A preliminary version of this paper appears in *Proceeding of Privacy Enhancing Technologies (PETS) 2024.*

Understanding Leakage in Searchable Encryption: a Quantitative Approach

Alexandra Boldyreva
Georgia Institute of Technology
sasha@gatech.edu

Zichen Gui
ETH Zürich
zichen.gui@inf.ethz.ch

Bogdan Warinschi
Dfinity & University of Bristol
bogdan.warinschi@dfinity.org

October 3, 2024

Abstract

Searchable encryption, or more generally, structured encryption, permits search over encrypted data. It is an important cryptographic tool for securing cloud storage. The standard security notion for structured encryption mandates that a protocol leaks nothing about the data or queries, except for some allowed leakage, defined by the *leakage* function. This is due to the fact that some leakage is unavoidable for efficient schemes. Unfortunately, it was shown by numerous works that even innocuous-looking leakage can often be exploited by attackers to undermine users' privacy and recover their queries and/or data, despite the structured encryption schemes being provably secure. Nevertheless, the standard security remains the go-to notion used to show the "security" of structured encryption schemes. While it is not likely that researchers will design practical structured encryption schemes with no leakage, it is not satisfactory that very few works study ways to assess leakage. This work proposes a novel framework to quantify leakage. Our methodology is inspired by the quantitative information flow, and we call our method q-leakage analysis. We show how q-leakage analysis is related to the standard security. We also demonstrate the usefulness of q-leakage analysis by analyzing the security of two existing schemes with complex leakage functions.

1 Introduction

Motivation and Prior Work. Cloud storage continues to experience explosive growth. It is predicted that 100 zettabytes will be stored on the cloud by 2025, or 50% of the world's data at that time. Even if the cloud storage provider is trusted, it may be subject to security compromise. Not surprisingly, privacy of data is often a big concern or even a core compliance requirement due to PCI DSS, HIPAA, or EU Data Protection regulations.

It is well known that simply using off-the-shelf encryption with the secret key held by the client does not "work" for cloud storage applications. This is because standard encryption is so strong, its ciphertexts are not searchable. Hence in the last decade and a half, numerous research works have addressed the problem of designing protocols for secure outsourced databases.

Unfortunately, researchers still cannot offer the practitioners great solutions that offer clear security–efficiency tradeoffs suitable for applications. And this is not to say we lack solutions that vary greatly in security and efficiency. We have protocols with extremely strong security, e.g., [George et al.(2021)], but these are impractical to a degree that no implementations has even been attempted. Conversely, we have extremely efficient solutions based on property-preserving encryption (PPE) [Pandey and Rouselakis(2012)], such as deterministic encryption (DE) [Amanatidis et al.(2007), Bellare et al.(2007)], where ciphertexts leak message equality, or order-preserving encryption (OPE) [Boldyreva et al.(2009)], where ciphertexts preserve the order of the plaintexts. But not surprisingly, security of PPE schemes is quite weak and the attacks are quite devastating [Naveed et al.(2015), Bindschaedler et al.(2018)].

And then we have searchable encryption protocols that offer decent performance and stronger security than PPE, e.g. [Curtmola et al.(2006), Kamara et al.(2012), Cash et al.(2013), Stefanov et al.(2014), Naveed et al.(2014), Bost(2016), Chamani et al.(2018), Demertzis et al.(2020), Bag et al.(2023)]. All of these schemes leak some information and many recent works [Islam et al.(2012), Cash et al.(2015), Lacharité et al.(2018), Grubbs et al.(2019), Blackstone et al.(2020), Gui et al.(2023a)] showed that this leakage, even seemingly innocuous, can be exploited by attackers violating privacy of users' data and queries. Moreover, it is often very hard to understand how serious the allowed leakage is, not only for practitioners, but for the protocol designers as well. So it is not clear whether a given solution is really suitable for a particular application, security-wise. The situation is even more pressing since following the practical demand, searchable encryption protocols have started to be widely deployed recently, by AWS [Services(2023)] and MongoDB [MongoDB(2023)]. Since their security guarantees are not fully clear, attacks already started to appear [Gui et al.(2023b)] and more are expected.

We believe it is the responsibility of fundamental research to clarify for practitioners (and academics) the extent and the implications of the leakage inherent in existing schemes, and to develop methodologies that make understanding and comparing the security of novel schemes possible.

A Closer Look. Let us take a closer look at how security is traditionally defined. A typical security definition for searchable or more general structured encryption, considers two "worlds." In the "real" world the adversary interacts with the protocol and observes whatever real attackers can know, e.g., the transcript of the communication, some partial knowledge about the data and the queries, etc. In the "ideal" world the attacker interacts with the simulator, who is trying to simulate the real environment for the adversary. The simulator does not know the secret key, but it is given some function of the data and the queries called *leakage*. The leakage function models the information that the protocol is *allowed to leak*, and it varies significantly depending on each particular protocol. The goal of the adversary is to output a bit, and the protocol is considered secure if for every efficient adversary, there is an efficient simulator so that the adversary outputs the same bit in both worlds with almost the same probability. The intuition is that the attacker cannot distinguish between the real and ideal worlds, hence the real protocol "behaves" like the ideal one that is simulated without secrets, and thus it cannot leak any secrets beyond the allowed leakage.

This is a classical approach to defining security and is used, for example, for secure multiparty computation (MPC). However, the difference is that in the classic MPC security definition, the simulator is not given any leakage besides what is output by the protocol's functionality. But when the simulator is given the leakage, like in the case of searchable encryption, the definition is saying that no information *other than the allowed leakage* is leaked. The adversary's goal is to output one bit of information, and this has almost nothing to do with exploiting the leakage. The definition does not attempt to answer the question of assessing the leakage, as it is deemed "allowed." Hence, the users of each protocol that has been proven secure under such definition are left on their own to weigh the risk of the leakage. Is the resulting security good? How good is it? Of course, the answers depend on the leakage function (aka leakage profile). The problem is that it is almost impossible to give a precise answer to the above questions except for somewhat trivial leakage functions.

Many protocols hide everything about the stored data but leak the access pattern, i.e. the attacker can tell if the same data is accessed twice. This does not sound like much, but subsequent attacks show that this can be dangerous [Islam et al.(2012), Cash et al.(2015), Lacharité et al.(2018), Blackstone et al.(2020), Gui et al.(2023a)]. Similarly, many works have shown that search pattern [Oya and Kerschbaum(2021), Oya and Kerschbaum(2022)] and volume leakage [Grubbs et al.(2018), Gui et al.(2019)] can also be exploited. Moreover, sometimes the leakage is so complex, like in protocols [Pappas et al.(2014), Cash et al.(2014)], one cannot even describe it in English within several paragraphs or at all. Providing provable security statements for protocols with allowed leakage that is hard to assess does not provide a good service for practitioners. They need to have a way to assess leakage and decide when it is acceptable and when it is not.

Prior Work. There are not that many works offering ways to assess leakage. In [Boldyreva et al.(2011)], the authors studied security of the ideal object for order-preserving encryption, - an order-preserving random permutation. The paper considers a couple of dedicated security notions (for the ideal object) where the adversary had to precisely decrypt a ciphertext or find the range within the plaintext lies.

A recent work [Kornaropoulos et al.(2022)] attempts to quantify privacy for searchable encryption and proposes the notion of leakage inversion. They observe that the leakage is a function and one can define its inverse which corresponds to the collection of databases that reveal structurally equivalent patterns to the original plaintext database. Studying the entropy of such "reconstruction space" may provide useful insight into the scheme's security. However, the approach has several drawbacks. Firstly, the computation of the reconstruction space is based on combinatorics and is not always easy to compute. Secondly, the reconstruction space only includes databases that agree with the leakage fully. This means that a scheme with many possible reconstructions may be falsely labelled as a secure scheme even if all of these reconstructions are close to the real database. Thirdly, the entropy of a reconstruction space is not directly related to how resilient a scheme is against an attack, making the entropy hard to interpret. Finally, the reconstruction space may be infinitely large for some schemes. In those cases, leakage inversion will not provide any meaningful insights.

There is also a work that proposes to perform leakage analysis with Bayesian networks [Kamara and Moataz(2023)]. The idea is that leakage functions can be modeled as Bayesian networks and leakage-abuse attacks can be viewed as statistical inference algorithms on these networks. The authors formalized a notion called coherence which, informally, measures the accuracy of the leakage abuse attacks. They studied the coherence of partial and full query recovery attacks of query equality pattern and volume leakage. The main drawback of the approach is that the analysis of coherence relies on combinatorial techniques. These techniques may not be applicable to complex leakage functions in existing schemes.

Then there are results of [Jurado and Smith(2019), Jurado et al.(2021)] which apply ideas from quantitative information flow theory to help understand leakage of the deterministic and order-revealing encryption schemes. *Quantitative information flow* theory offers novel non-cryptographic techniques for quantifying the amount of leakage of sensitive information caused by systems [Alvim et al.(2012), Smith(2015)]. In this setting a system is modeled as an information-theoretic channel C taking a secret input X to an observable output Y. The adversary's prior knowledge about X is modeled with a prior probability distribution π, and the goals of the adversary, for each possible value of X is modeled with a gain function g. The prior- and posterior g-vulnerabilities then measure the adversary's expected gain under g, before and after running C, and g-leakage is the multiplicative (or additive) increase in g-vulnerability.

The results of [Jurado and Smith(2019), Jurado et al.(2021)] applying these techniques to help understand PPE schemes are very interesting but the results require having or finding the ideal object for the primitive and possibly modeling it as a channel. For example, an ideal object for deterministic encryption is a random permutation and the one for order-preserving encryption is a random order-preserving function. The works [Jurado and Smith(2019), Jurado et al.(2021)] show how to model these as channels by specifying the information-theoretic channel matrices. This may be feasible for PPE schemes as the information leakage comes from the ciphertexts themselves. However, for more complex interactive searchable encryption protocols such as SEAL [Demertzis et al.(2020)], the encrypted database does not leak much information. Instead, most of the leakage comes when queries are answered. As far as we are aware, nobody has attempted to model both as an information channel simultaneously and doing so does not seem to be a straightforward task. We provide more detailed comparison with QIF in Section 5.1.

Motivating Examples. Let us consider two schemes, which are provably-secure yet have been subject to attacks. Accordingly, the security implications for both constructions are not very clear. We are also interested in these schemes because they have noisy leakage profiles which have not been studied before.

The first scheme is the differentially private volume-hiding encrypted multi-map (DP VHEMM) proposed by Patel et al. [Patel et al.(2019)]. To hide volume pattern, the scheme adds Laplacian noise during queries to smooth out the query response volumes. The authors of [Patel et al.(2019)] defined the differential privacy property for encrypted multi-maps and proved that DP VHEMM satisfies it. The notion is similar to the original differential privacy definition [Dwork et al.(2006)]. Briefly, the adversary should not be able to distinguish between two encrypted multi-maps when the two multi-maps only differ in two multi-map keys. For these two multi-map keys, the query response volumes should only differ by 1 between the two multi-maps. However, differential privacy does not say anything about query privacy and the data privacy implications are quite limited. In particular, differential privacy does not put any constraint on the leakage of different queries, so query reconstruction may be possible if the leakage is too "big". In addition, differential privacy only captures the security of similar multi-maps (these are known as neighbouring multi-maps in the literature). For multi-maps that are far apart, differential privacy only offers very weak security bounds (through generic composition) on data privacy.

Unfortunately, [Patel et al.(2019)] did not state or prove the standard indistinguishability security of DP VHEMM. We provide that analysis for completeness, identifying the leakage. Despite proven security, Oya and Kerschbaum [Oya and Kerschbaum(2021)] presented a query reconstruction attack against DP VHEMM. Their attack does not contradict the above security results. As we noted, differential privacy does not capture query privacy. The standard notion does, but in addition to the allowed leakage. And here the issue is that leakage permits query reconstruction attacks.

The second example is of SEAL [Demertzis et al.(2020)], which is a static searchable encryption scheme with adjustable leakage. The authors of SEAL proved the standard indistinguishability security of SEAL. Furthermore, they showed experimentally the resilience of SEAL against query recovery attacks under different parameters of SEAL. On the other hand, Oya and Kerschbaum [Oya and Kerschbaum(2021)] showed that SEAL may still be vulnerable to query reconstruction attacks for certain query distributions and databases if the (approximate) query distribution and the (approximate) data distribution are known to the attacker. In particular, Oya and Kerschbaum showed that they were able to recover 23% of the queries correctly by only making 500 queries on the Enron email corpus that contains 30,109 emails.

It is clear now, after the fact, that the leakage in each scheme enables the attacks. What is not clear is whether the leakages could have been analyzed before the attacks were found. The other open questions are whether other attacks are possible and in which settings some security could be guaranteed. We tried to apply the leakage assessment methodologies from prior works to answer these questions and we ran into some difficulties. The schemes and their leakages seem to be too complex for the techniques from [Kornaropoulos et al.(2022), Kamara and Moataz(2023), Jurado and Smith(2019), Jurado et al.(2021)] to apply. We do not see a good way to apply the combinatorial techniques of [Kornaropoulos et al.(2022), Kamara and Moataz(2023)]. Similarly, it does not seem possible to specify ideal objects for the above schemes, or model them as channels (to find information-theoretic channel matrices). Moreover, the reconstruction space computed as part of the leakage inversion method of [Kornaropoulos et al.(2022)] will be infinitely large for [Patel et al.(2019)] since DP VHEMM uses random variables with unbounded support sizes (see Remark in Section 6.1). Finally, the analyses in [Kornaropoulos et al.(2022), Kamara and Moataz(2023), Jurado and Smith(2019), Jurado et al.(2021)] focus on the ideal worlds/objects and it is not obvious if they transfer to the real world.

POTENTIAL USE CASES. We believe that q-leakage analysis can also be applied to more complex schemes, e.g., for boolean queries [Bag et al.(2023), Cash et al.(2013), Kamara and Moataz(2017)] and graph queries [Meng et al.(2015), Ghosh et al.(2021)]. We leave these analyses to future work.

Our Contributions. We propose a new methodology to quantify security implications of leakage in searchable encryption schemes. Our approach is inspired by the quantitative information flow theory and works [Alvim et al.(2012), Smith(2015)]. We believe our method is more applicable than previous approaches and show how it can be used to provide useful security insigts about DP VHEMM and SEAL .

In short, we suggest a method to define and quantify adversarial success in achieving various security goals when given the leakage and the ideal experiment. One can do so by considering various *quantitative leakage* functions and computing their expectations to analyze protocols' security. Special care is needed to ensure that these bounds translate to the real experiment with the protocol's execution, even when the standard security guarantees that the experiments are indistinguishable. We show the applicability of our methodology to obtain novel security results for two existing protocols. We now discuss our contributions in more detail.

q-LEAKAGE ANALYSIS. Recall that standard security notions ensure that the protocol does not leak more than some permitted leakage. To quantify the (in)security of the leakage we propose to use public functions which we call *quantitative leakage* or *ql* functions. Roughly speaking, the *ql*-function is a function of some secret information pertaining to the system under attack and the adversary's output, and reflect the "payout" that the adversary gets corresponding to his guess. Different ql-functions can be used to model different adversarial goals. For example, the goal of the adversary in [Naveed et al.(2015)] is to maximize the fraction of correct keyword guesses. We can easily imagine other important adversarial goals, such as the probability of a guess of any keyword or of all keywords, etc. Moreover, *ql*-functions can express much more fine-grained goals. For instance, one can define a ql-function measuring the attackers' ability to guess low-frequency keywords, if such keywords are more sensitive in the application. Such a ql-function will reward the attacker more if the guessed keyword has low frequency. We note that it does not seem possible to avoid considering various classes of attacks separately, since there cannot be an all-or-nothing leakage assessment of leakage. This is because there is always the ql-function asking to compute the whole leakage, but then the scheme is trivially insecure.

We believe that it is useful and natural to consider expectations of the ql-functions (that we call *expected quantitative leakage*) to analyze protocol security. This is because unlike leakage profiles that may be hard to understand, the expected quantitative leakage is a number that directly quantifies the damage of attacks. Having fixed a ql-function, the expected quantitative leakage of different schemes or a scheme instantiated with different parameters can be compared.

For known leakage, we quantitatively measure the ability of an adversary to compute some useful information defined by the ql-function. More concretely, there is an (ideal) experiment $\mathcal{Z}$ that generates a private key and a database, polynomially many queries and some auxiliary information aux about the database and the queries. Then the simulator $\mathcal{S}$, given the leakage, simulates the interaction with the attacker $\mathcal{A}$, who gets the auxiliary information aux that models the a-priori information. The adversary outputs a guess string ω, to maximize the expectation of the ql-function as the function of ω, the database and the queries. Security of a scheme with respect to a function ql is defined to be the supreme of expectation of the ql-function over all adversaries and experiments. Note that even though the adversary does not pick the database and the queries, we still model the prior knowledge with aux.

Now, one would expect that the above q-leakage analysis performed in the ideal world given the leakage carries over to the real world with the actual protocol execution, since the standard security shows that the worlds are indistinguishable. However, somewhat surprisingly, this is not always true, i.e., indistinguishability does not always imply that the expected quantitative leakages are the same in both worlds. In Section 5.1 we present a simple searchable encryption scheme and discuss its security and leakage. We then study a particular ql-function and show that the difference between the expected quantitative leakages in the real and ideal worlds is not negligible, even though the adversary cannot distinguish the worlds except with negligible probability.

Once we establish that the relation does not hold in general, we find under which conditions one can rely on already existing results. We show that a wide class of "practical" ql-functions, and for schemes proven to be indistinguishable, the expectation in the real world differs from the expectation in the ideal world by at most a negligible constant. Section 5 contains the formal definition and discusses the subtleties with regard to the order of quantifiers.

In Sections 5.2, 5.3 we discuss how one can choose ql-functions and how to interpret the concrete expected quantitative leakage bounds. We suggest comparing the expected quantitative leakage of the scheme with the expected quantitative leakage of an ideal scheme with no leakage. This provides a baseline measurement of how well the scheme hides the secrets captured by the ql-function.

q-Leakage Analysis Flow. To summarise, here are the steps of the analysis in our framework. First, a structured encryption scheme is proven secure with some formally defined leakage under the standard security definition. Then a set of ql-functions is chosen to capture some practical security goals. Also, one has to specify the environment capturing the application, such as known data or query distributions, etc. Next, a bound is proven for each ql-function in the ideal experiment. (This can be the most challenging part of the analyses.) Finally, by our results, if the ql-function satisfies the required property, the same bound is guaranteed in the real experiment where the attacker tackles the same goal interacting with the protocol. Our method permits both security and insecurity analyses. The results should allow practitioners to make more informed decisions regarding the security-efficiency trade-offs for their applications.

Case studies. Equipped with the new q-leakage analysis method, we show how it can be used to provide more fine-grained analysis for the protocols from our motivating examples. For the DP VHEMM [Patel et al.(2019)] analysis we could use our notion to formally capture that attack from [Oya and Kerschbaum(2021)] by defining the appropriate ql-function and calculating the lower bound on its expectation. But that would not be particularly useful after the fact. Instead, we show how our notion can be used to do a more fine-grained security analysis. The attack in [Oya and Kerschbaum(2021)] uses the fact that the query response volumes for the most frequent multi-map keys (for the dataset they have used) are far apart. However, not all multi-maps can be exploited this way. As extreme examples, consider a multi-map where the keys are student ids and the values are the genders of the students, or a multi-map where the keys are patient identifiers and the values are type of admission (see Texas Inpatient Public Use Data File (PUDF) [for Health Statistics(2023)] for a real-world example). For these multi-maps, the query response volumes will be 1 for every query since the multi-map keys are unique. We show that for multi-maps with "similar" query response volumes, DP VHEMM do offer a certain level of query privacy. We do this by picking an appropriate environment and *ql*- function that rewards guesses on the keys with large query response volumes more than the keys with small query response volumes, and providing its upper bound.

For some multi-maps, it may be okay to leak the identity of the multi-map keys with large query response volumes. For example, in a multi-map that holds medical records, it is not very impactful if an attacker manages to recover a query on common flu. On the other hand, it is a lot more detrimental if an attacker can learn that a query is on a cancer. To capture the scenario described above, we pick an environment and a *ql*-function that rewards correct guesses on low-frequent multi-map keys, and again, prove the upper bound.

To provide more insight about SEAL's security, we consider two *ql*-functions, one modeling generic query recovery attacks and the other modelling attacks that focus on the recovery of low-frequency keywords. For the first *ql*-function, we show a generic upper bound on the expected quantitative leakage. For the second *ql*-function, we show an upper bound for databases with keyword frequencies following Zipf distribution. Let x be the padding parameter of SEAL (x means that all query response volumes are padded to the next power of x). One surprising result we found is that for a database following Zipf distribution and a particular *ql*-function (ql_4 in Section 7), the expected quantitative leakage we computed is proportional to $\frac{x}{x-1}$. This suggests that padding is mostly effective when x is small and it has a diminishing return in terms of the expected quantitative leakage. We believe that this kind of result is only possible with approaches similar to our *q*-leakage analysis as a numerical output from the security notion is needed for such interpretations.

Summary. We hope future work will provide more useful analyses of practical schemes using our *q*-leakage framework, bringing new insights into their security.

2 Notation

We use the notation $(x_i)_{i=0}^{N}$ to represent a list (tuple) $(x_0, \ldots, x_N)$. To present running an interactive algorithm F between two parties, we use the notation $(\text{output}_\mathcal{A}, \text{output}_\mathcal{B}) \leftarrow [F_\mathcal{A}(\text{input}_\mathcal{A}),\ F_\mathcal{B}(\text{input}_\mathcal{B})]$. In the notation, $f_\mathcal{A}$ is the part of the interactive function run by party $\mathcal{A}$ and $f_\mathcal{B}$ is the part of the interactive function run by party $\mathcal{B}$. Let $T = (T_1, \ldots, T_n)$ be a tuple, we write $T[i]$ to denote the i-th component of T, i.e. $T[i] = T_i$. We write $\overrightarrow{v}$ to mean a list.

3 Preliminaries: Structured Encryption and its Security

Structured encryption is the generalization of searchable encryption, which in turn is the generalization of encrypted keyword search over documents [Goh(2003), Chang and Mitzenmacher(2005), Curtmola et al.(2006)]. Structured encryption was formalised in 2010 by Chase and Kamara [Chase and Kamara(2010)]. In this section, we recall the syntax and security definition for structured encryption. We start by recalling the definition of abstract data types.

Abstract Data Types. An abstract data type $\mathcal{S} = (\mathsf{Data}, \mathsf{Q}, \mathsf{RSPN}, \textbf{Query})$ is defined by a set of data objects Data, a set of query descriptions Q, a set of responses RSPN and a set **Query** of operations (queries) of the form $\mathsf{Data} \times \mathsf{Q} \rightarrow \mathsf{Data} \times \mathsf{RSPN}$.

Syntax for Structured Encryption. Our syntax is adapted from [Chase and Kamara(2010)] with slight modifications, explained later. The client **Clt** and server **Svr** in the syntax can be stateful.

Let $\mathcal{S} = (\mathsf{Data}, \mathsf{Q}, \mathsf{RSPN}, \textbf{Query})$ be an abstract data type. A private-key structured encryption scheme Σ for $\mathcal{S}$ with security parameter 1^λ and public parameter pub is defined by the following algorithms $\Sigma = (\textbf{Gen}, \textbf{Setup}, \textbf{EQuery})$:

- $\mathsf{sk} \leftarrow \mathbf{Gen}(1^\lambda, \mathsf{pub})$ is a probabilistic algorithm run by the client. It takes as input a security parameter 1^λ and a public parameter pub, and outputs a secret key sk.

- $(\perp, \mathsf{edata}) \leftarrow [\mathbf{Setup}_{\mathbf{Clt}}(\mathsf{pub}, \mathsf{sk}, \mathsf{data}), \mathbf{Setup}_{\mathbf{Svr}}()]$ is an interactive algorithm between the client and the server. The client takes as input a public parameter pub, a secret key sk and some data $\mathsf{data} \in \mathsf{Data}$, and the server does not take any input; after the interaction, the server outputs some encrypted data edata.

- $(\mathsf{rspn}, \mathsf{edata}') \leftarrow [\mathbf{EQuery}_{\mathbf{Clt}}(\mathsf{pub}, \mathsf{sk}, \mathsf{q}), \mathbf{EQuery}_{\mathbf{Svr}}(\mathsf{edata})]$ is an interactive query protocol between the client and server. The client takes as input a public parameter pub, a secret key sk and a query type $\mathsf{q} \in \mathsf{Q}$, and the server takes as input encrypted data edata; after the interaction, the client outputs a response rspn, and the server obtains updated data edata'.

Structured encryption Σ associated with $\mathcal{S} = (\mathsf{Data}, \mathsf{Q}, \mathsf{RSPN}, \mathbf{Query})$ is correct if for all $\lambda \in \mathbb{N}$, all pub, all sk output by $\mathbf{Gen}_{\mathbf{Clt}}$ $(1^\lambda$, $\mathsf{pub})$, for all $\mathsf{data}_0 \in \mathsf{Data}$, edata_0 output by the server after running $[\mathbf{Setup}_{\mathbf{Clt}}(\mathsf{pub}, \mathsf{sk}, \mathsf{data}_0), \mathbf{Setup}_{\mathbf{Svr}}()]$, for all query sequences $(\mathsf{q}_1, \ldots, \mathsf{q}_l) \in \mathsf{Q}^l$, it holds that $\mathsf{rspn}_i = \mathsf{rspn}'_i$ for all $i = 1, \ldots, l$, where $(\mathsf{data}_i, \mathsf{rspn}_i) \leftarrow \mathbf{Query}(\mathsf{data}_{i-1}, \mathsf{q}_i)$ and $(\mathsf{rspn}'_i, \mathsf{edata}_i) \leftarrow [\mathbf{EQuery}_{\mathbf{Clt}}(\mathsf{pub}, \mathsf{sk}, \mathsf{q}_i)$, $\mathbf{EQuery}_{\mathbf{Svr}}(\mathsf{edata}_i)]$, except for a negligible probability.

We say the scheme Σ is static if its corresponding plaintext query protocol **Query** never changes the data. Otherwise, we call the scheme dynamic. We define syntax specific to particular cases of encrypted multi-maps and searchable encryption below.

Encrypted Multi-maps (EMMs). A multi-map $\mathcal{S} = (\mathsf{MM}, \mathsf{Q}, \mathsf{RSPN}, \mathbf{Query})$ is an abstract data type such that the data is a set of key-value tuples $\mathsf{MM} = \{(\mathtt{key}_i, \overrightarrow{v_i})\}_{i \in [N]}$ where N is the number of multi-map keys in MM. Each multi-map key $\mathtt{key}_i$ is associated with a list of values $\overrightarrow{v_i}$. We write $\ell(\mathtt{key}_i)$ to mean the number of values associated to multi-map $\mathtt{key}_i$. For simplicity, we consider static multi-maps, where the only queries supported are key queries. For these queries, $\mathsf{Q} = \{\mathtt{key}_i\}_{i \in [N]}$ and $\mathsf{RSPN} \subset \{0,1\}^*$. A key query **Query** takes as input a multi-map MM and a key $\mathtt{key} \in \mathsf{Q}$, and outputs a list of values associated to the key $\mathtt{key}$, denoted as $\mathsf{MM}[\mathtt{key}] \in \mathsf{RSPN}$. An encrypted multi-map is a structured encryption scheme with multi-maps as the abstract data type.

Searchable Encryption. Document collection $\mathcal{S} = (\mathsf{DB}, \mathsf{Q}, \mathsf{RSPN}, \mathbf{Query})$ is a data type such that the data is a set of document-keywords tuples $\mathsf{DB} = ((\mathsf{d}_1, \{\mathsf{w}_{1,j}\}_{j=1}^{l_1}), \ldots, (\mathsf{d}_N, \{\mathsf{w}_{N,j}\}_{j=1}^{l_N}))$. For the purpose of this paper, we consider static document collections, that is, the only queries **Query** supported by $\mathcal{S}$ are keyword search queries. A keyword search query takes as input a document collection DB and a keyword $\mathsf{w} \in \mathsf{Q}$, and outputs a list of documents that contain the keyword $\left\{\mathsf{d}_j \mid \mathsf{w} \in \{\mathsf{w}_{j,i}\}_{i=1}^{l_j}\right\} \in \mathsf{RSPN}$. A structured encryption scheme that supports document collections is known as *searchable encryption.*

Security of Structured Encryption. We adapt the security definition from [Chase and Kamara(2010)] to our syntax (for the intuition behind this definition please see the Introduction).

Definition 1 (Adaptive Security of Interactive STE (SS-CQA-B))**.** Let $\mathcal{S} = (\mathsf{Data}, \mathsf{Q}, \mathsf{RSPN}, \mathbf{Query})$ be an abstract data type and let $\Sigma = (\mathbf{Gen}, \mathbf{Setup}, \mathbf{EQuery})$ be the associated structured encryption scheme. Let 1^λ be a security parameter and pub be a public parameter. Figure 1 presents the two games associated with a stateful semi-honest adversary $\mathcal{A}$ and a stateful simulator $\mathcal{S}$. The subscript $\mathcal{A}$ denotes that the algorithm is executed by the adversary, and hence it learns the corresponding randomness and communication. $\mathcal{L} = (\mathcal{L}_{\mathbf{Setup}}, \mathcal{L}_{\mathbf{EQuery}})$ is the leakage profile, where:

- $\mathcal{L}_{\mathbf{Setup}}$ is a function capturing leakage during **Setup**. $\mathcal{L}_{\mathbf{Setup}} : \{0,1\}^* \times \mathsf{Data} \rightarrow \{0,1\}^*$ takes as input a public parameter pub and data $\mathsf{data} \in \mathsf{Data}$, and outputs a description of the leakage (e.g. the size of data $|\mathsf{data}|$).

- $\mathcal{L}_{\mathbf{EQuery}}$ is a *stateful* function capturing leakage during queries. $\mathcal{L}_{\mathbf{EQuery}} : \{0,1\}^* \times \mathsf{Q}^* \times \mathsf{Data} \rightarrow \{0,1\}^*$ takes as input a public parameter pub, a sequence of queries $(\mathsf{q}_j)_{j=1}^l \in \mathsf{Q}^l$ and data $\mathsf{data} \in \mathsf{Data}$, and outputs a description of the leakage for the queries $(\mathsf{q}_j)_{j=1}^l$ (e.g. query equality which reveals which queries are the same). $\mathcal{L}_{\mathbf{EQuery}}$ is evaluated several times (see line 5 of $\text{Ideal}^{\text{SS-CQA-B}}_{\Sigma,\mathcal{A},\mathcal{L},\mathcal{S}}(1^\lambda, \mathsf{pub})$ in Figure 1).

```
$\mathrm{Real}^{\text{SS-CQA-B}}_{\Sigma,\mathcal{A}}(1^\lambda, \mathsf{pub})$
1: $\mathsf{data} \leftarrow \mathcal{A}(1^\lambda, \mathsf{pub})$
2: $\mathsf{sk} \leftarrow \mathbf{Gen}_{\mathbf{Clt}}(1^\lambda, \mathsf{pub})$
3: $(\perp, \mathsf{edata}) \leftarrow \big[\mathbf{Setup}_{\mathbf{Clt}}(\mathsf{pub}, \mathsf{sk}, \mathsf{data}), \mathbf{Setup}_{\mathcal{A}}()\big]$
4: $i \leftarrow 1$
5: while $\mathsf{q}_i \leftarrow \mathcal{A}()$ do
6:     $(\mathsf{rspn}_i, \mathsf{edata}) \leftarrow \big[\mathbf{EQuery}_{\mathbf{Clt}}(\mathsf{pub}, \mathsf{sk}, \mathsf{q}_i),\ \mathbf{EQuery}_{\mathcal{A}}(\mathsf{edata})\big]$
7:     $i \leftarrow i + 1$
8: $b \leftarrow \mathcal{A}()$
9: return $b$
```

```
$\mathrm{Ideal}^{\text{SS-CQA-B}}_{\Sigma,\mathcal{A},\mathcal{L},\mathcal{S}}(1^\lambda, \mathsf{pub})$
1: $\mathsf{data} \leftarrow \mathcal{A}(1^\lambda, \mathsf{pub})$
2: $(\perp, \mathsf{edata}) \leftarrow \big[\mathcal{S}(1^\lambda, \mathsf{pub}, \mathcal{L}_{\mathbf{Setup}}(\mathsf{pub}, \mathsf{data})),$
                     $\mathbf{Setup}_{\mathcal{A}}(1^\lambda)\big]$
3: $i \leftarrow 1$
4: while $\mathsf{q}_i \leftarrow \mathcal{A}()$ do
5:     $(\mathsf{rspn}_i, \mathsf{edata}) \leftarrow \big[\mathcal{S}(1^\lambda, \mathsf{pub}, \mathcal{L}_{\mathbf{EQuery}}(\mathsf{pub}, (\mathsf{q}_j)_{j=1}^{i}, \mathsf{data}), \mathbf{EQuery}_{\mathcal{A}}(\mathsf{edata})\big]$
6:     $i \leftarrow i + 1$
7: $b \leftarrow \mathcal{A}()$
8: return $b$
```

Figure 1: Experiments for SS-CQA-B definition.

We say that a structured encryption scheme Σ with security parameter 1^λ and public parameter pub is adaptive SS-CQA-B-BB secure (semantic secure against chosen query attack with bit output in the black-box model) with leakage $\mathcal{L}$ if there is a simulator $\mathcal{S}$ such that for every PPT adversary $\mathcal{A}$,

$$\left|\mathbf{Pr}\left[\mathrm{Real}^{\text{SS-CQA-B}}_{\Sigma,\mathcal{A}}(1^\lambda, \mathsf{pub}) = 1\right] - \mathbf{Pr}\left[\mathrm{Ideal}^{\text{SS-CQA-B}}_{\Sigma,\mathcal{A},\mathcal{S}}(1^\lambda, \mathsf{pub}) = 1\right]\right| \le \mathtt{negl}(\lambda).$$

We say that a structured encryption scheme Σ is adaptive SS-CQA-B-NBB secure (semantic secure against chosen query attack with bit output in the non-black-box model) with leakage $\mathcal{L}$ if for every PPT adversary $\mathcal{A}$, there is a simulator $\mathcal{S}$ such that the above inequality holds.

Comparison with Prior Definitions. The syntax in [Chase and Kamara(2010)] does not consider interactive protocols. Given that many recent schemes [Patel et al.(2019), Demertzis et al.(2020), Bag et al.(2023)] are interactive in nature, we opt for the most general syntax where both the setup protocol and the query protocol can be interactive. Also for generality, we do not consider tokens explicitly. Unlike [Chase and Kamara(2010)], we introduced the public parameter pub in the syntax. For example, the parameter for the Laplace noise and the amount of padding used in the construction from [Patel et al.(2019)] is an example of a public parameter. In prior works such as [Chen et al.(2018), Kamara and Moataz(2019), Patel et al.(2019), Demertzis et al.(2020)], the public parameter is assumed to be implicitly known, but we made it explicit to improve the clarity of the description of the syntax and the security notion.

Following [Chase and Kamara(2010)], we allow leakage in the security definition to be stateful, while this is not explicitly mentioned in many works, like [Kamara and Moataz(2017), Kamara and Moataz(2019), Patel et al.(2019), Demertzis et al.(2020), Minaud and Reichle(2022)]. Indeed, this causes a problem as all of these works require a stateful leakage function in their security proofs.

In addition, we note that most papers in the literature [Kamara et al.(2012), Kamara and Papamanthou(2013), Cash et al.(2014), Bost et al.(2017)] use the non-black-box security definition in the security statements they make, but the actual proofs are for the black-box security definition. This observation is important for connecting the standard notion to our new notion. More details can be found in Section 5.1.

4 Leakage and its Implications

Unlike many other cryptographic primitives, provably-secure structured encryption schemes are allowed to leak some information about the values they are supposed to protect: data or queries. This is permitted by the security notion we discussed above via the leakage function given to the simulator. More precisely, a security proof for a structured encryption scheme only guarantees that an adversary (e.g. an honest-but-curious server) cannot learn more information about the database and the queries than what is quantified by the leakage. We recall some of the most common leakage profiles permitted by many constructions [Curtmola et al.(2006), Kamara et al.(2012), Cash et al.(2013), Stefanov et al.(2014), Naveed et al.(2014), Bost(2016), Chamani et al.(2018), Demertzis et al.(2020), Bag et al.(2023)].

Common Leakage Profiles. We recall some common leakage profiles $\mathcal{L}_{\mathbf{EQuery}}$, which takes inputs $(\mathsf{pub}, (\mathsf{q}_i)_{i=1}^{l}, (\mathsf{data}_i)_{i=1}^{l}))$.

Access-pattern Leakage is typically modeled as a function that outputs a M-by-l matrix AP such that $\mathsf{AP}_{i,j} = 1$ if data element data_i is in the query response of query q_j and $\mathsf{AP}_{i,j} = 0$ otherwise.

Query Response Volume Leakage (or simply volume leakage) outputs a list L of length l such that $L[i] = |\mathsf{rspn}|$, where rspn is the second output of $\mathbf{Query}(\mathsf{data}, \mathsf{q})$.

Query Equality Pattern outputs an l-by-l matrix QE so that $\mathsf{QE}_{i,j} = 1$ if $\mathsf{q}_i = \mathsf{q}_j$ and $\mathsf{QE}_{i,j} = 0$ otherwise.

Leakage-Abuse Attacks. As we have alluded to in the introduction, the implications of the leakage are not easy to assess. Leakage profiles as above may seem unavoidable or innocuous. However, information leaked by many of the provably-secure schemes can be exploited by an adversary. Often, the results are devastating in that they reconstruct the plaintext database or the queries to a large degree. These attacks are known as *leakage-abuse attacks* in the literature.

Discussion. Note that all the leakage-abuse attacks in the literature do not violate SS-CQA-B-BB security. Clearly, it is a troublesome situation when a provably-secure structured encryption scheme can be subjected to attacks. To address this problem, one may attempt to build schemes with close to no leakage and argue that no attack is possible. However, this approach will likely lead to inefficient schemes. Alternatively, we can try to improve the security notion so that it captures the (in)feasibility of attacks directly. We opt for the latter approach.

5 Leakage Analysis Definition

We propose a novel methodology to assess the security implications of leakage in structured encryption. q-leakage analysis is inspired by the quantitative information flow techniques of Alvim et al. [Alvim et al.(2012)]. The core idea is to introduce a *quantitative leakage function* that measures how well an adversary can learn some private information. The ql-function can be chosen appropriately to capture certain classes of attacks, for example, query reconstruction attacks. Then bounding the ql-function's output would show resistance to the corresponding class of attacks. In contrast to the traditional advantage measures of adversarial success, our ql-function offers a much more fine-grained assessment. Our analysis allows us to reward the attacker for partial guesses and also for making more "valuable" guesses. We discuss this flexibility and how to choose ql-functions in Sections 5.2 and 5.3.

5.1 q-Leakage Analysis

We start with a natural notion capturing the ability of an attacker learning some secret information.

Real^{ql} Notion. The formal definition is in Definition 2, but we start with an informal description. We fix an abstract data type and the associated structured encryption scheme. Consider two parties, the environment $\mathcal{Z}$ and stateful PPT adversary $\mathcal{A}$. The environment $\mathcal{Z}$ generate secret key sk, some data data_0, polynomially many queries $(\mathsf{q}_j)_{j=1}^{l}$ and some auxiliary information aux on the data and the queries. Then the environment and the adversary run the setup protocol where the environment plays the role of the client and the adversary plays on behalf of the server. Next, the environment and the adversary run the query protocol with queries $(\mathsf{q}_j)_{j=1}^{l}$ where again the environment plays the role of the client and the adversary plays the role of the server. Finally, the adversary is given aux and outputs a guess ω. The output of the experiment is $ql(\omega, \mathsf{pub}, (\mathsf{data}_j)_{j=1}^{l}, (\mathsf{q}_j)_{j=1}^{l})$, where $\omega \in \{0,1\}^*$ is a string output by the adversary, pub is the public parameter, $(\mathsf{data}_j)_{j=1}^{l}$ are data generated at the start of the game and by the queries, and $(\mathsf{q}_j)_{j=1}^{l}$ are queries generated by the environment.

The supreme of the expected quantitative leakage of a function ql for scheme Σ with respect to environment $\mathcal{Z}$ is defined to be the supreme of expectation of the game over all PPT adversaries $\mathcal{A}$, i.e. $\sup_{\mathcal{A}} \mathbf{E}\left[\mathrm{Real}_{\Sigma,\mathcal{A},\mathcal{Z}}^{ql}(1^\lambda, \mathsf{pub})\right]$.

Definition 2 ($\mathbf{Real}^{ql}$). Let $\mathcal{S} = (\mathsf{Data}, \mathsf{Q}, \mathsf{RSPN}, \mathbf{Query})$ be an abstract data type and let $\Sigma = (\mathbf{Gen}, \mathbf{Setup}, \mathbf{EQuery})$ be the associated structured encryption scheme. Let 1^λ be a security parameter and pub be a public parameter. Figure 2 presents the experiment associated with the environment $\mathcal{Z}$ and the stateful semi-honest adversary $\mathcal{A}$. The environment plays the role of the client and the adversary plays the role of the server. The game is parametrized by the function ql.

Given a particular environment $\mathcal{Z}$ and function ql, we define the *expected quantitative leakage* as the supremum of the output of the game $\mathrm{Real}_{\Sigma,\mathcal{A},\mathcal{Z}}^{ql}(1^\lambda, \mathsf{pub})$ over all PPT adversaries $\mathcal{A}$, i.e. $\sup_{\mathcal{A}} \mathbf{E}\left[\mathrm{Real}_{\Sigma,\mathcal{A},\mathcal{Z}}^{ql}(1^\lambda, \mathsf{pub})\right]$.

```
$\mathrm{Real}_{\Sigma,\mathcal{A},\mathcal{Z}}^{ql}(1^\lambda, \mathsf{pub})$
1: $(\mathsf{data}_0, (\mathsf{q}_j)_{j=1}^{l}, \mathsf{aux}) \leftarrow \mathcal{Z}(1^\lambda, \mathsf{pub})$
2: $\mathsf{sk} \leftarrow \mathbf{Gen}_{\mathcal{Z}}(1^\lambda, \mathsf{pub})$
3: $(\perp, \mathsf{edata}_0) \leftarrow [\mathbf{Setup}_{\mathcal{Z}}(\mathsf{sk}, \mathsf{pub}, \mathsf{data}_0), \mathbf{Setup}_{\mathcal{A}}()]$
4: for $i \leftarrow 1, \ldots, l$ do
5:     $(\mathsf{rspn}_i, \mathsf{edata}_i) \leftarrow [\mathbf{EQuery}_{\mathcal{Z}}(\mathsf{sk}, \mathsf{pub}, \mathsf{q}_i),\ \mathbf{EQuery}_{\mathcal{A}}(\mathsf{edata}_{i-1})]$
6:     $(\mathsf{data}_i, \mathsf{rspn}'_i) \leftarrow \mathbf{Query}(\mathsf{data}_{i-1}, \mathsf{q}_i)$
7: $\omega \leftarrow \mathcal{A}(1^\lambda, \mathsf{pub}, \mathsf{aux})$
8: return $ql(\omega, \mathsf{pub}, (\mathsf{data}_j)_{j=0}^{l}, (\mathsf{q}_j)_{j=1}^{l})$
```

Figure 2: Real^{ql} experiment.

Remarks. The parties we consider are stateful though we do not explicitly specify their states.

We could have given ql aux, but this is not necessary as aux models some partial information about data and queries already given to it.

One can generalise our leakage analysis to other adversarial models. For example, the adversary may be allowed to pick the queries after it receives the encrypted database. This corresponds to an adversary who can execute non-adaptive queries on the server. A natural modification from there is to make the adversary adaptive, meaning that the adversary can choose the next query based on what it has seen so far. However, in this case, the ql-function needs to be crafted carefully to avoid letting the adversary win trivially (e.g. by guessing the queries he has chosen).

We note that our framework captures the worst-case security using the conditions specified by the environment. If a specific subset of samples is a concern, one can use an environment that generates these samples exclusively.

It can be quite hard to use the above notion directly to assess leakage of specific constructions. To make security proofs more feasible, we use the standard real-ideal paradigm and define the expected quantitative leakage in the ideal world, as it is easier to assess.

Ideal^{ql} Notion. The ideal experiment, or world, for our notion is presented in Figure 3. It is similar to the ideal experiment in the standard security notion [Chase and Kamara(2010)].

We consider three stateful parties: the environment $\mathcal{Z}$, the adversary $\mathcal{A}$ and the simulator $\mathcal{S}$. The experiment is also parametrized by the leakage profile $\mathcal{L} = (\mathcal{L}_{\mathbf{Setup}}, \mathcal{L}_{\mathbf{EQuery}})$ and the function ql. The adversary interacts with $\mathcal{S}$ who is given the output of the leakage functions. The game proceeds in the same way as the real game except that all real operations on the encrypted database (i.e. **Setup** and **EQuery**) are replaced by a simulator $\mathcal{S}$ simulating these operations with inputs given by the leakage function $\mathcal{L}$. By the end of the experiment, the adversary outputs a guess ω and the game outputs ql of ω (with all other relevant inputs). Similar to the real notion, we can define the supremum of the expected quantitative leakage as follows.

Given a particular environment $\mathcal{Z}$, a particular simulator $\mathcal{S}$ and function ql, we define the *expected quantitative leakage* as the supremum of the output of the game $\mathrm{Ideal}^{ql}_{\Sigma,\mathcal{A},\mathcal{Z},\mathcal{S}}(1^\lambda, \mathsf{pub})$ over all PPT adversaries $\mathcal{A}$, i.e. $\sup_{\mathcal{A}} \mathbf{E}\left[\mathrm{Ideal}^{ql}_{\Sigma,\mathcal{A},\mathcal{Z},\mathcal{S}}(1^\lambda, \mathsf{pub})\right]$.

We remark that for deterministic environments outputting the data and the queries the adversary could know those and could always “win” and hence the expected quantitative leakage may not be very informative. But we envision all environments describing practical attacks to contain some randomized challenges and the issue will not arise.

Indistinguishability does not Imply Equality in Expectation. Ideally, we want to show that for SS-CQA-B-BB secure schemes the expected quantitative leakage in the ideal world (Figure 3) and the expected quantitative leakage in the real world (Figure 2) are the same except for a negligible difference. Then, we can use the ideal experiment to compute the expected quantitative leakage. However, it turns out this does not hold true in general, even if the scheme is provably SS-CQA-B-BB secure.

We present a counterexample to show that the expectation in the real game cannot always be bounded by the expectation in the ideal game, even if the underlying scheme is provably secure. We then formalise conditions for which the expectation in the ideal world can be used to bound the expectation in the real world in Theorem 1.

```
$\mathrm{Ideal}^{ql}_{\Sigma,\mathcal{A},\mathcal{Z},\mathcal{S}}(1^\lambda, \mathsf{pub})$
1: $(\mathsf{data}_0, (\mathsf{q}_j)_{j=1}^{l}, \mathsf{aux}) \leftarrow \mathcal{Z}(1^\lambda, \mathsf{pub})$
2: $(\perp, \mathsf{edata}_0) \leftarrow [\mathcal{S}(1^\lambda, \mathsf{pub}, \mathcal{L}_{\mathbf{Setup}}(\mathsf{pub}, \mathsf{data}_0)),$
                      $\mathbf{Setup}_{\mathcal{A}}()]$
3: for $i \leftarrow 1, \ldots, l$ do
4:     $(\mathsf{rspn}_i, \mathsf{edata}_i) \leftarrow [\mathcal{S}(1^\lambda, \mathsf{pub}, \mathcal{L}_{\mathbf{EQuery}}(\mathsf{pub}, (\mathsf{q}_j)_{j=1}^{l}, \mathsf{data}_0)), \mathbf{EQuery}_{\mathcal{A}}(\mathsf{edata}_{i-1})]$
5:     $(\mathsf{data}_i, \mathsf{rspn}'_i) \leftarrow \mathbf{Query}(\mathsf{data}_{i-1}, \mathsf{q}_i)$
6: $\omega \leftarrow \mathcal{A}(1^\lambda, \mathsf{pub}, \mathsf{aux})$
7: return $ql(\omega, \mathsf{pub}, (\mathsf{data}_j)_{j=0}^{l}, (\mathsf{q}_j)_{j=1}^{l})$
```

Figure 3: Ideal^{ql} experiment.

Let 1^λ be the security parameter and pub be the public parameter. Consider the following structured encryption scheme Σ for some abstract data type $\mathcal{S} = (\mathsf{Data}, \mathsf{Q}, \mathsf{RSPN}, \mathbf{Query})$. The key generation algorithm $\Sigma.\mathbf{Gen}(1^\lambda, \mathsf{pub})$ generates a random key $\mathsf{sk} \leftarrow \$\{0,1\}^\lambda$. The setup algorithm $\Sigma.\mathbf{Setup}$ is non-interactive. The client encrypts data as $\mathsf{sk} \oplus \mathsf{data}$ when $\mathsf{sk} \neq 1^\lambda$ and data when $\mathsf{sk} = 1^\lambda$. Then ciphertext is sent to the server. We omit the query algorithm of Σ as it is not used in the example.

One can easily verify that the scheme (more specifically, the **Setup** component of it) is SS-CQA-B-BB secure with respect to the following leakage:

- For all $\mathsf{data} \in \{0,1\}^\lambda$, return a random string in $\{0,1\}^\lambda$.

The real experiment and the ideal experiment only differ when $\mathsf{sk} = 1^\lambda$. This key is chosen only with probability $2^{-\lambda}$, so the advantage of the adversary distinguishing the real experiment and the ideal experiment is upper bounded by $2^{-\lambda} = \mathtt{negl}(\lambda)$.

Now back to q-leakage analysis, consider an environment that generates data distributed uniformly in $\{0,1\}^\lambda$ and generates no query. We are interested in finding out how well the scheme Σ above hides data. We use the following contrived ql-function:

$$ql(\omega, \perp, (\mathsf{data}), \perp) = 2^\lambda \cdot \mathbb{1}\{\omega = \mathsf{data}\},$$

where $\mathbb{1}$ is an indicator function. Here, if the adversary manages to guess data correctly, it's ql-function yields 2^λ, otherwise, it gets 0.

The adversary can use the following strategy to maximise its guess. For any ciphertext c it sees, it guesses the data as c. In Real^g experiment, by using this strategy, the adversary can only win when $\mathsf{sk} = 0^\lambda$ or $\mathsf{sk} = 1^\lambda$. Its expected quantitative leakage with respect to the function ql is $2^\lambda \cdot 2^{-\lambda+1} = 2$.

Now using this leakage function stated above, we compute the expected quantitative leakage in the ideal world Ideal^{ql}. Since the ciphertext is totally random, the probability that the attacker can guess the data correctly is $2^{-\lambda}$. This means the expected quantitative leakage is 1.

Comparing the expected quantitative leakages in the real world and the ideal world, we see that they differ by 1 even though the scheme is SS-CQA-B-BB secure with the leakage function. This leads us to conclude that, in general, it is incorrect to say that SS-CQA-B-BB secure implies a negligible difference in the expectations between the real world and the ideal world in our new q-leakage analysis.

When Indistinguishability Holds. We note that the ql-function in our counter-example above is not very natural. For "reasonable" ql-functions we can expect a negligible difference in the expectations between the real world and the ideal world in our new q-leakage analysis. We give a quantification of such ql-functions in the following theorem.

Theorem 1 (SS-CQA-B-BB Security implies Security in our Analysis)**.** Let $\mathcal{L}$ be a leakage profile and Σ with security parameter 1^λ and public parameter pub be SS-CQA-B-BB secure with leakage $\mathcal{L}$. Let ql be a function such that:

1. $||ql|| \in \mathtt{poly}(\lambda)$,
2. $\sup_z |ql(z)| \in \mathtt{poly}(\lambda)$,

where $||ql||$ denotes the size of the support of ql where ql is non-zero. Then there is a simulator $\mathcal{S}$, such that for every PPT adversary $\mathcal{A}$, and for every environment $\mathcal{Z}$,

$$\mathbf{E}\left[\text{Real}^{ql}_{\Sigma,\mathcal{A},\mathcal{Z}}(1^\lambda, \mathsf{pub})\right] \leq \mathbf{E}\left[\text{Ideal}^{ql}_{\Sigma,\mathcal{A},\mathcal{Z},\mathcal{S}}(1^\lambda, \mathsf{pub})\right] + \mathtt{negl}(\lambda).$$

We give a high-level overview of the steps in the proof here. The full proof can be found in below. In the first step of our proof, we show that if we have two random variables that are computationally indistinguishable, we can bound the difference between the expected quantitative leakage of the two random variables with a general formula. This immediately implies that for "reasonable" ql-functions (with conditions stated in Theorem 1), the expected q-leakage of two computationally indistinguishable random variables can only differ by a negligible amount. Then, we create an intermediate security notion SS-CQA-S (where the adversary outputs a string instead of a bit in the end) and show that SS-CQA-B-BB security implies SS-CQA-S-BB security (Theorem 3), which can be of independent interest. Combining the results yield a proof for Theorem 1.

Computationally Indistinguishable ql-Functions. Let $X = (\Omega, \mathcal{F}, P_X)$ be a probability space. Let $|X|$ to be the number of outcomes of X, i.e. $|\Omega|$. For a function $g : \Omega \to \mathbb{R}$, we write $g(X)$ to mean a real-valued random variable which maps the outcomes of X to $g(\cdot)$ of it. We abuse the notation $z \in g(X)$ to mean elements in the image of $g(X)$. We formalise the criterion on the ql-function for it to give a negligible difference in expectations for the real world and the ideal world in our security game in Theorem 2.

Theorem 2 (Bound on Expectations with respect to g)**.** Let $X = (\Omega, \mathcal{F}, P_X)$ and $Y = (\Omega, \mathcal{F}, P_Y)$ be probability spaces and $g : \Omega \to \mathbb{R}$ be a function which maps the outcomes of the events to real numbers. Then

$$\mathbf{E}[g(X)] \leq \mathbf{E}[g(Y)] + |g(X)| \cdot \sup_z |g(z)| \cdot \Delta(g(X), g(Y)),$$

where $\Delta(g(X), g(Y))$ denotes the statistical distance (or total variation distance) between $g(X)$ and $g(Y)$, i.e.
$\sup_{z \in g(X)} |\mathbf{Pr}[g(X) = z] - \mathbf{Pr}[g(Y) = z]|$.

Proof. We start by expressing the difference between the expectations.

$$\begin{aligned}
&|\mathbf{E}[g(X)] - \mathbf{E}[g(Y)]| \\
&= \sum_{z \in g(X)} z \cdot \mathbf{Pr}[g(X) = z] - \sum_{z \in g(Y)} z \cdot \mathbf{Pr}[g(Y) = z] \\
&= \sum_{z \in g(X \cup Y)} z \cdot (\mathbf{Pr}[g(X) = z] - \mathbf{Pr}[g(Y) = z]) \\
&= \sum_{z \in g(X)} z \cdot (\mathbf{Pr}[g(X) = z] - \mathbf{Pr}[g(Y) = z]) .
\end{aligned}$$

So in particular, there is a z such that

$$|z \cdot (\mathbf{Pr}[g(X) = z] - \mathbf{Pr}[g(Y) = z])| \geq \frac{|\mathbf{E}[g(X)] - \mathbf{E}[g(Y)]|}{|g(X)|}.$$

We can take a weaker bound on the right hand side by replacing z on the left by $\sup_z |g(z)|$, and we get

$$\begin{aligned}
\left|\sup_z |g(z)| \cdot (\mathbf{Pr}[g(X) = z] - \mathbf{Pr}[g(Y) = z])\right| &\geq \tfrac{|\mathbf{E}[g(X)] - \mathbf{E}[g(Y)]|}{|g(X)|} , \\
\sup_z |g(z)| \cdot |\mathbf{Pr}[g(X) = z] - \mathbf{Pr}[g(Y) = z]| &\geq \tfrac{|\mathbf{E}[g(X)] - \mathbf{E}[g(Y)]|}{|g(X)|} , \\
|\mathbf{Pr}[g(X) = z] - \mathbf{Pr}[g(Y) = z]| &\geq \tfrac{|\mathbf{E}[g(X)] - \mathbf{E}[g(Y)]|}{|g(X)| \cdot \sup_z |g(z)|} .
\end{aligned}$$

The left hand side is bounded from above by statistical distance $\Delta(g(X), g(Y))$. Finally, by rearranging the terms, we have the desired result.

$$\Delta(g(X), g(Y)) \geq \frac{|\mathbf{E}[g(X)] - \mathbf{E}[g(Y)]|}{|g(X \cup Y)| \cdot \sup_z |g(z)|} \mathbf{E}[g(X)] \leq \mathbf{E}[g(Y)] + |g(X)| \cdot \sup_z |g(z)| \cdot \Delta(g(X), g(Y)).$$

□

It is straightforward to see that if two probability spaces are computationally indistinguishable, and a function g behaves nicely on the probability spaces (specified by the conditions in the corollary), we get Corollary 1.

Corollary 1 (Bound on Computationally Indistinguishable Probability Spaces with respect to g)**.** Let 1^λ be a security parameter. Let $X = (\Omega, \mathcal{F}, P_X)$ and $Y = (\Omega, \mathcal{F}, P_Y)$ be probability spaces and $g : \Omega \to \mathbb{R}$ be a function which maps the outcomes of the events to real numbers. Assume that the following conditions hold:

1. X and Y are computationally indistinguishable,
2. $|\{\omega \mid g(\omega) \neq 0, \omega \in \Omega\}| \in \mathtt{poly}(\lambda)$,
3. $\sup_z |g(z)| \in \mathtt{poly}(\lambda)$.

Then

$$\mathbf{E}[g(X)] \leq \mathbf{E}[g(Y)] + \mathtt{negl}(\lambda).$$

```
$\text{Real}^{\text{SS-CQA-S}}_{\Sigma,\mathcal{A}}(1^\lambda, \mathsf{pub})$
1: $\mathsf{data} \leftarrow \mathcal{A}(1^\lambda, \mathsf{pub})$
2: $\mathsf{sk} \leftarrow \mathbf{Gen}_{\mathbf{Clt}}(1^\lambda, \mathsf{pub})$
3: $(\perp, \mathsf{edata}) \leftarrow [\mathbf{Setup}_{\mathbf{Clt}}(\mathsf{sk}, \mathsf{pub}, \mathsf{data}), \mathbf{Setup}_{\mathcal{A}}()]$
4: $i \leftarrow 1$
5: while $\mathsf{q}_i \leftarrow \mathcal{A}()$ do
6:     $(\mathsf{rspn}_i, \mathsf{edata}) \leftarrow \big[\mathbf{EQuery}_{\mathbf{Clt}}(\mathsf{sk}, \mathsf{pub}, \mathsf{q}_i),\ \mathbf{EQuery}_{\mathbf{Svr}}(\mathsf{edata})\big]$
7:     $i \leftarrow i + 1$
8: $\omega \leftarrow \mathcal{A}()$
9: return $\omega$
```

```
$\text{Ideal}^{SS-CQA-S}_{\Sigma,\mathcal{A},\mathcal{S}}(1^\lambda, \mathsf{pub})$
1: $\mathsf{data} \leftarrow \mathcal{A}(1^\lambda, \mathsf{pub})$
2: $(\perp, \mathsf{edata}) \leftarrow \big[\mathcal{S}(1^\lambda, \mathsf{pub}, \mathcal{L}_{\mathbf{Setup}}(\mathsf{pub}, \mathsf{data})), \mathbf{Setup}_{\mathcal{A}}()\big]$
3: $i \leftarrow 1$
4: while $\mathsf{q}_i \leftarrow \mathcal{A}()$ do
5:     $(\mathsf{rspn}_i, \mathsf{edata}) \leftarrow \big[\mathcal{S}(1^\lambda, \mathsf{pub}, \mathcal{L}_{\mathbf{EQuery}}(\mathsf{pub}, (\mathsf{q}_j)_{j=1}^{i},\ \mathsf{data}), \mathbf{EQuery}_{\mathcal{A}}(\mathsf{edata})\big]$
6:     $i \leftarrow i + 1$
7: $\omega \leftarrow \mathcal{A}()$
8: return $\omega$
```

Figure 4: Games for SS-CQA-S.

Just as the standard notion we showed in Section 2, q-leakage analysis we described in Section 5.1 is quantified with a leakage function. In this section, we show that one can use the same leakage function for both notions if certain conditions are met. This helps with security proofs as it allows one to first quantify the leakage in the standard notion, and then calculate the expected q-leakage in the q-leakage analysis.

In more detail, we first propose an intermediate notion (Definition 3) that is the same as the standard notion except that the adversary returns a string as opposed to a bit in the real and ideal games. We then show that security in the standard notion (i.e. computational indistinguishability between the real and ideal games) implies security in this intermediate notion in Theorem 3. Finally, we show that if a scheme is secure in the intermediate notion with leakage $\mathcal{L}$, then the expected q-leakage in the real game of q-leakage analysis can be bounded by the expected q-leakage in the ideal game of q-leakage analysis with the same leakage $\mathcal{L}$.

We begin by stating the intermediate notions in Definition 3 below. One is in the black-box model and the other one is in the non-black-box model. The difference is the order of quantifiers.

Definition 3 (Modified Adaptive Security of Interactive STE (SS-CQA-S))**.** Let 1^λ be a security parameter and pub be a public parameter. Consider the probabilistic games in Figure 4 for structured encryption scheme Σ where $\mathcal{A}$ is a stateful semi-honest adversary which outputs a string in the end, $\mathcal{S}$ is a stateful simulator, **Clt** is the client played by the real game, **Svr** is the server played by the adversary, $\mathcal{L} = (\mathcal{L}_{\mathbf{Setup}}, \mathcal{L}_{\mathbf{Query}})$ is the leakage profile:

We say that scheme Σ with security parameter 1^λ and public parameter pub is adaptive SS-CQA-S-BB secure (semantic secure against chosen query attack with string output in the *black-box* model) with leakage $\mathcal{L}$ if there is a simulator $\mathcal{S}$ such that for every PPT adversaries $\mathcal{A}$ and distinguishers $\mathcal{D}$,

$$\left|\mathbf{Pr}\left[\mathcal{D}(\text{Real}^{\text{SS-CQA-S}}_{\Sigma,\mathcal{A}}(1^\lambda, \mathsf{pub})) = 1\right] - \mathbf{Pr}\left[\mathcal{D}(\text{Ideal}^{\text{SS-CQA-S}}_{\Sigma,\mathcal{A},\mathcal{S}}(1^\lambda, \mathsf{pub})) = 1\right]\right| \leq \mathtt{negl}(\lambda).$$

We say that scheme Σ with security parameter 1^λ and public parameter pub is adaptive SS-CQA-S-NBB secure (semantic secure against chosen query attack with string output in the *non-black-box* model) with leakage $\mathcal{L}$ if for every PPT adversaries $\mathcal{A}$ and distinguishers $\mathcal{D}$, there is a simulator $\mathcal{S}$ such that,

$$\left|\mathbf{Pr}\left[\mathcal{D}(\mathrm{Real}^{\text{SS-CQA-S}}_{\Sigma,\mathcal{A}}(1^\lambda,\mathsf{pub}))=1\right]-\mathbf{Pr}\left[\mathcal{D}(\mathrm{Ideal}^{\text{SS-CQA-S}}_{\Sigma,\mathcal{A},\mathcal{S}}(1^\lambda,\mathsf{pub}))=1\right]\right|\le \mathtt{negl}(\lambda).$$

The implications between the four notions are summarised in Theorem 3.

Theorem 3 (Implications between Different Notions.)**.** The implications between SS-CQA-B-BB, SS-CQA-B-NBB, SS-CQA-S-BB and SS-CQA-S-NBB are represented by Figure 5.

$$\begin{tikzcd}
\text{SS-CQA-B-BB} \arrow[r, shift left] \arrow[d] & \text{SS-CQA-S-BB} \arrow[l, shift left] \arrow[d] \\
\text{SS-CQA-B-NBB} & \text{SS-CQA-S-NBB} \arrow[l]
\end{tikzcd}$$

Figure 5: Implications between the notions.

Proof. The implications SS-CQA-B-BB $\Rightarrow$ SS-CQA-B-NBB and SS-CQA-S-BB $\Rightarrow$ SS-CQA-S-NBB are trivial.

SS-CQA-S-NBB $\Rightarrow$ SS-CQA-B-NBB. Suppose that the underlying scheme Σ is not secure in SS-CQA-B-NBB, then there is an adversary $\mathcal{A}$ for all simulators $\mathcal{S}$, the output of the real and ideal experiments differs with non-negligible probability. We construct an adversary against SS-CQA-S-NBB as follows. At the beginning of the experiment, the adversary outputs data data_0 and a sequence of queries $(\mathsf{q}_j)_{j=1}^l$ for some l. We run the query algorithm on the data an to obtain a bit b and set the output of the SS-CQA-S-NBB experiment to be $(b,(\mathsf{data}_j)_{j=1}^l,(\mathsf{q}_j)_{j=1}^l)$. Since the probability of obtaining b in the real world and the ideal world is different in SS-CQA-B-NBB, there must be a distinguisher who can distinguish $(b,(\mathsf{data}_j)_{j=1}^l,(\mathsf{q}_j)_{j=1}^l)$ from the real world and the ideal world, hence, scheme Σ must be insecure in SS-CQA-S-NBB.

SS-CQA-S-BB $\Rightarrow$ SS-CQA-B-BB. The proof is similar to SS-CQA-S-NBB $\Rightarrow$ SS-CQA-B-NBB so we do not repeat it here.

SS-CQA-B-BB $\Rightarrow$ SS-CQA-S-BB. For any simulator $\mathcal{S}$ in SS-CQA-B-BB, we construct an adversary that generates a non-negligible difference in probability using SS-CQA-S-BB adversaries as follows. Let $\mathcal{A}$ and $\mathcal{D}$ be some adversary and distinguisher that breaks scheme Σ in SS-CQA-S-BB for the simulator $\mathcal{S}$. The adversary against SS-CQA-B-BB works as follows. It uses adversary $\mathcal{A}$ to generate data data_0 and a sequence of queries $(\mathsf{q}_j)_{j=0}^l$. The queries are executed and a transcript $\mathcal{T}$ is kept. The transcript $\mathcal{T}$ is given to the adversary $\mathcal{A}$ to generate a string ω. This string is passed into distinguisher $\mathcal{D}$ and a bit b is produced. Since we assumed that SS-CQA-S-BB is insecure, the probability of getting b in the real world and ideal world is different, and scheme Σ is insecure in SS-CQA-B-BB. $\square$

Proof of Theorem 1. It suffices to show that the two games in q-leakage analysis are computationally indistinguishable if the underlying scheme is secure in the SS-CQA-B-BB model. Using Figure 5, we know that security in the SS-CQA-B-BB model implies security in the SS-CQA-S-BB model. Now, we modify the games in the SS-CQA-S-BB model by replacing the first line of the real and ideal worlds with that of q-leakage analysis, i.e. we let the environment sample the database, queries and auxiliary information:

$$(\mathsf{DB}_0,(\mathsf{q}_1,\ldots,\mathsf{q}_N),\mathsf{aux})\leftarrow \$\mathcal{Z}.$$

It is straightforward to see that security in the SS-CQA-S-BB model implies security in this modified model. The only difference between this model and q-leakage analysis is that the ql-function has not been applied to the output of the game yet. It can be easily shown that the two games in q-leakage analysis are indistinguishable if the modified model is secure with a simple reduction.

We can then use Corollary 1 to show the desired result. $\square$

REMARKS. We first remark on the order of the quantifiers for $\mathcal{S}$ and $\mathcal{A}$ in Theorem 1. As mentioned before, most of the constructions in the literature use SS-CQA-B-NBB as their security notion but the actual proofs are for SS-CQA-B-BB. In addition, we were only able to show the implication between SS-CQA-B-BB with our notion (see Figure 5). So it only makes sense to put $\mathcal{S}$ before $\mathcal{A}$.

To the best of our knowledge, most of the constructions in the literature [Curtmola et al.(2006), Kamara et al.(2012), Cash et al.(2013), Stefanov et al.(2014), Naveed et al.(2014), Bost(2016), Chamani et al.(2018), Demertzis et al.(2020), Bag et al.(2023)] use SS-CQA-B-NBB as their security notion but the actual proofs are for SS-CQA-B-BB. Hence, the leakage functions proved for those schemes can be used in our security notion directly.

We also note that the function ql from our counter-example does not satisfy the second condition of Theorem 1 since $\sup_z |ql(z)| = 2^\lambda \notin \texttt{poly}(\lambda)$.

Comparison with the QIF Framework. The quantitative information flow framework (QIF) [Alvim et al.(2012), Smith(2015)] considers a *gain* function g that takes the attacker's guess w and secret input x and output a number measuring how well the adversary guessed the information about x. Our functions are very similar in this regard. But the further treatment is different.

QIF models all possible (secret) inputs and (observable) outputs of the system as an information-theoretic channel matrix C that gives the conditional probabilities of outputs given the inputs. The information-theoretic essence of a channel matrix is a mapping from priors π to distributions on posterior distributions.

And then the (additive) g-leakage is computed as

$$\sum_y \max_w \sum_x \pi_x C_{x,y} g(w,x) - \max_w \sum_x \pi_x g(w,x) \ .$$

The QIF techniques apply well to property-preserving encryption (PPE) schemes such as deterministic or order-preserving encryption, as shown in [Jurado and Smith(2019), Jurado et al.(2021)]. This is because the PPE security definitions specify the ideal objects and those are feasible to model as a channel. Also, PPE leakage is independent of the queries.

The work [Boreale and Pampaloni(2014)] extended quantitative information flow to support generic leakage functions and adaptive adversaries. However, their results still do not apply to dynamic stateful schemes.

Our framework is not constrained to information channels and hence it provides more flexibility (e.g. leakage in SEAL is stateful and is hard to model as an information channel). Also, our framework includes a formal security model for searchable encryption. The security model is important to show how to model data, queries, and auxiliary information distributions. Moreover, our framework guarantees that the expected quantitative leakage computed in the ideal world can be transferred to the real world if the ql-function behaves nicely (Theorem 1). This is not investigated in [Alvim et al.(2012), Smith(2015)]. Finally, our framework focuses on the amount of damage an adversary can do to a system (i.e., posterior vulnerability which corresponds to the first term of the g-leakage formula) as opposed to g-leakage as we value the absolute security of the system more.

5.2 Choosing the ql-Function

The choice of the function ql is crucial in our q-leakage analysis since ql is used to capture classes of attacks. In this section, we provide a discussion on how the ql-functions can be chosen.

Capture the Goal of the Attacker. The ql-function should be picked to capture concrete classes of attacks. (As we discussed in the Introduction, it is impossible to meaningfully quantify leakage's damage wrt all possible attacks.) For example, we can capture query reconstruction attacks by awarding the adversary for every correctly guessed query. Formally, this function ql can be written as:

$$ql((\mathsf{w}_j)_{j=1}^l, \mathsf{pub}, (\mathsf{data}_j)_{j=0}^l, (\mathsf{q}_j)_{j=1}^l) = \frac{\sum_{j=1}^l \mathbb{1}\left(\mathsf{w}_j = \mathsf{q}_i\right)}{l},$$

where w_j is the plaintext the adversary guessed for query q_j. The ql-function returns the fraction of the correct guesses. In fact, the function ql match the query recovery rate in many attacks in the literature [Islam et al.(2012), Naveed et al.(2015), Zhang et al.(2016), Blackstone et al.(2020), Oya and Kerschbaum(2021), Oya and Kerschbaum(2022)].

Fine-grained quantitative Lakage. As mentioned above, some parts of the data and queries may be easier to attack than the other parts. For example, for a database containing medical records, there are significantly more queries on "flu" than on "cancer", so by pure guessing, an attacker can recover queries on "flu" more easily. However, the attacker may not be interested in queries on "flu" since it is a common disease.

The *ql*-function above cannot be used to encourage the attacker to make more guesses on "cancer". However, it can be easily modified to achieve that. Consider the following *ql*-function ql':

$$ql'((\mathsf{w}_j)_{j=1}^l, \mathsf{pub}, (\mathsf{data}_j)_{j=0}^l, (\mathsf{q}_j)_{j=1}^l) \frac{\sum_{j=1}^l \mathbb{1}(\mathsf{w}_j = \mathsf{q}_i) + \mathbb{1}(\mathsf{w}_j = \mathsf{q}_i, \mathsf{w}_j = \text{cancer})}{l} .$$

Here, we modified the function ql by adding an additional term $\mathbb{1}(\mathsf{w}_j = \mathsf{q}_i, \mathsf{w}_j = \text{cancer})$. The net effect of this is that whenever the attacker makes a correct guess and the guess is "cancer", it receives twice amount of gain than before. As a result, the attacker is more willing to make guesses on "cancer"since it is easier to maximize the expected quantitative leakage this way.

The function ql' can also be interpreted differently. Instead of considering the problem from the perspective of the attacker, we can consider it from the perspective of a defender. The goal of the defender is to minimize the gain an attacker can have on a database. So the gain on a query (when the attacker makes a correct guess) can be seen as the privacy loss of the query. Using ql' as an example, the *ql*-function suggests that the privacy loss of "cancer" is twice as that of "flu".

In general, the *ql*-function should be picked in a way to reflect real-world privacy concerns. The *ql*-function should reflect which parts of the data and queries are the adversary (or the defender) mostly interested in, so that the expected quantitative leakage computed in *q*-leakage analysis can accurately reflect the impact of attacks. We note that the complexity of the *q*-leakage analysis may depend on the a particular choice of the *ql*-function. For certain schemes and functions such analysis may be quite complex.

5.3 Interpreting the Expected Quantitative Leakage

With *q*-leakage analysis, whether a scheme is secure is no longer binary (i.e. a scheme can either be secure or insecure). Rather, security becomes relative. Different *ql*-functions can have very different semantic meanings, e.g., the expected loss of a company in cash and the success rate of query recovery attacks. We believe that it is hard to normalize different *ql*-functions into a unit interval. In this section, we discuss how one could still interpret the output of *q*-leakage analysis, i.e. the supremum of the expected quantitative leakage, or expected *q*-leakage for short.

Measure the Impact of an Attack. For *ql*-functions that directly model recovery rate of attacks with a certain goal (e.g. *ql* in Section 5.2 which models query recovery attacks), the expected *q*-leakage can be interpreted straightforwardly as an upper bound on the expected recovery rate of the attacks.In practice, practitioners who try to use structured encryption scheme to protect their data can use *q*-leakage analysis to compute the expected *q*-leakages for different schemes and pick the one with the smallest expected *q*-leakage (if the expected *q*-leakage is reasonably small). While there is no general guideline on how small the expected *q*-leakage needs to be for a scheme to be considered secure, we believe that using *q*-leakage analysis to find the expected *q*-leakage of different schemes is the first step in coming up with a consensus on the expected *q*-leakages for different classes of attacks.

Understand Fine-grained Security. Instead of using *q*-leakage analysis to capture recovery attacks directly, one can use a function such as ql' in Section 5.2 for a more fine-grained measure of security. In those cases, the *ql*-function should be chosen so that the expected *q*-leakage has a well-understood meaning. Then, the user can analyze the expected *q*-leakage of different schemes with their data and pick the scheme with the smallest expected *q*-leakage as before.

Comparing the Expected Quantitative Leakage to the Baseline. The expected quantitative leakage may be hard to interpret for some schemes and *ql*-functions. For that reason, we propose to compare the expected quantitative leakage to a baseline one, computed on a scheme with no leakage, i.e. $\mathcal{L}_{\mathbf{Setup}} = \perp$

and $\mathcal{L}_{\mathbf{EQuery}} = \perp$. Then, the difference in the expected q-leakages between the baseline and the scheme tells us how much security the scheme lost due to its leakage. More formally, we define the baseline leakage as $\sup_{\mathcal{A}} \mathbf{E}\left[\mathrm{Real}^{ql}_{\perp,\mathcal{A},\mathcal{Z}}(1^\lambda, \mathsf{pub})\right]$, where we abuse $\perp$ to mean a scheme with no leakage. We give concrete examples of baselines in Sections 6 and 7.

Efficiency and Security Trade-off. Some structured encryption schemes have parameters that affect their efficiency and security. It is generally well-understood how there parameters affect the efficiency of the schemes, but it is less clear how they affect the security. q-leakage analysis can be used to quantify security, and thus, allow users to make informed decisions on their parameter choices. As an example, we show in Section 7 how the parameters in Demertzis et al. [Demertzis et al.(2020)] affect security. This allows a user to pick parameters that achieve a specific target expected quantitative leakage, or help the user understand the security risks if they pick the parameters for certain efficiency goals.

6 Case Study: DP VHEMM Analysis

6.1 DP VHEMM Description and Prior Results

Overview of DP VHEMM. An encrypted multi-map is an encrypted collection of key-value pairs where the client can query values with multi-map keys. In this section, we study the security of differential-private volume-hiding encrypted multi-map (DP VHEMM) proposed by Patel et al. [Patel et al.(2019)]. We give a high-level description of DP VHEMM in this section and provide a more detailed description in Appendix A.

DP VHEMM is a static encrypted multi-map based on Cuckoo hashing. The main insight of the construction is that it is possible to encrypt and store the key-value pairs of a multi-map in the hash tables (two of them to be specific) used in Cuckoo hashing. Then, when the client wants to query for the keys later, it can hide the true query response volumes of the keys (i.e. the size of the value corresponding to the keys) by querying additional entries of the hash tables on top of the entries that it wants to retrieve. The authors argued that padding all query response volumes to the maximum query response volume (by querying enough dummy entries of the hash tables so that the total number of entries accessed is equal to the maximum size of the values) is expensive, so they proposed a differentially private way of hiding the query response volumes by adding Laplace noises to the true query response volumes (achieved by querying additional dummy entries of the hash tables).

Remark. Since DP VHEMM relies on Laplace noises with infinite support sizes, any true query response volume $\ell(\mathtt{key})$ for multi-map key $\mathtt{key}$ can produce the observed query response volume $\ell^*(\mathtt{key})$. This means that given the leakage of a particular database and a sequence of queries protected by DP VHEMM, the number of multi-maps that can produce the leakage is infinite in size. A direct consequence of this is that the reconstruction space studied in [Kornaropoulos et al.(2022)] will be infinite in size as well and it is unclear how the size of the reconstruction space helps with understanding the impact of the leakage of DP VHEMM.

Known Security of DP VHEMM. The authors of [Patel et al.(2019)] proved that DP VHEMM satisfies the differential privacy property they defined. The notion is similar to the original differential privacy definition [Dwork et al.(2006)]. Briefly, the adversary should not be able to distinguish between two encrypted multi-maps when the multi-maps differ only in two key-value pairs (i.e. the two multi-maps are the same except that for two of the keys, there is one more/less value associated to these keys). However, differential privacy does not say anything about query privacy and the data privacy implications are quite limited. In particular, differential privacy does not constrain the leakage of the encrypted queries, so query reconstruction attacks are not ruled out. In addition, differential privacy only guarantees the privacy of neighbouring multi-maps that only differ in two key-value pairs. The security bounds for multi-maps that differ significantly are too loose to be meaningful.

The authors of [Patel et al.(2019)] did not state or prove SS-CQA-B-(N)BB security of their scheme. We provide the analysis in Appendix B for completeness.

Despite the above security results, Oya and Kerschbaum [Oya and Kerschbaum(2021)] presented a query reconstruction attack against DP VHEMM. Their attack does not contradict the above two security statements. As we mentioned, differential privacy does not capture query privacy. SS-CQA-B notion does, but besides leakage. And here the issue is that leakage permits query reconstruction attacks.

Reproducing the results shown by Oya and Kerschbaum with q-leakage analysis would not be particularly useful. Instead, we show how q-leakage analysis can be used to perform a more fine-grained security analysis. The attack in [Oya and Kerschbaum(2021)] uses the fact that the query response volumes for the most frequent multi-map keys (for the dataset they have used) are far apart. However, not all multi-maps can be exploited this way. In this section, we show that for multi-maps with "similar" query response volumes (we will formally define what we mean by "similar" later), DP VHEMM do offer a certain level of query privacy.

6.2 Analysis of DP VHEMM Leakage

In this section, we show how our notion can be used to analyze the leakage of DP VHEMM in certain cases (for specific environments) and for particular security goals (for certain ql-functions). We begin by providing a general description of the environments we consider. We then refine the environment and consider two specific distributions of multi-map. We show that for some ql-functions, DP VHEMM is secure with respect to these environments.

Rewarding Multi-map Keys with Large Query Response Volumes. The attack by Oya and Kerschbaum [Oya and Kerschbaum(2021)] uses the fact that the query response volumes for the most frequent multi-map keys (for the dataset they have used) are far apart. However, not all multi-maps can be exploited this way. As an example, consider a multi-map where the keys are student ids and the values are the genders of the students. For this multi-map, the query response volumes will be 1 for every query since the multi-map keys are unique. We show that for multi-maps with "similar" query response volumes (we will formally define what we mean by "similar" later), DP VHEMM does offer a certain level of query privacy.

We use the following ql-function ql_1:

$$ql_1((i, \mathtt{key}_k), \mathsf{pub}, (\mathsf{MM}_j)_{j=0}^{l}, (\mathsf{q}_j)_{j=1}^{l}) = \mathbb{1}\{\mathtt{key}_k = \mathsf{q}_i\} \cdot \frac{k}{N},$$

where the guess is an index i and a key $\mathtt{key}_k$ ($\mathtt{key}_k$ is the k-th key in terms of query response volume, from the smallest to the largest) and N is the total number of keys in the multi-map. The ql-function outputs $\frac{k}{N}$ when the guessed key $\mathtt{key}_k$ matches with the i-th query q_i. In other words, the ql-function rewards guesses on the keys with large query response volumes more than the keys with small query response volumes.

We consider multi-maps with "similar" query response volumes. Let $\mathtt{key}_1, \ldots, \mathtt{key}_N$ be the keys of the multi-map such that $\mu_i \le \mu_j$ for all $i < j$ (recall that μ_i is the location parameter for the query response volume of $\mathtt{key}_i$). Let $\delta \in \mathbb{N}$ be a natural number such that for all i, $\mu_{i+\delta} - \mu_i < b$. That is, for every key $\mathtt{key}$, there are at least δ other keys that have similar location parameters (up to a difference of b) to it. The environment will generate a multimap with randomly chosen keys and values, and which satisfies the above conditions on the location parameter for the query response volumes. To simplify the analysis we consider the true query response volumes following the Laplace distribution. The environment also generates queries with an arbitrary distribution. The parameters describing the multi-map are given to the adversary as the auxiliary information. In more practical terms, this environment can be used to model a medical database where diseases (as keys of a multi-map) have a similar number of patients (patients as multi-map values).

For such an environment and the ql-function, the following theorem shows that DP VHEMM achieves a certain level of security. The proof is in Appendix B.

Theorem 4. Let 1^λ be a security parameter and $\mathsf{pub} =$ $(\alpha, \varepsilon, \lambda_{DP}, c(\cdot))$ be a public parameter. Let $N \in \mathbb{N}$ be the number of multi-map keys, $\delta \in \mathbb{N}$ be a natural number, $b \in \mathbb{R}^+, b \le 2/\varepsilon$ band $d \in \mathbb{N}$ be a natural number. Suppose $N \ge \delta \cdot \exp\left(\frac{-d \cdot \varepsilon}{2}\right)$ and $s < d$.

Let $\mathcal{Z}_1^{N,\delta,s,d}$ be the following environment:

1. It picks $\mu_1, \ldots, \mu_N \in \mathbb{N}$ such that $\mu_i \le \mu_j$ for all $i < j$ and for all i, $\mu_{i+\delta} - \mu_i < d$.
2. It generates a multi-map $\mathsf{MM} = \left\{(\mathtt{key}_i, (v_{i,j})_{j=1}^{l_i})\right\}_{i=1}^{N}$ such that:
   - $\mathtt{key}_i$ is picked randomly from $\{0,1\}^*$ (with an arbitrary distribution picked by the environment $\mathcal{Z}_1$) under the constraint that $\mathtt{key}_i \ne \mathtt{key}_j$ for all $i \ne j$.

- l_i are distributed as $\lfloor \mathbf{LapD}(\mu_i, s) \rfloor$.
- The values $v_{i,j}$ are randomly picked from $\{0,1\}^*$ under the constraint that all $v_{i,j}$ for a fixed i are unique. Repeated values are resampled until there are no more collisions.

3. It generates polynomially many queries $\mathsf{q}_1, \ldots, \mathsf{q}_l$ with an arbitrary query distribution chosen by the environment $\mathcal{Z}_1$.
4. It sets the auxiliary information as N, $\{(\mathtt{key}_i, \mu_i)\}_{i=1}^N$, δ, s, d.

Consider ql-function ql_1:

$$ql_1((i, \mathtt{key}_k), \mathsf{pub}, (\mathsf{MM}_j)_{j=0}^l, (\mathsf{q}_j)_{j=1}^l) = \mathbb{1}\{\mathtt{key}_k = \mathsf{q}_i\} \cdot \frac{k}{N}.$$

We get

$$\sup_{\mathcal{A}} \mathbf{E}\left[\mathrm{Real}^{ql_1}_{\text{DP VHEMM},\mathcal{A},\mathcal{Z}_1^{N,\delta,s,d}}(1^\lambda, \mathsf{pub})\right] \leq \frac{1}{\delta} \cdot \exp\left(\frac{d \cdot \varepsilon}{2}\right).$$

Proof. As $\mu_{i+\delta} - \mu_i < b$ for some δ, we can bound the posterior probability for identifying multi-map key $\mathtt{key}_i$ from above using Theorem 10. For simplicity, we assume $i + \delta$ is smaller or equal to the number of multi-map keys.

$$\begin{aligned}
&\mathbf{Pr}[\mathtt{key}_i \mid \ell^*] \\
\leq &\frac{1}{\sum_j \exp\left(-\frac{|\mu_i - \mu_j|}{2/\varepsilon}\right)} \\
\leq &\frac{1}{\sum_{j=i+1}^{i+\delta} \exp\left(-\frac{|\mu_i - \mu_j|}{2/\varepsilon}\right)} \\
\leq &\frac{1}{\sum_{j=i+1}^{i+\delta} \exp\left(-\frac{d \cdot \varepsilon}{2}\right)} \\
= &\frac{1}{\delta \exp\left(-\frac{d \cdot \varepsilon}{2}\right)} \\
= &\frac{1}{\delta} \cdot \exp\left(\frac{d \cdot \varepsilon}{2}\right) .
\end{aligned}$$

The multi-map keys $\mathtt{key}_i$ for i that have not been covered, i.e. $i + \delta$ is greater than the number of multi-map keys, have the same bound on the posterior probability by symmetry.

We can now consider the bound on the expected q-leakage in our security notion. As we have just shown above, no adversary can guess guess a multi-map key more than $\frac{e}{\delta}$ of the times. So a natural bound on the expected q-leakage can be derived as:

$$\begin{aligned}
&\mathbf{E}\left[\mathrm{Real}^{ql_1}_{\text{DP-VH EMM},\mathcal{A},\mathcal{Z}_1}(1^\lambda)\right] \\
\leq &\frac{1}{\delta} \cdot \exp\left(\frac{d \cdot \varepsilon}{2}\right) \cdot \max ql_1 \\
\leq &\frac{1}{\delta} \cdot \exp\left(\frac{d \cdot \varepsilon}{2}\right).
\end{aligned}$$

□

As a numerical example, let $N = 100$, $\delta = 20$, $s = 5$, and $d = 10$. This corresponds to a multi-map where there are 100 multi-map keys. For each multi-map key, there are at least 20 other multi-map keys with similar frequencies (the location parameter of the random variables that determines their frequencies are at most 10 apart). For $\varepsilon = 0.2$, Theorem 4 suggests that the expected q-leakage is upper bounded by 0.136.

In addition, the baseline expected q-leakage

$$\sup_{\mathcal{A}} \mathbf{E}\left[\text{Real}^{ql_1}_{\perp,\mathcal{A},\mathcal{Z}_1^{N,\delta,s,d}}(1^\lambda, \mathsf{pub})\right]$$

is upper bounded by $\frac{1}{N}$ where N is the number of multi-map keys. This is because the query distribution is not given to the adversary and no other auxiliary information allows it to make a guess that is better than a random guess. For the numerical example above, this means that the baseline gain is 0.01. While 0.136 expected q-leakage is much larger than the 0.01 baseline, the expected q-leakage is significantly smaller than the worst-case of 1. So the scheme should be considered reasonably secure for $\mathcal{Z}_1^{N,\delta,s,d}$ and ql_1 if the 0.136 expected q-leakage is acceptable.

Rewarding Multi-map Keys with Small Query Response Volumes. For some multi-maps, it may be okay to leak the identity of the multi-map keys with large query response volumes. For example, in a multi-map that holds medical records, it is not very impactful if an attacker manages to recover a query on common flue. On the other hand, it is a lot more detrimental if an attacker can recover a query on a cancer.

To capture the scenario described above, we need a ql-function that rewards correct guesses on low-frequent multi-map keys. We consider the following ql-function ql_2:

$$ql_2((i, \texttt{key}'_k), \mathsf{pub}, (\mathsf{MM}_j)_{j=0}^l, (\mathsf{q}_j)_{j=1}^l) = \mathbb{1}\left\{\texttt{key}'_k = \mathsf{q}_i\right\} \cdot \left(\frac{N-k}{N}\right)^3 .$$

In short, the function ql_2 outputs $\left(\frac{M-k}{M}\right)^3$ whenever the guess $\texttt{key}'_k$ matches the i-th query q_i. It is easy to see that the ql-function is inversely proportional to the rank of the keyword (the rank is the order of the keyword in terms of their true query response volumes; the keywords are ordered in increasing true query response volumes). We picked this ql-function as we can derive an analytical bound with it but other ql-functions are possible as well.

We consider an environment $\mathcal{Z}_2^{N,t,\delta,s}$. Let MM be a multi-map generated by environment $\mathcal{Z}_2^{N,t,\delta,s}$ with keys $\texttt{key}_1, \cdot, \texttt{key}_N$. As before, the keys satisfy $\mu_i \leq \mu_j$ for all $i < j$. In addition, there exists $k < N$ and $\delta \in \mathbb{R}^+$ such that for all $i, j < k$, $|\mu_i - \mu_j| < \delta \cdot s$. In other words, all keys from $\texttt{key}_1$ to $\texttt{key}_{k-1}$ have similar location parameters (and hence, similar query response volumes).

We can bound the security of DP VHEMM with respect to the environment $\mathcal{Z}_2^{N,t,\delta,s}$ and function ql_2 as follows:

Theorem 5. Let 1^λ be a security parameter and $\mathsf{pub} = (\alpha, \varepsilon, \lambda_{DP}, c(\cdot))$ be a public parameter. Let $N \in \mathbb{N}$ be the number of multi-map keys, $t < N$ be a natural number, $\delta \in \mathbb{R}^+$ be a positive real number, and $b \in \mathbb{R}^+, b \leq 2/\varepsilon$ be a positive real number.

Let $\mathcal{Z}_2^{N,t,\delta,s}$ be the following environment:

1. It picks $\mu_1, \ldots, \mu_N \in \mathbb{N}$ such that $\mu_i \leq \mu_j$ for all $i < j$ and $|\mu_i - \mu_j| < \delta \cdot s$ for all $i, j < t$.

2. It generates a multi-map $\mathsf{MM} = \left\{(\texttt{key}_i, (v_{i,j})_{j=1}^{l_i})\right\}_{i=1}^N$ such that:

   - $\texttt{key}_i$ is picked randomly from $\{0,1\}^*$ (with an arbitrary distribution picked by the environment $\mathcal{Z}_2$) under the constraint that $\texttt{key}_i \neq \texttt{key}_j$ for all $i \neq j$.
   - l_i are distributed as $\lfloor \mathbf{LapD}(\mu_i, s) \rfloor$.
   - The values $v_{i,j}$ are randomly picked from $\{0,1\}^*$ under the constraint that all $v_{i,j}$ for a fixed i are unique. Repeated values are resampled until there are no more collisions.

3. It generates polynomially many queries $\mathsf{q}_1, \ldots, \mathsf{q}_l$ with an arbitrary query distribution chosen by the environment $\mathcal{Z}_2$.

4. It sets the auxiliary information as N, $\{(\texttt{key}_i, \mu_i)\}_{i=1}^N$, t, δ, s.

Consider function ql_2:

$$ql_2((i, \texttt{key}_k), \mathsf{pub}, (\mathsf{MM}_j)_{j=0}^{l}, (\mathsf{q}_j)_{j=1}^{l}) = \mathbb{1}\{\texttt{key}_k = \mathsf{q}_i\} \cdot \left(\frac{N-k}{N}\right)^3 .$$

We have

$$\sup_{\mathcal{A}} \mathbf{E}\left[\mathrm{Real}^{ql_2}_{\text{DP VHEMM},\mathcal{A},\mathcal{Z}_2^{N,t,\delta,s}}(1^\lambda, \mathsf{pub})\right] \leq \max\left\{\left(\frac{N-t}{N}\right)^3, \frac{\exp(\delta)}{N}\right\}.$$

Proof. The expected q-leakage can be computed in two parts. The first part is for the high-frequent multi-map keys, i.e. $\texttt{key}_i$ for $i > t$. We can trivially bound as

$$\left(\frac{N-t}{N}\right)^3 .$$

The second part is for the low-frequent multi-map keys, i.e. $\texttt{key}_i$ for $i \leq t$. From Theorem 10 and use the constraint $|\mu_i - \mu_j| < 2\delta/\varepsilon$ for all $i, j < t$, we know that the probability of guessing the multi-map key $\texttt{key}_i$ for $i \leq t$ correctly is upper bounded by

$$\begin{aligned} &\frac{1}{\sum_{j=1}^{N} \exp\left(-\frac{|\mu_i - \mu_j|}{2/\varepsilon}\right)} \\ \leq &\frac{1}{\sum_{j=1}^{N} \exp\left(-\frac{2\delta/\varepsilon}{2/\varepsilon}\right)} \\ = &\frac{\exp(\delta)}{N}. \end{aligned}$$

Since the maximum gain for guess $\texttt{key}_i$ for $i \leq t$ correctly is 1, the expected q-leakage for the low-frequent multi-map keys is upper bounded by $\frac{\exp(\delta)}{N}$.

Combining the bound on the high-frequent and low-frequent multi-map keys together, we get

$$\begin{aligned} &\sup_{\mathcal{A}} \mathbf{E}\left[\mathrm{Real}^{ql_2}_{\text{DP-VH EMM},\mathcal{A},\mathcal{Z}_2}(1^\lambda, \mathsf{pub})\right] \\ &\leq \max\left\{\left(\frac{N-t}{N}\right)^3, \frac{\exp(\delta)}{N}\right\}. \end{aligned}$$

□

As a numerical example, consider parameters $N = 1000, t = 900$, $\epsilon = 0.2$, and $\delta = 5$. This corresponds to a multi-map with 1000 multi-map keys, of which 900 of them have query response volumes, and those 900 multi-map keys have true query response volumes that differ by at most 50. The expected q-leakage with these parameters is about 0.148.

In contrast, the baseline q-leakage with ql_2 is upper bounded by $\frac{1}{N} \cdot \sum_{i=1}^{N} \frac{i^3}{N^4} = \frac{(N+1)^2}{4N^3}$. This is because the best possible strategy for the adversary without any leakage is to guess randomly. It can succeed at most $\frac{1}{N}$ times and the expected q-leakage when it guess correctly is $\sum_{i=1}^{N} \frac{i^3}{N^4}$. Numerically, for $N = 1000$, the baseline q-leakage will be approximately 2.5×10^{-4}. This is significantly smaller than the expected q-leakage for DP VHEMM. However, this is to be expected because DP VHEMM is only designed to hide neighbouring databases (and as a result, queries with similar true query response volumes).

Still, we believe that DP VHEMM offers reasonable resilience against query reconstruction attacks that focus on low-frequency multi-map keys since the expected q-leakage is significantly smaller than the worst-case of 1. This suggests that DP VHEMM is useful in applications where most of the multi-map keys have small query response volumes (This is common since many real-world multi-maps follow Zipf's law[1]).

[1]Zipf's law states that empirically, the value of the nth entry in a list of measured values is inversely proportional to n.

7 Case Study: Analysis of SEAL

Overview of SEAL. A searchable encryption scheme is a scheme that supports encrypted keyword search on encrypted documents. In this section, we study the security of SEAL, which is a searchable encryption scheme proposed by Demertzis et al. [Demertzis et al.(2020)].

SEAL [Demertzis et al.(2020)] a static searchable encryption scheme with adjustable leakage. The construction uses two main techniques to suppress leakage. The first technique is the use of ORAM with adjustable leakage derived from a generic ORAM construction [Stefanov et al.(2013)]. The idea there is that instead of using a single ORAM to access all data, the data is permuted and partitioned into 2^α partitions. These partitions are then stored in individual ORAMs. When the client wants to retrieve an item, it first computes which ORAM it needs to access, and then accesses that ORAM to get the result. By increasing α, one can get a more efficient data retrieval scheme (because the individual ORAMs are smaller) at the cost of larger leakage.

The second technique used is padding. The authors suggest padding the frequencies of the keywords to the next power of a chosen integer x. This hides the exact keyword frequencies. In addition, if two keywords have the same padded frequency, they cannot be distinguished from each other.

Known results about SEAL security. With the description of the scheme, it is easy to see that the public parameter for SEAL is $\mathsf{pub} = (\alpha, x)$. The leakage of the scheme is as follows:

- $\mathcal{L}_{\mathbf{Setup}}(\mathsf{pub}, \mathsf{DB}) = \sum_{i=1}^{N} l_i$ where l_i is the number of keywords in the i-th document.
- $\mathcal{L}_{\mathbf{EQuery}}(\mathsf{pub}, (\mathsf{q}_i)_{i=1}^{l}, \mathsf{DB})$ is a stateful leakage function as follows. Before generating the leakage for any queries, the leakage function computes $M = \sum_{i=1}^{N} x^{\lceil \log_x |\mathsf{DB}(\mathsf{w}_i)| \rceil}$, where w_i are keywords in the database. Then, the leakage function creates 2^α partitions $P_1, \ldots, P_{2^\alpha}$. These partitions have a maximum size of $\lceil \frac{M}{2^\alpha} \rceil$ each. Then, the leakage function initialises a map $M : \mathsf{Q} \to \mathcal{P}(\mathbb{N})$ as an empty map. For every keyword w_i in the database (these keywords can also be viewed as elements in Q), the leakage function inserts $x^{\lceil \log_x |\mathsf{DB}(\mathsf{w}_i)| \rceil}$ elements $(\mathsf{w}_i, 1), \ldots, (\mathsf{w}_i, x^{\lceil \log_x |\mathsf{DB}(\mathsf{w}_i)| \rceil})$ into the partitions randomly. If a partition is full, the leakage function will try to insert the element into a different random partition. Let the indices of the partitions (the index of P_i is i) touched by the insertion process above (with repetition) be $\mathsf{idx}_1, \ldots, \mathsf{idx}_{x^{\lceil \log_x |\mathsf{DB}(\mathsf{w}_i)| \rceil}}$. The leakage function finally sets $M(\mathsf{w}_i) = (\mathsf{idx}_1, \ldots, \mathsf{idx}_{x^{\lceil \log_x |\mathsf{DB}(\mathsf{w}_i)| \rceil}})$.
  Then, the output of $\mathcal{L}_{\mathbf{EQuery}}(\mathsf{pub}, (\mathsf{q}_i)_{i=1}^{l}, \mathsf{DB})$ is simply $M(\mathsf{q}_1), \ldots, M(\mathsf{q}_l)$.

Oya and Kerschbaum [Oya and Kerschbaum(2021)] showed that SEAL is vulnerable to query reconstruction attacks for certain databases. In particular, the authors performed experiments on the Enron email corpus and the java-user mailing list from the Lucene project. In the experiments, these datasets are split into two halves, one half as the target dataset and the other half as the auxiliary dataset. The target dataset is encrypted using SEAL and queries are made on the encrypted database (the queries are generated with Google Trends). Then, the leakage observed from the setup and query phase of the protocol, along with the auxiliary dataset and Google Trends as a query distribution are given to the attacker. Oya and Kerschbaum showed that their attack algorithm can recover 23% of the queries correctly on the Lucene dataset, when the number of keywords used is set to 1000, the average number of queries is 250, and $x = 4$ (the authors did not specify the choice of α in the paper). The attack demonstrates that if the attacker has auxiliary information on the query distribution and keyword frequency distribution, query reconstruction can still work on the scheme.

Oya and Kerschbaum's attack does demonstrate the weaknesses of SEAL, but the assumptions on the power of the attacker are quite strong. In practice, it is hard to obtain an auxiliary query distribution. Without one, Oya and Kerschbaum's attack has very limited success (below 10% query recovery rate) in query reconstruction. We analyze the security implications of SEAL's leakage in the presence of auxiliary keyword frequency distribution but in the absence of auxiliary query distribution.

Analysis of SEAL's Leakage. We begin by making a general observation on the leakage of SEAL. SEAL hides the identity of the keywords by padding the frequencies of the keywords to the next power of x (x is chosen by the user) and hiding the keywords amongst the other keywords with the same padded frequency.

Given a database DB, we define $M_{x,i}$ to be the number of keywords in DB that has padded volume x^i. Then it is easy to observe that the best adversary who aims to recover a query with query response volume

x^i has $\frac{1}{M_{x,i}}$ chance of succeeding.

However, SEAL is designed to obfuscate keywords that already have similar frequencies. It is entirely possible that there exists an i such that $M_{x,i}$ is small (e.g. a keyword has a really high frequency and there are no other keywords with a similar frequency). In that case, SEAL will fail to protect the identity of those keywords. Hence, the security of SEAL depends on the distribution of keywords in the database. Our analysis below shows specific distributions of databases for which SEAL offers security.

Rewarding the most frequent Keywords. Although high-frequency keywords are less protected in general due to the distinct frequencies they have, there are databases where these keywords (and other keywords) are not vulnerable to query and data reconstruction attacks. For example, if the keywords are social security numbers (SSNs) and each SSN only appears once in the database, then $M_{x,1} = N$ where N is the size of the database, and the chance of any adversary guessing any SSN correctly is $\frac{1}{N}$.

More generally, for any query-guessing adversary that only has access to the padded frequencies of the keywords as auxiliary information (i.e. for keyword w, the adversary only knows that the true frequency of the keyword is between $x^{\lceil \log_x |\mathsf{DB}(\mathsf{w})| \rceil - 1}$ and $x^{\lceil \log_x |\mathsf{DB}(\mathsf{w})| \rceil}$), the expected q-leakage is upper bounded by $\min_i \frac{1}{M_{x,i}} \cdot \max ql$. We state this formally in Theorem 6.

Theorem 6. Let 1^λ be a security parameter and $\mathsf{pub} = (\alpha, x)$ be a public parameter. Let $l \in \mathbb{N}$ be a natural number and $M_{x,i} \in \mathbb{N}$ for $i = 0, \ldots, l$ be a list of natural numbers.

Let $\mathcal{Z}_3^{\{M_{x,i}\}_{i=0}^l}$ be the following environment:

1. For $i = 0, \ldots, l$, it randomly generates $M_{x,i}$ keywords $\mathsf{w}_{i,1}$, $\mathsf{w}_{i,M_{x,i}}$ (with an arbitrary distribution picked by the environment $\mathcal{Z}_3$) with the constraint that each keyword is unique (across all i's). For each keyword $\mathsf{w}_{i,j}$, generate a random number r between $x^{i-1} + 1$ and x^i inclusively. Then randomly generate r documents $\mathsf{d}_{i,j,k}$ for $k = 1, \ldots, r$ (with an arbitrary distribution picked by the environment $\mathcal{Z}_3$).
2. It sets the database DB as $(\mathsf{d}_{i,j,k}, \{\mathsf{w}_{i,j}\})_{i,j}$.
3. It sets the auxiliary information as $\{(\mathsf{w}_{i,j}, i)\}_{i,j}$. (Hence the adversary knows which "frequency bin" does each keyword fall into.)

Let $h : \{0,1\}^* \times \mathsf{Data}^* \times \mathsf{Q}^* \to \mathbb{R}$ be a function. Suppose ql_3 has the shape $ql_3((i, \mathtt{key}_k), \mathsf{pub}, (\mathsf{MM}_j)_{j=0}^l, (\mathsf{q}_j)_{j=1}^l) = h((i, \mathtt{key}_k), \mathsf{pub}, (\mathsf{MM}_j)_{j=0}^l, (\mathsf{q}_j)_{j=1}^l) \cdot \mathbb{1}\{\mathtt{key}_k = \mathsf{q}_i\}$. Then, given security parameter 1^λ and public parameter $\mathsf{pub} = (\alpha, x)$, for any $l \in \mathbb{N}$ and $M_{x,i} \in \mathbb{N}$ for $i = 0, \ldots, l$, for any function ql_3, the expected q-leakage of SEAL with respect to the environment $\mathcal{Z}_3^{\{M_{x,i}\}_{i=0}^l}$ and function ql_3 is upper bounded as:

$$\sup_{\mathcal{A}} \mathbf{E}\left[\mathrm{Real}^{\,ql_3}_{\mathrm{SEAL},\mathcal{A},\mathcal{Z}_3^{\{M_{x,i}\}_{i=0}^l}}(1^\lambda, \mathsf{pub})\right] \le \min_i \frac{1}{M_{x,i}} \cdot \sup_z |h(z)|.$$

It is easy to see why the bound works. The supremum of the expected q-leakage is the same as the supremum of ql_3. But ql_3 can be decomposed into h and an indicator function $\mathbb{1}\{\mathtt{key}_k = \mathsf{q}_i\}$. For the indicator function, we know that if there are $M_{x,i}$ keywords with the same padded frequency, the adversary cannot distinguish these keywords. So the indicator function can only be 1 $\frac{1}{M_{x,i}}$ of the times. Hence, the $\min_i \frac{1}{M_{x,i}}$ component in the bound. The other half of the bound comes from the fact that the supremum of ql_3 is upper bounded by the supremum of h.

This bound is useful when $\min_i \frac{1}{M_{x,i}}$ is small. Or equivalently, all M_{x_i} are large. This corresponds to a database where all keyword frequencies are similar (e.g. a database of SSNs where each SSN only appear once), or a database where there are several clusters of keyword frequencies and all clusters have a reasonably large size (e.g. a database of products by a company where groups of products are produced in similar quantities).

Rewarding Low-frequent Keywords. Conversely, an adversary may be interested in recovering the keywords of the queries with low frequency. An example of a ql-function that rewards low-frequent keywords more is

$$ql_4((i, \mathsf{w}), \mathsf{pub}, (\mathsf{DB}_j)_{j=0}^l, (\mathsf{q}_j)_{j=1}^l) = \frac{\mathbb{1}\{\mathsf{q}_i = \mathsf{w}\}}{|\mathsf{DB}(\mathsf{q}_i)|}.$$

This function achieves the goal since it is inversely proportional to the frequency of the guessed keyword.

For databases where the frequency of the keywords following a Zipf distribution, we can prove the following bound on the expected q-leakage.

Theorem 7. Let 1^λ be a security parameter and $\mathsf{pub} = (\alpha, x)$ be a public parameter. Let $n \in \mathbb{N}$ be the number of keywords in the database.

Let $\mathcal{Z}_4^n$ be the following environment:

1. For $i = 0, \dots, n$, it samples r_i from the Zipf distribution with probability mass function $\mathbf{Pr}[r] = \frac{1}{H_n}\frac{1}{r}$, where H_n is the n-th harmonic number. After that, it randomly generates keyword w_i and r documents $\mathsf{d}_{i,1}, \dots, \mathsf{d}_{i,r}$(with an arbitrary distribution picked by the environment $\mathcal{Z}_4$).
2. It sets the database DB as $(\mathsf{d}_{i,j}, \{\mathsf{w}_i\})_{i,j}$.
3. It sets the auxiliary information as $\{(\mathsf{w}_i, \lceil \log_x r_i \rceil)\}_i$.

Given security parameter 1^λ and public parameter $\mathsf{pub} = (\alpha, x)$, for any $n \in \mathbb{N}$, $n \gg x$, for any function ql_4, the expected q-leakage of SEAL with respect to the environment $\mathcal{Z}_4^n$ and function ql_4 is upper bounded as:

$$\sup_{\mathcal{A}} \mathbf{E}\left[\mathrm{Real}^{ql_4}_{\mathrm{SEAL},\mathcal{A},\mathcal{Z}_4^n}(1^\lambda, \mathsf{pub})\right] \le \frac{H_n}{n}\frac{x}{x-1}.$$

Proof. We begin by computing the expectation of $M_{x,i}$. Since the keyword frequencies follow Zipf distribution with probability mass function $\mathbf{Pr}[r] = \frac{1}{H_n}\frac{1}{r}$, the expectation of $M_{x,i}$ can be calculated as

$$\begin{aligned}\mathbf{E}[M_{x,i}] &= n \cdot \left(\frac{1}{H_n}\frac{1}{x^{i-1}+1} + \ldots + \frac{1}{H_n}\frac{1}{x^i}\right)\\ &\ge \frac{n}{H_n}\frac{x^i - x^{i-1}}{x^i}\\ &= \frac{n}{H_n}\frac{x-1}{x}\ .\end{aligned}$$

Recall that guessing a keyword with frequency r correct is awarded $\frac{1}{r}$ gain. It means that if r after padding becomes x^i for some i, the gain is upper bounded by $\frac{1}{x^{i-1}}$.

Since the adversary is given all the plaintext keywords with the floors of $\log_x$ of their frequencies, upon observing the query response volumes, the best the adversary can do is to randomly pick a keyword with matching padded volume as its guess. This means the expected gain is upper bounded as follows:

$$\begin{aligned}&\sup_{\mathcal{A}} \mathbf{E}\left[\mathbf{Real}^{ql_4}_{\mathrm{SEAL},\mathcal{A},\mathcal{Z}_4^n}(1^\lambda, \mathsf{pub})\right]\\ &\le \max_i \frac{1}{x^{i-1}} \cdot \frac{H_n}{n}\frac{x}{x-1}\\ &\le \frac{H_n}{n}\frac{x}{x-1}\ .\end{aligned}$$

□

As an example, if $n = 1000$ (1000 keywords in total where the query response volumes r are distributed with the probability mass function $\mathbf{Pr}[r] = \frac{1}{H_n}\frac{1}{r}$), we get the bound on the expected q-leakage as $0.0075\frac{x}{x-1}$.

On the contrary, the baseline for ql_4 is $\frac{H_n}{n^2}$. This is easy to see since the best guessing strategy of the adversary is to guess randomly and its expected q-leakage when it guesses correctly is $\frac{1}{n} \cdot \sum_{i=1}^{n} \frac{1}{n} = \frac{H_n}{n}$. Numerically, this means that the baseline for the example above is 7.49×10^{-6} which is significantly smaller than the expected q-leakage of SEAL. However, this does not mean SEAL is ineffective in suppressing volume leakage since the expected q-leakage of SEAL is significantly better than the worst-case bound of 1.

Interestingly, we observe that the bound on the expected q-leakage for SEAL grows proportional to $\frac{x}{x-1}$. This suggests that increasing x is not efficient in reducing the bound (since the storage/communication overhead grows linearly in x but the security bound only reduces proportional to $\frac{x}{x-1}$).

8 Conclusions

We proposed a new methodology to quantify leakage of information inherent in most practical structured encryption schemes. We exemplified its use with two practical protocol analyses that provided novel security insights. For DP VHEMM, we showed that for multi-maps that have similar query response volumes from different multi-map keys, DP VHEMM can offer sufficient query privacy. For SEAL, we discovered a surprising result where padding has a diminishing return in terms of the query privacy it provides. These kinds of fine-grained security analysis are not possible with the standard security notion. We leave it to future works to study more existing and new schemes' leakage with respect to various ql-functions and environments. We note that as the first step, we only considered semi-honest security. We hope that future works will generalize our framework to active adversaries. We believe our framework will prove to be useful for more fine-grain analyses of practical structured encryption schemes and help their wider adoption.

9 Acknowledgments

We thank the anonymous PETS reviewers for their useful comments. Alexandra Boldyreva was supported in part by Cisco Research Award and the National Science Foundation Award No.1946919. Zichen Gui was supported by Zurich Information Security and Privacy Center (ZISC).

References

[Alvim et al.(2012)] M. S. Alvim, K. Chatzikokolakis, C. Palamidessi, and G. Smith. 2012. Measuring Information Leakage Using Generalized Gain Functions. In *2012 IEEE 25th Computer Security Foundations Symposium*. 265–279. `https://doi.org/10.1109/CSF.2012.26`

[Amanatidis et al.(2007)] Georgios Amanatidis, Alexandra Boldyreva, and Adam O'Neill. 2007. Provably-Secure Schemes for Basic Query Support in Outsourced Databases. In *Data and Applications Security XXI, 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Redondo Beach, CA, USA, July 8-11, 2007, Proceedings (Lecture Notes in Computer Science, Vol. 4602)*, Steve Barker and Gail-Joon Ahn (Eds.). Springer, 14–30. `https://doi.org/10.1007/978-3-540-73538-0\_2`

[Bag et al.(2023)] Arnab Bag, Debadrita Talapatra, Ayushi Rastogi, Sikhar Patranabis, and Debdeep Mukhopadhyay. 2023. TWo-IN-one-SSE: Fast, Scalable and Storage-Efficient Searchable Symmetric Encryption for Conjunctive and Disjunctive Boolean Queries. *Proceedings on Privacy Enhancing Technologies* 2023, 1 (Jan. 2023), 115–139. `https://doi.org/10.56553/popets-2023-0008`

[Bellare et al.(2007)] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. 2007. Deterministic and Efficiently Searchable Encryption. In *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings (Lecture Notes in Computer Science, Vol. 4622)*, Alfred Menezes (Ed.). Springer, 535–552. `https://doi.org/10.1007/978-3-540-74143-5\_30`

[Bindschaedler et al.(2018)] Vincent Bindschaedler, Paul Grubbs, David Cash, Thomas Ristenpart, and Vitaly Shmatikov. 2018. The Tao of Inference in Privacy-Protected Databases. *Proc. VLDB Endow.* 11, 11 (jul 2018), 1715–1728. `https://doi.org/10.14778/3236187.3236217`

[Blackstone et al.(2020)] Laura Blackstone, Seny Kamara, and Tarik Moataz. 2020. Revisiting Leakage Abuse Attacks. In *ISOC Network and Distributed System Security Symposium – NDSS 2020*. The Internet Society, San Diego, CA, USA.

[Boldyreva et al.(2009)] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. 2009. Order-Preserving Symmetric Encryption. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings (Lecture Notes in Computer Science, Vol. 5479)*, Antoine Joux (Ed.). Springer, 224–241. `https://doi.org/10.1007/978-3-642-01001-9\_13`

[Boldyreva et al.(2011)] Alexandra Boldyreva, Nathan Chenette, and Adam O'Neill. 2011. Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings (Lecture Notes in Computer Science, Vol. 6841)*, Phillip Rogaway (Ed.). Springer, 578–595. `https://doi.org/10.1007/978-3-642-22792-9\_33`

[Boreale and Pampaloni(2014)] Michele Boreale and Francesca Pampaloni. 2014. Quantitative Information Flow under Generic Leakage Functions and Adaptive Adversaries. In *Formal Techniques for Distributed Objects, Components, and Systems*, Erika Ábrahám and Catuscia Palamidessi (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 166–181.

[Bost(2016)] Raphael Bost. 2016. $\Sigma o\phi o\varsigma$: Forward Secure Searchable Encryption. In *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi (Eds.). ACM Press, Vienna, Austria, 1143–1154. `https://doi.org/10.1145/2976749.2978303`

[Bost et al.(2017)] Raphaël Bost, Brice Minaud, and Olga Ohrimenko. 2017. Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives. In *ACM CCS 2017: 24th Conference on Computer and Communications Security*, Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM Press, Dallas, TX, USA, 1465–1482. `https://doi.org/10.1145/3133956.3133980`

[Cash et al.(2015)] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. 2015. Leakage-Abuse Attacks Against Searchable Encryption. In *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, Indrajit Ray, Ninghui Li, and Christopher Kruegel (Eds.). ACM Press, Denver, CO, USA, 668–679. `https://doi.org/10.1145/2810103.2813700`

[Cash et al.(2014)] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. 2014. Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation. In *ISOC Network and Distributed System Security Symposium – NDSS 2014*. The Internet Society, San Diego, CA, USA.

[Cash et al.(2013)] David Cash, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. 2013. Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries. In *Advances in Cryptology – CRYPTO 2013, Part I (Lecture Notes in Computer Science, Vol. 8042)*, Ran Canetti and Juan A. Garay (Eds.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 353–373. `https://doi.org/10.1007/978-3-642-40041-4_20`

[Chamani et al.(2018)] Javad Ghareh Chamani, Dimitrios Papadopoulos, Charalampos Papamanthou, and Rasool Jalili. 2018. New Constructions for Forward and Backward Private Symmetric Searchable Encryption. In *ACM CCS 2018: 25th Conference on Computer and Communications Security*, David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang (Eds.). ACM Press, Toronto, ON, Canada, 1038–1055. `https://doi.org/10.1145/3243734.3243833`

[Chang and Mitzenmacher(2005)] Yan-Cheng Chang and Michael Mitzenmacher. 2005. Privacy Preserving Keyword Searches on Remote Encrypted Data. In *ACNS 05: 3rd International Conference on Applied Cryptography and Network Security (Lecture Notes in Computer Science, Vol. 3531)*, John Ioannidis, Angelos Keromytis, and Moti Yung (Eds.). Springer, Heidelberg, Germany, New York, NY, USA, 442–455. `https://doi.org/10.1007/11496137_30`

[Chase and Kamara(2010)] Melissa Chase and Seny Kamara. 2010. Structured Encryption and Controlled Disclosure. In *Advances in Cryptology – ASIACRYPT 2010 (Lecture Notes in Computer Science, Vol. 6477)*, Masayuki Abe (Ed.). Springer, Heidelberg, Germany, Singapore, 577–594. `https://doi.org/10.1007/978-3-642-17373-8_33`

[Chen et al.(2018)] G. Chen, T. Lai, M. K. Reiter, and Y. Zhang. 2018. Differentially Private Access Patterns for Searchable Symmetric Encryption. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 810–818. `https://doi.org/10.1109/INFOCOM.2018.8486381`

[Curtmola et al.(2006)] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. 2006. Searchable symmetric encryption: improved definitions and efficient constructions. In *ACM CCS 2006: 13th Conference on Computer and Communications Security*, Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati (Eds.). ACM Press, Alexandria, Virginia, USA, 79–88. `https://doi.org/10.1145/1180405.1180417`

[Demertzis et al.(2020)] Ioannis Demertzis, Dimitrios Papadopoulos, Charalampos Papamanthou, and Saurabh Shintre. 2020. SEAL: Attack Mitigation for Encrypted Databases via Adjustable Leakage. In *USENIX Security 2020: 29th USENIX Security Symposium*, Srdjan Capkun and Franziska Roesner (Eds.). USENIX Association, 2433–2450.

[Dwork et al.(2006)] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC 2006: 3rd Theory of Cryptography Conference (Lecture Notes in Computer Science, Vol. 3876)*, Shai Halevi and Tal Rabin (Eds.). Springer, Heidelberg, Germany, New York, NY, USA, 265–284. `https://doi.org/10.1007/11681878_14`

[for Health Statistics(2023)] Texas Health Care Information Collection Center for Health Statistics. 2023. Texas Inpatient Public Use Data File (PUDF). `https://www.dshs.texas.`

gov/texas-health-care-information-collection/health-data-researcher-information/texas-inpatient-public-use. [Online; accessed 29-Nov-2023].

[George et al.(2021)] Marilyn George, Seny Kamara, and Tarik Moataz. 2021. Structured Encryption and Dynamic Leakage Suppression. In *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part III (Lecture Notes in Computer Science, Vol. 12698)*, Anne Canteaut and François-Xavier Standaert (Eds.). Springer, 370–396. https://doi.org/10.1007/978-3-030-77883-5_13

[Ghosh et al.(2021)] Esha Ghosh, Seny Kamara, and Roberto Tamassia. 2021. Efficient Graph Encryption Scheme for Shortest Path Queries. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security* (Virtual Event, Hong Kong) *(ASIA CCS '21)*. Association for Computing Machinery, New York, NY, USA, 516–525. https://doi.org/10.1145/3433210.3453099

[Goh(2003)] Eu-Jin Goh. 2003. Secure Indexes. Cryptology ePrint Archive, Report 2003/216. https://eprint.iacr.org/2003/216.

[Grubbs et al.(2018)] 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 CCS 2018: 25th Conference on Computer and Communications Security*, David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang (Eds.). ACM Press, Toronto, ON, Canada, 315–331. https://doi.org/10.1145/3243734.3243864

[Grubbs et al.(2019)] Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson. 2019. Learning to Reconstruct: Statistical Learning Theory and Encrypted Database Attacks. In *2019 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, San Francisco, CA, USA, 1067–1083. https://doi.org/10.1109/SP.2019.00030

[Gui et al.(2019)] Zichen Gui, Oliver Johnson, and Bogdan Warinschi. 2019. Encrypted Databases: New Volume Attacks against Range Queries. In *ACM CCS 2019: 26th Conference on Computer and Communications Security*, Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz (Eds.). ACM Press, London, UK, 361–378. https://doi.org/10.1145/3319535.3363210

[Gui et al.(2023a)] Zichen Gui, Kenneth G. Paterson, and Sikhar Patranabis. 2023a. Rethinking Searchable Symmetric Encryption. In *2023 IEEE Symposium on Security and Privacy (SP)*. 1401–1418. https://doi.org/10.1109/SP46215.2023.10179460

[Gui et al.(2023b)] Zichen Gui, Kenneth G. Paterson, and Tianxin Tang. 2023b. Security Analysis of MongoDB Queryable Encryption. In *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, Joseph A. Calandrino and Carmela Troncoso (Eds.). USENIX Association, 7445–7462. https://www.usenix.org/conference/usenixsecurity23/presentation/gui

[Islam et al.(2012)] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. 2012. Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation. In *ISOC Network and Distributed System Security Symposium – NDSS 2012*. The Internet Society, San Diego, CA, USA.

[Jurado et al.(2021)] Mireya Jurado, Catuscia Palamidessi, and Geoffrey Smith. 2021. A Formal Information-Theoretic Leakage Analysis of Order-Revealing Encryption. In *34th IEEE Computer Security Foundations Symposium, CSF 2021, Dubrovnik, Croatia, June 21-25, 2021*. IEEE, 1–16. https://doi.org/10.1109/CSF51468.2021.00046

[Jurado and Smith(2019)] Mireya Jurado and Geoffrey Smith. 2019. Quantifying Information Leakage of Deterministic Encryption. In *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop, CCSW@CCS 2019, London, UK, November 11, 2019*, Radu Sion and Charalampos Papamanthou (Eds.). ACM, 129–139. https://doi.org/10.1145/3338466.3358915

[Kamara and Moataz(2017)] Seny Kamara and Tarik Moataz. 2017. Boolean Searchable Symmetric Encryption with Worst-Case Sub-linear Complexity. In *Advances in Cryptology – EUROCRYPT 2017, Part III (Lecture Notes in Computer Science, Vol. 10212)*, Jean-Sébastien Coron and Jesper Buus Nielsen (Eds.). Springer, Heidelberg, Germany, Paris, France, 94–124. https://doi.org/10.1007/978-3-319-56617-7_4

[Kamara and Moataz(2019)] Seny Kamara and Tarik Moataz. 2019. Computationally Volume-Hiding Structured Encryption. In *Advances in Cryptology – EUROCRYPT 2019, Part II (Lecture Notes in Computer Science, Vol. 11477)*, Yuval Ishai and Vincent Rijmen (Eds.). Springer, Heidelberg, Germany, Darmstadt, Germany, 183–213. https://doi.org/10.1007/978-3-030-17656-3_7

[Kamara and Moataz(2023)] Seny Kamara and Tarik Moataz. 2023. Bayesian Leakage Analysis: A Framework for Analyzing Leakage in Encrypted Search. Cryptology ePrint Archive, Report 2023/813. https://eprint.iacr.org/2023/813.

[Kamara and Papamanthou(2013)] Seny Kamara and Charalampos Papamanthou. 2013. Parallel and Dynamic Searchable Symmetric Encryption. In *FC 2013: 17th International Conference on Financial Cryptography and Data Security (Lecture Notes in Computer Science, Vol. 7859)*, Ahmad-Reza Sadeghi (Ed.). Springer, Heidelberg, Germany, Okinawa, Japan, 258–274. `https://doi.org/10.1007/978-3-642-39884-1_22`

[Kamara et al.(2012)] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. 2012. Dynamic searchable symmetric encryption. In *ACM CCS 2012: 19th Conference on Computer and Communications Security*, Ting Yu, George Danezis, and Virgil D. Gligor (Eds.). ACM Press, Raleigh, NC, USA, 965–976. `https://doi.org/10.1145/2382196.2382298`

[Kornaropoulos et al.(2022)] Evgenios M. Kornaropoulos, Nathaniel Moyer, Charalampos Papamanthou, and Alexandros Psomas. 2022. Leakage Inversion: Towards Quantifying Privacy in Searchable Encryption. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi (Eds.). ACM, 1829–1842. `https://doi.org/10.1145/3548606.3560593`

[Lacharité et al.(2018)] Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson. 2018. Improved Reconstruction Attacks on Encrypted Data Using Range Query Leakage. In *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, San Francisco, CA, USA, 297–314. `https://doi.org/10.1109/SP.2018.00002`

[Meng et al.(2015)] Xianrui Meng, Seny Kamara, Kobbi Nissim, and George Kollios. 2015. GRECS: Graph Encryption for Approximate Shortest Distance Queries. In *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, Indrajit Ray, Ninghui Li, and Christopher Kruegel (Eds.). ACM Press, Denver, CO, USA, 504–517. `https://doi.org/10.1145/2810103.2813672`

[Minaud and Reichle(2022)] Brice Minaud and Michael Reichle. 2022. Dynamic Local Searchable Symmetric Encryption. In *Advances in Cryptology – CRYPTO 2022, Part IV (Lecture Notes in Computer Science, Vol. 13510)*, Yevgeniy Dodis and Thomas Shrimpton (Eds.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 91–120. `https://doi.org/10.1007/978-3-031-15985-5_4`

[MongoDB(2023)] MongoDB. 2023. Queryable Encryption: MongoDB Manual. `https://www.mongodb.com/docs/manual/core/queryable-encryption/`.

[Naveed et al.(2015)] Muhammad Naveed, Seny Kamara, and Charles V. Wright. 2015. Inference Attacks on Property-Preserving Encrypted Databases. In *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, Indrajit Ray, Ninghui Li, and Christopher Kruegel (Eds.). ACM Press, Denver, CO, USA, 644–655. `https://doi.org/10.1145/2810103.2813651`

[Naveed et al.(2014)] Muhammad Naveed, Manoj Prabhakaran, and Carl A. Gunter. 2014. Dynamic Searchable Encryption via Blind Storage. In *2014 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, Berkeley, CA, USA, 639–654. `https://doi.org/10.1109/SP.2014.47`

[Oya and Kerschbaum(2021)] Simon Oya and Florian Kerschbaum. 2021. Hiding the Access Pattern is Not Enough: Exploiting Search Pattern Leakage in Searchable Encryption. In *USENIX Security 2021: 30th USENIX Security Symposium*, Michael Bailey and Rachel Greenstadt (Eds.). USENIX Association, 127–142.

[Oya and Kerschbaum(2022)] Simon Oya and Florian Kerschbaum. 2022. IHOP: Improved Statistical Query Recovery against Searchable Symmetric Encryption through Quadratic Optimization. In *USENIX Security 2022: 31st USENIX Security Symposium*, Kevin R. B. Butler and Kurt Thomas (Eds.). USENIX Association, Boston, MA, USA, 2407–2424.

[Pandey and Rouselakis(2012)] Omkant Pandey and Yannis Rouselakis. 2012. Property Preserving Symmetric Encryption. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings (Lecture Notes in Computer Science, Vol. 7237)*, David Pointcheval and Thomas Johansson (Eds.). Springer, 375–391. `https://doi.org/10.1007/978-3-642-29011-4\_23`

[Pappas et al.(2014)] Vasilis Pappas, Fernando Krell, Binh Vo, Vladimir Kolesnikov, Tal Malkin, Seung Geol Choi, Wesley George, Angelos D. Keromytis, and Steven M. Bellovin. 2014. Blind Seer: A Scalable Private DBMS. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*. IEEE Computer Society, 359–374. `https://doi.org/10.1109/SP.2014.30`

[Patel et al.(2019)] Sarvar Patel, Giuseppe Persiano, Kevin Yeo, and Moti Yung. 2019. Mitigating Leakage in Secure Cloud-Hosted Data Structures: Volume-Hiding for Multi-Maps via Hashing. In *ACM CCS 2019: 26th Conference on Computer and Communications Security*, Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz (Eds.). ACM Press, London, UK, 79–93. `https://doi.org/10.1145/3319535.3354213`

[Services(2023)] Amazon Web Services. 2023. AWS Database Encryption SDK. `https://docs.aws.amazon.com/database-encryption-sdk/latest/devguide/searchable-encryption.html`.

[Smith(2015)] Geoffrey Smith. 2015. Recent Developments in Quantitative Information Flow (*Invited Tutorial*). In *Proc. LICS 2015: 30th ACM/IEEE Symposium on Logic in Computer Science*. 23–31.

[Stefanov et al.(2014)] Emil Stefanov, Charalampos Papamanthou, and Elaine Shi. 2014. Practical Dynamic Searchable Encryption with Small Leakage. In *ISOC Network and Distributed System Security Symposium – NDSS 2014*. The Internet Society, San Diego, CA, USA.

[Stefanov et al.(2013)] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher W. Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. 2013. Path ORAM: an extremely simple oblivious RAM protocol. In *ACM CCS 2013: 20th Conference on Computer and Communications Security*, Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung (Eds.). ACM Press, Berlin, Germany, 299–310. `https://doi.org/10.1145/2508859.2516660`

[Zhang et al.(2016)] 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 2016: 25th USENIX Security Symposium*, Thorsten Holz and Stefan Savage (Eds.). USENIX Association, Austin, TX, USA, 707–720.

A Full Description of DP VHEMM

In this section, we give a detailed description of DP VHEMM.

Basic Volume-Hiding Encrypted Multi-map. In [Patel et al.(2019)], Patel et al. proposed a differentially-private volume-hiding EMM based on Cuckoo hashing. To simplify its description, we first present the basic volume-hiding construction (we refer to the solution as VHEMM from now on), and later present the scheme enhanced with differential privacy.

Let 1^λ be the security parameter, $\mathsf{F} : \{0,1\}^\lambda \times \{0,1\}^m \to \{0,1\}^n$ be a pseudorandom function family, and $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ be a symmetric encryption scheme with message space $\{0,1\}^*$ and key space $\{0,1\}^\lambda$. Let $\mathsf{MM} = \left\{(\mathtt{key}_i, (v_{i,j})_{j=1}^{l_i})\right\}_{i\in[N]}$ be a multi-map. The scheme has a public parameter $\alpha \in \mathbb{R}^+$ where α is used to determine the size of the hash tables. The scheme VHEMM=(**Gen**, **Setup**, **EQuery**) is as follows. **Gen** samples two keys, $\mathsf{sk_F}$ for F and $\mathsf{sk_{Enc}}$ for Π. **Setup** initialises two hash tables T_0 and T_1 of size $(1+\alpha) * \sum_{i\in[N]} l_i$ such that $n = \lceil \log_2((1+\alpha) * \sum_{i\in[N]} l_i)\rceil$. Then, for every key-value pair $(\mathtt{key}_i, (v_{i,j})_{j=1}^{l_i})$, the client breaks the key-value pair down into $(\mathtt{key}_i, v_{i,1}), \ldots, (\mathtt{key}_i, v_{i,l_j})$. For $j = 1, \ldots, l_i$, the client inserts value $v_{i,j}$ into the position $\mathsf{F}_{\mathsf{sk_F}}(\mathtt{key}_i||j||0)$ of hash table T_0. Similarly, it inserts value $v_{i,j}$ into the position $\mathsf{F}_{\mathsf{sk_F}}(\mathtt{key}_i||j||1)$ of hash table T_1. If the position in the hash table has already been occupied, the client follows the insertion procedure of Cuckoo hashing and moves the preoccupied item to a different location.

Once all elements have been inserted, the algorithm fills the empty locations of the hash tables with dummy values. The dummy values are chosen in a way that they can be distinguished from real values. Then all entries are encrypted using $\Pi.\mathsf{Enc}$ under $\mathsf{sk_{Enc}}$, and the ciphertexts are sent to the server. In addition, the client finds the maximum query response volume, i.e. $\max_i l_i$ and stores it as state.

The **EQuery** protocol is as follows. To query a multi-map key $\mathtt{key}$, the client computes addresses $\mathsf{F}_{\mathsf{sk_F}}(\mathtt{key}||j||b)$ for $j = 1, \ldots, \max_i l_i$ and $b = 0, 1$. These addresses are sent to the server and the server responds with the encrypted contents stored in these addresses in the hash tables. These encrypted contents are then decrypted using Dec and $\mathsf{sk_{Enc}}$ and the dummy query responses are filtered out.

Differentially Private Volume-Hiding Encrypted Multi-map. The VHEMM scheme described above can be quite inefficient as the client has to retrieve encrypted contents from $2 * \max_i l_i$ addresses for every query. As an alternative, Patel et al. [Patel et al.(2019)] proposed another scheme based on *differential privacy*. We refer to the solution as DP VHEMM, and it will be the focus of our analysis.

DP VHEMM uses function **Lap** : $\mathbb{R}^+ \times \{0,1\}^* \to \mathbb{R}$ to sample Laplace random variables. It takes as input a scale parameter s of a Laplace random variable and a seed seed, and outputs a realisation of the Laplace random variable with location 0 and scale s. We write **Lap**$(s; \mathsf{seed})$ to emphasize that seed fully determines the randomness of **Lap**.

The public parameter of DP VHEMM contains the following:

- $\alpha \in \mathbb{R}^+$: A parameter determining the size of the hash tables.
- $\varepsilon \in \mathbb{R}^+$: The scale parameter of a Laplace random variable.

- $\lambda_{DP} \in \mathbb{R}^+$: A parameter determening the amount of fixed padding (on top of the random padding introduced by a Laplace random variable) in the scheme.
- $c(\lambda_{DP}, \varepsilon)$: A function that determines the amount of additional padding based on λ_{DP} and ε.

Gen samples $\mathsf{sk}_1, \mathsf{sk}_2$ for F, $\mathsf{sk}_{\mathsf{Enc}} \in \{0,1\}^\lambda$ for Π, and $\mathsf{sk}_3 \in \{0,1\}^\lambda$ to compute seed for **Lap**.

Setup. The client runs the setup protocol of VHEMM just as before, with sk_1 and $\mathsf{sk}_{\mathsf{Enc}}$. The client also builds a multi-map $\mathsf{MM}' = \{(\mathtt{key}_i, l_i)\}_{i=1}^N$ where N is the total number of multi-map keys which maps the multi-map keys to their *true query response volumes* l_i. The client then runs the setup protocol of VHEMM again on multi-map MM' (with new hash tables), now with sk_2 and $\mathsf{sk}_{\mathsf{Enc}}$. The client does not have to maintain a state.

EQuery. To query a multi-map key $\mathtt{key}$, the client first runs the **EQuery** protocol of VHEMM on the hash tables that store the true query response volumes to obtain $\ell(\mathtt{key}) = |\mathsf{MM}[\mathtt{key}]|$. Next, the client computes the padded query response volume $\ell^*(\mathtt{key}) = \lfloor \ell(\mathtt{key}) + \mathbf{Lap}(2/\varepsilon; \mathsf{F}_{\mathsf{sk}_3}(\mathtt{key})) + c(\lambda_{DP}, \varepsilon) \rfloor$[2] [3]. Then, the client queries the hash table MM that stores the actual key-value pairs on the server with addresses $\mathsf{F}_{\mathsf{sk}_2}(\mathtt{key}||j||b)$ for $j = 1, \ldots, \ell^*(\mathtt{key})$ and $b = 0, 1$. The query responses are decrypted with key $\mathsf{sk}_{\mathsf{Enc}}$ and the dummy values are filtered out.

B Theoretical Results on DP VHEMM

Definition 4 (Secure Pseudorandom Laplace Random Number Generator)**.** Let $\mathbf{G} : \mathcal{R} \times \{0,1\}^\lambda \to \mathcal{Z}$ be a family of deterministic polynomial time computable functions. Let $\mathbf{LapD}(x, b)$ be a mathematical Laplace random variable with location x and scale b. We say that $\mathbf{G}$ is a secure pseudorandom Laplace random number generator if for any $b \in \mathcal{R}$ and polynomial time algorithm $\mathcal{A}$ which outputs 0 or 1 as a distinguisher,

$$\left|\mathbf{Pr}_{\mathsf{sk} \leftarrow \$\{0,1\}^\lambda}[\mathcal{A}(\mathbf{G}(b, \mathsf{sk}))] - \mathbf{Pr}_{r \leftarrow \$\mathbf{LapD}(0,b)}[\mathcal{A}(r)]\right| \leq \mathtt{negl}(\lambda).$$

Note that in DP VHEMM, the **Lap** function has three inputs, namely the scale $2/\varepsilon$, a cryptographic key sk_3 and a multi-map key $\mathtt{key}$. This is slightly different from the expected inputs for a secure pseudorandom Laplace random number generator. In Theorem 8, we show that for any input ε and $\mathtt{key}$ and for a random key sk, the output of function $\mathbf{Lap}(2/\varepsilon; \mathsf{F}_{\mathsf{sk}}(\mathtt{key}))$ is computationally indistinguishable from $\mathbf{Lap}(2/\varepsilon; \mathtt{key})$ if F is a secure pseudorandom function.

Theorem 8 ($\mathbf{Lap}(2/\varepsilon; \mathsf{F}_{\mathsf{sk}}(\mathtt{key}))$ and $\mathbf{Lap}(2/\varepsilon; \mathtt{key})$ Indistinguishability)**.** Let F is a secure pseudorandom function. Let $\mathtt{key} \leftarrow \$\{0,1\}^\lambda$, $f(\varepsilon, \mathsf{sk}, \mathtt{key}) = \mathbf{Lap}(2/\varepsilon; \mathsf{F}_{\mathsf{sk}}(\mathtt{key}))$ and $g(\varepsilon, \mathsf{sk}) = \mathbf{Lap}(2/\varepsilon; \mathtt{key})$. For all $\varepsilon \in \mathcal{R}$, multi-map key $\mathtt{key} \in \{0,1\}^*$ and polynomial time algorithm $\mathcal{A}$ which outputs 0 or 1 as a distinguisher,

$$|\mathbf{Pr}_{\mathsf{sk} \leftarrow \$\{0,1\}^\lambda}[\mathcal{A}(\mathbf{Lap}(2/\varepsilon; \mathsf{F}_{\mathsf{sk}}(\mathtt{key}))] - \mathbf{Pr}_{\mathsf{sk} \leftarrow \$\{0,1\}^\lambda}[\mathbf{Lap}(2/\varepsilon; \mathsf{sk})]| \leq \mathtt{negl}(\lambda).$$

Proof of Theorem 8. Suppose the adversary can distinguish $\mathbf{Lap}(2/\varepsilon; \mathsf{F}_{\mathsf{sk}}(\mathtt{key}))$ from $\mathbf{Lap}(2/\varepsilon; \mathsf{sk})$. We can use this to distinguish $\mathsf{F}_{\mathsf{sk}}(\mathtt{key})$ from sk by simply applying $\mathbf{Lap}(2/\varepsilon; \cdot)$ over the two inputs and calling the distinguisher. This will be a violation of the PRF security of F and the statement is proven. ☐

Security of DP VHEMM. In this section, we give a formal security statement of DP VHEMM and its proof. The proof uses the standard technique of constructing a simulator $\mathcal{S}$ which can produce a transcript that is computationally indistinguishable from a real execution of the scheme.

[2] In [Patel et al.(2019)], the authors did not specify how to compute $\mathbf{Lap}(2/\varepsilon)$ deterministically with $\mathtt{key}$ as the input. We propose to do this with $\mathbf{Lap}(2/\varepsilon; \mathsf{F}_{\mathsf{sk}_3}(\mathtt{key}))$ in our paper. We provide a security definition for $\mathbf{Lap}()$ in Appendix B that is needed for the security of the protocol.

[3] In addition, in [Patel et al.(2019)], the padded query response volume for multi-map $\mathtt{key}$ is computed as $\ell^*(\mathtt{key}) = \ell(\mathtt{key})) + \mathbf{Lap}(2/\varepsilon; \mathsf{F}_{\mathsf{sk}_3}(\mathtt{key})) + c(\lambda_{DP}, \varepsilon)$. This is not mathematically correct as the Laplace distribution is a continuous distribution but query response volumes need to be integral. We add the floor function and show in Lemma 1 that the modified scheme still satisfies the differential-private property proposed in the original paper. In addition, we bound the probability that an adversary can recover the identity of a multi-map key in Theorem 10.

```
$\mathcal{L}_{\text{len}}(\mathsf{pub}, (\mathsf{q}_1, \ldots, \mathsf{q}_m), \mathsf{MM}; \mathbf{st})$
1: $(\alpha, \varepsilon, \lambda_{DP}, c(\cdot)) \leftarrow \mathsf{pub}$
2: if $\mathsf{q}_m \in \mathbf{st}$ then
3:     return $\mathbf{st}[\mathsf{q}_m]$
4: $\ell^* \leftarrow |\mathsf{MM}[\mathsf{q}_m]| + \mathbf{LapD}(0, 2/\varepsilon) + c(\lambda_{DP}, \varepsilon)$ ▷ $\mathbf{LapD}(0, 2/\varepsilon)$ is a mathematical Laplace distribution
   with location 0 and scalar $2/\varepsilon$.
5: $\mathbf{st}[\mathsf{q}_m] \leftarrow \ell^*$
6: return $\mathbf{st}[\mathsf{q}_m]$
```

Figure 6: Response volume leakage function of DP VHEMM.

Theorem 9 (Security of DP VHEMM)**.** Let 1^λ be a security parameter and $\mathsf{pub} = (\alpha, \varepsilon, \lambda_{DP}, c(\cdot))$ be a public parameter. Let DP VHEMM be the encrypted multi-map scheme defined above. Let $\mathsf{F} : \{0,1\}^\lambda \times \{0,1\}^m \to \{0,1\}^n$ be a secure pseudorandom function family, $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ be an IND-CPA secure symmetric encryption scheme with message space $\{0,1\}^*$ and key space $\{0,1\}^\lambda$, and $\mathbf{Lap} : \mathcal{R} \times \{0,1\}^\lambda \to \mathbb{N}$ be a secure Laplace random number generator (see Definition 4) used by the scheme.

Define setup leakage as $\mathcal{L}_{\mathbf{Setup}}(\mathsf{pub}, \mathsf{MM}) = \sum_{i=1}^N l_i$ where N is the number of multi-map keys in MM. For arbitrary queries q_1 up to q_m, define query leakage as $\mathcal{L}_{\mathbf{Query}} = (\mathcal{L}_{\text{QEq}}, \mathcal{L}_{\text{len}})$. $\mathcal{L}_{\text{QEq}}$ is a query equality leakage function $\mathcal{L}_{\text{QEq}}(\mathsf{pub}, (\mathsf{q}_1, \ldots, \mathsf{q}_m), \mathsf{MM}) = \mathsf{QE}$ where QE is a matrix where $\mathsf{QE}_{i,j} = 1$ if $\mathsf{q}_i = \mathsf{q}_j$; $\mathsf{QE}_{i,j} = 0$ otherwise. $\mathcal{L}_{\text{len}}$ is a *stateful* volume leakage function shown in Figure 6.

Then DP VHEMM is SS-CQA-B-BB secure with leakage
$(\mathcal{L}_{\mathbf{Setup}}, \mathcal{L}_{\mathbf{Query}})$.

Proof. Without loss of generality, assume that the multi-map is $\mathsf{MM} = \left\{(\mathtt{key}_i, (v_{i,j})_{j=1}^{l_i})\right\}_{i \in [N]}$ and the (adaptive) queries are $\mathsf{q}_1, \ldots, \mathsf{q}_k$.

Let $\mathcal{S}$ be the simulator we construct. There are two algorithms $\mathcal{S}$ has to simulate, namely the setup algorithm **Setup** and the query algorithm **EQuery**. We show how these can be done below:

- **Setup**: Given the setup leakage $\mathcal{L}_{\mathbf{Setup}}(\mathsf{MM}) = \sum_{i=1}^N l_i$, the simulator constructs two hash tables T_0 and T_1 of size $(1+\alpha)\sum_{i=1}^N l_i$ each. The entries of the hash table are encryptions of the zero string under a randomly picked key.

- **Query**: The simulator is given two leakage components. The first component is the query equality leakage
$\mathcal{L}_{\text{QEq}}(\mathsf{q}_1, \ldots, \mathsf{q}_m, \mathsf{MM}) = \mathsf{QE}$ where QE is a matrix such that $\mathsf{QE}_{i,j} = 1$ if $\mathsf{q}_i = \mathsf{q}_j$; $\mathsf{QE}_{i,j} = 0$ otherwise. The second component is the volume leakage $\mathcal{L}_{\text{len}}(\mathsf{q}_1, \ldots, \mathsf{q}_m, \mathsf{MM}; \mathbf{st}) = \ell^*$. To simulate the transcript, the simulator works as follows. It begins by checking if there exists $j \neq m$ such that $\mathsf{QE}_{m,j} = 1$. Suppose that is false. Then, the simulator generates ℓ^* random accesses on hash tables T_0 and T_1 each and stores the accessed addresses in its state. On the other hand, if there exists $j \neq m$ such that $\mathsf{QE}_{m,j} = 1$. The simulator looks up for the addresses accessed by query q_j in its state and replays the accesses.

We argue that the simulator above produces a transcript that is computationally indistinguishable from a real execution of the scheme. To do this, we use the following sequence of games:

- $\mathbf{Game}_0$ is identical to $\mathbf{Real}^{\text{SS-CQA-B}}_{\text{DP VHEMM},\mathcal{A}}(1^\lambda)$.

- $\mathbf{Game}_1$ replaces the IND-CPA secure ciphertexts in the hash tables with encryptions of the zero string under a random key.

- $\mathbf{Game}_2$ replaces the PRF used to generate the hash table addresses with a random function.

- $\mathbf{Game}_3$ replaces the padded query response volumes (i.e. $\ell^*(\mathtt{key}) = \lfloor \ell(\mathtt{key}) + \mathbf{Lap}(2/\varepsilon; \mathsf{F}_{\mathsf{sk}_3}(\mathtt{key})) + c(\lambda_{DP}, \varepsilon)\rfloor$) by ones generated in the leakage $\mathcal{L}_{\text{len}}()$ (Figure 6).

Game$_0$ and **Game**$_1$ are only negligibly-distinguishable as it breaks IND-CPA of the encryption scheme otherwise.

Game$_1$ and **Game**$_2$ are computationally indistinguishable for two reasons. Firstly, **Game**$_1$ and **Game**$_2$ may be different when the random function used in **Game**$_2$ does not produce a valid cuckoo hashing. However, this happens with negligible probability when $\sum_{i=1}^{N} l_i = \omega(1)$. Secondly, an adversary may be able to distinguish the random function used in **Game**$_2$ from the PRF used in **Game**$_1$. Though this only happens with negligible probability as it breaks PRF security otherwise.

Finally, **Game**$_2$ and **Game**$_3$ are computationally indistinguishable since **Lap**() is a secure Laplace random number generator (see Theorem 8) and it has only been called polynomially many times.

It is easy to see that **Game**$_3$ is identical to
$\text{Ideal}^{\text{SS-CQA-B}}_{\text{DP VHEMM},\mathcal{A},(\mathcal{L}_{\textbf{Setup}},\mathcal{L}_{\textbf{Query}}),\mathcal{S}}(1^\lambda)$ and the proof is complete. □

Generic Bounds on the Success Rate of Query Reconstruction Attack. In this section, we give generic bounds on the probability that an adversary can recover the multi-map key of a query for the setup above. We use these bounds to bound the expectations of the *ql*-functions we consider below. On a high level, the bounds are calculated in two steps. In the first step, we compute the likelihood of the multi-map key of a given query `q` being `key` for any multi-map key `key`. Then, we bound the success rate of an adversary in recovering the multi-map key of a particular query in a sequence of queries.

We begin by deriving the probability mass function of the floored Laplace distribution. For simplicity, we only consider the case where the parameter μ of the Laplace distribution is an integer.

Lemma 1. Let $X \sim \lfloor \mathbf{Lap}(\mu, b) \rfloor$ be a floored Laplace random variable where $\mu \in \mathbb{Z}$. Then the probability mass function (defined over integers) of X is

$$P(X = x) = \frac{1}{2}\exp\left(-\frac{|x-\mu|}{b}\right)\left(1 - \exp\left(\frac{1}{b}\right)\right).$$

Proof of Lemma 1. The probability density function of Laplace random variable $\mathbf{Lap}(\mu, b)$ is defined to be

$$\mathbf{Pr}\left[\mathbf{Lap}(\mu, b) = x\right] = \frac{1}{2b}\exp\left(-\frac{|x-\mu|}{b}\right).$$

This can be easily integrated to give the cumulative distribution function

$$\mathbf{Pr}\left[\mathbf{Lap}(\mu, b) < x\right] = \frac{1}{2} + \frac{1}{2}\mathtt{sgn}(x-\mu)\left(1 - \exp\left(-\frac{|x-\mu|}{b}\right)\right).$$

We are now ready to compute the probability mass function $\mathbf{Pr}\left[\lfloor \mathbf{Lap}(\mu, b) \rfloor = x\right]$ of the floored Laplace distribution. As $\mu \in \mathbb{Z}$ and the support of the floored Laplace distribution only contain integers, we only need to consider the cases $x \geq \mu$ and $x < \mu$.

Case 1: $x \geq \mu$.

$$\begin{aligned}
&\mathbf{Pr}\left[\lfloor \mathbf{Lap}(\mu, b) \rfloor = x\right] \\
=&\mathbf{Pr}\left[x \leq \mathbf{Lap}(\mu, b) < x+1\right] \\
=&\mathbf{Pr}\left[\mathbf{Lap}(\mu, b) < x+1\right] - \mathbf{Pr}\left[\mathbf{Lap}(\mu, b) < x\right] \\
=&\frac{1}{2}\left(1 - \exp\left(-\frac{|x+1-\mu|}{b}\right)\right) - \frac{1}{2}\left(1 - \exp\left(-\frac{|x-\mu|}{b}\right)\right) \\
=&\frac{1}{2}\exp\left(-\frac{|x-\mu|}{b}\right) - \frac{1}{2}\exp\left(-\frac{|x-\mu|}{b}\right)\exp\left(\frac{1}{b}\right) \\
=&\frac{1}{2}\exp\left(-\frac{|x-\mu|}{b}\right)\left(1 - \exp\left(\frac{1}{b}\right)\right).
\end{aligned}$$

Case 2: $x < \mu$.

$$\begin{aligned}
&\mathbf{Pr}\left[\lfloor \mathbf{Lap}(\mu, b) \rfloor = x\right] \\
=&\mathbf{Pr}\left[x \leq \mathbf{Lap}(\mu, b) < x+1\right] \\
=&\mathbf{Pr}\left[\mathbf{Lap}(\mu, b) < x+1\right] - \mathbf{Pr}\left[\mathbf{Lap}(\mu, b) < x\right] \\
=&-\frac{1}{2}\left(1 - \exp\left(-\frac{|x+1-\mu|}{b}\right)\right) + \frac{1}{2}\left(1 - \exp\left(-\frac{|x-\mu|}{b}\right)\right) \\
=&-\frac{1}{2}\exp\left(-\frac{|x-\mu|}{b}\right)\exp\left(\frac{1}{b}\right) + \frac{1}{2}\exp\left(-\frac{|x-\mu|}{b}\right) \\
=&\frac{1}{2}\exp\left(-\frac{|x-\mu|}{b}\right)\left(1 - \exp\left(\frac{1}{b}\right)\right) \quad .
\end{aligned}$$

□

Given an observed query response volume $\ell^*(\texttt{key})$ (the multi-map key `key` is unknown to the attacker), we are interested in the likelihood of it coming from multi-map key $\texttt{key}'$. An adversary that can compute these can simply pick the guess of the multi-map key that is the most rewarding (with the consideration of the ql-function). We show how the likelihood can be bounded and how it relates to the information-theoretical bounds on the success rate of query reconstruction.

We show the following proposition which will be useful to bound the ratio between likelihoods later.

Proposition 1 (Ratio of Probabilities between Laplace Random Variables)**.** Let $X_i \sim \lfloor \mathbf{Lap}(\mu_i, b) \rfloor$ be independent floored Laplace random variables for $i = 1, 2$ where:

- $\mu_i \in \mathbb{Z}\ \forall\ i$,
- $|\mu_1 - \mu_2| = c$.

Then for any $d \in \mathbb{N}$,

$$\mathbf{Pr}\left[X_1 = d\right] \leq \exp\left(\frac{c}{b}\right) \cdot \mathbf{Pr}\left[X_2 = d\right].$$

Proof of Proposition 1. We re-write one of the probabilities into the other.

$$\begin{aligned}
&\mathbf{Pr}\left[X_2 = d\right] \\
=&\frac{1}{2}\exp\left(-\frac{|d-\mu_2|}{b}\right)\left(1 - \exp\left(\frac{1}{b}\right)\right) \\
\geq&\frac{1}{2}\exp\left(-\frac{|d-\mu_1| + |\mu_1-\mu_2|}{b}\right)\left(1 - \exp\left(\frac{1}{b}\right)\right) \\
=&\frac{1}{2}\exp\left(-\frac{|\mu_1-\mu_2|}{b}\right)\exp\left(-\frac{|d-\mu_1|}{b}\right)\left(1 - \exp\left(\frac{1}{b}\right)\right) \\
=&\exp\left(-\frac{c}{b}\right)\mathbf{Pr}\left[X_1 = d\right] \quad .
\end{aligned}$$

To second line is simply a rewriting of the probability using Lemma 1. The third line uses triangle inequality. The fourth line rewrites the first exponential into two components. Finally, the last line converts the terms into the desired form in the proposition. We obtain the inequality in the proposition by re-arranging the terms. □

We are now ready to show the bounds on the posterior probability $\mathbf{Pr}\left[\texttt{key} \mid \ell^*\right]$ where `key` is the guess of the queried multi-map key given that the observed query response volume is ℓ^*. We note that this probability is information-theoretically optimal – no adversary can guess the multi-map key of a query correctly with a probability higher than the posterior probability. As a consequence, we can use the posterior probability to argue about the maximum expected q-leakage in our security notion.

Theorem 10 (Bounds on the Posterior Probabilities)**.** Let $\{\texttt{key}_i\}$ be the set of multi-map keys of the multi-map and their true multi-map key volumes distributed as $\ell(\texttt{key}_i) \sim \lfloor \mathbf{Lap}(\mu_i, b) \rfloor$. Let $Y_i \sim \mathbf{Lap}(0, 2/\varepsilon)$ be the Laplace random noise with location 0 and scale $2/\varepsilon$ used on $\ell(\texttt{key}_i)$. Assume $2/\varepsilon > b$, i.e. the amount of Laplace noise added is larger than the uncertainty of the actual volumes.

Then, we can write the observed query response volumes as $\ell^*(\texttt{key}_i) \sim \lfloor 2\ell(\texttt{key}_i) + Y_i + c(\lambda_{DP}, \varepsilon) \rfloor$, where $c(\lambda_{DP}, \varepsilon)$ is a constant that depends on λ_{DP} and ε.

Given an observed query response volume ℓ^*, the posterior probability $\mathbf{Pr}[\texttt{key}_i \mid \ell^*]$ is defined as

$$\frac{\mathbf{Pr}[\texttt{key}_i]\,\mathbf{Pr}[\ell^* \mid \texttt{key}_i]}{\sum_j \mathbf{Pr}\left[\texttt{key}_j\right]\mathbf{Pr}\left[\ell^* \mid \texttt{key}_j\right]}.$$

This probability is bounded from below by

$$\frac{1}{\sum_j \exp\left(\frac{|\mu_i - \mu_j|}{2/\varepsilon}\right)}$$

and from above by

$$\frac{1}{\sum_j \exp\left(-\frac{|\mu_i - \mu_j|}{2/\varepsilon}\right)}\;.$$

Proof of Theorem 10. The proof is a simple manipulation of likelihood functions. Without loss of generality, we consider the relation between $\mathbf{Pr}[\ell^* \mid \texttt{key}_i]$ and $\mathbf{Pr}\left[\ell^* \mid \texttt{key}_j\right]$ for any choice of i and j.

$$\begin{aligned}
&\mathbf{Pr}\left[\ell^* \mid \texttt{key}_j\right] \\
=&\mathbf{Pr}\left[\lfloor 2\ell(\texttt{key}_j) + \mathbf{Lap}(0, 2/\epsilon) + c(\lambda_{DP}, \varepsilon)\rfloor = \ell^*\right] \\
=&\sum_k \mathbf{Pr}\left[\ell(\texttt{key}_j) = k\right] \cdot \\
&\mathbf{Pr}\left[\lfloor \mathbf{Lap}(0, 2/\epsilon)\rfloor = \ell^* - 2k - c(\lambda_{DP}, \varepsilon)\right] \\
\leq&\sum_k \exp\left(\frac{|\mu_i - \mu_j|}{2/\varepsilon}\right) \cdot \\
&\mathbf{Pr}\left[\ell(\texttt{key}_i) = k\right] \cdot \mathbf{Pr}\left[\lfloor \mathbf{Lap}(0, 2/\epsilon)\rfloor = \ell^* - 2k - c(\lambda_{DP}, \varepsilon)\right] \\
=&\exp\left(\frac{|\mu_i - \mu_j|}{2/\varepsilon}\right)\sum_k \\
&\mathbf{Pr}\left[\ell(\texttt{key}_i) = k\right] \cdot \mathbf{Pr}\left[\lfloor \mathbf{Lap}(0, 2/\epsilon)\rfloor = \ell^* - 2k - c(\lambda_{DP}, \varepsilon)\right] \\
=&\exp\left(\frac{|\mu_i - \mu_j|}{2/\varepsilon}\right)\mathbf{Pr}\left[\ell^* \mid \texttt{key}_i\right]\;.
\end{aligned}$$

The second line of the equation is a simple rewrite of the first one. The third line expands the probability into a convolution. The fourth line uses Proposition 1 to upper-bound the likelihood. The last two lines are rewrites of the expression in reverse.

By using this relation on the posterior probability directly, we obtain the desired upper bound.

The lower bound can be derived in the same way by using the relation

$$\mathbf{Pr}\left[\ell(\texttt{key}_j) = k\right] \geq \exp\left(-\frac{|\mu_i - \mu_j|}{2/\varepsilon}\right)\mathbf{Pr}\left[\ell(\texttt{key}_i) = k\right]$$

for any k. We omit the details here.

$\square$