Recovery Rate	IKK [22]	SAP [33]	IHOP [35]
10 %	0.90	0.91	0.82
15 %	0.92	0.96	0.85
20 %	0.93	0.99	0.87

1.1 Contributions

In this paper, we design a metric for statistical closeness of keyword data collections and re-evalute sampled-data attacks on encrypted keyword search. Throughout our measurements of our metric and attack performance we also utilize new data that is intuitively similar but from distinct sources. These contributions are crucial to the following research question we pose (given the potentially very large attack surface of sampled-data attacks):

Can real-world ESAs be successfully attacked with auxiliary data from a dataset source different to the one of the target data?

We provide a differentiated answer: There are successful instances of these attacks, but, *according to our metric, auxiliary and target data need to be very close for attacks to be able to uncover more than* 10% *of the plaintext queries*. In fact, despite being semantically very similar, our new data from distinct sources does not reach this level and the corresponding attack instances do not see success beyond 10% recovery. Moreover, we show that further evaluation aspects, such as using a real-world log of queries issued on the data collection by actual users of the instance (called a "query log"), can heavily influence results and conclusions.

Detailed Contributions. More concretely, our contributions are as follows:

(1) We **extend the open-source leakage attack framework LEAKER** [26], which allows to easily re-evaluate known-data attacks on arbitrary data, to support sampled-data keyword attacks. We will open a pull request of our changes upon acceptance of our paper.
(2) We **uncover additional data** for the empirical foundation of leakage cryptanalysis. Using data from mailing lists of the Ubuntu and Debian distributions, we create data collections from *separate* sources that are semantically similar and can be used as a more realistic model of auxiliary information than using one dataset/source.

(3) We **design a metric** for statistical closeness of data collections. Using established and our new data, we **empirically demonstrate an exponential relation** between closeness and success of the attacks of [22, 33, 35]. The metric can hence be used to indicate attack success on ESA instances with a candidate auxiliary data collection, e.g., to determine if countermeasures are necessary because a publicly available data collection is too close to the private potential deployment. We present a summary in Tab. 1: In order to correctly uncover even 10% of the user's queries correctly, according to our experiments the attacker's auxiliary information and the target data need a statistical closeness of at least 0.82. Our Ubuntu-Debian combination has a measured metric of 0.65 and our experiments validate that it is not close enough to reach the cut-off value of 10% attack success. We observe such high values of closeness only in cases where both data stem from the same source. Furthermore, a comparison with the *co-occurrence similarity* [11, 37] shows that our metric is superior.

(4) We **re-evaluate the sampled-data leakage attacks** of [33, 35] on established and our new data and under various conditions to identify which evaluation aspects influence results. Most significantly, we for the first time use a query log of real user queries for evaluating sampled-data attacks, which so far has only been done for known-data attacks [26]. This is also the most influential aspect of evaluations: We find that recovery rates of the attacks on a query log are vastly higher compared to the original works. However, in these cases the auxiliary information does stem from the same source. Due to the scarce availability of relevant data, it remains open how successful sampled-data attacks are under real-world query logs when the adversary does not have access to data from the same source.

2 RELATED WORK

We review related work in the general area of encrypted search, its cryptanalysis via leakage attacks, and the quantification of leakage.

2.1 Encrypted Search

The area of encrypted search began with the work of Song, Wagner and Perrig [38]. Definitions for ESAs are found in Goh [16] and Chang and Mitzenmacher [7]. The standard security definition for Searchable Symmetric Encryption (SSE) was given by Curtmola et al. [10]. This was then generalized as Structured Encryption (STE) by Chase and Kamara [8]. Other building blocks for ESAs are Property-Preserving Encryption (PPE) [1, 3], Fully-Homomorphic Encryption (FHE) [15], Functional Encryption (FE) [5], Oblivious RAM (ORAM) [18], and Secure Multi-Party Computation (MPC) [17, 41]. We refer to [14] for a survey on ESAs.

2.2 Leakage Attacks

The first leakage attack was described by Islam et al. [22] and became known as the IKK attack. It was described as a sampled-data attack, though subsequent works [6, 26, 37] also used it as a known-data version. Known-data attacks were given by [4, 6] and sampled-data attacks by [11, 20, 21, 33, 35]. All these works (and ours) consider the user's queries to be drawn independently of each other. Recently, the case of dependend queries for query equality leakage has been considered by [25, 35]. We refer to Kamara et al. [26] for more information and a survey on leakage attacks. In Sect. 4, we present more details on the attacks subject to our work. Furthermore, [26] provides the LEAKER framework for evaluating (known-data) leakage attacks and a re-evaluation of known-data attacks. Our work is concerned with sampled-data attacks and measuring their success.

2.3 Leakage Quantification

Wright and Pouliot [40] quantify the success of leakage attacks against deterministic and order-revealing encryption, which are considered Property-Preserving Encryption (PPE). Frameworks to analyze deterministic and order-revealing encryption are provided by Jurado and Smith [24] and Jurado et al. [23], respectively. Furthermore, Grubbs et al. [19] provide a framework to analyze leakage w.r.t. data reconstruction attacks, applying to range search.

Leakage Inversion. Kornaropoulos et al. [28] propose a framework called *leakage inversion* that can provide a quantification of privacy in ESAs based on theoretical considerations of the size of the reconstruction space (to invert an observed leakage profile). Regarding the keyword ESAs we are concerned with, the metric can quantify the privacy loss of certain leakages and adversarial information—more precisely, knowledge of all possible keywords contained in the data collection and knowledge of the response length of all possible keywords contained in the data collection. The metric is not directly applicable to sampled-data attacks.

Coherence. Recently, Kamara and Moataz [27] porposed a theoretical framework based on Bayesian leakage analysis leading to a quantifiable notion called *coherence*. The authors show how modeling a leakage profile as a Bayesian network allows to bound the coherence given various distributions on data, queries, and adversarial information. Such bounds are, among others, provided for query recovery attacks using query equality and/or response length information.

We only cover sampled-data query reconstruction attacks making use of the query equality, response length, and/or co-occurrence patterns, i.e., applying to ESAs built on SSE/STE or ORAM techniques. More importantly, our approach is entirely empirical and based on the performance of existing sampled-data attacks. Instead of bounds, we provide concrete measurements that give empirical indications of attack success. It is our belief that both empirically-ground approaches and theoretically-grounded approaches are necessary for understanding privacy in ESAs.

3 PRELIMINARIES

We first provide some basic notations and fundamentals of ESAs. Then, we overview different leakage patterns and profiles, followed by the most common adversary models and leakage attack types.

3.1 Notation

We denote the set of integers $\{1, \ldots, n\}$ as $[n]$. The corresponding power set is represented by $2^{[n]}$. Let a be a sequence of n elements, then a_i refers to its ith element. For a set S, $|S|$ is its cardinality.

Let a data collection D be a collection of n documents d_i, where $i \in [1, \ldots, n]$ over a keyword space $\mathbb{W} = \{w_1, w_2, \ldots, w_m\}$ of size m.

Each document d_i has an unique identifier i and can be modeled as a set of keywords attached to it: $d_i \subseteq \mathbb{W}$. With $\mathbb{D}$ we describe the space of all document collections over a keyword space $\mathbb{W}$. The query space $\mathbb{Q} \subseteq \mathbb{W}$ is the space of all keywords that are queried. The result of a keyword query $w \in \mathbb{Q}$ are all documents that contain w, which we write as $D(w) = \{d \in D : w \in d\}$. Each keyword query w has a frequency that refers to the number of documents in D that contain w: $|D(w)|$. The identifier function $ids : \mathbb{W} \to 2^{[n]}$, $ids(w) \mapsto \{i \in [n] : d_i \in D(w)\}$ maps a keyword w to the identifiers of the documents that contain w.

3.2 Encrypted Search Algorithms (ESAs)

An ESA enables a client to search on encrypted data stored on a server without needing to decrypt it. In this work, we only consider ESAs supporting keyword queries on document collections. We can model a system consisting of two parties:

A client owning a data collection they wish to store and query remotely on a server.

A server offering the storage services, e.g. a cloud storage provider.

Generally speaking, an ESA is composed of the following operations:

Setup: In the setup protocol, the client encrypts the data collection D and sends the encrypted data collection $\overline{D}$ to the server.

Query: The query protocol runs between the client and the server. The client generates a query for a specific keyword, which they send in encrypted form (often called a *token*) to the server. The server evaluates it on the encrypted data collection to retrieve the requested documents and returns them to the client.

Update: With a dynamic (non-static) ESA, the client can modify the data stored on the server after the setup step by removing documents from the data collection or adding new documents.

3.3 Leakage

Each operation of an ESA can reveal some information. In the initialization, for instance, the server learns at least a bound on the size of the data collection. During query operations, the server may learn, for example, the number of returned documents. This disclosed information is called leakage. The respective *leakage profile* of the encryption scheme is composed of different *leakage patterns*, depending on which information is leaked. As is common in the literature of sampled-data attacks [22, 33, 35], we focus on leakage patterns in static ESAs associated with the query operation.

Leakage Patterns. Often times, two types of leakage stemming from the realm of SSE are prominently discussed, namely search pattern leakage and access pattern leakage. The search pattern describes whether two queries are identical, meaning they request the same keyword. The access pattern corresponds to which documents match a particular query. In the following, we describe a more detailed classification of ESA leakage patterns based on [4] which the attacks subject to our work can exploit.

The Response Identitiy (RID) pattern represents the full access pattern. It reveals the identifiers of the queried documents and can be described by a function $RID : \mathbb{D} \times \mathbb{W}^t \to [2^{[n]}]^t$, $RID(D, w_1, \ldots, w_t) = (ids(w_1), \ldots, ids(w_t))$.

The Co-Occurrence (CO) pattern indicates how often two keywords jointly appear in the same document. It can be described by a function $CO : \mathbb{D} \times \mathbb{W}^t \to [n]^{t \times t}$, which is defined as $CO(D, w_1, \ldots, w_t) = M^{co}$ that returns a $t \times t$ matrix M^{co} with entries $M^{co}_{i,j} = |ids(w_i) \cap ids(w_j)|$. The CO pattern can be derived from the RID pattern.

The Response Length (RLEN) pattern discloses the number of documents that match a given query, i.e., the length of the response. It can be described by the function $RLEN : \mathbb{D} \times \mathbb{W}^t \to \mathbb{N}$, $RLEN(D, w_1, \ldots, w_t) = (|D(w_1)|, \ldots, |D(w_t)|)$. The RLEN pattern can be derived from the CO pattern.

The Query Equality (QEQ) pattern represents the search pattern. It exposes whether two queries are equal and can be expressed by a function $QEQ : \mathbb{D} \times \mathbb{W}^t \to \{0,1\}^{t \times t}$, $QEQ(D, w_1, \ldots, w_t) = M^{qeq}$ that returns a binary $t \times t$ matrix M^{qeq} with entries $M^{qeq}_{i,j} = \begin{cases} 1, & \text{if } w_i = w_j \\ 0, & \text{otherwise} \end{cases}$.

The Query Frequency (QFREQ) pattern indicates how often a certain query occurs in relation to the total number of queries. It is represented by the function $QFREQ : \mathbb{D} \times \mathbb{W}^t \to \mathbb{R}^t$, $QFREQ(D, w_1, \ldots, w_t) = f$ that returns a vector f of length t with elements $f_i = \sum_j M^{qeq}_{i,j}/t$. The QFREQ pattern can be derived from the QEQ pattern.

3.4 Adversary Models

There are two main types of adversaries when it comes to attacks on ESA [2]:

Persistent adversaries have access to the encrypted data collection and associated ciphertexts and see all query and update tokens as well as the answers returned by the server. To illustrate, this model represents a scenario in which the client does not trust an honest-but-curious server. Another option might be a corrupted server, where the attacker can read its memory and observe all interactions.

Snapshot adversaries, on the other hand, only see the encrypted data collection but do not have access to the query or update tokens. In this case we can differentiate between the server and the attacker. The scenario could be described by an attacker hacking the server at one point in time and capturing a snapshot of its memory, hence gaining access to the encrypted data collection, but not observing the interactions between server and client.

Furthermore, we can distinguish between active and passive adversaries:

Passive adversaries can observe the interaction of client and server. They exploit this observed leakage to attack the ESA instance.

Active adversaries, on the other hand, are also able to interact with the system. They can tamper with the data collection by injecting their own files. This allows them to observe the queries and exploit leakage corresponding to files they already know. Attacks assuming an active adversary like this are also called file-injection attacks [4, 36, 42].

We are concerned with *persistent, passive* adversaries.

3.5 Leakage Attacks

A leakage attack utilizes some auxiliary knowledge and the observed leakage to uncover private information. Commonly, attacks use the leakage of a sequence of queries to uncover the queries. When evaluating an attack, its performance is measured by the *recovery rate*, referring to the fraction of correctly uncovered queries. Depending on the kind of auxiliary knowledge required by the adversary, we can classify attacks on ESA as the following type:

Known-Data attacks, which are also known as ground-truth attacks, require the strong assumption that the adversary has access to a subset of the indexed documents on the server. This knowledge might stem from a previous data breach and can be utilized by an attacker to target the remainder of the data collection.

Sampled-Data attacks are based on the weaker assumption that the adversary has access to a statistically close, yet not identical dataset. These attacks are also known as inference attacks or statistical attacks. To follow up on the data breach example from the known-data case, we could think of an attack after a data breach, where the breached documents were removed from the data collection. In recent literature [33, 35], the sampled-data scenario is often simulated by sampling two disjunct subsets from one dataset, where one subset is subsequently given to the attacker as auxiliary information and the other one is used as attack target. However, this does not significantly weaken the assumption on the auxiliary knowledge, since the adversary still needs access to data of the original dataset (like in said data breach scenario). A more interesting scenario would be an attack on a private, encrypted data collection from one dataset, where the adversary uses data from another dataset as auxiliary information. This motivates the design of the attacks but until now was not used in the evaluations.

In our work, we are concerned with *sampled-data* attacks and provide evaluations with data from separate dataset sources.

4 CONSIDERED SAMPLED-DATA ATTACKS

We detail the sampled-data attacks subject to our work here.

4.1 IKK

The so-called IKK attack [22] utilizes the co-occurrence of keywords (CO; cf. Sect. 3.3) in the data collection as well as a fraction of known queries. The authors prove that the optimization problem of minimizing the distance between the observed and real co-occurrence matrices is NP-complete. Hence, they propose using a heuristic approximation technique known as simulated annealing to solve it. Simulated annealing iteratively tries to find a global optimum for a target function from an arbitrary initial state, where in each iteration the current state may change to a neighboring state, ultimately converging towards a satisfactory state or aborting until a specified amount of iterations is reached.

4.2 SAP

Oya and Kerschbaum [33] propose SAP, a Search and Access Pattern-based attack that relies on a Maximum Likelihood Estimation (MLE) approach. The authors consider ESAs that leak both the search and the access pattern (cf. Sect. 3.3). From the former, they can infer the query equality (QEQ), allowing them to calculate the query frequency (QFREQ), which refers to how often the client performs a specific query. From the latter, the response length (RLEN) is computed, i.e., the number of documents returned in response to a query. Using their auxiliary information, the attacker can compute the same metrics for the keywords. The proposed attack solves a Linear Assignment Problem (LAP) to arrive at a mapping that maximizes the probability of observing the described leakage given the auxiliary information.

4.3 IHOP

While Linear Assignment Problems (LAPs; as utilized in SAP) have efficient solvers to find an optimal solution, they cannot use quadratic terms that contain the co-occurence (CO) pattern information (cf. Sect. 3.3). These require solving a Quadratic Assignment Problem (QAP). Therefore, Oya and Kerschbaum [35] present IHOP, a statistical query recovery attack that follows an Iterative Heuristic algorithm to solve a quadratic Optimization Problem. The attack can combine different information and may use the response length (RLEN), the co-occurence (CO), and/or the frequency computed from the query equality (QEQ/QFREQ). However, QAPs are difficult to solve, so IHOP iteratively solves multiple LAPs as described in the following. First, it generates an initial mapping of tokens to keywords. Then, it performs a certain number of the following iterations: The existing mappings are randomly divided into two sets, namely the set of fixed tokens and the set of free tokens. It now computes a new mapping for the free tokens, with which the initial mapping is updated. Choosing the number of iterations is a trade-off between performance and accuracy.

5 USED DATA SOURCES

We describe the datasets we use for our evaluation of our metric (cf. Sect. 7) and for re-evaluating sampled-data attacks (cf. Sect. 8). All data has been pre-processed using LEAKER [26].

5.1 Email Data

This type of data has long been a main motivation for the study of ESAs and is prominent in the leakage attack literature.

The Enron Email Corpus [9] is the predominant evaluation target in ESA cryptanalysis [4, 6, 11, 20, 22, 33, 35, 37]. As in previous works, we use all emails sent by all employees as data collection.

The Debian security-announce Mailing List [12] is a public newsletter about security advisories. We chose it since many Linux distributions are built on the same architecture, so we expect corresponding mailing lists to be semantically similar. Hence, we can use these lists *from distinct sources* to model settings where the attacker uses one dataset obtained from a source to attack another ESA instance from another source. We used a simple script to retrieve the contents of the emails from the Debian website and filtered out hashes, fingerprints, and commands. We do not distinguish between different users and use 18 years of data, from October 2004 to September 2022, to form a data collection. This data we uncovered has not been used in the context of ESAs before.

The Ubuntu security-announce Mailing List [39] is an equivalent to the Debian mailing list. We consider data of the same 18-year period as for the Debian list as data collection and filter similarly. This data has not been used in the context of ESAs before.

5.2 Genetic Data

Due to high sensitivity, genetic data is an important ESA use case.

The Arabidopsis Information Resource (TAIR) Data Collection [29] contains genetic data of the model higher plant Arabidopsis Thaliana. The data collection is obtained and indexed exactly as in [26], where this data was first discovered for the evaluation of known-data leakage attacks.

The TAIR Query Log [13] includes all queries per user on the TAIR database from January 1, 2012 through April 30, 2013. We use this query log as query data on the TAIR data collection.

5.3 Search Engine Data

As ESAs are used for private search, search engine data is frequently used to motivate ESAs and to evaluate leakage attacks.

Google Trends [31] contains query data on the Google search engine, which was used in previous evaluations on attacks that also exploit the QEQ pattern [30, 33, 35]. We use the pre-processed trends data [33, 34] for 3 000 keywords extracted for the 260 weeks from May 2015 to May 2020. Contrary to a query log that displays a list of keywords issued by a user, this data provides the number of occurrences of a query in a certain timeframe (weeks). However, both types of data can be used to sample queries for evaluations and as auxiliary information for attacks exploiting QEQ[2] and QFREQ. Hence, we refer to both of them as *query data*.

6 OUR STATISTICAL CLOSENESS METRIC

In the following, we design a metric for measuring the statistical closeness of two data collections, which we will evaluate with collections introduced in Sect. 5.1 w.r.t. attack performance in Sect. 7.

6.1 General Design

We aim that our metric design reflects the intuitive closeness of datasets, e.g., with our datasets of Sect. 5, samples of Enron should be very close, the Ubuntu and Debian collections should be close, and Ubuntu and Enron should be noticeable less close than the other combinations. Then, we will be able to evaluate the correlation between attack success and data closeness quantified by our metric.

To find such a metric for any two data collections D^I and D^{II}, we investigate several statistics which we call *criteria*. We will then combine these criteria into our final metric.

Additional Notation. Before defining our criteria, we need to define some more notation. $f^I(w) = \frac{|D^I(w)|}{n^I}$ defines the normalized frequency of keyword w in D^I, where n^I is the amount of documents in D^I. $\overline{w}_i^I$ denotes the i-th keyword of D^I sorted according to decreasing frequency in D^I, i.e., $\forall i < j : f^I(\overline{w}_i^I) \geq f^I(\overline{w}_j^I)$. Consequently, we define the ordered set of the i-th most frequent keywords in D^I as $KW_i^I = \{\overline{w}_j^I : j \in [i]\}$. f^{II}, $\overline{w}_i^{II}$, and KW_i^{II} are defined analogously for D^{II}.

Data Collection Criteria. In the following, we present the criteria we consider for our metric and their formal definitions, which we denote by $CR^{(1)}, CR^{(2)}$, and $CR^{(3)}$ and depend on a variable k:

[CR 1] The mean of the normalized sizes of the intersections of the most frequent keywords in D^I and D^{II}:

$$CR^{(1)}(k) = \frac{1}{k}\sum_{i=1}^{k}\frac{|KW_i^I \cap KW_i^{II}|}{i}$$

[CR 2] The mean of the normalized amounts of occurrences of the most frequent keywords of D^I in D^{II}:

$$CR^{(2)}(k) = \frac{1}{k}\sum_{i=1}^{k}\frac{|KW_i^I \cap \mathbb{W}^{II}|}{i}$$

[CR 3] The mean of the relative differences of frequencies of the most frequent keywords of D^I within D^I and their frequencies within D^{II}:

$$CR^{(3)}(k) = \frac{1}{k}\sum_{w \in KW_k^I}\left(1 - \frac{|f^I(w) - f^{II}(w)|}{\max(f^I(w), f^{II}(w))}\right)$$

Our presented criteria comprise the means over different statistics of the most frequent keywords, calculated up to the k most frequent keywords. Since we noticed high variability for low values of k, a sufficiently large value of k needs to be set to obtain consistent results (we set $k = 10^4$ in our experiments). This additionally allows to determine the statistical error of each criterion by calculating the standard deviation of the sets given as input to the mean computation.

It should also be noted that criteria [CR 2] and [CR 3] are not symmetrical, but differ in which data collection serves as the training data and thus for generating the set of most frequent keywords.

Final Metric. We combine the three criteria into our final statistical closeness metric, which we denote by SC:

$$\mathrm{SC}(k) = \frac{1}{3}\left(CR^{(1)}(k) + CR^{(2)}(k) + CR^{(3)}(k)\right)$$

For the determination of the statistical uncertainty, the use of error propagation is necessary. The standard deviation between the three criteria can be considered as systematic uncertainty. We observed noticeably large uncertainties with [CR 3], which is why we additionally evaluated a version of SC that only takes the mean of $CR^{(1)}$ and $CR^{(2)}$. However, our detailed evaluation conducted in the following Sect. 6.2 indicates that the combination of all three criteria provides the best separation between our various datasets.

Runtime. Note that $KW^{I/II}$ can be pre-computed. Thus, $CR^{(1)}$ and $CR^{(2)}$ both require $O(k^2)$ operations due to k set intersection operations of minimum size i each and $CR^{(3)}$ requires $O(k)$ operations as the outputs of the functions f^I and f^{II} can already be pre-computed. Overall, computation of our metric SC hence requires $O(k^2)$ and is independent of the size of the data collection.

[2]While information on frequency such as Google Trends does not provide a direct way to evaluate QEQ attacks as with a query log, the widespread assumption that queries are drawn independently enables such evaluations, ultimately relying on QFREQ. Dependent-query QEQ attacks are given by [25, 35] and do not apply here.

6.2 Detailed Evaluation of Our Metric

We evaluate the criteria proposed in Sect. 6.1 on the data presented in Sect. 5.1. We first analyze the Enron mailset in the sampled data setting by splitting it in half into two disjoint subsets uniformly at randomly, as is often done in the literature. Since the sampling is random, we perform it ten times and then determine the mean for each criterion. Additionally, we study the Debian and Ubuntu mailing lists as an example of data that comes from different sources, yet are intuitively similar. Finally, one of the security mailing lists and the Enron corpus serve as examples of unrelated datasets.

We present the evaluation of our overall metric in Fig. 1. Overall, our metric reflects the intuition behind the data set combinations: Knowing the other half of Enron yields the most close dataset according to the metric (scores around 0.85; see blue points in Fig. 1). The semantically very similar Ubuntu and Debian datasets still surpass a score of 0.6 while the intuitively dissimilar score between Ubuntu and Enron is low (around 0.35).

Criteria Combinations. In Fig. 1, we look at both, all three criteria combined, as well as only criteria [CR 1] and [CR 2] combined. As mentioned, the statistical uncertainties are generally smaller for the second model that does not use [CR 3]. However, there is no clear trend for the systematic uncertainties when using only [CR 1] and [CR 2] compared to all three criteria. Furthermore, within errors statistical or systematic, the two variants coincide. Therefore, we decided to include [CR 3] in the final computation of our metric, as it is the only criterion that takes the distribution of keywords over the documents in the dataset into account. This kind of information is used by many attacks, making it a valuable distinguisher for predicting attack success with our metric.

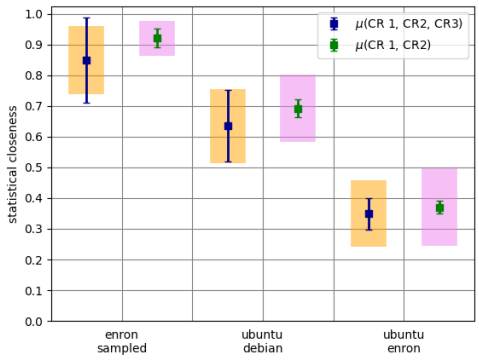

Figure 1: Our statistical closeness metric for the Enron mailset in the sampled data setting, the Ubuntu and Debian mailing lists, as well as the Ubuntu and Enron data. The blue data points show our metric computed from all three criteria, while the green data points take only criteria [CR 1] and [CR 2] into account. The errorbars denote the respective statistical uncertainty stemming from the statistical uncertainties of each individual criterion. The systematic uncertainties are represented by the colored boxes and stem from the deviation between the used criteria.

7 EVALUATION OF OUR STATISTICAL CLOSENESS METRIC

In the following sections we will compute the statistical closeness of different datasets and evaluate it against the recovery rate of various sampled-data attacks. Subsequently, we will compare our metric to the co-occurrence metric suggested by [11, 37].

Implementation in LEAKER [26]. We rely on the LEAKER framework [26] written in Python and extend it as follows. Regarding LEAKER's pre-processing capabilities, we added classes for the Ubuntu, Debian, and Google Trends data. We incorporated re-implementations of the SAP [33] and IHOP [35] attacks, which we verified to be correct by reproducing the original results. Our extensions include modifications to the internal evaluator class of LEAKER and its data collection representation to allow for the evaluation of sampled-data attacks, which was not possible before. Finally, we extended the module for statistical analysis with our metric computation and the visualizer module to be able to plot the metric results. All of this resulted in 1 765 added or altered lines of code. We have opened a pull request in the main LEAKER repository[3] to open-source our code and facilitate inclusion of our extensions into the main framework.

7.1 Evaluating Statistical Closeness Against Recovery Rate

We evaluate the sampled-data attacks of [22, 33, 35]. Our goal is not only to demonstrate that our metric is a very useful indicator of attack success but also to determine the necessary quality of adversarial knowledge. Thus, we use both the existing approach of sampling testing and training sets from one dataset (Enron) and our new data from different datasets (Ubuntu and Debian; cf. Sect. 5).

7.1.1 Experiment Setup. All of our evaluations are performed on a `Debian GNU/Linux 11` machine with 24 *GB* of RAM, 12 CPU cores, and 64 *GB* of hard disk storage.

Sampled Training and Test Sets. For an accurate analysis, we sample training and test subsets from Enron of various sizes. E.g., a 10 % sampling here means that both training and test data are disjunct 10% of the data collection sampled uniformly at random. The intuition is that the smaller the sampled subsets are, the smaller their similarity, i.e., their statistical closeness. The number of sample points is denoted by s in the following, e.g., sampling 10 %, 20 %, 30 %, 40 %, and 50 % yields $s = 5$. After sampling the datastets, we restrict all of them to the same number of keywords n_{kw} for the evaluation. This number of keywords is also used as the parameter k for computing the statistical closeness (cf. Sect. 6.1). Since the sampling is random, we repeat it t times. Each time, the training set is given as attacker knowledge and the testing set is used for an evaluation (sample the attacked queries and compute their leakage).

Distinct Datasets. In addition to sampled training and training sets, we use pairs of datasets from *distinct sources*. We use Ubuntu and Debian as a pair that intuitively provides closely-related adversarial information and Ubuntu and Enron that intuitively provides unrelated adversarial information (besides being emails of the English language). Our aim is to include corresponding data points as

[3] https://encrypto.de/code/LEAKER

Table 2: Summary of the attack parameters for our evaluations. s is the number of sampled datasets, n_{kw} is the number of keywords we restrict each dataset to, and n_q is the number of queries sampled from the query space of size Q. We repeat the sampling process t times and evaluate r runs of each attack on all data collections.

	s	n_{kw}	n_q	$\|Q\|$	t	r
IKK [22]	7	500	15	50	2	3
SAP [33]	9	1 000	100	150	10	20
IHOP [35]	9	1 000	100	150	5	5

a way of seeing both that the intuition is reflected by the statistical closeness and how it relates to attack success.

Evaluation Runs. For each attack, we evaluate r runs. For each run, a new set of n_q keywords is sampled from the query space Q of the test set. If not otherwise specified, the query space consists of the 150 most frequent keywords[4] and we sample $n_q = 100$ keywords at random from it. Hence, in total, for each of the s sampled datasets, we have $t \times r$ recovery rates, for which we compute the mean recovery rate and display it together with the minimum and maximum recovery rates as uncertainty range. For datasets from different sources, we get r recovery rates to average. We choose the values of evaluation parameters depending on the runtime of the corresponding attack for feasibility. A summary of the parameters is given in Tab. 2.

7.1.2 Evaluating Statistical Closeness With IKK [22]. We present the results in Fig. 2. Note that the experimental parameters in this instance consist of a comparatively low number of queries, keywords, and evaluations. This stems from the computational complexity of the attack; we had to restrict the data collections in order to obtain a sensible trade-off between experiment runtime and ideal parameters.

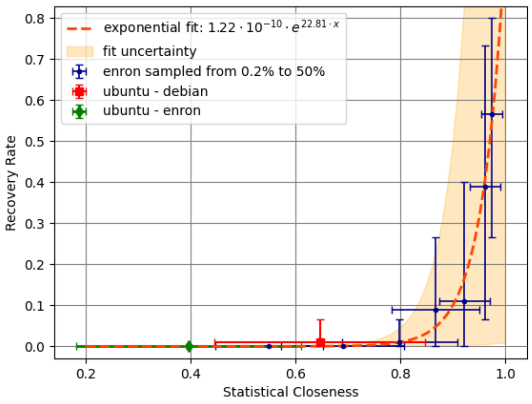

Figure 2: Evaluation of the recovery rate of the IKK attack [22] against our statistical closeness metric.

[4]This is a common evaluation in ESA cryptanalysis, experimentally verified by [26] in the sense that real-world query logs contain high frequency keywords.

The relation between the recovery rate and the statistical closeness can be described by an exponential fit (cf. fit curve of Fig. 2). At a statistical closeness of around 0.9, the recovery rate of the IKK attack goes down to around 10 %, although the recovery rates are much higher for closer data collections, with a best case of 80 % and an average of 57 % for the largest sampled data collections. For the Ubuntu and Debian mailing lists we only see an average recovery rate of only 2 % and a best case of 7 % (cf. Fig. 2). The IKK attack was not able to recover even a single query for the Ubuntu and Enron data in any run.

7.1.3 Evaluating Statistical Closeness With SAP [33]. The results of this evaluation are shown in Fig. 3. We evaluate the attacks only on response length information, as we cannot consider query equality information for our Ubuntu/Debian data.

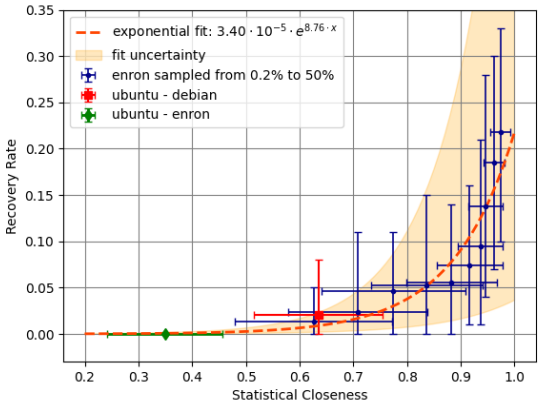

Figure 3: Evaluation of the recovery rate of the SAP [33] attack against our statistical closeness metric using only response length information.

As for IKK, we observe a strong exponential relation between the statistical closeness of the training and testing data and the achieved recovery rate of the SAP attack. In the best case with a statistical closeness of 0.97, SAP recovers up to one third of the queried keywords. However, the recovery rate drops fast for data collections less close to each other. At a statistical closeness of around 0.9, it is already below 10 % recovery on average. The Ubuntu and Debian datasets have a statistical closeness of almost 0.65, which allows the attack to only recover around 2 – 3 % of the queried keywords on average. Attacking Enron with Ubuntu training data results in less than 1 % recovery.

7.1.4 Evaluating Statistical Closeness With IHOP [35]. The results of this evaluation are shown in Fig. 4. For reasons similar to the case of SAP, we evaluate the attacks only on co-occurrence information.

We again observe a strong exponential relation. IHOP achieves higher recovery rates than IKK and SAP, e.g., over 35 % for a statistical closeness of 0.93. However, the overall relation between recovery and closeness is similar to IKK and SAP, since these higher rates are only achieved once a closeness of around 0.82 is reached. A lower closeness results in less than 10% recovery. Performance of

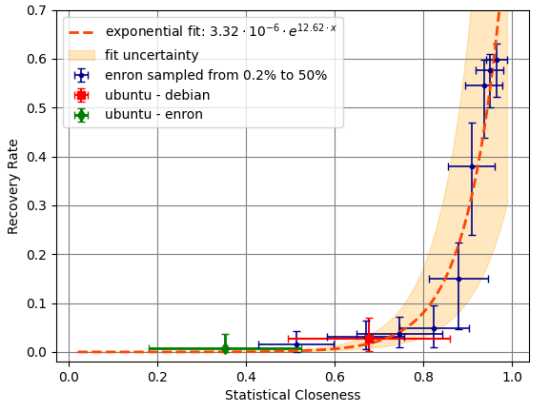

Figure 4: Evaluation of the recovery rate of the IHOP [35] attack against our statistical closeness metric using only co-occurrence information.

the attack in the Ubuntu-Debian setting is very similar to that of SAP. For Ubuntu-Enron, it is slightly more successful than SAP but the mean recovery stays close to 0.

7.1.5 Summary. In short, the experimental relation between our statistical closeness metric and attack performance is exponential. We refer to Tab. 1 in Sect. 1.1 as a summary of our experiments in the form of statistical closeness values necessary to correctly uncover more than specific fractions of queries. If this fraction of queries is determined to be a level of significance, our evaluations demonstrate that instances with a corresponding statistical closeness are vulnerable to attacks. E.g., the evaluated attacks require a statistical closeness of more than 0.8 to recover about 10 % of the queries. Crucially, we only encountered such statistically close data when sampling from the same data source (the Enron dataset). The intuitively similar and statistically close datasets Ubuntu and Debian (statistical closeness of around 0.65), however, are not close enough to be subject to significant recovery rates. *We thus demonstrate an empirical exponential relationship between attack success and statistical closeness and show that related data from separate sources does not reach significant closeness.*

Outlook. Of course, future attacks may change this empirical basis, but our metric can serve as an indication of attack performance of the current state-of-the-art in sampled-data attacks. Subsequent attack improvements (e.g., IHOP [35] improving upon IKK [22]) raise the overall recovery rates in our experiments. However, the general exponential relationship between recovery rate and statistical closeness holds for all evaluated attacks. Assuming this relationship remains in place for future attack improvements, these may likely only lower the cut-off values at which points the recovery reaches significant levels. This is already visible in Tab. 1, where IHOP lowers the cut-off values by up to 8% compared to IKK. Hence, it remains open to develop attacks that can lower the cut-off values to the levels of data collections from differently sourced datasets (like Ubuntu and Debian), especially for use cases where one of these sources is public and the other one is private and target of attacks.

7.2 Comparison to Co-Occurrence Similarity

In this section, we compare our newly developed metric with the co-occurrence simiarity measurement used by [11, 37]. The co-occurrence similarity computes the absolute distance between the co-occurrence matrices of two datasets.

Experiment Setup. We use the formulas given by [37] for the same datasets of our metric evaluation in Sect. 7.1 w.r.t. the ability to predict the recovery rate of the IKK attack [22] for samples of different sizes from the Enron mailset. We focus on the IKK attack since it was the subject of the co-occurrence similarity design and evaluation of [37]. The parameters for the attack are chosen exactly like in Sect. 7.1.

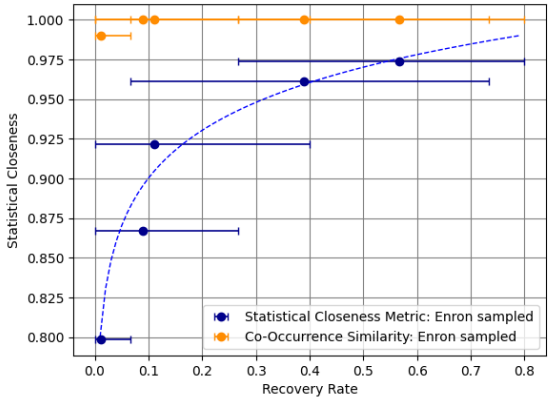

Figure 5: Comparison of co-occurrence similarity [37] and our statistical closeness metric. The plot shows the values of both similarity measures for data collections sampled from the Enron mailset at sizes between 0.8 % and 50 % of the original dataset on the y-axis. The x-axis shows the corresponding recovery rate of the IKK attack [22].

Results. We present the results of our comparison in Fig. 5. While the x-axis (attack performance) values remain the same, the different metrics attain vastly different values (y-axis). Our first observation is that the co-occurrence similarity stays constant for sample sizes from below 2 % up to 50 % of the original dataset size, although the recovery rate for those samples drastically changes. While we can recover almost 60 % of the queries on average when splitting the Enron dataset in half, the recovery rate drops below 10 % for the smaller sample sizes. This change in recovery rate cannot be predicted by the co-occurrence similarity which shows an agreement of almost 100 % for all of those data points. Only for sample sizes smaller than 1 % of the original dataset size the co-occurrence similarity decreases slightly. However, it is still extremely high with a value of almost 99 %, despite the fact that on average only 1 % of the queries can be reconstructed for this data.

Comparison to Our Metric. In contrast, our metric provides a much more fine-grained measure of the statistical closeness of different datasets. The values of our metric are clearly distinguishable for different sample sizes, which agrees well with the differences in

recovery rate. As the relation between the recovery rate and the statistical closeness of the datasets follows an exponential function, we can fit a logarithmic curve to our data points. The fit nicely describes all the evaluated data points and allows to make predictions on the necessary statistical closeness of two datasets in order to achieve a certain recovery rate. Therefore, for measuring sampled-data attack success on various ESA attack instances, our statistical closeness metric can give consistent estimates. Furthermore, it is computationally much less intense (independent of collection size vs. quadratic costs of co-occurrence similarity; cf. Sect. 6.1).

Although [37] also discovered an exponential correlation between the recovery rate of IKK and the co-occurrence similarity, we were not able to find a suitable fit function.

8 RE-EVALUATION OF THE ATTACKS

The evaluations conducted in Sect. 7 show a clear relation between our statistical closeness metric and attack efficacy. However, further evaluation aspects may influence the way one can interpret ESA attacks, some of which may be specific to certain attacks. Therefore, in this section, we re-evaluate the prominent sampled-data attacks of [33, 35][5] with varying parameters and data to uncover further relations between the way attacks are evaluated and their efficacy. With our Debian and Ubuntu datasets, we are for the first time in the position to investigate sampled-data attack parameters on testing and training data from *distinct* yet semantically similar datasets. With the TAIR data, we for the first time can evaluate sampled-data attacks with a real-world query log.[6] In the following we use this data to obtain evaluation results regarding parameters that significantly affect attack performance compared to the original manners of evaluating them.

8.1 Experiment Setup

We use the same machine as in Sect. 7.1.1. We fix parameters to obtain feasible runtimes while varying those parameters that influence attack performance. While some parameters could also be chosen differently, we had to make a trade-off between breadth and (runtime) feasiblity of our evaluations to obtain useful results.

Training and Testing Sets. If we use a data collection from a single dataset source (Enron or TAIR), we sample and split it (50%) into testing and training data collections as in Sect. 7.1.1 and denote it as "sampled". For the combination of the Ubuntu and Debian datasets, we use Ubuntu as training set and Debian as testing set. We always use partial query sampling, i.e., the queries have to occur in the adversary's knowledge. We use query data from Google Trends and the TAIR query log to form testing and training query sets. We use Google Trends on all data collections. Additionally, we use the TAIR query log on the TAIR data collection. When using Google Trends data, we split the testing and training data to comprise n_w weeks of query data for varying values of n_w. There is a 5 week offset between testing and training data, i.e., the last n_w weeks of data of Google Trends are used as testing set, and the last $2n_w + 5$ to $n_w + 5$ weeks of the data are used as training set. For each week in the testing set, we sample n_q/n_w queries as the user's queries, resulting in a total number of n_q queries. As the queries are sampled weekly, the queries can repeat. When using TAIR data, we use the testing set to populate a query space size of 500 and sample $n_q = 100$ queries (unless specified otherwise) with repetition from it per run according to the query frequencies in the testing set.

Evaluation Runs. We sample fresh training and testing data t times. Each time, we further sample a new set of queries from the query space r times. The concrete values for t and r are the same as in Sect. 7.1.1 (cf. Tab. 2). The attacker in each instance is given the training set and the leakage computed on the queries from the testing set. From all runs of an attack, we compute the mean recovery rate and display it together with the minimum and maximum recovery rates as uncertainty range.

8.2 Evaluating SAP

The SAP attack [33] uses the RLEN and QEQ/QFREQ patterns. As static evaluation parameters, we chose $n_q = 250$ and $n_w = 50$.

Attack Parameters. The primary attack parameter is α, which refers to the weight assigned to the response length and query frequency information utilized by the attack. More formally, SAP solves the following equation depending on α, whereby P is a permutation matrix assigning query tokens to uncovered plaintexts, C_{RLEN} is a cost metric relying on differences between *RLEN* leakage and auxiliary information, and C_{QFREQ} is an analogous cost metric relying on *QFREQ* leakage:

$$\text{argmin}_P \left(P \left[(1-\alpha) C_{RLEN} + \alpha C_{QFREQ} \right] \right)$$

Hence, $\alpha = 0$ means only RLEN information is used, whereas $\alpha = 0.5$ means both information is weighted equally. Secondly, the attack has a number of chosen keywords n_c. The adversary chooses a list of n_c keywords for which it has auxiliary information. The attack is then only evaluated on these keywords. Thus, the attack has the strong assumption that the user only queries keywords from the adversarially chosen keyword list. While a similar assumption that we also make in parts (partial query sampling) is prominent in the literature, e.g., [4, 26], the difference here is that the attacker can choose this set. In contrast, for partial query sampling the keyword list is just the keyword universe of the training data. To evaluate its influence, we will vary the value of n_c.

Further Evaluation Aspects. We additionally evaluate the attack on the TAIR data collection together with its query log. The query log can either be used solely for the evaluation given $\alpha = 0$ or it can be used both for the evaluation and as auxiliary QEQ information by splitting into training and testing data. When using the query log, we choose all keywords in the training data as chosen keywords.

8.2.1 Results. Fig. 6 shows our results for various α on all datasets. First, the combination of both leakage patterns yields the highest recovery rate for each setting, which confirms the result of the original paper [33]. Secondly, we observe that a lower number of chosen keywords allows to recover more of the queries, which was also mentioned by [33]. With a high $n_c = 3\,000$, performance is heavily degraded (compare the right plot of Fig. 6 to the left one where $n_c = 100$). Thus, the less information an attacker has on what keywords the user may query for, the less successful it will be. Another interesting observation is that while there are settings of

[5] We omit a re-evaluation of IKK [22] because this was already provided by [37].
[6] While [26] used query logs, they only considered known-data attacks in their re-evaluations.

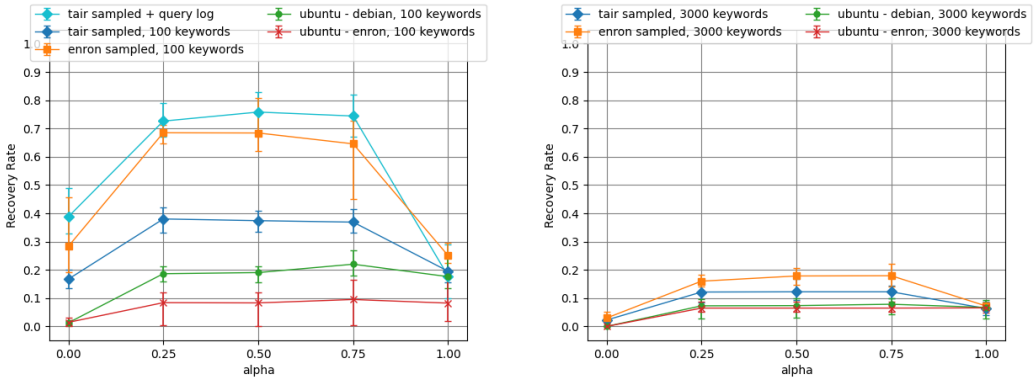

Figure 6: Evaluation of the SAP attack [33] for different values of α against different combinations of datasets. The right plot shows the recovery rates of SAP for 3 000 chosen keywords and different combinations of datasets. The same is depicted in the left part of the figure for 100 chosen keywords. In addition, the recovery rate of SAP against the sampled TAIR database using the corresponding query log is plotted (cyan diamonds), where the chosen keywords consist of all keywords in the training set.

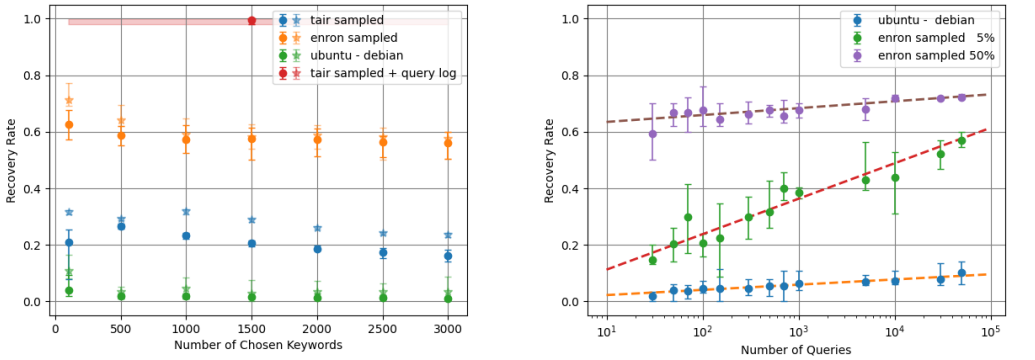

Figure 7: Evaluation of the IHOP attack [35] against different combinations of datasets. The left plot shows the recovery rates of IHOP for varying numbers of chosen keywords and different combinations of datasets. The circles show the recovery rates in the co-occurrence-only mode and the stars denote the recovery rates in the combined mode (using co-occurrence and query equality information). In addition, the plot shows the recovery rate of TAIR evaluated with its query log. In the right plot, the recovery rate of IHOP in combined mode is depicted for different numbers of queries.

the attack in which we can recover a significant amount of queries using actually different data sources (e.g., for 100 chosen keywords 26.8 % recovery on Ubuntu-Debian; cf. left plot of Fig. 6), these are solely achieved when the auxiliary query information is from the same source as the testing data (Google Trends). For $\alpha = 0$ the recovery rates are almost 0.

Influence of a Query Log. Interestingly, the recovery rate of the attack run against the sampled TAIR data collection with the TAIR query log (cf. left plot of Fig. 6) is the highest of all with a mean recovery rate of 75.8 % for $\alpha = 0.5$, despite having over 39 000 keywords in the keyword space (and, hence, the chosen keyword list). A stark difference to the evaluation without the query log[7] can be observed (below 10% recovery for 3 000 chosen keywords; cf. right part of Fig. 6). We believe this stems from the fact that in contrast to Google Trends, the queries of the query log were actually performed on the TAIR data collection. Therefore, *evaluating* queries that are more closely related to the data collection rather than queries from a different source results in higher recovery rates and higher rates than those of the original evaluations. It is, however, not to be confused with the evaluation aspect of emulating *adversary information*. For TAIR, this information is from the same data source as the private information *for both the evaluations with and without its query log*. It, hence, remains open how the attack would perform in an evaluation on queries issued on the target data collection while having no adversarial information that stems from the same source as the target data and/or queries. Unfortunately,

[7]Recall that without the query log, the evaluation uses queries and auxiliary information from Google Trends as in the original evaluation [33] (cf. Sect. 8.1), which were not originally performed on the TAIR data collection.

we cannot evaluate this, as we could not find a query log for our Ubuntu and Debian datasets.

8.3 Evaluating IHOP

Like SAP [33], the IHOP attack [35] combines different information (co-occurrence and query equality).[8] Due to its computational complexity, we restrict each data collection to the 5 000 most common keywords and set n_w to either be 10 or 50, depending on the resource demands of the evaluation. We evaluate parameters for a fixed numer of queries n_q, which in all cases except TAIR we set to be 1 000. For TAIR, we can only generate $n_q = 100$ queries due to resource limitations. Additionally, we evaluate the attack for varying n_q on the Enron, Ubuntu, and Debian datasets, because we observed that varying this parameter can influence results.

Attack Parameters. For simplicity, we also use the same parameter α as in SAP to denote the modes, using $\alpha = 0$ to refer to pure co-occurrence mode and $\alpha = 0.5$ for the combination (which we call "combined mode"). IHOP also operates on a list of n_c keywords chosen by the adversary. Hence, in our re-evaluation we vary all these parameters and additionally investigate the usage of a query log (either as a means for evaluation with $\alpha = 0$ or also split up into training and testing data for $\alpha = 0.5$). For fixed n_q, we vary the number of chosen keywords n_c. For varying n_q, we fix $n_c = 100$. When using the TAIR query log, we again set the chosen keyword list to be equal to the keyword universe of the training set.

8.3.1 Results. Fig. 7 shows the results of our re-evaluation of IHOP. Our first observation (cf. left plot of Fig. 7) is that combining CO and QEQ information leads to slightly better recovery rates for most numbers of chosen keywords. Furthermore, we see that the dependence of the recovery rate on the number of chosen keywords is negligible. Although the recovery rate decreases slightly when more keywords are selected, this decrease is not significant. In contrast to the SAP attack (cf. Sect. 8.2.1), where a higher number of selected keywords resulted in almost no recovery, this is a huge improvement. The recovery rates on the Ubuntu and Debian combination of training and testing sets remain low (almost exclusively below 15 %). Like for the SAP attack, the recovery rate of the IHOP attack run against the sampled TAIR data collection with the TAIR query log is the highest of all. The mean recovery rate is 99.4 % for both the co-occurrence-only mode and the combined mode, despite having no restriction on the number of chosen keywords. This stands in contrast to recovery below 40% for TAIR in our setting without query log, i.e., the setting of previous evaluations where unrelated Google Trends data is used for queries. As in the case of SAP (cf. Sect. 8.2.1), we attribute this to the difference in evaluation data, whereby the attack performs much better on queries related to the data. It, however, also still remains open how the attack fares on data collections from different sources but with evaluated queries actually issued on the data collection.

Number of Queries. The right part of Fig. 7 shows the recovery rate of IHOP for different numbers of queries in combined mode. We see that the recovery rate increases with the number of queries, following a logarithmic function. This is especially visible for the 5 % sample of Enron, where the recovery climbs from around 20 % at $n_q = 50$ to almost 60 % at $n_q = 50\,000$. Thus, a larger number of observed queries leads to a higher recovery rate.

Table 3: Influence of evaluation choices on the recovery rate of SAP [33] and IHOP [35]. ∼ denotes approximately constant behavior. +/− denotes a positive/negative correlation and the number of +/− signs the correlation strength.

	SAP [33]	IHOP [35]
Evaluation on query log	++	+++
Number of queries n_q	∼	+
Number of chosen keywords n_c	−−	∼

8.4 Summary

Tab. 3 summarizes the most important evaluation choices and their influence on the recovery rate of the attacks. While SAP [33] suffers from diminishing performance if the number of chosen keywords is large, this is not the case for IHOP [35]. A noticeable positive influence on attack performance is usage of a query log, both for usage as only testing set or for usage as testing and training set. However, *these instances of high attack performance with query logs are only observed on data from a single source (the TAIR dataset).*

Outlook. An interesting open problem emerges from our results, namely evaluating sampled-data attacks on query logs whereby the training and testing data stem from different sources and the query log has been issued on the testing data.[9] Such an instance is hard to find with publicy available data. However we expect corresponding evaluations, for example with the help of providers with access to relevant data for research purposes, to be a further important contribution in understanding how realistic successful sampled-data leakage attacks are.

9 CONCLUSIONS

We proposed a new metric for the statistical closeness of two data collections and demonstrated that it can be used to predict success of sampled-data leakage attacks on corresponding instances of ESAs. We re-evaluated sampled-data attacks against new data (query logs and collections from distinct sources) and found that query logs improve attack accuracy while the evaluation on distinct sources diminishes it. To fully model the common persistent, passive adversary type, evaluations on data from *both* distinct sources and with corresponding query logs are necessary. This remains an open problem, as we did not have access to such data.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their suggestions. 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 cofunded by the Deutsche Forschungsgemeinschaft (DFG) within SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/251805230.

[8] [35] also presents a QEQ-only attack assuming dependent queries. Like most other works, we only consider independent queries.

[9] Ideally, another query log issued on the training data may further be supplied to the adversary to exploit QEQ information.

REFERENCES

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

[2] Ghous Amjad, Seny Kamara, and Tarik Moataz. 2019. Breach-Resistant Structured Encryption. In *Proceedings on Privacy Enhancing Technologies (PoPETS)*, Vol. 2019.

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

[4] Laura Blackstone, Seny Kamara, and Tarik Moataz. 2020. Revisiting Leakage Abuse Attacks. In *Network and Distributed System Security Symposium (NDSS)*.

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

[6] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. 2015. Leakage-Abuse Attacks Against Searchable Encryption. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

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

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

[9] William W. Cohen. 2015. Enron Corpus. Accessed 2023-05-07, https://www.cs.cmu.edu/~./enron/.

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

[11] Marc Damie, Florian Hahn, and Andreas Peter. 2021. A Highly Accurate Query-Recovery Attack against Searchable Encryption using Non-Indexed Documents. In *USENIX Security Symposium (USENIX Security)*.

[12] Debian Mailing Lists. 2022. debian-security-announce. Accessed 2022-10-04, https://lists.debian.org/debian-security-announce/.

[13] Maria Esch, Jinbo Chen, Stephan Weise, Keywan Hassani-Pak, Uwe Scholz, and Matthias Lange. 2014. A Query Suggestion Workflow for Life Science IR-Systems. *Journal of Integrative Bioinformatics* (2014).

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

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

[16] Eu-Jin Goh. 2003. Secure Indexes. *IACR ePrint* 216 (2003).

[17] Oded Goldreich, Silvio Micali, and Avi Wigderson. 1987. How to play any mental game. In *ACM Symposium on Theory of Computing (STOC)*.

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

[19] Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G Paterson. 2018. Pump up the volume: Practical database reconstruction from volume leakage on range queries. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[20] Zichen Gui, Kenneth G Paterson, and Sikhar Patranabis. 2023. Rethinking Searchable Symmetric Encryption. In *IEEE Symposium on Security and Privacy (S&P)*.

[21] Zichen Gui, Kenneth G Paterson, and Tianxin Tang. 2023. Security Analysis of {MongoDB} Queryable Encryption. In *32nd USENIX Security Symposium (USENIX Security 23)*. 7445–7462.

[22] Islam Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. 2012. Access pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation. In *Networkand Distributed System Security Symposium (NDSS)*.

[23] Mireya Jurado, Catuscia Palamidessi, and Geoffrey Smith. 2021. A formal information-theoretic leakage analysis of order-revealing encryption. In *IEEE Computer Security Foundations Symposium (CSF)*.

[24] Mireya Jurado and Geoffrey Smith. 2019. Quantifying information leakage of deterministic encryption. In *ACM SIGSAC Conference on Cloud Computing Security Workshop (CCSW)*.

[25] Seny Kamara, Abdelkarim Kati, Tarik Moataz, Jamie DeMaria, Andrew Park, and Amos Treiber. 2023. MAPLE: MArkov Process Leakage attacks on Encrypted Search. *IACR ePrint* 810 (2023).

[26] Seny Kamara, Abdelkarim Kati, Tarik Moataz, Thomas Schneider, Amos Treiber, and Michael Yonli. 2022. SoK: Cryptanalysis of Encrypted Search with LEAKER - A framework for LEakage AttacK Evaluation on Real-world data. In *IEEE European Symposium on Security and Privacy (EuroS&P)*.

[27] Seny Kamara and Tarik Moataz. 2023. Bayesian Leakage Analysis: A Framework for Analyzing Leakage in Encrypted Search. *IACR ePrint* 813 (2023).

[28] Evgenios M Kornaropoulos, Nathaniel Moyer, Charalampos Papamanthou, and Alexandros Psomas. 2022. Leakage Inversion: Towards Quantifying Privacy in Searchable Encryption. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[29] Philippe Lamesch, Kate Dreher, David Swarbreck, Rajkumar Sasidharan, Leonore Reiser, and Eva Huala. 2010. Using The Arabidopsis Information Resource (TAIR) to Find Information About Arabidopsis Genes. *Current Protocols in Bioinformatics* 30, 1 (6 2010).

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

[31] Google LLC. 2023. Google Trends. Accessed 2023-05-07, https://trends.google.com/trends.

[32] MongoDB. 2023. MongoDB Manual. Accessed 2023-07-10, https://www.mongodb.com/docs/manual/core/queryable-encryption/.

[33] Simon Oya and Florian Kerschbaum. 2020. Hiding the Access Pattern is Not Enough: Exploiting Search Pattern Leakage in Searchable Encryption. In *USENIX Security Symposium (USENIX Security)*.

[34] Simon Oya and Florian Kerschbaum. 2020. Pre-Processed Google Trends Data. Accessed 2023-05-07, https://github.com/simon-oya/USENIX21-sap-code/blob/master/datasets_pro/enron_db.pkl.

[35] Simon Oya and Florian Kerschbaum. 2022. IHOP: Improved Statistical Query Recovery against Searchable Symmetric Encryption through Quadratic Optimization. In *USENIX Security Symposium (USENIX Security)*.

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

[37] Ruben Groot Roessink, Andreas Peter, and Florian Hahn. 2021. Experimental Review of the IKK Query Recovery Attack: Assumptions, Recovery Rate and Improvements. In *International Conference on Applied Cryptography and Network Security (ACNS)*.

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

[39] Ubuntu Mailing Lists. 2022. The ubuntu-security-announce Archives. Accessed 2022-10-12, https://lists.ubuntu.com/archives/ubuntu-security-announce/.

[40] Charles V Wright and David Pouliot. 2017. Early detection and analysis of leakage abuse vulnerabilities. *IACR ePrint* 1052 (2017).

[41] Andrew C Yao. 1982. Protocols for secure computations. In *Annual Symposium on Foundations of Computer Science (FOCS)*.

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