# Quantifying Information Leakage of Deterministic Encryption

Mireya Jurado
Florida International University
School of Computing and Information Sciences
Miami, Florida, USA
mjurado@fiu.edu

Geoffrey Smith
Florida International University
School of Computing and Information Sciences
Miami, Florida, USA
smithg@cis.fiu.edu

## ABSTRACT

In order to protect user data while maintaining application functionality, encrypted databases can use specialized cryptography such as property-revealing encryption, which allows a property of the underlying plaintext values to be computed from the ciphertext. One example is deterministic encryption which ensures that the same plaintext encrypted under the same key will produce the same ciphertext. This technology enables clients to make queries on sensitive data hosted in a cloud server and has considerable potential to protect data. However, the security implications of deterministic encryption are not well understood.

We provide a leakage analysis of deterministic encryption through the application of the framework of *quantitative information flow*. A key insight from this framework is that there is no single "right" measure by which leakage can be quantified: information flow depends on the operational scenario and different operational scenarios require different leakage measures. We evaluate leakage under three operational scenarios, modeled using three different gain functions, under a variety of prior distributions in order to bring clarity to this problem.

## CCS CONCEPTS

• **Security and privacy → Information-theoretic techniques**; **Management and querying of encrypted data**; *Formal security models*.

## KEYWORDS

Quantitative Information Flow; Deterministic Encryption; Leakage

**ACM Reference Format:**
Mireya Jurado and Geoffrey Smith. 2019. Quantifying Information Leakage of Deterministic Encryption. In *2019 Cloud Computing Security Workshop (CCSW'19), November 11, 2019, London, United Kingdom.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3338466.3358915

## 1 INTRODUCTION

Sensitive user data continues to be stolen in large-scale data breaches. While standard encryption protects data confidentiality, it restricts database application functionality. An alternative solution is to use specialized cryptography to enable *encrypted databases* such that some application functionality is preserved. One approach to encrypted databases is property-revealing encryption which allows a property of the underlying plaintext to be computed from the ciphertext. An example of property-revealing encryption is deterministic encryption which ensures that the same plaintext encrypted under the same key will produce the same ciphertext. A client can deterministically encrypt a column with sensitive information and then host the data in a remote location, such as a cloud database. The client can then create a query, encrypt the query's keywords locally, retrieve the encrypted column's matching items from the cloud, and decrypt them locally. Another example of property-revealing encryption is order-revealing encryption, which allows the order of two plaintexts to be computed from the ciphertexts. Order-revealing encryption facilitates sorting and range queries.

Property-revealing encryption is controversial. The CryptDB system, introduced by Popa, Redfiled, Zeldovich, and Balakrishnan in 2011, implements property-revealing encryption to enable the functionality of database management systems [30]. In 2015, Naveed, Kamara, and Wright presented attacks on columns encrypted under deterministic and order-revealing encryption, based on the design of CryptDB [28]. Naveed et al. demonstrated that an attacker with a well-correlated auxiliary database can perform an inference attack on encrypted database columns in order to recover private data. In response, Popa, Zeldovich, and Balakrishnan provided guidelines for the safe use of the CryptDB system in which they claimed deterministic encryption is safe to use for a sensitive field if every value in a column appears only once; deterministic encryption for non-unique fields is described as "allowing some leakage" [31]. But there does not currently exist a clear understanding of the leakage of these encryption schemes.

There is considerable interest in the field of encrypted databases: many cryptographic constructions are being considered [1, 7, 8, 13, 15, 16], different sophisticated attacks continue to be developed [9, 18, 21, 22, 32], and several academic workshops that focus on this topic have been established [19, 34]. Furthermore, there are existing commercial products that encrypt data to preserve application functionality such as Bitglass [5], CipherCloud [14], McAfee MVISION Cloud [24], Microsoft Always Encrypted [27], Netskope [29], and Symantec CloudSOC [36]. While encrypted database solutions are extremely attractive, at present their security implications are not well understood.

### 1.1 Contributions

In this paper, we undertake a detailed analysis of the leakage associated with deterministic encryption through the framework of *quantitative information flow* (QIF). QIF is an information-theoretic framework used to quantify the amount of information flow in a


*CCSW'19, November 11, 2019, London, United Kingdom*

ACM ISBN 978-1-4503-6826-1/19/11…$15.00
https://doi.org/10.1145/3338466.3358915

system. In contrast with machine learning approaches, QIF can provide concrete upper bounds that reflect the best results that an optimal adversary is able to achieve. A key insight from this framework is that there is no one "correct" way to measure leakage. Information flow depends on the operational scenario and different operational scenarios require different leakage measures. We contribute a leakage analysis under three different operational scenarios in order to develop a much clearer understanding of the information leakage associated with deterministic encryption.

Imagine a medical database in which there is a single column with one row per patient that consists of the patient's disease diagnosis. Let there be $n$ patients and $k$ possible diagnosis; assume $k$ is small. With strong encryption, the adversary is unable to determine anything about the ciphertext. With deterministic encryption, the adversary can see which entries are the same and possibly perform some type of inference attack. Given this scenario, we would like to quantify the leakage under different adversarial goals. For example, the adversary may try to guess the entire column of diseases; in Section 4, we see this corresponds to Bayes vulnerability. She could also try to guess the disease of a patient, either a particular patient $i$ or an arbitrary patient. These operational scenarios are evaluated in Section 5. We analyze leakage with respect to these three operational scenarios and under various prior distributions and as we will see, they are quite different. In some cases, there is no leakage at all and in others, there is quite a bit.

- In the scenario where the adversary attempts to guess the entire column, regardless of the distribution on diseases, there is a large amount of leakage for large $n$. For some disease distributions, the adversary can achieve a success probability close to 1, while for others, the adversary's success probability is far less than 1.
- In both scenarios where the adversary tries to guess the disease of a single patient, under a uniform distribution on diseases, there is no leakage at all.

### 1.2 Related Work

How to meaningfully discuss leakage within the field of encrypted databases is an open question. The approach to leakage began when Curtmola, Garay, Kamara and Ostrovsky defined a *trace* as the information one is willing to leak about the interaction between the client and the server [15]. Chase and Kamara then defined more precise *leakage functions* as a way to precisely capture what is being leaked by ciphertext and tokens [12]. This approach is first applied to order-preserving encryption by Chenette, Lewi, Weis, and Wu [13]. More recently, leakage is discussed in terms of *leakage patterns* that describe what data is revealed at a specific operation; these compose to form a scheme's *leakage profile* [20, 23]. The leakage profile is fundamental to a scheme's security definition, but the threat posed by a leakage profile is unclear [10, 23].

Despite the existence of leakage profiles, cryptoanalysis research has exploited different sets of intentionally revealed information to accomplish adversarial goals such as data or query recovery [9, 18, 21, 22, 28, 38]. The following is only a sample of cryptoanalysis research on property-revealing encryption. As discussed previously, Naveed et al. develop attacks in which a well-correlated auxiliary database is used to infer ciphertext values [28]. The adversarial goal is to recover as many plaintext values in a single encrypted column as possible, but the success of the attack depends on the degree to which the auxiliary database is correlated. Grubbs, Sekniqi, Bindschaedler, Naveed, and Ristenpart empirically evaluate order-revealing and order-preserving encryption but explicitly "leave as an open question providing a more formal analysis of inference attacks" [21]. Durak, DuBuisson, and Cash explore the behavior of leakage profiles on data under a non-uniform distribution and cross-column correlations but could not quantify the attacks theoretically [18]. Bindschaedler, Grubbs, Cash, Ristenpart, and Shmatikov also evaluate cross-column correlations and provide an attack that functions as a maximum likelihood estimator which maps plaintexts to ciphertexts given some prior distribution from an auxiliary database. [4]. This attack corresponds to the operational scenario in which the adversary attempts to guess as many positions correctly as possible; in this work, we consider three different scenarios.

Prior work has analyzed leakage and quantified security in the encrypted search domain. Sedghi, Doumen, Hartel, and Jonker provide an information-theoretic analysis of searchable encryption in which they evaluate three seminal schemes [35]. De Capitani Di Vimercati, Foresti, Jajodia, Paraboschi, and Samarati examine the leakage associated with data indexing over fragments, where indexing is defined as a function mapping plaintext values to obfuscated values. They enumerate ways in which combinations of indexing and adversary knowledge can recover sensitive information [17]. Ceselli et al. provide a graph-based approach to quantify the security provided by indexing [11]. Our work is the first application of the QIF framework to this domain.

*Organization.* Section 2 provides an overview of the QIF framework and Section 3 explains how deterministic encryption is modeled within this framework. Sections 4 and 5 evaluate deterministic encryption with respect to three different operational scenarios. Section 6 then discusses computational challenges specific to our leakage calculations and gives a detailed look at posterior vulnerability. Lastly, Section 7 discusses future directions and concludes.

## 2 QUANTITATIVE INFORMATION FLOW

In this section, we provide a brief overview of the key concepts of quantitative information flow (QIF). An excellent short introduction can be found in McIver [25] and a book-length treatment can be found in Alvim et al. [2]; this book includes all definitions and theorems mentioned in this section, as well as proofs for these theorems.

QIF is an information-theoretic approach that assumes that an adversary's prior knowledge about a secret input $X$ drawn from a finite set of possible secret values $\mathcal{X}$ is modeled as a probability distribution $\pi$. Intuitively, a uniform $\pi$ means that $X$ has more secrecy while a non-uniform $\pi$ means the secret has less secrecy. A system takes the secret $X$ as input and produces observable output $Y$, which may help the adversary achieve some goal, such as guessing the secret or a property of the secret.

We can statically model all possible inputs and outputs of the system as an information-theoretic channel matrix C that gives the conditional probabilities $p(y|x)$.

*Definition 2.1 (Channel Matrix).* Let $\mathcal{X}$ and $\mathcal{Y}$ be finite sets, intuitively representing input values and observable output values.

| $\pi$ | C | $y_1$ | $y_2$ | $y_3$ | $y_4$ |
|---|---|---|---|---|---|
| 1/4 | $x_1$ | 1/2 | 1/2 | 0 | 0 |
| 1/2 | $x_2$ | 0 | 1/4 | 1/2 | 1/4 |
| 1/4 | $x_3$ | 1/2 | 1/3 | 1/6 | 0 |

$\rightarrow$

| J | $y_1$ | $y_2$ | $y_3$ | $y_4$ |
|---|---|---|---|---|
| $x_1$ | 1/8 | 1/8 | 0 | 0 |
| $x_2$ | 0 | 1/8 | 1/4 | 1/8 |
| $x_3$ | 1/8 | 1/12 | 1/24 | 0 |

$\rightarrow$

| $[\pi \triangleright \mathsf{C}]$ | 1/4 | 1/3 | 7/24 | 1/8 |
|---|---|---|---|---|
| $x_1$ | 1/2 | 3/8 | 0 | 0 |
| $x_2$ | 0 | 3/8 | 6/7 | 1 |
| $x_3$ | 1/2 | 1/4 | 1/7 | 0 |

**Figure 1: An example of how a prior $\pi$ and a channel C are mapped to joint matrix J and then to hyper-distribution $[\pi \triangleright \mathsf{C}]$**

.

A *channel matrix* C *from* $\mathcal{X}$ *to* $\mathcal{Y}$ is a matrix, indexed by $\mathcal{X} \times \mathcal{Y}$, whose rows give the distribution on outputs corresponding to each possible input. That is, entry $\mathsf{C}_{x,y}$ denotes $p(y|x)$, the conditional probability of getting output $y$ given input $x$. Note that all entries in a channel matrix are between 0 and 1 and each row sums to 1. □

We assume the adversary knows C and can update her knowledge about $X$ to a posterior distribution $p_{X|y}$ given each output $y$. Each output $y$ also has a probability $p(y)$ of occurring. The fundamental insight is that the information-theoretic essence of a channel matrix C is a mapping from priors $\pi$ to *distributions on posterior distributions*, which we call hyper-distributions and denote $[\pi \triangleright \mathsf{C}]$. This mapping is called the *abstract channel* denoted by C.

Figure 1 provides an example of how channel matrix C maps a prior $\pi$ to hyper-distribution $[\pi \triangleright \mathsf{C}]$. First we form the joint matrix J, where $\mathsf{J}_{x,y}$ gives the joint probability of each input-output pair. Note that J is computed by multiplying row $x$ of C by the prior probability $\pi_x$. Next we find the marginal distribution $p_Y$ by summing the columns of J; thus here we get $p_Y = (1/4, 1/3, 7/24, 1/8)$. Now each possible value of $Y$ gives a posterior distribution on $X$, by Bayesian updating. Those posterior distributions can be calculated by normalizing the columns of J; the posterior distributions here are $(1/2, 0, 1/2)$, $(3/8, 3/8, 1/4)$, $(0, 6/7, 1/7)$, and $(0, 1, 0)$.

We can imagine these posterior distributions as *worlds* that an adversary seeing the output of C could end up in; we also refer to these posterior distributions as *inner distributions*. It is important to realize these worlds are not equally likely. These inner distributions themselves have probabilities of occurring, which we call the *outer distribution*; in this example, the outer distribution is $(1/4, 1/3, 7/24, 1/8)$. Notice in the last world, the adversary knows that the secret is $x_2$. However, this world only occurs with probability 1/8. It is more likely that the adversary ends up in a different world in which she is less sure about the secret. The fundamental effect of the channel is to enable each output $y$ to provide the adversary with different knowledge about the secret $X$. Finally, we observe that the particular output labels $y_1, y_2, \ldots$ do not matter at all, as renaming them to $z_1, z_2, \ldots$ would have no effect on the leakage. As a result, the essence of the effect of channel C on prior $\pi$ is simply the hyper-distribution $[\pi \triangleright \mathsf{C}]$ shown on the right-hand side of Figure 1.

## 2.1 Gain Functions and $g$-Vulnerability

We can use gain functions to measure the *vulnerability* of $X$ with respect to specific operational scenarios. We define gain functions in the following way:

*Definition 2.2 (Gain function).* Given a finite, non-empty set $\mathcal{X}$ (of possible secret values) and a non-empty set $\mathcal{W}$ (of possible actions), a *gain function* is a function $g : \mathcal{W} \times \mathcal{X} \to \mathbb{R}$. □

We let $g(w, x)$ dictate the adversary's gain for performing action $w$ when the secret's value is $x$. Often, actions are guesses the adversary could make about the secret, but sometimes the best action is to not make any guess at all, as in a scenario where making an incorrect guess triggers a penalty.

An obvious and often relevant gain function addresses the adversarial goal of guessing the entire secret correctly in one try; we refer to this as the identity gain function denoted $g_{\mathrm{id}}$. In this scenario, the set of actions available to the adversary is equal to the set of secrets $\mathcal{W} = \mathcal{X}$. The adversary will receive a gain of 1 if she guesses the entire secret correctly and 0 otherwise. The gain function $g_{\mathrm{id}}$ is defined in the following way:

*Definition 2.3 (Identity gain function).* The *identity* gain function $g_{\mathrm{id}} : \mathcal{X} \times \mathcal{X} \to \{0, 1\}$ is given by

$$g_{\mathrm{id}}(w, x) := \begin{cases} 1, & \text{if } w = x, \\ 0, & \text{if } w \neq x \end{cases}$$

□

More generally, gain functions allow us to express a variety of operational scenarios. For example, we can model the case in which the adversary is allowed to make $k$ guesses to correctly guess the value of $X$. Or we can model the case in which the adversary must guess whether or not $X$ satisfies some property. We can also model the case in which the adversary is penalized for making an incorrect guess.

Given a gain function $g$, an optimal adversary will choose an action that maximizes her expected gain with respect to $\pi$. We define $g$-vulnerability in the following way:

*Definition 2.4 ($g$-Vulnerability).* The $g$-vulnerability of $\pi$ is defined as

$$V_g(\pi) := \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \pi_x g(w, x)$$

□

We refer to this as *prior $g$-vulnerability* as it represents how vulnerable the secret is before the channel is run. For example, notice that the $g_{\mathrm{id}}$-vulnerability of $\pi$ is $\max_x \pi_x$. Intuitively, an adversary attempting to guess the secret will guess a value with highest probability of occurring. Because $V_{g_{\mathrm{id}}}(\pi)$ represents such a basic security concern, we have for it a special terminology *Bayes vulnerability* and special notation $V_1(\pi)$; this notation emphasizes that we are focusing on guessing the secret in one try.

To measure the vulnerability of the secret after the channel is run, we define *posterior $g$-vulnerability* as the average $g$-vulnerability in the hyper-distribution:

*Definition 2.5 (Posterior g-Vulnerability).* Given a prior $\pi$, gain function $g$ and channel matrix C from $\mathcal{X}$ to $\mathcal{Y}$, the *posterior g-vulnerability* $V_g[\pi \triangleright \mathsf{C}]$ is

$$V_g[\pi \triangleright \mathsf{C}] := \sum_{\substack{y \in \mathcal{Y} \\ p(y) \neq 0}} p(y) V_g(p_{X|y})$$

□

It is a theorem (from Alvim et al. [3]) that posterior $g$-vulnerability can be calculated directly from the channel matrix without calculating the posterior distributions:

Theorem 2.6. *Given prior $\pi$, gain function $g$, and channel matrix* C *from $\mathcal{X}$ to $\mathcal{Y}$, we have*

$$V_g[\pi \triangleright \mathsf{C}] = \sum_{y \in \mathcal{Y}} \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \pi_x \mathsf{C}_{x,y} g(w, x)$$

□

## 2.2 Leakage

The prior $g$-vulnerability $V_g(\pi)$ represents how vulnerable the secret is without information from the channel. The posterior $g$-vulnerability $V_g[\pi \triangleright \mathsf{C}]$ represents how vulnerable the secret will be to an adversary who can see the output of the channel. Given that, it is natural to measure the leakage of the channel by comparing the prior $g$-vulnerability with the posterior $g$-vulnerability. This comparison can be done additively or multiplicatively.

*Definition 2.7 (Multiplicative and Additive g-Leakage).* Given prior probability distribution $\pi$, a gain function $g$, and channel C, the *multiplicative g-leakage* is

$$\mathcal{L}^{\times}_g(\pi, \mathsf{C}) := \frac{V_g[\pi \triangleright \mathsf{C}]}{V_g(\pi)}$$

and the *additive g-leakage* is

$$\mathcal{L}^{+}_g(\pi, \mathsf{C}) := V_g[\pi \triangleright \mathsf{C}] - V_g(\pi)$$

In the case of $g_{\mathrm{id}}$, we use the name *Bayes leakage* and the notation $\mathcal{L}^{\times}_1(\pi, \mathsf{C})$ and $\mathcal{L}^{+}_1(\pi, \mathsf{C})$. □

It is a theorem (from Alvim et al. [3]) that posterior $g$-vulnerability is always greater than or equal to prior $g$-vulnerability:

Theorem 2.8. *Posterior g-vulnerability is always greater than or equal to prior g-vulnerability: for any prior $\pi$, channel* C *and gain function g, we have $V_g[\pi \triangleright \mathsf{C}] \geq V_g(\pi)$.* □

This implies that multiplicative leakage is always greater or equal to 1[1] and additive leakage is greater or equal to 0. If equality holds, there is no leakage.

A crucial insight of QIF is that there is not a single "right" way to measure leakage. There are many possible leakage measures captured by the $g$-leakage family. The precise leakage is dependent on the gain function $g$ and the prior distribution $\pi$.

[1]We assume that gain functions are restricted so that $g$-vulnerability is always non-negative.

## 2.3 Refinement

A common situation when comparing two channels A and B is that under some conditions channel A leaks more, and in others channel B leaks more. Hence for robustness it is desirable to determine when one channel *never* leaks more than another. This leads to the notion of *refinement*:

*Definition 2.9 (Refinement).* Given channels A and B over input space $\mathcal{X}$, we say that A is *refined by* B, written $\mathsf{A} \sqsubseteq \mathsf{B}$, if for any prior $\pi$ and gain function $g$ we have $V_g[\pi \triangleright \mathsf{A}] \geq V_g[\pi \triangleright \mathsf{B}]$. □

Remarkably, the *Coriaceous Theorem* (from McIver et al. [26], though actually dating from the 1950's and the work of David Blackwell [6]) shows that refinement has a structural characterization:

Theorem 2.10 (Coriaceous Theorem). *For channel matrices* A *and* B *over input space $\mathcal{X}$, we have* $\mathsf{A} \sqsubseteq \mathsf{B}$ *iff there exists a post-processing channel matrix* R *such that* $\mathsf{B} = \mathsf{AR}$. □

As an example, consider the following channels (from [26]):

| A | $z_1$ | $z_2$ | $z_3$ |
|---|---|---|---|
| $x_1$ | 2/5 | 0 | 3/5 |
| $x_2$ | 1/10 | 3/4 | 3/20 |
| $x_3$ | 1/5 | 1/2 | 3/10 |

| B | $y_1$ | $y_2$ | $y_3$ |
|---|---|---|---|
| $x_1$ | 1 | 0 | 0 |
| $x_2$ | 1/4 | 1/2 | 1/4 |
| $x_3$ | 1/2 | 1/3 | 1/6 |

While A and B look very different, it turns out that they actually refine each other—we have both $\mathsf{A} \sqsubseteq \mathsf{B}$ and $\mathsf{B} \sqsubseteq \mathsf{A}$, as shown here:

| B | $y_1$ | $y_2$ | $y_3$ |
|---|---|---|---|
| $x_1$ | 1 | 0 | 0 |
| $x_2$ | 1/4 | 1/2 | 1/4 |
| $x_3$ | 1/2 | 1/3 | 1/6 |

=

| A | $z_1$ | $z_2$ | $z_3$ |
|---|---|---|---|
| $x_1$ | 2/5 | 0 | 3/5 |
| $x_2$ | 1/10 | 3/4 | 3/20 |
| $x_3$ | 1/5 | 1/2 | 3/10 |

| $R_1$ | $y_1$ | $y_2$ | $y_3$ |
|---|---|---|---|
| $z_1$ | 1 | 0 | 0 |
| $z_2$ | 0 | 2/3 | 1/3 |
| $z_3$ | 1 | 0 | 0 |

and

| A | $z_1$ | $z_2$ | $z_3$ |
|---|---|---|---|
| $x_1$ | 2/5 | 0 | 3/5 |
| $x_2$ | 1/10 | 3/4 | 3/20 |
| $x_3$ | 1/5 | 1/2 | 3/10 |

=

| B | $y_1$ | $y_2$ | $y_3$ |
|---|---|---|---|
| $x_1$ | 1 | 0 | 0 |
| $x_2$ | 1/4 | 1/2 | 1/4 |
| $x_3$ | 1/2 | 1/3 | 1/6 |

| $R_2$ | $z_1$ | $z_2$ | $z_3$ |
|---|---|---|---|
| $y_1$ | 2/5 | 0 | 3/5 |
| $y_2$ | 0 | 1 | 0 |
| $y_3$ | 0 | 1 | 0 |

As a result, we have by Theorem 2.10 that B never leaks more than A, and also that A never leaks more than B. As a result, A and B always have exactly the *same* leakage.[2] Note that when we model deterministic encryption in Section 3, we will encounter exactly this phenomenon of two channels that look very different but actually have the same leakage.

# 3 DETERMINISTIC ENCRYPTION MODEL

*Secret.* We define $X$ to be a database column consisting of plaintext values that are independently chosen at each index according to a distribution $\delta$. We therefore assume the distribution $\pi$ on columns comes from an underlying distribution of values $\delta$.

[2]In fact, because refinement is a *partial order* on abstract channels [26], A and B actually denote the *same* abstract channel.

*Channel.* A dilemma that we face is that QIF is information-theoretic, while encryption assumes a computationally-bounded adversary. As a result, we cannot directly model the deterministic encryption channel that maps a column of plaintexts $(x_1, x_2, \ldots, x_n)$ to the column of ciphertexts $(c_1, c_2, \ldots, c_n)$ under some randomly-chosen key — the trouble is that the resulting channel matrix leaks everything, since each of its columns contains just one non-zero entry (as is necessary for decryption to be possible).

Fortunately, cryptographers often consider an "ideal object" that is conjectured to be computationally indistinguishable from the actual cryptographic scheme, and the ideal object may be suitable for QIF analysis. For deterministic encryption of $b$-bit strings, the ideal object is a *random permutation* of type $\{0,1\}^b \to \{0,1\}^b$, so that all such functions are equally likely to be chosen.[3] As far as implementation is concerned, note that if our plaintexts are 128 bits long, then it suffices to encrypt them with AES under a randomly-chosen key. Dealing with longer plaintexts requires more care.[4]

The ideal object is hence a probabilistic channel I that maps a column $(x_1, x_2, \ldots, x_n)$ of plaintexts to a column $(v_1, v_2, \ldots, v_n)$ of random independent values, subject only to the constraint that equality is preserved: $x_i = x_j$ iff $v_i = v_j$.

While we could analyze the leakage of I directly, we can make the analysis easier by observing that I's leakage is exactly the *same* as a deterministic channel C that maps $(x_1, x_2, \ldots, x_n)$ to a *partition* of the indices $\{1, 2, \ldots, n\}$ where each block in the partition consists of those indices for which the corresponding $x$-values are equal. (For example, $(a, b, b, c, a)$ maps to the partition $\{\{1, 5\}, \{2, 3\}, \{4\}\}$.) For we observe that I and C refine each other: $\mathsf{I} \sqsubseteq \mathsf{C}$ and $\mathsf{C} \sqsubseteq \mathsf{I}$. For C can be factored into a cascade of I followed by a post-processing channel R that maps a tuple $(v_1, v_2, \ldots, v_n)$ to the partition of $\{1, 2, \ldots, n\}$ based on the equalities among the $v_i$'s. Similarly, $\mathsf{I} = \mathsf{CS}$, where S maps a partition into a tuple $(v_1, v_2, \ldots, v_n)$ of independent random values that respects the partition (i.e. $v_i = v_j$ iff $i$ and $j$ belong to the same block). Because I and C refine each other, by Theorem 2.10 their leakage is always exactly the same.[5]

*Running Example.* Our running example is a database column of medical diagnoses where every index corresponds to a patient. We depict this database column as a tuple; for example, $(a, c, b, a)$ would indicate that the first patient has disease $a$, the second patient has $c$, and so forth. Let $n$ be the length of the column (the number of patients) and let there be $k$ possible values (the number of possible diseases). Figure 2 illustrates the channel matrix given three possible diseases $a$, $b$, $c$ ($k = 3$) and three patients ($n = 3$). The channel matrix has 27 rows and 5 columns.

| C | $y \in \mathcal{Y}$ | | | | |
|---|---|---|---|---|---|
| $x \in \mathcal{X}$ | {1, 2, 3} | {1, 2} {3} | {1, 3} {2} | {2, 3} {1} | {1} {2} {3} |
| (a, a, a) | 1 | 0 | 0 | 0 | 0 |
| (a, a, b) | 0 | 1 | 0 | 0 | 0 |
| (a, a, c) | 0 | 1 | 0 | 0 | 0 |
| (a, b, a) | 0 | 0 | 1 | 0 | 0 |
| (a, b, b) | 0 | 0 | 0 | 1 | 0 |
| (a, b, c) | 0 | 0 | 0 | 0 | 1 |
| (a, c, a) | 0 | 0 | 1 | 0 | 0 |
| (a, c, b) | 0 | 0 | 0 | 0 | 1 |
| (a, c, c) | 0 | 0 | 0 | 1 | 0 |
| (b, a, a) | 0 | 0 | 0 | 1 | 0 |
| (b, a, b) | 0 | 0 | 1 | 0 | 0 |
| (b, a, c) | 0 | 0 | 0 | 0 | 1 |
| (b, b, a) | 0 | 1 | 0 | 0 | 0 |
| (b, b, b) | 1 | 0 | 0 | 0 | 0 |
| (b, b, c) | 0 | 1 | 0 | 0 | 0 |
| (b, c, a) | 0 | 0 | 0 | 0 | 1 |
| (b, c, b) | 0 | 0 | 1 | 0 | 0 |
| (b, c, c) | 0 | 0 | 0 | 1 | 0 |
| (c, a, a) | 0 | 0 | 0 | 1 | 0 |
| (c, a, b) | 0 | 0 | 0 | 0 | 1 |
| (c, a, c) | 0 | 0 | 1 | 0 | 0 |
| (c, b, a) | 0 | 0 | 0 | 0 | 1 |
| (c, b, b) | 0 | 0 | 0 | 1 | 0 |
| (c, b, c) | 0 | 0 | 1 | 0 | 0 |
| (c, c, a) | 0 | 1 | 0 | 0 | 0 |
| (c, c, b) | 0 | 1 | 0 | 0 | 0 |
| (c, c, c) | 1 | 0 | 0 | 0 | 0 |

**Figure 2: Channel matrix C when $n = 3$ and $k = 3$**

## 4 BAYES VULNERABILITY ANALYSIS

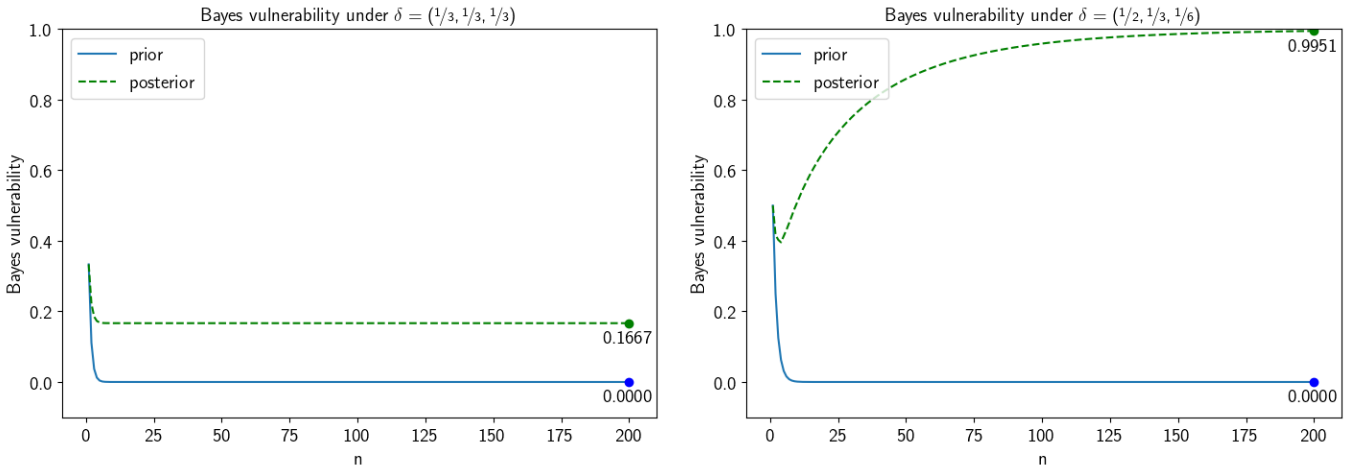
The first operational scenario we consider is Bayes vulnerability in which the adversary attempts to guess the entire column. As an intuition behind how an adversary would be able to do this, suppose that the distribution on diseases is non-uniform such that $\delta(a) = 1/2$, $\delta(b) = 1/3$, and $\delta(c) = 1/6$. For large $n$, we would expect that the partition would have one block that is large, one that is medium sized, and another that is small. Provided with this distribution on diseases, there is an obvious guessing strategy: that the largest block corresponds to $a$, the medium block corresponds to $b$, and the smallest block corresponds to $c$. We can understand Bayes vulnerability precisely by graphing the prior and posterior Bayes vulnerability as a function of $n$. In order to limit the complexity of the graphs while providing some insights, we limit ourselves to three diseases ($k = 3$) which we call $a$, $b$, and $c$.

While we could explicitly graph the additive or multiplicative leakage, a graph of the prior and posterior Bayes vulnerability is more revealing. In a real world scenario, we expect one would encounter a variety of distributions that illustrate many characteristics. The distributions seen here were chosen because they are simple, easy to understand, and illustrate the phenomena we expect with more complicated distributions. In Figures 3 and 4, we graph the prior and posterior Bayes vulnerabilities under four disease distributions $\delta$: 1) a uniform distribution, 2) an arbitrarily chosen non-uniform distribution, 3) a distribution in which two diseases have the same probability, and 4) a distribution in which

[3] This was an explicit criterion in the NIST AES design competition: "Algorithms will be judged on … the extent to which the algorithm output is indistinguishable from a random permutation."

[4] Section 3.1 of the CryptDB paper [30] describes a way to do this correctly.

[5] Like channels A and B in Section 2, I and C actually denote the same abstract channel.




(a) (b)

**Figure 3: Prior and posterior Bayes vulnerability under (a) a uniform distribution $\delta = (1/3, 1/3, 1/3)$ and (b) a non-uniform distribution $\delta = (1/2, 1/3, 1/6)$**

two diseases are very close but not the same. (Please note that the graphs are annotated with rounded values.)

## 4.1 Uniform vs Non-uniform

Let us first examine the prior and posterior Bayes vulnerability under a uniform distribution of diseases $\delta = (1/3, 1/3, 1/3)$ depicted in Figure 3(a). The solid blue line indicates an adversary who only knows the prior distribution attempting to guess the entire column. At $n = 1$, the prior Bayes vulnerability is $1/3$. Given a single disease to guess about, the adversary can guess $a$, $b$, or $c$ and be correct with equal probability. As $n$ increases, the adversary will have to guess the disease at every index and her probability of success quickly goes towards 0, exactly $(1/3)^n$. The dashed green line represents the posterior Bayes vulnerability which corresponds to how well the adversary can do given the output which indicates the way the column is broken into blocks. We can observe that the posterior curve is going to $1/6$. Once $n$ is large, we can generally expect three blocks. The adversary must guess which is which and there are 6 possible ways the diseases could be allocated to the three blocks.[6] There is leakage under a uniform prior because the output has made the adversary's task much easier.

We can contrast this graph with Figure 3(b) which displays the prior and posterior Bayes vulnerability under a non-uniform distribution of diseases $\delta = (1/2, 1/3, 1/6)$. The prior Bayes vulnerability again goes towards 0, but on the posterior side, as $n$ grows, the output is likely to consist of a large block, a medium block, and a small block which the adversary will be able to reliably map to $a$, $b$, and $c$ and her success will go to 1.

[6]While most index partitions where $k = 3$ can result from six possible plaintext columns, the one exception is the partition with a single block which corresponds to only three possible columns. This results in a posterior Bayes vulnerability that is slightly higher than $1/6$. For example, at $n = 4$, the posterior Bayes vulnerability is roughly 0.173.

Comparing Figure 3(a) and Figure 3(b), we can see that additive Bayes leakage is clearly higher in Figure 3(b). Surprisingly, the multiplicative Bayes leakage is higher in Figure 3(a), although this is not visible. While prior Bayes vulnerabilities under both priors are close to zero, the prior vulnerability under the uniform prior is much smaller than the prior vulnerability under the non-uniform prior.

Let us examine when $n = 200$. Under the non-uniform prior, the prior Bayes vulnerability is $(1/2)^{200}$ and the posterior Bayes vulnerability is close to 1 therefore the multiplicative leakage is close to $2^{200} \approx 1.61 \times 10^{60}$. In contrast, under the uniform prior, the prior Bayes vulnerability is $(1/3)^{200}$ and the posterior Bayes vulnerability is around $1/6$. Therefore, the multiplicative leakage under the uniform prior is close to $1/6 \times 3^{200} \approx 4.43 \times 10^{94}$ and orders of magnitude higher than under the non-uniform prior.

## 4.2 Same vs Close

Now let us compare the prior and posterior Bayes vulnerability when the probabilities of two diseases are the same and when they are very close, illustrated in Figure 4 .

Under $\delta = (1/2, 1/4, 1/4)$, the posterior Bayes vulnerability in Figure 4(a) approaches $1/2$. In terms of blocks, we expect one large block and two roughly equal-sized blocks. The adversary will likely be able to distinguish disease $a$ as it is expected to be the largest block, but she will have an equal chance of guessing which small block is disease $b$ and which is $c$. However, because it is possible that the block sizes will not conform to the distribution, the posterior Bayes vulnerability at $n = 200$ is slightly less than 0.5 at roughly 0.499986.

In comparison, Figure 4(b) shows that under $\delta = (1/2, 26/100, 24/100)$, the posterior Bayes vulnerability slowly continues to grow, apparently converging to 1. The largest $n$ value for which we have done the calculation is $n = 5000$, at which point the posterior Bayes

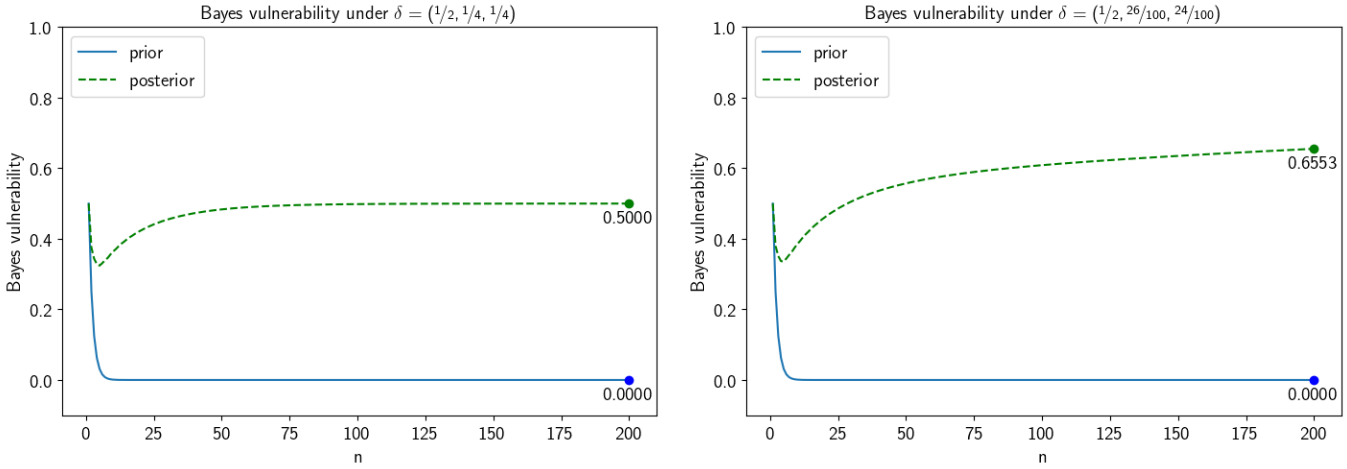



**Figure 4: Prior and posterior Bayes vulnerability when (a) two diseases have the same probability $\delta = (1/2, 1/4, 1/4)$ and (b) two diseases have close probabilities $\delta = (1/2, 26/100, 24/100)$**

**Table 1: Top 10 major diagnostic categories at a large inpatient acute palliative care service [33].**

| Diagnosis | Percentage |
|---|---|
| Cancer | 41.3% |
| Cardiac Disease | 17.4% |
| Pulmonary Disease | 14.0% |
| Stroke | 9.4% |
| Renal Disease | 3.5% |
| Dementia | 2.4% |
| Liver Disease | 1.7% |
| AIDS | 0.4% |
| Lou Gehrig's Disease | 0.2% |
| Other | 9.6% |

vulnerability is 0.977.[7] As $n$ grows, the small probability differences between $b$ and $c$ will become more visible in the block sizes and the adversary will be more likely to correctly assign them to their respective diseases.

### 4.3 Towards a More Realistic Distribution

As a gesture towards considering more realistic distributions, Table 1 shows a distribution of the top 10 most prevalent diagnostic categories at a large inpatient acute palliative care service [33] and Figure 5 graphs the prior and posterior Bayes vulnerability under that distribution.

The phenomenon we observed from the simple non-uniform distribution in Section 4.1 is visible in this more complicated distribution: as $n$ increases, the adversary's ability to correctly guess

[7]The computation for such large $n$ values becomes very expensive due to combinatorial explosion. This is discussed further in Section 6.

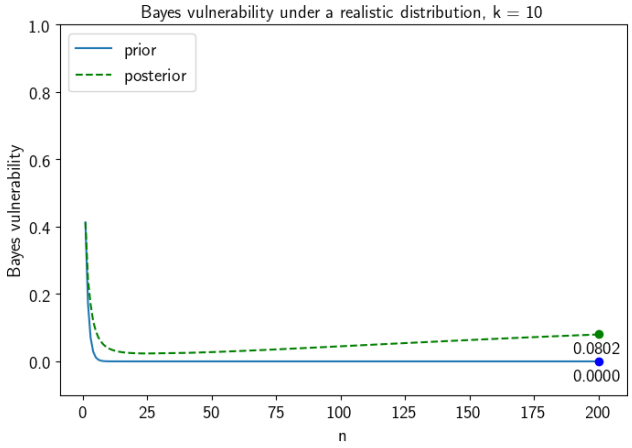



**Figure 5: Prior and posterior Bayes vulnerability of ten diagnostic categories when $\delta$ is set according to Table 1.**

the column increases. As can be seen in Figure 5, when $n = 200$ the adversary has an 8% chance of guessing the entire column correctly.

## 5 SINGLE INDEX GAIN FUNCTION ANALYSIS

Next, we examine the case in which an adversary attempts to guess the disease corresponding to a single index representing a single patient. We evaluate two variations of this scenario: (1) the adversary is free to choose any index corresponding to a patient and guess their disease and (2) the adversary is forced to guess the disease at a given index belonging to a particular patient.

For the *free* gain function, the adversary is free to choose an index in the column and a disease $d \in \mathcal{D}$. Intuitively, this scenario reflects the case when an adversary does not care about discovering

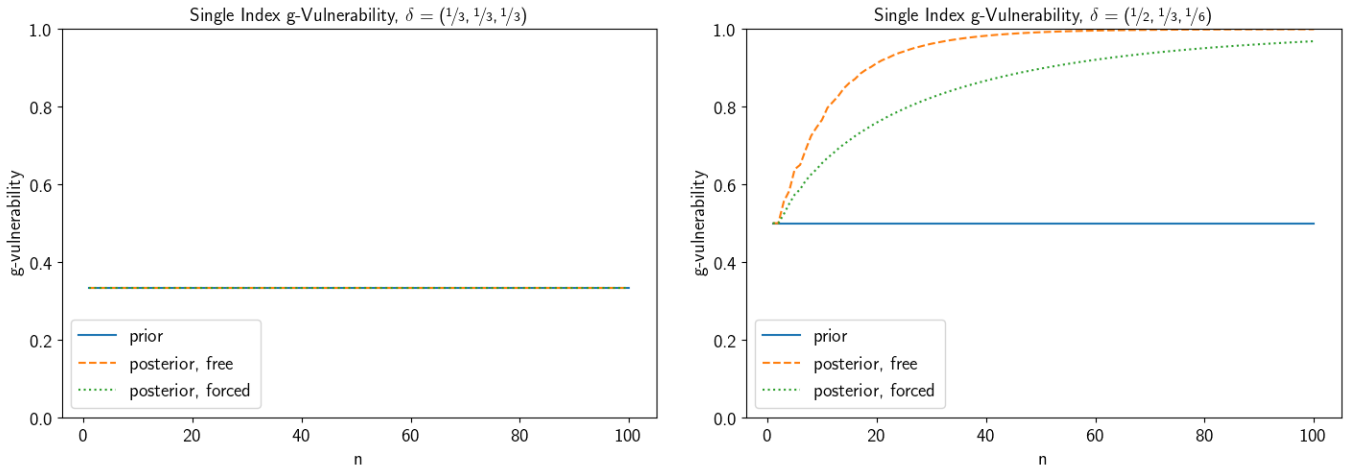



**Figure 6: Prior and posterior $g$-vulnerability for both single index gain functions under (a) a uniform distribution $\delta = (1/3, 1/3, 1/3)$ and (b) a non-uniform distribution $\delta = (1/2, 1/3, 1/6)$**

a particular patient's disease but instead is content with discovering anyone's diagnosis.

*Definition 5.1 (Free gain function).* The *free* gain function $g_{\text{free}} : \mathcal{W} \times \mathcal{X} \to \{0, 1\}$ when $\mathcal{W} := \{(i, d) \mid 1 \leq i \leq n \wedge d \in \mathcal{D}\}$ is given by

$$g_{\text{free}}((i, d), x) := \begin{cases} 1 & \text{if } x_i = d \\ 0 & \text{otherwise} \end{cases}$$

□

For the *forced* variation of this gain function, the adversary is given a specific index $i$ where $1 \leq i \leq n$ and is forced to guess a disease at this index. Intuitively, this gain function reflects the scenario in which the adversary wants to discover the disease of a high-value patient.

*Definition 5.2 (Forced gain function).* The *forced* gain function $g_{\text{forced}} : \mathcal{W} \times \mathcal{X} \to \{0, 1\}$ when $\mathcal{W} := \{d \mid d \in \mathcal{D}\}$ is given by

$$g_{\text{forced}}(d, x) := \begin{cases} 1 & \text{if } x_i = d \\ 0 & \text{otherwise} \end{cases}$$

□

The prior $g$-vulnerability for both variations is always the same. The adversary's strategy in both scenarios is simply to guess the most likely disease; the only difference is that she can choose an index for the free variation, but without insight from the channel, it makes no difference as to which index she picks. The probability that she is correct is equal to the probability of the most likely disease and will be the same in both cases. Therefore, we can graph the prior $g$-vulnerability alongside the posterior $g$-vulnerability for both operational scenarios.

Note that under both distributions, the prior $g$-vulnerability in Figure 6 is much higher than the prior Bayes vulnerability in Figure 3. Intuitively, this is because the task of guessing about a single index correctly in one try is much easier than guessing correctly about all $n$ indices.

As Figure 6(a) demonstrates, under a uniform prior the three lines coincide; this indicates that deterministic encryption leaks nothing with respect to both single index gain functions. The channel provides the adversary with no additional information to determine which block corresponds to which disease, and so whether she is forced to guess about a particular index or is free to choose one, she will only be correct with probability 1/3. As discussed previously, given a uniform distribution on diseases, the additive leakage in the Bayes scenario was approximately 1/6 but the additive leakage in the single index scenarios is 0.

In the non-uniform case represented by Figure 6(b), the solid blue line indicates that there is a 1/2 chance of guessing an index correctly a priori. In the free scenario, the adversary can guess about whatever block is most advantageous to guess about and the posterior $g$-vulnerability goes up to 1 very quickly. But in the forced scenario, the adversary may be forced to guess about a block she is not sure about, and so the vulnerability goes up more slowly.

As we can see, the prior distribution can drastically affect the leakage of the channel. The two prior distributions chosen here show many aspects of the story but one could examine different priors or a greater number of diseases which could exhibit other details.

## 6 DISCUSSION

The following section describes some of the computational challenges involved with calculating prior and posterior $g$-vulnerability, the differences in technique behind calculating the free and forced $g$-functions, as well as a more detailed look at what comprises the $g$-vulnerability for a given $n$.

**Table 2: Number of corresponding integer and index partitions per $n$ where $k = 3$**

| $n$ | Integer Partitions | Index Partitions |
|---|---|---|
| 1 | 1 | 1 |
| 50 | 234 | $1.20 \times 10^{23}$ |
| 100 | 884 | $8.59 \times 10^{46}$ |
| 150 | 1,951 | $6.17 \times 10^{70}$ |
| 200 | 3,434 | $4.23 \times 10^{94}$ |

## 6.1 Computational Challenges

As discussed in Section 2, prior and posterior $g$-vulnerability can be calculated from the channel matrix. However, the channel matrices given a reasonably large $n$ become enormous. For example, let $n = 100$ and let $k = 3$. The resulting channel matrix has $3^{100} \approx 5.15 \times 10^{47}$ rows. To calculate the number of columns in the channel matrix (which represent all possible channel outputs), we need to count the number of ways that $n$ items can be partitioned into at most $k$ non-empty subsets. To this end, we can use a summation of Stirling Numbers of the Second Kind [37]. The number of columns in the channel matrix can then be calculated as follows:

$$|\mathcal{Y}| = \sum_{i=1}^{\min(n,k)} \left\{ {n \atop i} \right\}$$

Therefore, the number of columns in C when $n = 100$ and $k = 3$ is approximately $8.59 \times 10^{46}$. Given the size of this matrix, the computation of prior and posterior $g$-vulnerability cannot be done naïvely.

We can avoid addressing each index partition in isolation by observing that the position of specific indices (the location of each disease) does not affect the adversary's strategy. Instead, the adversary's strategy is reliant on just the *sizes* of each block. We can represent these block sizes as an *integer partition* of $n$ into at most $k$ integers.

Briefly, let us examine the case with 4 patients ($n = 4$) and 3 unique diseases ($k = 3$). An adversary could observe the following blocks that indicate the first two diseases are the same and the second two are the same: $\{1, 2\}\{3, 4\}$. She could also observe $\{1, 3\}\{2, 4\}$ or $\{1, 4\}\{2, 3\}$. All of these observations correspond to the integer partition [2, 2, 0] and will generate the same probabilities.

Since index partitions that correspond to the same integer partition generate the same probabilities, the key strategy to calculate prior and posterior $g$-vulnerability is to use integer partitions instead of index partitions. In the previous example where $n = 100$ and $k = 3$, only 884 integer partitions can account for all index partitions. Table 2 illustrates that as the number of index partitions grows explosively, the number of corresponding integer partitions remains manageable. However, it should be acknowledged that as $k$ grows, the number of index partitions and integer partitions grow even more explosively. For example, when $k = 25$ and $n = 100$, there are 139,620,591 integer partitions that account for $4.37 \times 10^{114}$ index partitions. But for small values of $k$, integer partitions provide a mechanism by which we can work out the probabilities in a tractable fashion.

## 6.2 Free vs Forced

For both single index $g$-functions, given an integer partition that represents block sizes, we determine the best guess for every block and its probability of being correct. For the free variation, we choose the maximum over all probabilities, but in the forced variation, we calculate a weighed average that addresses the probability that the adversary will be asked to guess about an index in that block. One can view the forced variation as restricting the adversary's choices; instead of being able to guess the optimal answer given a particular channel output, she may be forced to guess about a block she is not sure about.

## 6.3 A Detailed Look For a Given $n$

If we examine the uppermost dashed orange line in Figure 6(b), we can see that when $n = 12$ and $\delta = (1/2, 1/3, 1/6)$, the posterior $g$-vulnerability in the free scenario is slightly above 0.8. How this value is calculated is quite complicated. There are many possible outputs that can arise and these outputs have different threats that they represent. Figure 7 illustrates the complexity behind this value by depicting all possible outputs and their respective threats to the secret.

Given 12 patients and 3 diseases, there are 88,574 possible index partitions that can be output by the channel but these can be accounted for with only 19 integer partitions. These integer partitions represent the block structures that the adversary could observe. Figure 7 visually represents these block structures as stacked bars such that the first blue bar represents the largest block, the next orange bar represents the second largest block, and the last green bar represents the smallest block.

For every integer partition, annotated purple circles indicate which block and disease guess the adversary should make given that output while the position of these circles indicate the adversary's probability of being correct. For example, let us examine the integer partition [12, 0, 0] which represents the case that every patient has the same disease. Given this partition, the adversary should guess about any index in this block and she should guess that this index corresponds to disease $a$; this guess will be correct 99.2% of the time. In contrast, if the adversary observes the integer partition [6, 5, 1], her best option is to guess that the index represented by smallest block of size 1 corresponds to disease $c$.

The purple line at the bottom of the graph indicates the probability of each block structure actually occurring. As stated previously, given the integer partition [12, 0, 0], the adversary is very likely to correctly guess $a$. Unfortunately for the adversary, this outcome is very unlikely; she is much more likely to see an integer partition that more closely reflects the distribution such as [6, 4, 2].

Posterior $g$-vulnerability summarizes this table by calculating the adversary's probability of being correct (the purple circles) weighed by the probability that the output occurs (the purple line). In this case where $n = 12$, the posterior $g$-vulnerability for the free variation is approximately 0.813. Recall that the prior $g$-vulnerability at $n = 12$ was 0.5. The posterior $g$-vulnerability is greater than the prior, which is consistent with the theorem that states the posterior $g$-vulnerability is always greater than or equal to prior $g$-vulnerability. Yet while the overall posterior $g$-vulnerability can

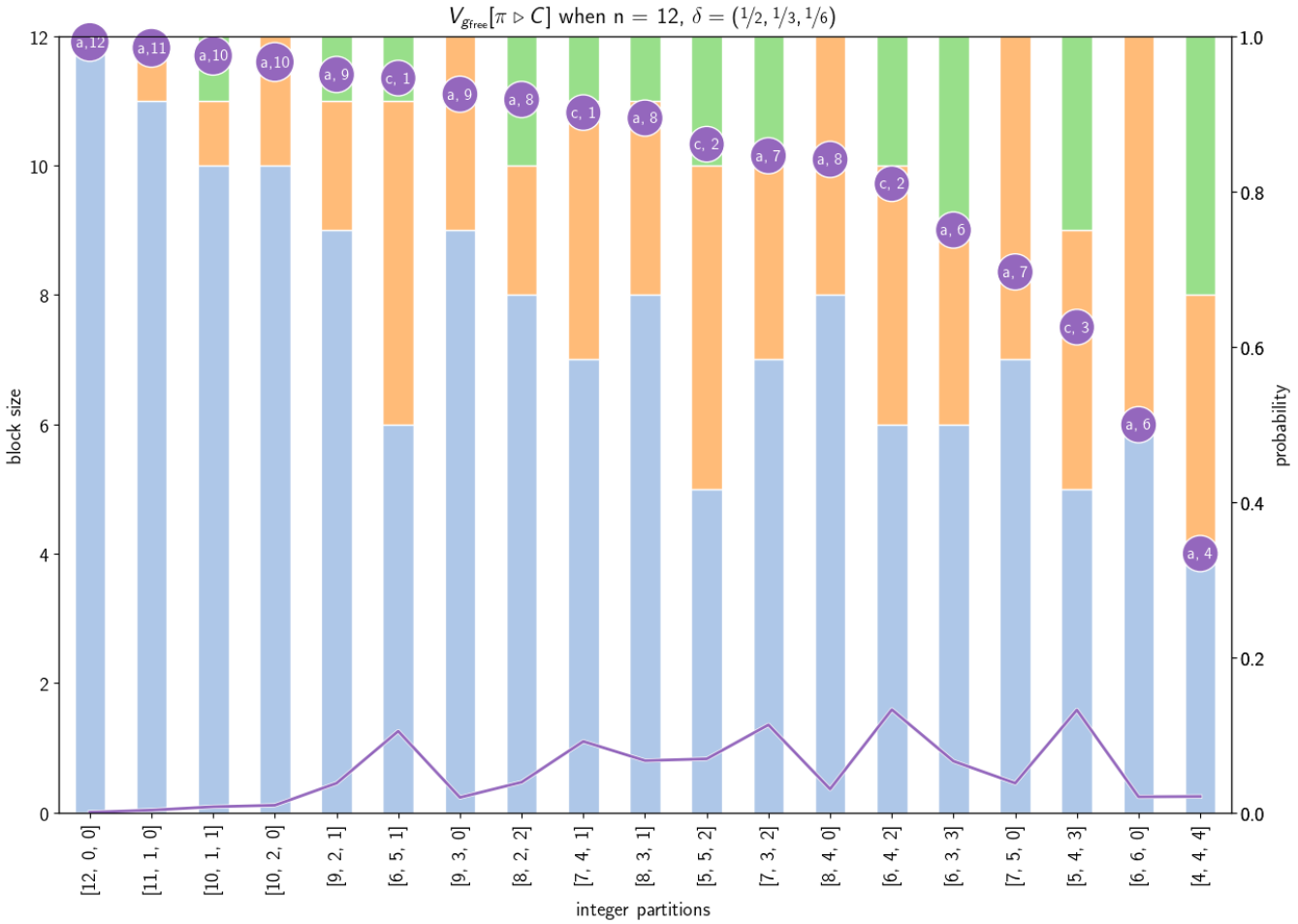



**Figure 7: An in-depth view of the posterior $g$-vulnerability in the free scenario when $n = 12$ under the prior $\delta = (1/2, 1/3, 1/6)$**

only increase, the $g$-vulnerability of certain posteriors can be less, as illustrated by the [4, 4, 4] case which has a $g$-vulnerability of 1/3.

## 7 CONCLUSION AND FUTURE WORK

This paper provides a leakage analysis of deterministic encryption under three operational scenarios: (1) the adversary must guess the entire column, (2) the adversary is free to guess about an arbitrary patient, and (3) the adversary is forced to guess about a particular patient. We see that in some scenarios and under various priors, there is considerable leakage while in others, there is no leakage at all. This application of the quantitative information flow framework provides more nuance than the coarse discussion of the leakage of deterministic encryption that exists in the literature and contributes a much clearer understanding of information leakage associated with deterministic encryption.

We leave to future work considering correlations across columns. There have already been inference attacks that address this scenario [4, 18], and we would like to formally analyze leakage in this setting. From the QIF perspective, leakage of a secret and another correlated secret via a joint distribution is called *Dalenius leakage*. Another important future direction is to analyze the leakage of various order-revealing encryption schemes. We would like to explore if there exists a refinement order among schemes such that one scheme is not more dangerous than another, regardless of prior or gain function. Lastly, we would like to understand mitigation strategies, such as inserting fake entries prior to uploading the database to the cloud. Given that leakage is dependent on the prior, we would like to know if one can functionally alter the prior by inserting fake data to reduce the amount of leakage.

## ACKNOWLEDGMENTS

We are grateful to Alexandra Boldyreva and Adam O'Neill for early discussions of this work and to the anonymous reviewers for their helpful comments. This work was partially supported by the National Science Foundation under grant CNS-1749014.

## REFERENCES

[1] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. 2004. Order Preserving Encryption for Numeric Data. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data (SIGMOD '04)*. ACM, New York, NY, USA, 563–574. https://doi.org/10.1145/1007568.1007632

[2] Mário S Alvim, Konstantinos Chatzikokolakis, Annabelle McIver, Carroll Morgan, Catuscia Palamidessi, and Geoffrey Smith. 2019. *The Science of Quantitative Information Flow*. Springer International Publishing.

[3] Mário S Alvim, Kostas Chatzikokolakis, Catuscia Palamidessi, and Geoffrey Smith. 2012. Measuring information leakage using generalized gain functions. In *Computer Security Foundations Symposium (CSF), 2012 IEEE 25th*. IEEE, 265–279.

[4] 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 (July 2018), 1715–1728. https://doi.org/10.14778/3236187.3236217

[5] Bitglass. 2014. *Bitglass*. https://www.bitglass.com/

[6] David Blackwell. 1951. Comparison of Experiments. In *Proc. Second Berkeley Symposium on Mathematical Statistics and Probability*. 93–102.

[7] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. 2009. Order-Preserving Symmetric Encryption. In *Advances in Cryptology - EUROCRYPT 2009*, Antoine Joux (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 224–241.

[8] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. 2015. Semantically Secure Order-Revealing Encryption: Multi-input Functional Encryption Without Obfuscation. In *Advances in Cryptology - EUROCRYPT 2015*, Elisabeth Oswald and Marc Fischlin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 563–594.

[9] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. 2015. Leakage-Abuse Attacks Against Searchable Encryption. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15)*. ACM, New York, NY, USA, 668–679. https://doi.org/10.1145/2810103.2813700

[10] David Cash, Feng-Hao Liu, Adam O'Neill, and Cong Zhang. 2016. Reducing the Leakage in Practical Order-Revealing Encryption. *IACR Cryptology ePrint Archive* (2016), 661. https://eprint.iacr.org/2016/661.pdf

[11] Alberto Ceselli, Ernesto Damiani, Sabrina De Capitani Di Vimercati, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. 2005. Modeling and Assessing Inference Exposure in Encrypted Databases. *ACM Transactions on Information and System Security* 8, 1 (Feb 2005), 119–152. https://doi.org/10.1145/1053283.1053289

[12] Melissa Chase and Seny Kamara. 2010. Structured Encryption and Controlled Disclosure. In *Advances in Cryptology - ASIACRYPT 2010*, Masayuki Abe (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 577–594.

[13] Nathan Chenette, Kevin Lewi, Stephen A. Weis, and David J. Wu. 2016. Practical Order-Revealing Encryption with Limited Leakage. In *Fast Software Encryption*, Thomas Peyrin (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 474–493.

[14] CipherCloud. 2010. *CipherCloud*. http://www.ciphercloud.com/

[15] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. 2006. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*. ACM, New York, NY, USA, 79–88. https://doi.org/10.1145/1180405.1180417

[16] Dawn Xiaoding Song, D. Wagner, and A. Perrig. 2000. Practical techniques for searches on encrypted data. In *Proceeding 2000 IEEE Symposium on Security and Privacy. S P 2000*. 44–55. https://doi.org/10.1109/SECPRI.2000.848445

[17] Sabrina De Capitani Di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. 2013. On Information Leakage by Indexes over Data Fragments. (2013), 94–98. https://doi.org/10.1109/ICDEW.2013.6547434

[18] F. Betül Durak, Thomas M. DuBuisson, and David Cash. 2016. What Else is Revealed by Order-Revealing Encryption?. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. ACM, New York, NY, USA, 1155–1166. https://doi.org/10.1145/2976749.2978379

[19] ESSA2. 2018. *Second Workshop on Encryption for Secure Search and other Algorithms*. https://www.cc.gatech.edu/~aboldyre/ESSA/

[20] B. Fuller, M. Varia, A. Yerukhimovich, E. Shen, A. Hamlin, V. Gadepally, R. Shay, J. D. Mitchell, and R. K. Cunningham. 2017. SoK: Cryptographically Protected Database Search. In *2017 IEEE Symposium on Security and Privacy (SP)*. 172–191. https://doi.org/10.1109/SP.2017.10

[21] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart. 2017. Leakage-Abuse Attacks against Order-Revealing Encryption. In *2017 IEEE Symposium on Security and Privacy (SP)*. 655–672. https://doi.org/10.1109/SP.2017.44

[22] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. 2012. Access Pattern Disclosure on Searchable Encryption: Ramification, Attack and Mitigation.. In *Network and Distributed System Security Symposium (NDSS'12)*.

[23] Seny Kamara, Tarik Moataz, and Olya Ohrimenko. 2018. Structured Encryption and Leakage Suppression. In *Advances in Cryptology – CRYPTO 2018*, Hovav Shacham and Alexandra Boldyreva (Eds.). Springer International Publishing, Cham, 339–370.

[24] McAfee. 2018. *McAfee MVISION Cloud*. https://www.mcafee.com/enterprise/en-us/products/mvision-cloud.html

[25] Annabelle McIver. 2019. Experiments in Information Flow Analysis. In *Mathmatics of Program Construction*. Springer International Publishing.

[26] Annabelle McIver, Carroll Morgan, Geoffrey Smith, Barbara Espinoza, and Larissa Meinicke. 2014. Abstract Channels and Their Robust Information-Leakage Ordering. In *Principles of Security and Trust*, Martín Abadi and Steve Kremer (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 83–102.

[27] Microsoft. 2016. *Always Encrypted (Database Engine)*. https://msdn.microsoft.com/en-us/library/mt163865.aspx

[28] Muhammad Naveed, Seny Kamara, and Charles V. Wright. 2015. Inference Attacks on Property-Preserving Encrypted Databases. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15)*. ACM, New York, NY, USA, 644–655. https://doi.org/10.1145/2810103.2813651

[29] Netskope. 2013. *Netskope*. https://www.netskope.com/

[30] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. 2011. CryptDB: Protecting Confidentiality with Encrypted Query Processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles (SOSP '11)*. ACM, New York, NY, USA, 85–100. https://doi.org/10.1145/2043556.2043566

[31] Raluca Ada Popa, Nickolai Zeldovich, and Hari Balakrishnan. 2015. Guidelines for Using the CryptDB System Securely. *IACR Cryptology ePrint Archive* (2015). https://eprint.iacr.org/2015/979.pdf

[32] David Pouliot and Charles V. Wright. 2016. 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)*. ACM, New York, NY, USA, 1341–1352. https://doi.org/10.1145/2976749.2978401

[33] Philip H. Santa-Emma, Ralph Roach, Mary Ann Gill, Pam Spayde, and Robert M. Taylor. 2002. Development and Implementation of an Inpatient Acute Palliative Care Service. *Journal of Palliative Medicine* 5, 1 (2002), 93 – 100.

[34] Encrypted Search. 2019. *Encrypted Search Workshop*. https://icerm.brown.edu/topical_workshops/tw19-1-es/

[35] Saeed Sedghi, Jeroen Doumen, Pieter Hartel, and Willem Jonker. 2008. Towards an Information Theoretic Analysis of Searchable Encryption. In *Information and Communications Security*, Liqun Chen, Mark D. Ryan, and Guilin Wang (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 345–360.

[36] Symantec. 2016. *Symantec CloudSOC*. https://www.symantec.com/products/cloud-application-security-cloudsoc

[37] J. H. van Lint and R. M. Wilson. 2001. *A Course in Combinatorics* (2 ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511987045

[38] 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 *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 707–720.