Early Detection and Analysis of Leakage Abuse Vulnerabilities

Charles V. Wright and David Pouliot

Portland State University {cvwright,dpouliot}@cs.pdx.edu

October 27, 2017

Abstract

In order to be useful in the real world, efficient cryptographic constructions often reveal, or "leak," more information about their plaintext than one might desire. Up until now, the approach for addressing leakage when proposing a new cryptographic construction has focused entirely on qualifying exactly what information is leaked. Unfortunately there has been no way to predict what the real-world impact of that leakage will be.

In this paper, we argue in favor of an analytical approach for quantifying the vulnerability of leaky cryptographic constructions against attacks that use leakage to recover the plaintext or other sensitive information. In contrast to the previous empirical and ad-hoc approach for identifying and assessing such vulnerabilities, analytical techniques can be integrated much earlier in the design lifecycle of a new construction, and the results of the analysis apply much more broadly across many different kinds of data.

We applied the proposed framework to evaluate the leakage profiles of five recent constructions for deterministic and order-revealing encryption. Our analysis discovered powerful attacks against every construction that we analyzed, and with only one possible exception, the attack allows the adversary to recover virtually any plaintext with only an exponentially small probability of error. We hope that these results, together with the proposed analytical framework, will help spur the development of new efficient constructions with improved leakage profiles that meaningfully limit the power of leakage abuse attacks in the real world.

1 Introduction

In order to be useful in the real world, efficient cryptographic constructions often reveal, or "leak," more information about their plaintext than one might desire. The leakage can be as simple as the length of a message, or the number of messages in some interval. Other schemes intentionally leak more about the messages themselves, e.g. revealing which messages are duplicates of one another or the relative ordering of the messages. This leakage can sometimes be exploited by adversaries in order to learn something—or in some cases, everything—about the plaintext. Unfortunately, progress toward understanding the real-world impact of leakage on security and the power of inference attacks [22] that it enables—sometimes also called *leakage abuse attacks* [12]—has been haphazard and ad-hoc.

For designers of efficient cryptographic constructions, the predominant approach to characterizing leakage has been entirely qualitative, meaning that one first describes exactly what information is revealed by any new proposed leaky scheme, and then one proves that nothing more is leaked. Up until now, designers have lacked quantitative methods for evaluating or understanding the impact of the leakage. It has been mostly through empirical studies demonstrating specific attacks on specific data sets [22, 31, 12, 17, 36, 19] that understanding of the impact of leakage has advanced.

As a result, the value of the qualitative analysis for predicting the real-world security of any new construction has been limited. In many ways, recent history in this area echoes the development of cryptographic protocols before the advent of provable security. In some cases, devastating inference attacks have gone undiscovered for several years after the publication of a new construction. In other cases, similar attacks have appeared quickly. The absence of known leakage attacks does not guarantee that no such vulnerabilities exist; a new attack might be discovered at any time. Similarly, the existence of one known attack does not mean that there are not also other, more serious vulnerabilities waiting to be discovered.

In practice, the problem is compounded even further by the complexities of dealing with noisy, random data from the real world. Just because an attack works well (or poorly) on one set of data does not necessarily mean that it will perform similarly on a different dataset. Designers of systems built on leaky constructions have used this fact to argue that schemes which have been demonstrated to be insecure with one kind of data may still be safe for use with other kinds of data [35]. Although we should be skeptical of such claims, evaluating them in a meaningful way can be very difficult and time-consuming when the only mechanism to do so involves first collecting representative data samples and then performing experiments for each known attack and each new type of data in question. To make matters worse, in some cases, no representative data sets are publicly available for such work.

A more important weakness of the current empirical approach is that it is only useful for finding flaws in insecure constructions. No amount of experiments can ever be sufficient to verify that a given leakage profile is secure. Without a more systematic approach to understanding the problem, it is almost impossible to evaluate the real-world value of any new leaky scheme.

1.1 Contributions

In this paper, we propose a new, systematic, analytical approach to the study of leakage abuse and other inference vulnerabilities. We frame the leakage abuse attack as a standard statistical inference problem, where the observable leakage is a (probabilistic) function of some protected information that the adversary seeks to recover. In doing so, we can evaluate the impact of the leakage analytically rather than empirically. This approach brings a number of advantages.

First, we can use well known results on the performance of various statistical techniques that have been developed over many years in the probability and statistics literature. For example, in many cases we can easily identify the *optimal* attack strategy for the adversary. Specifically, for recovering discrete-valued secrets, we know that maximum *a posteriori* (MAP) estimation is optimal in that it minimizes the chance that the adversary guesses incorrectly. Similarly, for real-valued secrets, maximum likelihood estimation (MLE) minimizes the expected error in the adversary's guess. In either case, we can use the optimal strategy to construct a concrete attack on a given construction, and then we know that there can be no other, more powerful attack using the same leakage.

Second, we argue that a more principled, analytical approach helps to make the study of inference attacks more accessible to a broader community. Up until now, evaluating the security of a construction against inference attacks has required basic proficiency not only in applied cryptography but also in statistics, optimization, and machine learning, as well as a fair amount of intuition and luck. By clearly delineating all the steps in the identification and evaluation of potential vulnerabilities, we hope to make our framework useful to designers of new leaky constructions. Then potential problems may be detected, analyzed, and, if necessary, remedied much earlier in the design process. Eventually, proofs of security against leakage abuse attacks might even become standard practice when proposing any new "leaky" scheme. Of course, this paper is only the first step (out of many) toward such a goal.

In this work, we show how our analytical framework can be used to identify leakage abuse attacks on leaky constructions and to predict their impact without the need for time-consuming data collection and experimentation. In principle, this analytical framework could also be used for evaluation of network protocols or any other application of leaky cryptography. Because encrypted databases have been a hot area for inference attacks in recent years, we focus our evaluation on recent encryption schemes intended to support outsourced database-style queries, namely deterministic encryption (DET) and various flavors of order-revealing encryption (ORE).

Overall, our analysis shows that current schemes leave much room for improvement in security. Every scheme we have analyzed is vulnerable to either online or offline attacks that enable the adversary to recover the plaintext with high probability. As far as we know, the vulnerabilities and attacks that our analysis uncovered for POPE [37] and LW16 [30] are the first known attacks on these constructions. For all but one scheme that we analyzed (POPE), the effectiveness of the adversary's attack is limited only by (1) the distribution of the target attribute, (2) the quality of the adversary's auxiliary information, and (3) the number of records in the encrypted database. In the most common cases, the adversary's chance of success increases with the size of the encrypted database, and he succeeds with all but an exponentially small probability as the number of records increases. This result is somewhat ironic, since the original goal of deterministic and order-revealing encryption was to trade off a little security in order to efficiently handle queries on very large databases. It is unfortunate then that these schemes can only hope to achieve security on smaller databases, where more conservative approaches like oblivious RAM [38] and standard symmetric searchable encryption [15, 13] may also offer acceptable performance with stronger security.

1.2 Assumptions

Our analysis relies on a set of standard statistical assumptions and conservative cryptographic assumptions. We model records in the database as an independent and identically distributed (i.i.d.) sample from a discrete probability distribution over a finite set of plaintexts. From the attacker's perspective, this assumption is also conservative: if the records in some real application are not independent, then the adversary could leverage their relatedness to craft an even more powerful attack.

Cryptographically, we consider only the ideal version of each type of construction, where no partial information is leaked beyond the intended leakage profile. In practice, real schemes may reveal substantially more [10]. In this paper, we limit our analysis to attacks that consider only a single database column, and we show that these are sufficient to demonstrate the insecurity of current schemes. In reality, attackers can also leverage relationships between encrypted columns to recover information, even when each column by itself leaks nothing [17].

1.3 Outline

We begin with a review of related work in leaky encryption schemes, leakage abuse and inference attacks, and analytical frameworks in Section 2. We describe our framework for reasoning about leakage and the attacks that leverage it in Section 3.

We begin our analysis in Section 4 by introducing two abstractions that we will use to simplify our analysis of several schemes. We call these abstractions the *biased coin game* (BCG) and the *loaded dice game* (LDG). In Section 5 we use the BCG to derive a very simple offline data recovery attack against deterministic encryption (DET). This simple attack illustrates the vulnerability of the scheme without the need for more sophisticated statistical machinery like those used in earlier, empirical works like [31].

In Section 6 we evaluate several flavors of order-preserving and order-revealing encryption. In Section 6.1 we show how the simple attack on DET generalizes to an attack on ideal order revealing encryption (ORE) [33, 11]. In addition we describe a new, optimal offline attack on ORE based on the loaded dice game abstraction. In Section 6.2, we adapt the ORE attack to construct an online query recovery attack against the left-right ORE construction of Lewi and Wu [30], which is semantically secure against offline attacks. In Section 6.3 we show how the theory of order statistics leads naturally to an optimal data recovery attack on the ideal version of randomized, frequency-hiding ORE schemes like [24]. We also show that the bucketing attack of [19] gives an efficient alternative that is also nearly guaranteed to guess the correct plaintext for most encrypted records. In Section 6.4 we show how the bucketing attack can be generalized to handle the partial order preserving encoding (POPE) scheme of Roche et al [37]. Finally, in Section 6.5 we adapt the attack on randomized ORE to give an alternate analysis of standard ORE with a new, exponential lower bound on the adversary's accuracy.

We conclude the paper in Section 7 with some thoughts on directions for future work and some recommendations for how our analytical framework might be used to improve the security of leaky constructions going forward.

2 Related Work

Efficiently Queryable Database Encryption Deterministic encryption was first studied formally by Bellare, Boldyreva and O'Neill [8]. Order-preserving encryption (OPE) was introduced by Agrawal, Kiernan, Srikant and Xu [4] and first studied formally by Boldyreva, Chenette, Lee and O'Neill [9]. Popa, Li and Zeldovich [33] proposed an interactive order-revealing encryption (ORE) protocol which was improved by Kerschbaum and Schroepfer [25]. Non-interactive ORE was first achieved by Boneh, Lewi, Raykova, Sahai, Zhandry and Zimmerman [11] and improved in recent works [30, 14]. While ORE reveals only the order of two ciphertexts, OPE is known to leak more information about the plaintexts [9, 10, 17]. Recently, Kadhem, Amagasa, and Kitagawa [23] and Kerschbaum [24] proposed randomized ORE constructions, and Roche et al proposed a partial order revealing construction, POPE [37].

Efficient Encrypted Database Systems The CryptDB system of Popa, Redfield, Zeldovich and Balakrishnan [34] first demonstrated the use of a combination of efficiently queryable encryption techniques to support a large subset of SQL over encrypted data on a standard database server. Similar approaches from industry include Google's encrypted BigQuery (EBQ) demo [2], SAP's

SEEED project [3] and Microsoft SQL Server's Always Encrypted (AE) [1]. In a similar vein, Lau, Chung, Chengyu, Jang, Lee and Boldyreva [29] and He, Akhawe, Jain, Shi and Song [20] developed constructions for easily deployable encryption supporting efficient keyword search.

Leakage abuse and inference attacks Islam, Kuzu and Kantarcioglu described the first inference attack on encrypted databases [22]. The IKK attack was a query-recovery attack on a common leakage profile found in searchable symmetric encryption (SSE) schemes. Later work by Cash, Grubbs, Perry and Ristenpart [12] showed that the effectiveness of the IKK attack may be more limited in practice. Inference attacks against deterministic and order-revealing encryption were first considered by Naveed, Kamara and Wright [31]. Pouliot and Wright gave inference attacks [36] against the constructions of Lau et al. and of He et al. [29, 20], improving on previous active attacks described in [12]. The NKW attacks have also been improved recently by Grubbs, Sekniqi, Bindschaedler, Naveed and Ristenpart [19] and by Betul Durak, DuBuisson and Cash [17].

Statistical Cryptanalysis Baigneres, Junod, and Vaudenay [6], working in the context of linear cryptanalysis on symmetric primitives, analyzed the adversary's chance of success in distinguishing between two distributions D_0 and D_1 that are statistically close. Working from the Neyman-Pearson lemma [32], they show that the number of samples that the adversary needs is inversely proportional to the Kullback-Leibler divergence [28], sometimes called the *relative entropy*, of D_0 and D_1. In the context of our biased coin game, their result gives us the adversary's expected accuracy for a game with two coins. Our analysis in Section 4.1 shows that the adversary can still win with high probability even in games with more than two coins.

Analytical Frameworks The most similar analytical framework to ours is Kifer and Machanavajjhala's pufferfish framework [26] for reasoning about privacy. The framework of coupled-worlds privacy from Bassily, Groce, Katz, and Smith [7] is similar. Both of these give extensions to Dwork's notion of differential privacy [18], and they focus on evaluating the privacy of irreversible database redaction mechanisms rather than the secrecy of reversible encryption schemes as in our work.

3 Analytical Framework

Our approach is based on the observation that inference attacks on leaky cryptographic constructions are at their core nothing more than standard statistical inference problems. We model the secret "target" data that the adversary wishes to recover as a random sample from some probability distribution. The leakage that the adversary observes is a function of the secret data, so therefore we represent the leakage also as a random sample, drawn from a distribution which is parameterized in part by the target data. The adversary's task is to make an informed guess about the encrypted target data, given the observed leakage and his auxiliary information. Our framework is general and can be used to design and analyze inference attacks against any leaky cryptographic construction.

By framing the attack as a statistical inference problem, we can make use of a rich literature that has been developed over many years in the probability and statistics communities. In particular, this approach makes it easy to identify the adversary's optimal guessing strategy for many classes of problems. This also makes it feasible to predict the success of inference attacks analytically for whole classes of data, where previously our understanding of the power of these attacks was ad-hoc and limited to empirical results obtained for particular data sets.

Furthermore, because we can now identify and analyze *optimal* attack strategies that give the adversary the best possible chance of success, we can begin to talk about proving security against inference attacks in a meaningful way. If we can bound the edversary's success rate analytically, then an upper bound gives a proof of security—no adversary can hope to do any better with the same information.

On the other hand, if we can establish an analytical lower bound for *any* inference attack, even if it is a particularly poor one, then we have a proof of *in*security for the given construction. It follows that, no matter what the source of the target data, the adversary is still likely to win. Such a construction merits no further evaluation, only (perhaps) additional refinement to reduce its leakage to a safer level.

Overview of the framework Our framework gives a principled approach for discovering and evaluating leakage abuse vulnerabilities. The process that we propose is roughly the same as that of any other statistical inference problem. In our experience, the designers of attacks tend to take a more heuristic approach, often lumping together several steps into one. We argue that a more careful, methodical approach is warranted. Among other things, this makes it easier to identify optimal attack strategies and to establish analytical bounds on the attacks' performance. We formalize the process of discovering and evaluating a new inference vulnerability in five discrete steps.

I. Write down a parametric model that describes how the leakage is generated as a probabilistic function of the hidden target data (eg, the plaintext). As in all statistical inference problems, this step requires a certain amount of expertise, and it may involve some difficult tradeoffs. However, in practice the task is often relatively straightforward. In the following sections, we give a variety of examples using our current attacks as well as several others from recent work.

II. Write down a concrete description of the information that the adversary seeks to extract from the encrypted data. More formally, write this as a set $\mathcal{H}$ of competing hypotheses about the target data. For example, if the adversary's goal is to recover the plaintext for a given ciphertext c, he would define his hypotheses as the set $\mathcal{H} = \{H_i\}$ where $H_i : \mathsf{Dec}(k, c) = m_i$ for each plaintext $m_i \in \mathbb{M}_k$.

III. Using the relationships exposed in the parametric model above, select an objective function $U : \mathcal{H} \to \mathcal{R}$ for use in evaluating the adversary's hypotheses and choosing the best one, $\hat{H} = \arg\max_{H \in \mathcal{H}} U(H)$.

IV. Find an algorithm for efficiently finding the hypothesis in $\mathcal{H}$ that maximizes the objective function U. This maximization is a often combinatorial optimization problem. Many such problems are in NP, so it may be possible in the future to prove a lower bound on the hardness of attacking some constructions. However, for the adversary it typically suffices to find an approximate solution, and efficient approximate solvers for many of the most important problems are readily available in off-the-shelf software.

V. Use the parametric model derived in step I to predict various measures of the adversary's success. If it is possible to establish an upper bound on the success rate of the optimal adversarial strategy, then such a bound serves as a proof of security. On the other hand, if we can establish a lower bound on the success of some attack, then we have a negative result: we know that the construction is insecure, without the need for any further analysis. More generally, we can use analytical bounds on the adversary's success metrics to compare constructions in a principled way, for example to evaluate trade-offs between performance and security.

We elaborate on each of the steps in the framework in the following sections.

3.1 Parametric models for leakage

The first step in our framework is to design or select a formal probabilistic model that captures the relationship between the leakage that the adversary can observe and the hidden information that he wants to recover: the encrypted records, the queries, or the key, etc.

In this paper, we use a general and very conservative statistical model that makes the adversary's task as hard as reasonably possible. We assume that the plaintext records in the database $X = (X_1, X_2, \ldots, X_n)$ are an independent and identically distributed, or *i.i.d.*, sample from a Categorical distribution over a discrete set of plaintexts $\mathcal{M} = (m_1, m_2, \ldots, m_M)$. In a Categorical distribution, each message m_i occurs with probability p_i, where $\sum_{i=1}^{M} p_i = 1$.

If the plaintext records are not independent, then the adversary could use some knowledge about their interrelationships to improve his guessing accuracy. Similarly, if the adversary knows something more about the shape of the distribution, for example that it has a bell-curve shape (e.g. Poisson) or a power law shape (e.g. Zipf), then he may be able to estimate the probability of each plaintext more accurately from a smaller amount of auxiliary information.

The leakage profile of the particular construction will determine how the leakage is related to the plaintext. For example, deterministic encryption reveals the (unlabeled) set of sample frequencies of the plaintexts in the encrypted column, Order-revealing encryption leaks the frequencies as well as the number of records greater than and less than each plaintext.

3.2 Defining hypotheses for inference

The second step in designing or analyzing an inference attack is to clearly define what information the adversary seeks to extract from the target data.

In an offline attack on an encrypted database, the adversary uses leakage from the encrypted database and outputs a hypothesis about some encrypted record(s). For example, the adversary might want to guess the plaintext for some ciphertext c_j; we call this a data recovery attack. Then his candidate hypotheses would take the form $H_i : \mathsf{Dec}(k, c_j) = m_i$.

In an online attack, the adversary uses leakage both from the encrypted database itself and from one or more queries. His hypotheses can be about either the database or the queries, or both. For example, in a query recovery attack, the adversary observes an encrypted query q_j. Given a set of candidate keywords W, he attempts to determine which keyword $w \in W$ is in the query. His hypotheses take the form $H_i : \mathsf{Dec}(k, q_j) = w_i$.

3.3 Selecting an objective function.

After defining the relationship between the leakage and the encrypted data, and defining the information that he wants to extract, the adversary's next step is to define an *objective*, or "goodness," function to evaluate his candidate hypotheses and choose the best one. In practice, this is usually one of the most heuristic and ad-hoc steps. Attack designers may even skip it entirely and instead jump directly to Step IV, selecting an attack technique based on intuition or familiarity without necessarily thinking or worrying too much about what the chosen attack algorithm is actually maximizing. We argue that a more principled approach is warranted.

In this paper, we show how the adversary can use two standard objective functions to craft powerful attacks. Maximum *a posteriori* (MAP) estimation and maximum likelihood estimation (MLE) are standard statistical inference techniques.

Recommended Workflow In general, we propose the following approach for evaluating a new leaky scheme. First, ask *Can it be provably broken by a very simple heuristic attack?* If so, then the designer must go "back to the drawing board" and attempt to reduce the leakage profile. (Note: At the time of this writing, we are here. In Sections 5 and 6, we show the existence of highly effective inference attacks against every construction that we have analyzed.)

If the simplest attacks are insufficient to break a new construction, then it is worth considering the adversary's optimal attack strategy and obtaining a (loose) lower bound on its effectiveness. If the optimal attack does not guarantee the adversary an unacceptable level of success, then the system designer should pursue an upper bound on the optimal attack's success rate or some other proof of security.

We acknowledge that this approach feels somewhat backwards compared to the typical provable security approach, where constructions are designed from the start to be as secure as possible. However, we must remember that leakage exists in efficient constructions out of necessity, in order to satisfy some practical real-world design constraint. Therefore any effort to remove or minimize it will likey involve significant trade offs and increased overhead in terms of computation, storage space, and/or network traffic.

3.3.1 Maximum likelihood estimation (MLE)

Maximum likelihood estimation (MLE) says that we should choose the hypothesis that makes the observed data most probable. More formally, given some observed event E and a set of hypotheses $\mathcal{H} = \{H_i\}$ competing to explain the event, MLE says that we should choose the hypothesis that maximizes the *likelihood function.* The likelihood of the hypothesis H given the event E is simply the probability of the event given the hypothesis:

$$\begin{aligned}\hat{H}_{MLE} &= \arg\max Ł(H_i|E) \\ &= \arg\max \Pr\left[\, E|H_i \,\right]\end{aligned}$$

Because the MLE objective function does not include the priors, it is often less challenging to maximize compared to MAP. Maximum likelihood also has a number of nice statistical properties of its own. In particular, it is an asymptotically efficient estimator, meaning that as the sample size increases toward infinity, MLE achieves the minimum possible mean squared error (the Cramer-Rao lower bound). Thus, for attacks on numeric data, MLE minimizes the expected difference between the target data and the adversary's guess.

3.3.2 Maximum a posteriori (MAP)

Maximum *a posteriori* (MAP) estimation can be viewed as a Bayesian extension of the maximum likelihood approach. MAP says that we should choose the hypothesis that has the highest probability under the *posterior* distribution. The posterior probability of hypothesis H given event E incorporates both the observed evidence as well as a Bayesian prior for the hypothesis.

$$\hat{H}_{MAP} = \arg\max \Pr[\, H_i | E \,] * \Pr[\, H_i \,]$$

MAP estimation is especially useful for attacks on cryptographic constructions because, for discrete-valued data, MAP minimizes the probability that the adversary guesses incorrectly. In other words, given the same data, there can be no other attack that guesses more accurately than MAP. An interesting direction for future work might be to prove security for leaky constructions by establishing an upper bound on the MAP adversary's success.

3.4 Implementation

To have practical impact, an inference attack must have an efficient implementation. This may pose a challenge for the adversary when the maximization of his objective function is an NP problem. In practice, however, many of the most relevant problems have been well studied in the statistics and operations research literature, and even NP optimization problems often have good polynomial time approximate solutions.

For example, Islam, Kuzu, and Kantarcioglu [22] prove that their optimization problem is in NP, but they use simulated annealing to find a solution for realistic problem sizes in a matter of hours. Pouliot and Wright reduce a similar frequency matching attack to a weighted graph matching problem [36]. Because weighted graph matching is in NP, they rely on approximation techniques to derive a best guess for the permutation matrix P that maps plaintexts to ciphertexts. In [31], Naveed, Kamara, and Wright reduce their frequency matching attack to a linear sum assignment problem, which can be solved exactly in cubic time using either integer linear programming or the Hungarian algorithm [27]. Later work from Grubbs et al [19] improves on the NKW attack on OPE by reducing it to a maximum weight non-crossing matching problem, which can be solved exactly in quadratic time.

3.5 Analytical Evaluation

For any cryptanalytic task, it is important to precisely define what is considered a successful attack. The most conservative notion of success for an inference attack would be to show the existence of an attack and of auxiliary data such that, given an encrypted column and the auxiliary dataset, the adversary can recover a single bit of the plaintext column with non-negligible advantage; that is, with non-negligible probability in k over $1/2$. While such an attack would certainly constitute a break from a cryptanalytic perspective, practitioners may question whether the auxiliary dataset can be found in practice and whether the recovery of one bit is really damaging enough.

We stress that the attacks analyzed in this work, like earlier inference attacks [22, 31, 12, 19], satisfy a much stronger notion of success. In fact, we show that they can recover entire plaintext messages with high probability given the encrypted column and given auxiliary data that is sampled from any distribution that is statistically-close to the message distribution.

Specifically for data recovery attacks let $\hat{X} = (\hat{X}_1, \ldots, \hat{X}_n)$ be the random variables representing the adversary's best guess for the plaintext records $X = (X_1, \ldots, X_n)$. Previous works on inference attacks have focused on simple measures like the adversary's overall guessing accuracy.

Accuracy is the probability that the adversary's inference is correct.

$$\mathsf{Acc} = \Pr\left[\hat{X}_i = X_i\right]$$

However, accuracy is a somewhat crude metric for the adversary's success, because he can achieve high accuracy by guessing only a small number of very common plaintexts. For example, the NKW attacks achieved greater than 80% accuracy for the "length of stay" attribute in a hospital data set by correctly guessing only the three most common values ("1 day," "2 days," "3 days") out of a total of 365. Given previous work, it might be tempting to conclude that deterministic and order-revealing encryption schemes provide reasonable security for rarer values, for example a hospital stay of 7 or 14 days. On the contrary, in the following sections, we show that these schemes admit inference attacks that are much more powerful. To capture the power of these attacks, we give bounds on the adversary's effectiveness in terms of his accuracy for each record in the database (c.f. Section 6.3) or in terms of his true positive rate and false positive rate for each plaintext $m \in \mathcal{M}$.

True Positive Rate for plaintext m is the probability that the adversargy guesses m when it appears in the target data.

$$\mathsf{TPR}_m = \Pr\left[\hat{X}_i = m \mid X_i = m\right]$$

False Positive Rate for plaintext m is the probability that the adversary guesses m when some other plaintext appears.

$$\mathsf{FPR}_m = \Pr\left[\hat{X}_i = m \mid X_i \neq m\right]$$

In the following sections, we use our analytical framework to establish lower bounds on the adversary's accuracy, TPR, and FPR for attacks on deterministic and order-revealing encryption. We show that in the general case, as the number of records in the encrypted database increases, the lower bounds for each measure of the adversary's success approaches one. We begin in the next section by defining two game abstractions to help with the analysis of DET and ORE.

4 Games with Biased Coins and Loaded Dice

In the following sections we will show how the analysis of common leakage profiles can be reduced to well known statistical inference problems involving biased coins and loaded (unfair) dice. We are not aware of standard names for these problems, but they are sometimes assigned as homework in advanced undergraduate courses in statistics and machine learning. Here we refer to them simply as the biased coin game (Section 4.1) and the loaded dice game (Section 4.2), which is a simple generalization of the biased coin game. In each game, the player (ie, the adversary) is challenged to identify one of several unfair coins (dice) by observing its behavior in a set of trials.

In this section, we define the games formally, and we give the optimal solution strategy as well as several simpler heuristic strategies for playing each game. We use the heuristic strategies to prove lower bounds on the adversary's success rate when he uses the optimal strategy. Then in the following sections, we show how the exploitation of leakage from recent encrypted database schemes can be reduced to either the BCG or the LDG, so that the adversary can use the same strategies to break their secrecy with the same (high) chance of success.

4.1 The Biased Coin Game

The BCG is a game between a challenger and an adversary that works as follows. The challenger holds m biased coins such that the i^{th} coin lands heads with probability p_i. The challenger chooses one coin according to some prior distribution π, where coin i is selected with probability π_i. He tosses the selected coin n times and records the outcomes of the trials in an n-bit vector $\mathbf{s}$ with a 1 for heads and 0 for tails. Finally, he provides the adversary with the prior probabilities $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_m)$, the probabilities of the coins $\mathbf{p} = (p_1, \ldots, p_m)$, and the sample $\mathbf{s}$. The adversary outputs its guess as to which coin the challenger picked. The adversary wins if it guesses the correct coin. We refer to this as an (m, n)-biased coin game. In practice, real-world attacks do not reduce exactly to the BCG but to a variant where the adversary is only provided with *approximations* of the coins' probabilities and priors, $\widetilde{\mathbf{p}}$ and $\widetilde{\boldsymbol{\pi}}$, where for all $1 \leq i \leq m$,

$$p_i - \delta \leq \widetilde{p}_i \leq p_i + \delta$$

and

$$\pi_i - \delta \leq \widetilde{\pi}_i \leq \pi_i + \delta.$$

We refer to this as the (m, n, δ)-approximate BCG.

A Statistical Model of Leakage Strategies for winning the BCG rely on the basic observation that the data revealed to the adversary (the sequence of coin tosses) are a random function of the hidden information that he aims to recover (the identity of the coin). Specifically, the outcome of each coin toss is distributed according to a Bernoulli distribution with parameter $\alpha \in \mathbf{p}$, where α is the bias of the selected coin. Therefore the total number of heads

$$h \overset{def}{=} \sum_{i=1}^{n} s_i,$$

is distributed according to a Binomial distribution with parameters (n, α). The linkage between these two distributions will serve as the basis for our attacks in the next section where, intuitively, the Bernoulli models the message distribution and the Binomial models the leakage distribution.

Defining Hypotheses It should be obvious that the BCG adversary's task is to choose from a set of m competing hypotheses, where $\mathcal{H} = \{H_i : \text{Coin } i \text{ was selected}\}$.

Choosing the Best Hypothesis As we discussed in Section 3.5, the strategy that maximizes the adversary's chance of guessing correctly is maximum *a posteriori* estimation.

$$
\begin{aligned}
\hat{H}_{MAP} &= \arg\max_{H_i \in \mathcal{H}} \ \Pr[\, H_i \mid \mathbf{s} \,] \\
&= \arg\max_{H_i \in \mathcal{H}} \ \frac{\pi_i \Pr[\, \mathbf{s} \mid H_i \,]}{\sum_j \pi_j \Pr[\, \mathbf{s} \mid H_j \,]} \\
&= \arg\max_{H_i \in \mathcal{H}} \ \pi_i \Pr[\, \mathbf{s} \mid H_i \,] \\
&= \arg\max_{H_i \in \mathcal{H}} \ \pi_i \ (p_i)^h \ (1 - p_i)^{n-h}
\end{aligned}
$$

The last step follows from the definition of the Binomial distribution.

In order to make our analysis easier, we also consider the simpler MLE strategy. The maximum likelihood hypothesis is the one that maximizes the probability of the observed data.

$$
\begin{aligned}
\hat{H}_{MLE} &= \arg\max_{H_i \in \mathcal{H}} \ \Pr[\, \mathbf{s} \mid H_i \,] \\
&= \arg\max_{H_i \in \mathcal{H}} \ (p_i)^h (1 - p_i)^{n-h}
\end{aligned}
$$

Implementing the Optimization For each of the three guessing strategies described above, it suffices for the adversary to compute the utility function for each of the m coins and take the one that scores best. This takes $O(m)$ time.

Evaluation Winning the biased coin game is not typically a hard problem. With a little analyis from the fundamental principles of statistics, we can now show that most instances of the BCG can be won with surprisingly high probability, even if the adversary uses a sub-optimal guessing strategy.

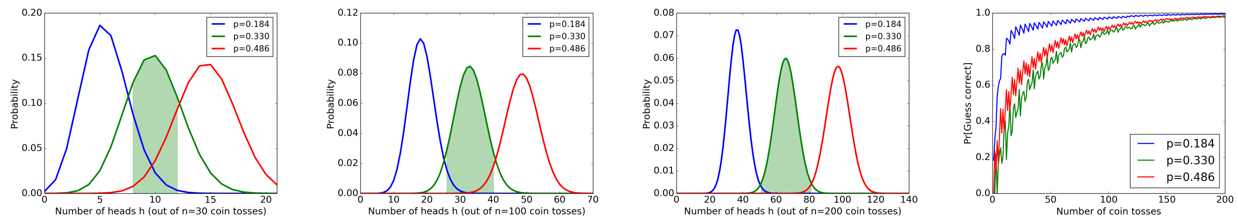

Figure 1: Expected results for biased coin experiments

To help develop the reader's intuition, in Figure 1 we plot the probability of obtaining a certain number of heads when tossing three biased coins having $\mathbf{p} = (0.184, 0.33, 0.486)$. The first three plots show the expected "bell curve" shape for the Binomial PDF; for each coin i, the probability that the number of heads h is near its expected value of $p_i n$ is relatively large. The probability drops off quickly as h diverges from its expected value.

These plots also suggest an intuitive explanation for the maximum likelihood (MLE) guessing strategy. Given that h heads resulted from n tosses of the coin, the adversary should look to the point where $x = h$ on the graph and pick whichever coin's curve is on top. For example, the shaded region shows where the adversary should choose the middle coin (the one having $p = 0.33$).

Recall from the definition of a probability distribution function that the total area under each curve is equal to 1. Then the area of the shaded region gives the probability that the adversary is successful in identifying the middle coin. As the number of coin tosses, n, increases, the shaded region comprises a larger fraction of the area under the curve. In other words, the adversary's chance of winning increases. The fourth plot shows how the adversary's chance of success increases for the three coins as n grows from 1 to 200. For each coin, the adversary's sucess rate approaches 100%.

Given basic probability and statistics, this result should not be surprising. The law of large numbers says that, given sufficiently many trials, the average result from a series of experiments should be very close to its expected value. More concretely, two famous results from Hoeffding [21], known as Hoeffding's inequality and the Chernoff-Hoeffding theorem, give upper bounds on the probability that h will be outside the shaded winning region for a given coin. In this paper, we use the simpler of Hoeffding's two inequalities to obtain bounds on the adversary's expected performance, because it gives an expression that is algebraically simpler than one derived from the tighter Chernoff-Hoeffding bound.

Theorem 1 (Hoeffding's Inequality [21])**.** *Let $H(n)$ be the number of heads obtained on n tosses of a biased coin with probability of heads equal to p, and let $\varepsilon > 0$. Then*

$$\Pr\left[\, H(n) \geq (p+\varepsilon)n \,\right] \leq \exp(-2\varepsilon^2 n)$$

and

$$\Pr\left[\, H(n) \leq (p-\varepsilon)n \,\right] \leq \exp(-2\varepsilon^2 n).$$

The Boundary Point In order to apply Hoeffding's inequality to obtain a bound on our adversary's success rate, all that remains is to determine the correct ε that corresponds to the boundary of the winning region. In this version of the paper, we show how this can be done for the maximum likelihood strategy, and we leave the similar analysis of the MAP strategy for a future version of the paper. (Again, recall that since MAP is the optimal strategy, any lower bound on any other strategy's performance is also a lower bound on the success rate of the MAP strategy.)

Definition 1 (Binomial Boundary Point)**.** *For probabilities p_0 and p_1 with $p_0 < p_1$, we define the* boundary point *$\phi(p_0, p_1)$ as the point where*

$$\binom{n}{k} p_0^k (1-p_0)^{n-k} > \binom{n}{k} p_1^k (1-p_1)^{n-k}$$

for all $n > 0$ and all $k \in [0, \phi(p_0,p_1)n)$, and

$$\binom{n}{k} p_0^k (1-p_0)^{n-k} < \binom{n}{k} p_1^k (1-p_1)^{n-k}$$

for all $n > 0$ and all $k \in (\phi(p_0,p_1)n, n]$.

We give a closed-form expression for the boundary point and its derivation in Appendix A. For purposes of this section, it suffices to state that $\phi(p_0, p_1)$ is a function of only p_0 and p_1, independent of the number of samples n, and that the boundary always falls in the range between the two probabilities.

True Positive Rate Let α denote one of the challenger's m coins, and let $p \in \mathbf{p}$ be the heads probability of coin α, let $q \in \mathbf{p}$ be the largest coin probability smaller than p, and let $r \in \mathbf{p}$ be the smallest coin probability greater than p. Then the adversary will guess that the challenger tossed coin α whenever the number of heads, h, falls in the open interval $\big(\phi(q,p)n, \phi(p,r)n\big)$ between the boundary for q and p, and the boundary for p and r.

When the challenger does select coin α, the expected number of heads is $p \cdot n$. The probability that h falls into the winning region — that is, the true positive rate for the coin—is bounded by Hoeffding's inequality as

$$\begin{aligned} TPR(\alpha) \geq 1 &- \exp\Bigg(-2\Big(p - \phi(q,p)\Big)^2 n\Bigg) \\ &- \exp\Bigg(-2\Big(\phi(r,p) - p\Big)^2 n\Bigg) \end{aligned}$$

In the approximate version of the game, where the adversary may have error up to δ in his estimate for each coin's probability, the lower bound of the region where he guesses coin α may be as tight as $\phi(q+\delta, p+\delta)$ and the upper bound may be as tight as $\phi(p-\delta, r-\delta)$. We are now ready to state our first result in Theorem 2.

Theorem 2 (True Positive Rate for Coin α)**.** *Let α denote one of the m coins in an (m,n,δ)-approximate biased coin game where each coin has a unique probability of heads. Let $p \in \mathbf{p}$ be the heads probability of the given coin, let $q \in \mathbf{p}$ be the largest coin probability smaller than p, let $r \in \mathbf{p}$ be the smallest coin probability greater than p, and let $\varepsilon > 0$. If $\varepsilon < p - \phi(q+\delta, p+\delta)$ and $\varepsilon < \phi(p-\delta, r-\delta) - p$, then the true positive rate of the frequency matching strategy is at least*

$$1 - 2\exp(-2\varepsilon^2 n).$$

The proof follows by plugging in ε to the formula for the true positive rate given above.

Note that Theorem 2 holds only when the probabilities $p \in \mathbf{p}$ are unique. If there are N_p coins with probability p, then the adversary must guess randomly among them, and so his true positive rate is $\frac{1}{N_p}$ times the quantity above. Also, note that if $p = \min(\mathbf{p})$ or $p = \max(\mathbf{p})$, then the adversary has an even greater chance of success, because the coin's winning region is bounded by other coins on only one side.

False Positive Rate A false positive for coin α occurs whenever the challenger selects some coin other than α, but the number of heads falls into the range where the adversary guesses α. We can apply Hoeffding's inequality again to show that the probability of a false positive is small, and it decreases with n and with the difference in the coin probabilities.

Suppose the challenger selects a coin $\beta \neq \alpha$ having probability p_β. If $p_\beta > p$, then the number of heads is likely greater than the upper end of the range for coin α. The probability that there are so few heads as to be confused with coin α is less than

$$\begin{aligned} \Pr\left[\, h < \phi(p, p_\beta)n \,\right] &\leq \exp\Bigg(-2\Big(\phi(p,p_\beta) - p\Big)^2 n\Bigg) \\ &\leq \exp\Bigg(-2\Big(\phi(p,r) - p\Big)^2 n\Bigg) \end{aligned}$$

The second line follows from the first because $r \leq p_\beta$. (See Theorem 7 in Appendix A for more details.)

Similarly, if $p_\beta < p$, then the number of heads is most likely below (ie, to the left of) the region for coin α. The probability that coin β results in so many heads as to be confused with coin α is less than

$$\begin{aligned}\Pr\left[\, h > \phi(q,p)n \,\right] &\leq \exp\left(-2\Big(p - \phi(p_\beta, p)\Big)^2 n\right)\\ &\leq \exp\left(-2\Big(p - \phi(q, p)\Big)^2 n\right)\end{aligned}$$

Theorem 3 (False Positive Rate for Coin α)**.** *Let α denote one of the m coins in an (m,n,δ)-approximate biased coin game where each coin has a unique probability of heads. Let $p \in \mathbf{p}$ be the heads probability of the given coin, let $q \in \mathbf{p}$ be the largest coin probability smaller than p, let $r \in \mathbf{p}$ be the smallest coin probability greater than p, and let $\varepsilon > 0$. If $\varepsilon < p - \phi(q+\delta, p+\delta)$ and $\varepsilon < \phi(p-\delta, r-\delta) - p$, then the false positive rate for the frequency matching strategy is at most*

$$\begin{aligned}FPR(\alpha) &\leq \sum_{p_j \neq p} \pi_j \exp\left(-2\varepsilon^2 n\right)\\ &\leq \exp\left(-2\varepsilon^2 n\right)\end{aligned}$$

Note that Theorem 3 only holds when the probabilities $p \in \mathbf{p}$ are unique. If there are N_p coins with probability p, then the adversary must guess randomly among them. In this case, his FPR is at least $\frac{N_p - 1}{N_p}$.

4.2 The Loaded Dice Game

The LDG is a generalization of the biased coin game, where each experiment can have more than two outcomes. More formally, the challenger holds m unfair d-sided dice, such that the i^{th} die lands with the j^{th} side up with probability $p_{i,j}$. As in the BCG, the challenger chooses one die according to the prior distribution π, and he rolls it n times. He records the outcome of each trial in the sample $\mathbf{s}$, where $s_i \in [1, d]$. Finally, he provides the adversary with the prior probabilities $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_m)$, the probability vector for each of the dice $\mathbf{p}_i = (p_{i,1}, \ldots, p_{i,d})$, and the sample $\mathbf{s}$. The adversary outputs its guess as to which die the challenger picked, and wins if the guess is correct. We call this an (m,d,n)-loaded dice game.

As in the BCG, when the adversary's probability estimates are only accurate within some factor δ, we call it a (m,d,n,δ)- approximate loaded dice game.

Statistical Model of Leakage In the LDG, the adversary learns the total number of dice rolls that resulted in each face of the die. More formally, let $h = (h_1, \ldots, h_d)$ be the vector where

$$h_j \overset{def}{=} \sum_i I_{s_i = j}.$$

In other words, when the challenger chooses die α, the list $\mathbf{s}$ of dice rolls is sample of size n from a Categorical distribution with parameter $p_\alpha = (p_{\alpha,1}, \ldots p_{\alpha,d})$. Similarly, the vector of counts $(h_1, \ldots, h_d)$ is a sample from a Multinomial distribution with parameters $(n; p_{\alpha,1}, \ldots, p_{\alpha,d})$.

Defining Hypotheses Just like in the biased coin game, the adversary has one hypothesis for each of the challenger's m dice, namely $\mathcal{H} = \{H_i : \text{Die } i \text{ was selected}\}$.

Choosing the Best Hypothesis The maximum *a posteriori* strategy chooses the die that maximizes the posterior probability.

$$\begin{aligned}\hat{H}_{MAP} &= \arg\max_{H_i \in \mathcal{H}} \ \Pr[\, H_i \mid \mathbf{s} \,] \\ &= \arg\max_{H_i \in \mathcal{H}} \ \pi_i \Pr[\, \mathbf{s} \mid H_i \,] \\ &= \arg\max_{H_i \in \mathcal{H}} \ \pi_i \ (p_{i,1})^{h_1} (p_{i,2})^{h_2} \ldots (p_{i,d})^{h_d}\end{aligned}$$

The maximum likelihood strategy is similar to MAP, but it ignores the priors.

$$\begin{aligned}\hat{H}_{MLE} &= \arg\max_{H_i \in \mathcal{H}} \ \Pr[\, \mathbf{s} \mid H_i \,] \\ &= \arg\max_{H_i \in \mathcal{H}} \ (p_{i,1})^{h_1} (p_{i,2})^{h_2} \ldots (p_{i,d})^{h_d}\end{aligned}$$

Evaluation Analysis of the Multinomial distribution, similar to our analysis in the previous section, is much more complicated than for the Binomial. Fortunately for our purposes it suffices to show that the adversary can achieve a good success rate in the LDG by reducing it to an instance of the BCG.

The reduction is straightforward. Suppose the adversary receives a challenge in the (m, d, n) loaded dice game where h_j of the dice rolls landed with side j for $j \in [d]$. First, the adversary chooses just one side of the die for his analysis and treats all other sides as if they were the same. Without loss of generality, call the chosen side k. The adversary then constructs an instance of the BCG as follows.

Let the heads probability for coin i be the probability that die i rolls a k. In the sample of coin tosses, let the lth coin toss be heads if the lth roll of the die was a k, and tails otherwise. The LDG adversary then uses the optimal BCG adversary to pick the best coin for a BCG challenge with h_k heads on n coin tosses.

Theorem 4 (Reduction of LDG to BCG)**.** *If the LDG adversary focuses his analysis on a single side $k \in [d]$ of the dice, then his performance in the LDG is equivalent to that of an adversary in the biased coin game with a set of m coins having probabilities $\mathbf{p} = \{p_{1,k}, \ldots, p_{m,k}\}$.*

The proof follows from the fact that each component $k \in [d]$ in a d-dimensional Multinomial distribution with parameters $(n; p_1, \ldots, p_d)$ follows a Binomial distribution with parameters (n, p_k). In other words, for a d-sided loaded die having probabilities $(p_1, \ldots, p_d)$, the number of times a k is rolled out of n trials is Binomial with probability p_k—just like the number of heads for a biased coin with the same probabiilty. Consequently, the bounds on the adversary's true positive rate and false positive rate from Theorem 2 and Theorem 3 apply to the loaded dice game as well.

5 Attacks on Deterministic Encryption

Here we show how the problem of guessing the plaintext for an encrypted record in a database with deterministic encryption reduces to the biased coin game. We reiterate that we chose this approach not because it gives the most powerful attack on deterministic encryption (it does not), but because it makes the success of the attack easy to predict analytically.

Furthermore, this simple attack is sufficient to show the insecurity of deterministic encryption for almost all real-world databases. The only databases for which the scheme is not obviously insecure are those where (i) the data is drawn from a distribution where all the plaintexts have very nearly the same probability, i.e. the high min-entropy requirement from [8, 5], or (ii) the number of encrypted records is very small, or (iii) for some reason the adversary cannot obtain reliable estimates of the plaintext probabilities.

Let $Y = (y_1, \ldots, y_n)$ be the n encrypted records from a given column in an encrypted database, where each record y_i is the deterministic encryption of the plaintext record x_i. Without loss of generality, we model the plaintexts as n i.i.d. random variables $X = (X_1, X_2, \ldots, X_n)$ drawn from a Categorical distribution over M messages $(m_1, \ldots, m_M)$, where $\Pr[\, X_i = m_j \,] = p_j$ for all $i \in [n], j \in [M]$, and $\sum_j p_j = 1$. Let $Z = (z_1, \ldots, z_N)$ be an independent sample from the same (or nearly the same) distribution as X. The adversary uses Z as his auxiliary information in order to guess X based on the leakage from Y.

Because deterministic encryption reveals the equality of records, ie $y_i = y_j$ iff $x_i = x_j$, the adversary learns how many times the plaintext records appear in the database, but he does not (immediately) learn which count goes with which plaintext. Here we show how he can use his auxiliary data and the observed ciphertext counts in the BCG to guess the plaintext for each ciphertext independently. This technique is not guaranteed to produce a consistent set of answers; it may guess that two or more ciphertexts all decrypt to the same plaintext, which is not possible with DET. However, the results of our analysis show he has only a small chance of guessing wrong.

To guess the plaintext $X_i = \mathsf{Dec}(y_i)$ for some ciphertext $y_i \in Y$, the DET adversary constructs an instance of the BCG as follows. In the game that he sets up, each coin represents one of the M possible plaintext messages, and the number of heads represents the number of times his target ciphertext appears in the database. More formally, let $\widetilde{p}_j = \frac{1}{N} \sum_{i=1}^{N} I_{z_i = m_j}$, and let $h = \frac{1}{n} \sum_{i=1}^{n} I_{y_i = y}$. The DET adversary poses a challenge to the BCG adversary with M coins having approximate probabilities $(\widetilde{p}_1, \ldots, \widetilde{p}_M)$, and h heads on n tosses of the selected coin. When the BCG adversary guesses coin j, the DET adversary guesses that the plaintext $\mathsf{Dec}(y_i) = m_j$.

Therefore, according to the analysis of the BCG in Section 4.1, if the error in the adversary's estimates of the plaintext probabilities is small, ie $|p_j - \widetilde{p}_j| \leq \delta$, then the adversary can identify each plaintext with a true positive rate that is exponentially close to 100% (by Thm. 2), as well as with an exponentially small false positive rate (by Thm. 3).

As a concrete example, suppose some plaintext m occurs with probability $p = 1\%$, and the next-smallest and next-largest plaintext frequencies are $q = 0.9\%$ and $r = 1.1\%$, respectively. Then our analysis predicts that an adversary with no error in his estimated probabilities (ie, $\delta = 0$) can identify m in an encrypted datbase of 1000 records with at least 12.5% true positive rate. If the database has 100 thousand records, his success rate will be greater than 88%.

6 Attacks on Order-Revealing Encryption

6.1 Order-Preserving and Order-Revealing Encryption

In order-preserving encryption [4, 9], the ciphertexts reveal not only the equality of ciphertexts but also their relative ordering. Given two ciphertexts $y_i = \mathsf{Enc}(x_i)$ and $y_j = \mathsf{Enc}(x_j)$, $y_i < y_j$ iff $x_i < x_j$.

Order-revealing encryption [33, 11], also sometimes called "efficiently orderable encryption" [10], gives a more general definition, wherein y_i and y_j reveal the ordering of x_i and x_j through some efficiently computable function. (The comparison function may or may not be a simple comparison of the ciphertexts). When the ORE reveals that y_i and y_j represent the same plaintext, we write this as $y_i \equiv y_j$.

Given an ORE-encrypted column $Y = (y_1, \ldots, y_n)$ and an auxiliary data set $Z = (z_1, \ldots, z_N)$, the adversary can use the loaded dice game to guess the plaintexts $X = (x_1, \ldots, x_n)$ by reducing the attack to a game with 3-sided dice. Each die represents one of the M plaintext messages. The three sides of the die for plaintext m_j represent the three possible outcomes when comparing another plaintext with m_j; either the other message is less than, equal to, or greater than m_j. The adversary uses his auxiliary data to set up the dice probabilities as follows:

$$\begin{aligned} \widetilde{p}_{j,1} &= \Pr\left[\, z_i < m_j \,\right] \\ \widetilde{p}_{j,2} &= \Pr\left[\, z_i = m_j \,\right] \\ \widetilde{p}_{j,3} &= \Pr\left[\, z_i > m_j \,\right] \end{aligned}$$

Given a target ciphertext $y \in Y$, the adversary sets the observed counts $h = (h_1, h_2, h_3)$ as the number of ciphertexts in Y that are less than y, equal to y, and greater than y, respectively. Then he can use the LDG adversary to choose the best die. When the LDG adversary guesses die j, the ORE adversary guesses that $x = \mathsf{Dec}(y)$ is the corresponding plaintext m_j.

By Theorem 4, the ORE adversary will perform at least as well as the DET adversary. And depending on the data distribution, he may be able to do much better. For example, notice that when the plaintext distribution is the discrete Uniform, the DET adversary learns nothing from the ciphertext because all plaintexts are equally likely. However, the ORE adversary can use the relative ordering of the ciphertexts to make better guesses, eg by using the BCG reduction and focusing on either side 1 or side 3 of his dice.

6.2 Lewi and Wu's Left-Right ORE

Lewi and Wu [30] recently described a new ORE scheme where the ciphertexts are semantically secure against offline attacks. However, once the adversary observes a single encrypted query, he can learn a great deal about it (and, consequently, about the encrypted records in the database).

In the LW ORE scheme, encrypting a plaintext gives two ciphertexts, called the *left* and *right* ciphertext. We write this as $(L_x, R_x) \leftarrow \mathsf{Enc}(x)$. A collection of only left ciphertexts or only right ciphertexts reveals nothing about the plaintexts, but comparing a left ciphertext to a right ciphertext reveals the ordering of the two plaintexts. Therefore the untrusted server stores only the right ciphertexts of the encrypted database records, and when the client submits a query it sends only left ciphertexts.

To perform a range query, the client encrypts the top T and bottom B of the range, and it sends the two left ciphertexts L_T and L_B to the server. The server compares each right ciphertext in the database to the left ciphertexts to determine whether the encrypted record is greater, equal, or less than each end of the range. This allows it to service the query, but doing so also reveals a great deal of information about T and B.

The query leakage about L_T and about L_B reveals at least as much information as the static leakage in standard ORE in the previous section. That is, the adversary learns the number of records in the database that are less than, equal to, and greater than B, and he learns the same information about T. Therefore one viable attack strategy is to simply use the attack in the previous section to make a guess about B using the 3-sided loaded dice game, then repeat the same process to make a guess about T. Another alternative is to use our attack on ORE using order statistics (see Section 6.5) to guess T and B independently. Either approach recovers each end of the range with only an exponentially small chance of error.

We can construct the optimal attack, which is potentially more powerful, from a 5-sided dice game. Given a total of M possible unique plaintexts, we define a total of $\frac{M^2-M}{2}$ loaded dice, one for each possible query (l, u) with $l < u$. The five sides of the dice represent the five possible categories for database records relative to l and u. Any plaintext record must be either:

1. Less than l
2. Equal to l
3. Greater than l but less than u
4. Equal to u
5. Greater than u

We set the probabilities on the die representing query (l, u) to be the fraction of the auxiliary data that are in each category relative to l and u. Then, for an encrypted query (L_b, L_t), we set the outcome of the ith dice roll $\mathbf{s}_i$ to 1 if the ith database record $x_i < b$, we set it to 2 if $x_i = b$, etc. Finally, we use the adversary for the loaded dice game to pick the die that best explains the observed outcomes.

Using the maximum *a posteriori* strategy in this 5-sided loaded dice game gives the adversary the optimal attack for recovering a single query in the Lewi-Wu scheme. Obtaining a tight bound on the optimal adversary's success rate is complicated. Fortunately doing so would only be necessary if we wanted to prove security of the scheme for some kind(s) of data. For now, it suffices to know that the simpler attacks outlined above can recover both t and b with high probability as the number of records increases.

6.3 Randomized Order Revealing Encryption

Randomized ORE constructions, such as those proposed by Kerschbaum [24] or Kadhem et al [23], reveal partial ordering of the plaintexts but not equality. Given $y_i = \mathsf{Enc}(x_i)$ and $y_j = \mathsf{Enc}(x_j)$, if $x_i < x_j$ then $y_i < y_j$, but the converse statement does not necessarily hold. Instead, if $y_i < y_j$, then we only know that $x_i \leq x_j$.

Our analysis of randomized ORE schemes does not rely on the biased coin game or loaded dice game, but instead on the theory of order statistics [16]. Given an encrypted column of n R-ORE ciphertexts $Y = (y_1, \ldots, y_n)$, when some encrypted record y_i is greater than exactly $r - 1$ other elements of Y, we say that it is the order statistic of order r. We write this as $y_i = y_{(r)}$, and we call r the *rank* of y_i in Y. Here we conservatively assume an ideal version of the R-ORE construction, where no two ciphertexts are identical; therefore each ciphertext has a unique rank in Y.

By the definition of randomized ORE, the ciphertexts leak the order statistics of the plaintext records. That is, if $y_i = y_{(r)}$, then $x_i = x_{(r)}$. As it turns out, the rank of a number in a random sample is a surprisingly powerful predictor for the number itself. Intuitively, the idea is that a number's frequency and magnitude give a good predictor of the position(s) it occupies in a sorted list. A small number should appear near the front of the list, while a large number should appear toward the end. Similarly, a frequent number should occupy a large span of ranks, while an infrequent number should appear in only a few positions. We show two strategies that the adversary can apply to use the leaked ranks in guessing the plaintexts.

For a ciphertext $y \in Y$ with rank r out of n, the adversary has M competing hypotheses about the plaintext, $H_i : \mathsf{Dec}(y) = m_i$. The optimal MAP strategy chooses the plaintext with the greatest posterior probability.

$$\begin{aligned}\hat{H}_{MAP} &= \arg\max_{H_i \in \mathcal{H}} \Pr\left[\, H_i \mid Y \,\right] \\ &= \arg\max_{H_i \in \mathcal{H}} \Pr\left[\, \mathsf{Dec}(y) = m_i \mid \mathit{Rank}(y, Y) = r \,\right] \\ &= \arg\max_{H_i \in \mathcal{H}} \Pr\left[\, x_{(r)} = m_i \,\right]\end{aligned}$$

David and Nagaraja show how this probability can be computed exactly for each candiate plaintext m_i (see Sec 2.4 of [16]). Although this approach gives a formula that can be computed on a modern CPU in a reasonable amount of time, the resulting expression is somewhat complicated, and it is not intuitively clear from the formula whether the resulting quantity will be large or small. Therefore we give the formula and its derivation in Appendix B but do not include it here.

Instead, we focus on establishing a clear and concise lower bound on the optimal strategy's performance. To do so, we describe and analyze a much simpler attack strategy that also guesses the plaintext with near-perfect success. To help develop the reader's intuition, in Figure 2 we show an example with three plaintexts $m_1 < m_2 < m_3$ having probabilities $0.184, 0.33, 0.486$, respectively. The figure shows the posterior probability that each entry in the list of sorted database records takes on each plaintext value. Even when the number of records in the database is relatively small, for most ranks there is a single obvious best guess for the plaintext—namely, the one whose posterior probability is much greater than all the others. The dotted vertical lines correspond to the cumulative frequencies of the plaintexts; notice that the locations of these lines are very closely related to changes in which plaintext has the greatest probability.

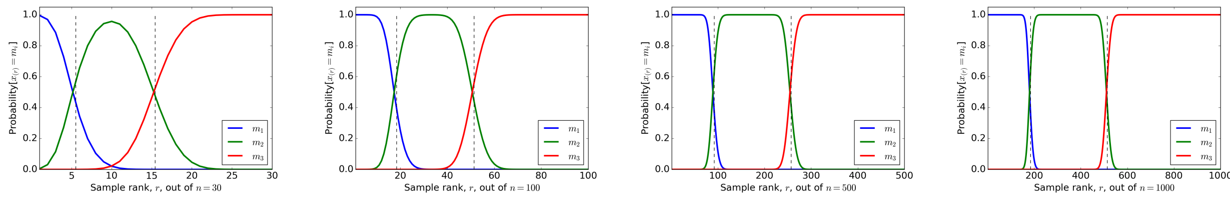

Figure 2: Posterior probabilities from order statistics

In the following analysis, we show that the *bucketing attack* of Grubbs et al [19] works well because it leverages the relationship between the cumulative frequencies and the (optimal) maximum *a posteriori* guess for the plaintext at each rank. Given an encrypted column of n records and M candidate plaintexts $m_1 < m_2 < \cdots < m_M$, the adversary partitions the list of ranks $[1, n]$ into up to M "buckets." The dotted lines in Figure 2 give the edges of the buckets for our example distribution. More concretely, let $F_X(m) = \Pr\left[\, x \le m \,\right], x \in X, m \in \mathcal{M}$ be the cumulative distribution function (CDF) of the plaintext distribution. If $n \cdot p_i \ge 1$, then the bucket for plaintext m_i includes all ranks from $\lceil n \cdot F_X(m_{i-1}) \rceil$ up to $\lfloor n \cdot F_X(m_i) \rfloor$. When the rank r falls in bucket i, the adversary guesses that $x_{(r)} = m_i$.

Analysis Using the theory of order statistics we can easily show that, for an encrypted column with sufficiently many records, the bucketing attack almost always guesses correctly. For each plaintext $m_i \in \mathcal{M}$, let $g(m_i)$ be the probability that a plaintext record is strictly less than m_i, ie

$$\begin{aligned} g(m_i) &= \Pr[\, x < m_i \,] \\ &= \sum_{j<i} p_j \end{aligned}$$

Note that, if $i > 1$, then $g(m_i) = F(m_{i-1})$; otherwise $g(m_i) = 0$. Our analysis starts from the observation that

$$\Pr\left[\, x_{(r)} = m \,\right] = 1 - \Pr\left[\, x_{(r)} < m \,\right] - \Pr\left[\, x_{(r)} > m \,\right].$$

It suffices to show that both of the probabilities on the right-hand side are small. Notice that the event $x_{(r)} < m$ occurs if and only if the plaintext column X contains at least r records smaller than m. Let S_m be the number of these records in X, ie

$$S_m = \sum_{i=1}^{n} I_{x_i < m}.$$

Then S_m has a Binomial distribution with parameters n and $g(m)$, and its expected value is $n \cdot g(m)$. For $r > n \cdot g(m)$, Hoeffding's inequality gives an exponential upper bound on the probability that S_m is at least r.

$$\begin{aligned} \Pr\left[\, x_{(r)} < m \,\right] &= \Pr[\, S_m \geq r \,] \\ &\leq \exp\left(-2\Big(g(m) - \frac{r}{n}\Big)^2 n \right) \end{aligned}$$

In other words, as the rank r increases, it becomes exponentially unlikely that the plaintext $x_{(r)}$ will be a small number.

Similarly, the event $x_{(r)} > m$ occurs only when X contains fewer than r records which are less than or equal to m. For $r < n \cdot F(m)$, we can apply Hoeffding's inequality again to obtain an upper bound on the probability.

$$\Pr\left[\, x_{(r)} > m \,\right] \leq \exp\left(-2\Big(F(m) - \frac{r}{n}\Big)^2 n \right)$$

That is to say, as the rank r decreases, the the plaintext $x_{(r)}$ is exponentially less likely to be a large number.

Notice that both Hoeffding bounds apply simultaneously only when $g(m) < \frac{r}{n} < F(m)$, and that the only plaintext which satisfies this condition is the one chosen by the bucketing attack. Combining the two Hoeffding bounds, we are now ready to state our main result about the bucketing attack on R-ORE.

Theorem 5 (Bucketing Attack Accuracy for Rank r)**.** *Let $r \in [n]$ be the rank of a ciphertext $y \in Y$, and let $m \in \mathcal{M}$ be the plaintext guessed by the bucketing attack for y. Let $\varepsilon > 0$. If $\varepsilon < \min(\frac{r}{n} - g(m), F(m) - \frac{r}{n})$, then the adversary's guess is correct with probability at least*

$$Acc(r) \geq 1 - 2\exp(-2\varepsilon^2 n)$$

6.4 Partial Order Preserving Encoding

With a small amount of extra work, the attack in the previous section on randomized ORE can be generalized into an attack on the partial order preserving encoding (POPE) construction of Roche et al [37]. In POPE, the server stores encrypted search terms (also called "labels") in a tree structured index, similar to a B-tree. Each non-leaf node contains a *sorted* list of encrypted labels, one for each of its child nodes. Every node in the tree also contains a buffer of *unsorted* encrypted data.

POPE enforces the following partial ordering property on all nodes in the tree. Let u be a node in a POPE tree $\mathcal{T}$, having child nodes u_j and u_{j-1} with labels ℓ_j and ℓ_{j-1}, respectively. Then for any label ℓ in the sub-tree $\mathcal{T}_{u_j}$ rooted at u_j, we have $\ell_{j-1} < \ell \leq \ell_j$. From this ordering constraint, when an adversary considers a target node u in the POPE tree, he learns:

- n_1, the number of encrypted records that are stricly less than those in the target node. These are the records stored in nodes "to the left of" u in the POPE tree.
- n_2, the number of encrypted records in the target node.
- n_3, the number of encrypted records that are strictly greater than those in the target node. These are the records stored in nodes "to the right of" u in the POPE tree.
- n_4, the number of encrypted records whose partial ordering cannot be determined relative to those in the current node These are the records stored in nodes that are either parents or children of node u in the POPE tree.

Let n be the total number of records whose ordering relative to node u can be determined, plus those in the target node, ie $n = n_1 + n_2 + n_3$, and let Y be the collection of these n encrypted records, and let X be their plaintexts. For a record $y \in Y$ in the target node, the adversary can determine an estimate of the rank of its plaintext $x = \mathsf{Dec}(y)$, up to the granularity allowed by the size of the encrypted buffer at node u. Specifically, he knows that $n_1 < \mathit{Rank}(x, X) \leq n_1 + n_2 + n_3$.

Assumptions Here, we assume that some queries have already been processed; otherwise POPE hides all ordering information. Furthermore, we assume that the distribution of the queries is approximately uniform over the range of the plaintexts, so that X is a random sample over roughly the same distribution as the entire set of plaintexts in the database.

Attacks The MAP attack strategy is to choose the plaintext $m \in \mathcal{M}$ with the greatest posterior probability. The only information the adversary knows about the rank of the plaintext is that it falls within the half-open interval $(n_1, n_1+n_2]$; therefore all n_2 ranks in this range are equally likely, and the posterior probability is simply the weighted sum over these ranks.

$$\begin{aligned}
\Pr\left[\, x = m \,\right] &= \sum_{r=n_1+1}^{n_1+n_2} \Pr\left[\, \mathit{Rank}(x, X) = r \,\right] \cdot \Pr\left[\, x_{(r)} = m \,\right] \\
&= \sum_{r=n_1+1}^{n_1+n_2} \frac{1}{n_2} \cdot \Pr\left[\, x_{(r)} = m \,\right] \\
&= \frac{1}{n_2} \sum_{r=n_1+1}^{n_1+n_2} \Pr\left[\, x_{(r)} = m \,\right]
\end{aligned}$$

In practice, the attack may be much simpler. If the block is small, then the adversary's best guess may be the same for each rank in the entire target block. A sufficient condition for this event to occur is that the entire block falls into the range where the bucketing attack from the previous section guesses the same answer for every plaintext rank contained in the block. This happens when there exists some plaintext m with $n \cdot g(m) < n_1$ and $n_1 + n_2 \leq n \cdot F(m)$. In this case, the accuracy bound from Theorem 5 applies to every item in the block. One possible countermeasure against such attacks is to keep the block size larger than the expected number of copies of any one plaintext.

Even then, if the adversary cannot guess the plaintext exactly, he may still be able to eliminate many candidate plaintexts from consideration. Suppose that for some plaintext $m \in \mathcal{M}$, the adversary sees that $n \cdot F(m) < n_1$. Then by Hoeffding's bound, he knows that there is only an exponentially small chance that the plaintexts in his target block are less than or equal to m. Similarly, if $n_1 + n_2 < n \cdot g(m)$, then there is only a small chance that the records are greater than or equal to m. After eliminating the plaintexts that are too large or too small, the only plaintexts that the adversary should expect to find in the block are those with $n_1 < n \cdot F(m) \leq n_1 + n_2$.

6.5 An Alternative Analysis of ORE

A basic understanding of order statistics also gives rise to an alternative approach for the analysis of OPE and ORE. Given a (non-frequency-hiding) ORE ciphertext column $Y = (y_1, y_2, \ldots, y_n)$, the adversary can always choose to ignore the frequencies of the y_i's and treat them as if they came from a randomized ORE. He first sorts the y_i's into increasing order to determine a unique rank $r \in [1, n]$ for each ciphertext record. Then he runs the bucketing attack from Section 6.3 to guess the plaintext for each record independently. By Theorem 5, using this strategy gives him a very high chance of guessing correctly for any rank that is not too close to the edge of a bucket, that is, not too close to any $F(m_i) \cdot n$ for any plaintext $m_i \in \mathcal{M}$. On the other hand, for ranks that are close to the edge of the bucket at $F(m_i) \cdot n$, he may not be able to reliably distinguish between m_i and m_{i+1}.

The adversary can improve the effectiveness of this attack strategy for non-frequency hiding ORE if he also takes into account the leakage that tells him which encrypted records are equal. Intuitively, Theorem 5 says that the bucketing attack works best for ciphertexts whose rank r is near the center of a bucket, where the plaintext $x_{(r)}$ is both very unlikely to be larger than the adversary's guess and simultaneously very unlikely to be smaller than the guess.

The Bucket Centroid Attack The bucketing attack in Section 6.3 defines the bucket for plaintext $m_i \in \mathcal{M}$ as the interval of ranks from $\lceil n \cdot g(m_i) \rceil$ up to $\lfloor n \cdot F(m_i) \rfloor$. We define the *centroid* of bucket i as the middle rank in the bucket, and we write this as $C_i = n\frac{g(m_i)+F(m_i)}{2}$. In the bucket centroid attack on ORE, the adversary uses only the bucket centroids to make his guesses about the plaintexts. If the centroid C_i for bucket i is an integer, he uses the bucketing attack to guess the plaintext for rank C_i. He then guesses that the same plaintext is also the decryption for all other $y \in Y$ such that $y \equiv y_{(C_i)}$. If the centroid C_i is not an integer, the adversary uses the bucketing attack to guess the plaintext for the two ranks $\lfloor C_i \rfloor$ and $\lceil C_i \rceil$. If the bucketing attack returns the same plaintext for both ranks, then the adversary guesses that this plaintext is also the decryption of all other ciphertext records $y \in Y$ with $y \equiv y_{(\lfloor C_i \rfloor)}$ or $y \equiv y_{(\lceil C_i \rceil)}$.

Like the simple BCG-inspired attack on deterministic encryption in Section 5, this strategy is not guaranteed to give a consistent set of answers. For example, if there exists some $y \in Y$ where the set of ciphertexts equal to y includes two bucket centroids, $y_{(C_i)}$ and $y_{(C_j)}$, corresponding to two different plaintexts m_i and m_j, then this strategy will guess that $\mathsf{Dec}(y)$ is both m_i and m_j, which is impossible. However, a straightforward application of Theorem 5 shows that the chance of guessing incorrectly is exponentially small, and it decreases with the number of encrypted records.

The BCA correctly identifies the plaintext m_i whenever the bucketing attack on randomized ORE guesses correctly for the record with rank C_i. Therefore we can obtain a bound on the BCA's accuracy for plaintext m_i by plugging in C_i for the rank r in Theorem 5.

Theorem 6 (Accuracy of the BCA for Plaintext m_i)**.** *Let $Y = (y_1, \ldots, y_n)$ be the ORE encryption of a plaintext column $X = (x_1, \ldots, x_n)$ sampled from a set of plaintexts $\mathcal{M} = (m_1, \ldots, m_M)$. The adversary's accuracy for recovering plaintext m_i is at least*

$$Acc(m_i) \geq 1 - 2\exp\Big(-\frac{1}{2}n(p_i)^2\Big).$$

The proof follows from Theorem 5 and the fact that $F(m_i) - \lfloor C_i \rfloor \geq \frac{p_i}{2}$ and $\lceil C_i \rceil - g(m_i) \geq \frac{p_i}{2}$.

For a concrete example, suppose some plaintext m occurs with probability $p = 0.01$. In a database of 1000 records, Theorem 6 says that the BCA can identify m with probability at least 4.88%, and if the database has 10000 records, then his chance increases to 39%. With 100 thousand records, his success rate is greater than 99%.

7 Conclusion

In this paper we presented a new approach for predicting leakage abuse vulnerabilities and for evaluating their severity analytically, rather than empirically as in previous work. We proposed a new analytical framework for reasoning about leakage and its consequent attacks, and we demonstrated the use of this framework for the analysis of several recent schemes for encrypting records in relational databases. In each case, our analysis showed that the current schemes are vulnerable to attacks that recover the plaintext for encrypted records or queries with high probability as the size of the database increases.

Future Work In this paper, we have not yet attempted to prove security for any construction. This is in part because every scheme that we analyzed (with the possible exception of some configurations of POPE [37]) allows the adversary to succeed with probability arbitrarily close to 100%. Still, we hope that our framework will be useful in the future for designers of new or modified schemes, who seek to show that a reduced leakage profile limits the adversary's power in some way. One approach to such a proof of security might be to show that the adversary's optimization task of choosing the best plaintext is an NP-hard problem with no polynomial time approximate solution. Another approach would be to first define the adversary's optimal guessing strategy, e.g. maximum *a posteriori*, for the given leakage profile and then to prove some upper bound on its accuracy in recovering the target information.

Acknowledgements

The authors would like to thank Muhammad Naveed for his help at the beginning of this project and Scott Coull for introducing us to the analytical frameworks for differential privacy referenced in Section 2. We thank Seny Kamara for his involvment in this project, including many rounds of discussion and feedback, and especially for his help in focusing the ideas in this paper.

References

[1] Always Encrypted. `https://msdn.microsoft.com/en-us/library/mt163865(v=sql.130).aspx`.

[2] Google Encrytped Big Query. `https://github.com/google/encrypted-bigquery-client`.

[3] SAP SEEED. `https://www.sics.se/sites/default/files/pub/andreasschaad.pdf`.

[4] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *SIGMOD*, pages 563–574, 2004.

[5] G. Amanatidis, A. Boldyreva, and A. O'Neill. Provably-secure schemes for basic query support in outsourced databases. In *Working conference on Data and applications security*, pages 14–30, 2007.

[6] T. Baigneres, P. Junod, and S. Vaudenay. How far can we go beyond linear cryptanalysis? In *Asiacrypt*, volume 3329, pages 432–450. Springer, 2004.

[7] R. Bassily, A. Groce, J. Katz, and A. Smith. Coupled-worlds privacy: Exploiting adversarial uncertainty in statistical data privacy. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 439–448, Oct 2013.

[8] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In *CRYPTO*, pages 535–552, 2007.

[9] A. Boldyreva, N. Chenette, Y. Lee, and A. O'neill. Order-preserving symmetric encryption. In *EUROCRYPT*, pages 224–241, 2009.

[10] A. Boldyreva, N. Chenette, and A. O'Neill. Order-preserving encryption revisited: improved security analysis and alternative solutions. In *CRYPTO*, pages 578–595, 2011.

[11] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, and J. Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In *EUROCRYPT*, pages 563–594, 2015.

[12] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. Leakage-abuse attacks against searchable encryption. In *Proceedings of the ACM Conference on Communications and Computer Security (CCS '15)*. ACM, 2015.

[13] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. *IACR Cryptology ePrint Archive*, 2014:853, 2014.

[14] N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu. Practical order-revealing encryption with limited leakage. In *Fast Software Encryption (FSE '16)*, 2016.

[15] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *CCS*, pages 79–88, 2006.

[16] H. David and H. Nagaraja. *Order Statistics.* Wiley Series in Probability and Statistics. Wiley, 2004.

[17] F. B. Durak, T. M. DuBuisson, and D. Cash. What else is revealed by order-revealing encryption? In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1155–1166, New York, NY, USA, 2016. ACM.

[18] C. Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[19] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart. Leakage-abuse attacks against order-revealing encryption. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 655–672, May 2017.

[20] W. He, D. Akhawe, S. Jain, E. Shi, and D. Song. Shadowcrypt: Encrypted web applications for everyone. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 1028–1039, New York, NY, USA, 2014. ACM.

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

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

[23] H. Kadhem, T. Amagasa, and H. Kitagawa. MV-OPES: Multivalued-order preserving encryption scheme: A novel scheme for encrypting integer value to many different values. *IEICE TRANSACTIONS on Information and Systems*, 93(9):2520–2533, 2010.

[24] F. Kerschbaum. Frequency-hiding order-preserving encryption. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 656–667, New York, NY, USA, 2015. ACM.

[25] F. Kerschbaum and A. Schroepfer. Optimal average-complexity ideal-security order-preserving encryption. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 275–286, New York, NY, USA, 2014. ACM.

[26] D. Kifer and A. Machanavajjhala. A rigorous and customizable framework for privacy. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '12, pages 77–88, New York, NY, USA, 2012. ACM.

[27] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97, 1955.

[28] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 03 1951.

[29] B. Lau, S. Chung, C. Song, Y. Jang, W. Lee, and A. Boldyreva. Mimesis aegis: A mimicry privacy shield–a system's approach to data privacy on public cloud. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 33–48, San Diego, CA, Aug. 2014. USENIX Association.

[30] K. Lewi and D. J. Wu. Order-revealing encryption: New constructions, applications, and lower bounds. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1167–1178, New York, NY, USA, 2016. ACM.

[31] M. Naveed, S. Kamara, and C. V. Wright. Inference attacks on property-preserving encrypted databases. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 644–655, New York, NY, USA, 2015. ACM.

[32] J. Neyman and E. S. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231:289–337, 1933.

[33] R. A. Popa, F. H. Li, and N. Zeldovich. An ideal-security protocol for order-preserving encoding. In *2013 IEEE Symposium on Security and Privacy*, pages 463–477, May 2013.

[34] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan. CryptDB: Protecting confidentiality with encrypted query processing. In *SOSP*, pages 85–100, 2011.

[35] R. A. Popa, N. Zeldovich, and H. Balakrishnan. Guidelines for using the cryptdb system securely. Cryptology ePrint Archive, Report 2015/979, 2015. `http://eprint.iacr.org/2015/979`.

[36] D. Pouliot and C. V. Wright. The shadow nemesis: Inference attacks on efficiently deployable, efficiently searchable encryption. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1341–1352, New York, NY, USA, 2016. ACM.

[37] D. S. Roche, D. Apon, S. G. Choi, and A. Yerukhimovich. Pope: Partial order preserving encoding. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1131–1142, New York, NY, USA, 2016. ACM.

[38] E. Stefanov, M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas. Path ORAM: An extremely simple oblivious RAM protocol. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer &#38; Communications Security*, CCS '13, pages 299–310, New York, NY, USA, 2013. ACM.

A The Binomial Boundary Point

In the biased coin game, suppose we have two biased coins c_0 and c_1 with probabilities of heads $p_0 < p_1$. Definition 1 defines the *boundary point* $\phi(p_0, p_1)$ as the proportion of coin tosses that land "heads," above which coin c_1 is more likely than c_0, and below which c_0 is more likely than c_1.

We can derive a closed form expression for the boundary point as follows. If we begin with the condition that coin c_1 is more likely than c_0, i.e.

$$\binom{n}{k} p_0^k (1-p_0)^{n-k} < \binom{n}{k} p_1^k (1-p_1)^{n-k}$$

then the combinations cancel out

$$p_0^k (1-p_0)^{n-k} < p_1^k (1-p_1)^{n-k}.$$

Taking the log of both sides and re-arranging to isolate k, we obtain

$$\frac{\log \frac{1-p_0}{1-p_1}}{\log \frac{p_1}{p_0} + \log \frac{1-p_0}{1-p_1}} \cdot n < k.$$

Similarly, if we require $\mathcal{L}(c_0) > \mathcal{L}(c_1)$, i.e.

$$\binom{n}{k} p_0^k (1-p_0)^{n-k} > \binom{n}{k} p_1^k (1-p_1)^{n-k}$$

then we arrive at

$$\frac{\log \frac{1-p_0}{1-p_1}}{\log \frac{p_1}{p_0} + \log \frac{1-p_0}{1-p_1}} \cdot n > k.$$

Therefore $\phi(p_0, p_1) = \frac{\log \frac{1-p_0}{1-p_1}}{\log \frac{p_1}{p_0} + \log \frac{1-p_0}{1-p_1}}$ is the desired boundary point for p_0 and p_1.

Theorem 7. *If $p_0 < p_1 < p_2$, then $\phi(p_0, p_1) \leq \phi(p_0, p_2)$.*

Proof Sketch The proof of Theorem 7 follows from the shape of the Binomial probability

$$B(n, p, k) = \binom{n}{k} p^k (1-p)^{n-k}$$

As a function of k, with fixed n and p, the probability is maximized at $k = \lfloor pn \rceil$; it is increasing to the left of that point and decreasing to the right. Similarly, if we hold k and n fixed and view the Binomial probability as a function of p, it is maximized at $p = \frac{k}{n}$. It is increasing on the interval $[0, \frac{k}{n})$ and decreasing on the interval $(\frac{k}{n}, 1]$.

Let $k' = \lfloor \phi(p_0, p_1) n \rfloor$ be the largest number of heads for which p_0 is more likely than p_1, and let $p' = \frac{k'}{n}$. Because the Binomial with fixed n and $k = k'$ is decreasing on the interval $(p', 1]$, and because $p_2 > p_1 > p'$, it follows that $B(n, p_2, k') < B(n, p_1, k')$. Therefore $\phi(p_0, p_2) > \frac{k'}{n}$. As $n \to \inf$, the fraction $\frac{k'}{n}$ approaches $\phi(p_0, p_1)$, and the result holds.

Theorem 8. *If $p_0 < p_1 < p_2$, then $\phi(p_0, p_2) \leq \phi(p_1, p_2)$.*

Proof Sketch The proof for Theorem 8 is similar to that of Theorem 7. Because the Binomail probability at $k'' = \lceil \phi(p_1, p_2) \rceil$ is increasing for p in $[0, \phi(p_1, p_2))$ and because $p_0 < p_1 < phi(p_1, p_2)$, we have $B(n, p_0, k'') < B(n, p_1, k'')$. Therefore $\phi(p_0, p_2) < \frac{k''}{n}$, and as $n \to \inf$, the result holds.

B Posterior Probabilities from Order Statistics

David and Nagaraja [16] describe a technique for computing the posterior distribution of an item x_i in an i.i.d. random sample $X = (x_1, x_2, \ldots, x_n)$ based on the item's rank in the sample and the distribution of the sample. Specficially, let r be the rank of x_i in X, i.e. it is the order statistic of order r, which we write as $x_i = x_{(r)}$. Let $\mathcal{M} = (m_1, m_2, \ldots, m_M)$ be the domain of the items, with $m_1 < m_2 < \cdots < m_M$. Let F_X be the cumulative distribution function of the items in the sample, where $F_X(m) = \Pr[\, x \le m \,]$ for $m \in \mathcal{M}$.

Then we can compute the posterior distribution for the item $x_{(r)}$ at any rank $r \in [1, n]$ as follows. We begin by noting that

$$\Pr\left[\, x_{(r)} = m_j \,\right] = \Pr\left[\, x_{(r)} \le m_j \,\right] - \Pr\left[\, x_{(r)} \le m_{j-1} \,\right].$$

The event that $x_{(r)} \le m_j$ occurs exactly when the sample X contains r or more items that are less than or equal to m_j. Because the items in X are drawn independently from F_X, each entry in the sample will be $\le m_j$ with probability $F_X(m_j)$. Therefore the probability that there are exactly k records less than or equal to m_j is given by the Binomial probability

$$\Pr[\, k \text{ items } \le m_j \,] = \binom{n}{k} F_X(m_j)^k (1 - F_X(m_j))^{n-k}.$$

The probability that there are r or more such items is then

$$\Pr\left[\, x_{(r)} \le m_j \,\right] = \sum_{k=r}^{n} \binom{n}{k} F_X(m_j)^k (1 - F_X(m_j))^{n-k}.$$

Similarly, for the next-smallest value m_{j-1}, we have

$$\Pr\left[\, x_{(r)} \le m_{j-1} \,\right] = \sum_{k=r}^{n} \binom{n}{k} F_X(m_{j-1})^k (1 - F_X(m_{j-1}))^{n-k}.$$

We arrive at the desired quantity by subtracting the second summation from the first.