Estimating g-Leakage via Machine Learning

Marco Romanelli, Konstantinos Chatzikokolakis, Catuscia Palamidessi, Pablo Piantanida

Estimating g-Leakage via Machine Learning

Marco Romanelli
marco.romanelli@inria.fr
Inria, École Polytechnique, IPP, Università di Siena
Palaiseau, France

Konstantinos Chatzikokolakis
kostas@chatzi.org
University of Athens
Athens, Greece

Catuscia Palamidessi
catuscia@lix.polytechnique.fr
Inria, École Polytechnique, IPP
Palaiseau, France

Pablo Piantanida
pablo.piantanida@centralesupelec.fr
CentraleSupelec, CNRS, Université Paris Saclay
Gif-sur-Yvette, France

ABSTRACT

This paper considers the problem of estimating the information leakage of a system in the black-box scenario, i.e. when the system's internals are unknown to the learner, or too complicated to analyze, and the only available information are pairs of input-output data samples, obtained by submitting queries to the system or provided by a third party. The *frequentist* approach relies on counting the frequencies to estimate the input-output conditional probabilities, however this method is not accurate when the domain of possible outputs is large. To overcome this difficulty, the estimation of the Bayes error of the ideal classifier was recently investigated using Machine Learning (ML) models, and it has been shown to be more accurate thanks to the ability of those models to learn the input-output correspondence. However, the Bayes vulnerability is only suitable to describe *one-try* attacks. A more general and flexible measure of leakage is the *g-vulnerability*, which encompasses several different types of adversaries, with different goals and capabilities. We propose a novel approach to perform black-box estimation of the g-vulnerability using ML which does not require to estimate the conditional probabilities and is suitable for a large class of ML algorithms. First, we formally show the learnability for all data distributions. Then, we evaluate the performance via various experiments using k-Nearest Neighbors and Neural Networks. Our approach outperform the frequentist one when the observables domain is large.

CCS CONCEPTS

• **Security and privacy → Formal security models**; **Formal methods and theory of security**; **Information flow control**; • **Computing methodologies → Neural networks**; **Machine learning**.

KEYWORDS

g-vulnerability estimation; machine learning; neural networks

1 INTRODUCTION

The information leakage of a system is a fundamental concern of computer security, and measuring the amount of sensitive information that an adversary can obtain by observing the outputs of a given system is of the utmost importance to understand whether such leakage can be tolerated or must be considered a major security flaw. Much research effort has been dedicated to studying and proposing solutions to this problem, see for instance [2, 4, 7, 10, 11, 20, 30, 38]. So far, this area of research, known as quantitative information flow (QIF), has mainly focused on the so-called white-box scenario, i.e. assuming that the channel of the system is known, or can be computed by analyzing the system's internals. This channel consists of the conditional probabilities of the outputs (observables) given the inputs (secrets).

The white-box assumption, however, is not always realistic: sometimes the system is unknown, or anyway it is too complex, so that an analytic computation becomes hard if not impossible to be performed. Therefore, it is important to consider also black-box approaches where we only assume the availability of a finite set of input-output pairs generated by the system. A further specification of a black box model is the "degree of inaccessibility", namely whether or not we are allowed to interact with the system to obtain these pairs, or they are just provided by a third party. Both scenarios are realistic, and in our paper we consider the two cases.

The estimation of the internal probabilities of a system's channel have been investigated in [17] and [19] via a frequentist paradigm, i.e., relying on the computation of the frequencies of the outputs given some inputs. However, this approach does not scale to applications for which the output space is very large since a prohibitively large number of samples would be necessary to achieve good results and fails on continuous alphabets unless some strong assumption on the distributions are made. In order to overcome this limitation, the authors of [14] exploited the fact that Machine Learning (ML) algorithms provide a better scalability to black-box measurements. Intuitively, the advantage of the ML approach over the frequentist one is its generalization power: while the frequentist method can only draw conclusions based on counts on the available samples, ML is able to extrapolate from the samples and provide better prediction (generalization) for the rest of the universe.

In particular, [14] proposed to use k-Nearest Neighbors (k-NN) to measure the basic QIF metrics, the Bayes vulnerability [38]. This is the expected probability of success of an adversary that has exactly one attempt at his disposal (one-try), and tries to maximize the chance of guessing the right secret. The Bayes vulnerability corresponds to the converse of the error of the ideal Bayes classifier that, given any sample (observable), tries to predict its corresponding class (secret). Hence the idea is to build a model that approximates such classifier, and estimate its expected error. The main takeaway is that any ML rule which is *universally consistent* (i.e., approximates the ideal Bayes classifier) has a guarantee on the accuracy of the estimation, i.e., the error in the estimation of the leakage tends to vanish as the number of training samples grows large.

The method of [14], however, is limited to the Bayes vulnerability, which models only one particular kind of adversary. As argued in [4], there are many other realistic ones. For instance, adversaries whose aim is to guess only a part of the secret, or a property of the secret, or that have multiple tries at their disposal. To represent a larger class of attacks, [4] introduced the so-called *g-vulnerability*. This metric is very general, and in particular it encompasses the Bayes vulnerability.

Symbol	Description
$x \in \mathcal{X}$	a secret
$w \in \mathcal{W}$	a guess
$y \in \mathcal{Y}$	an observable output by the system
X	random var. for secrets taking values $x \in \mathcal{X}$
W	random var. for guesses taking values $w \in \mathcal{W}$
Y	random var. for observables taking values $y \in \mathcal{Y}$
$\lvert\mathcal{S}\rvert$	size of a set $\mathcal{S}$
$\mathcal{P}(\mathcal{S})$	Distribution over a set of symbols $\mathcal{S}$
$\mathcal{H}$	class of learning functions f
π, P_X	prior distribution over the secret space
$\hat{\pi}$, $\widehat{P}_X$	empirical prior distribution over the secret space
C	Channel matrix
$\pi \triangleright C$	joint distribution from prior π and channel C
P_{XY}	joint probability distribution
$\widehat{P}_{XY}$	empirical joint probability distribution
$P_{Y\|X}$	conditional probability of Y given X
$\widehat{P}_{Y\|X}$	empirical conditional probabilities
$\mathbb{P}$	probability measure
$\mathbb{E}[\cdot]$	expected value
$g(w, x)$	gain function: guess w and secret x as inputs
G	gain matrix of size $\lvert\mathcal{W}\rvert \times \lvert\mathcal{X}\rvert$ for a specific g
V_g	g-vulnerability
$V(f)$	g-vulnerability functional
$\widehat{V}_n(f)$	empirical g-vuln. functional evaluated on n samples

Table 1: Table of symbols.

In this paper, we propose an approach to the black-box estimation of g-vulnerability via ML. The idea is to reduce the problem to that of approximating the Bayes classifier, so that *any universally consistent ML algorithm can be used for the purpose*. This reduction essentially takes into account the impact of the gain function in the generation of the training data, and we propose two methods to obtain this effect, which we call *channel pre-processing* and *data pre-processing*, respectively. We evaluate our approach via experiments on various channels and gain functions. In order to show the generality of our approach, we use two different ML algorithms, namely k-NN and Artificial Neural Networks (ANN), and we compare their performances. The experimental results show that our approach provides accurate estimations, and that it outperforms by far the frequentist approach when the observables domain is large.

1.1 Our contribution

- We propose a novel approach to the black-box estimation of g-vulnerability based on ML. To the best of our knowledge, this is the first time that a method to estimate g-vulnerability in a black-box fashion is introduced.
- We provide statistical guarantees showing the learnability of the g-vulnerability for all distributions and we derive distribution-free bounds on the accuracy of its estimation.
- We validate the performance of our method via several experiments using k-NN and ANN models. The code is available at the URL https://github.com/LEAVESrepo/leaves.

1.2 Related work

One important aspect to keep in mind when measuring leakage is the kind of attack that we want to model. In [30], Köpf and Basin identified various kinds of adversaries and showed that they can be captured by known entropy measures. In particular it focussed on the adversaries corresponding to Shannon and Guessing entropy. In [38] Smith proposed another notion, the Rényi min-entropy, to measure the system's leakage when the attacker has only one try at its disposal and attempts to make its best guess. The Rényi min-entropy is the logarithm of the Bayes vulnerability, which is the expected probability of the adversary to guess the secret correctly. The Bayes vulnerability is the converse of the Bayes error, which was already proposed as a measure of leakage in [12].

Alvim et al. [4] generalized the notion of Bayes vulnerability to that of g-vulnerability, by introducing a parameter (the gain function g) that describes the adversary's payoff. The g-vulnerability is the expected gain of the adversary in a one-try attack.

The idea of estimating the g-vulnerability in the using ML techniques is inspired by the seminal work [14], which used k-NN algorithms to estimate the Bayes vulnerability, a paradigm shift with respect to the previous frequentist approaches [9, 17, 19]. Notably, [14] showed that universally consistent learning rules allow to achieve much more precise estimations than the frequentist approach when considering large or even continuous output domains which would be intractable otherwise. However, [14] was limited to the Bayes adversary. In contrast, our approach handles any adversary that can be modeled by a gain function g. Another novelty w.r.t. [14] is that we consider also ANN algorithms, which in various experiments appear to perform better than the k-NN ones.

Bordenabe and Smith [6] investigated the indirect leakage induced by a channel (i.e., leakage on sensitive information not in the domain of the channel), and proved a fundamental equivalence between Dalenius min-entropy leakage under arbitrary correlations and g-leakage under arbitrary gain functions. This result is similar to our Theorem 4.2, and it opens the way to the possible extension of our approach to this more general leakage scenario.

2 PRELIMINARIES

In this section, we recall some useful notions from QIF and ML.

2.1 Quantitative information flow

Let $\mathcal{X}$ be a set of secrets and $\mathcal{Y}$ a set of observations. The adversary's initial knowledge about the secrets is modeled by a prior distribution $\mathcal{P}(\mathcal{X})$ (namely P_X). A system is modeled as a probabilistic *channel* from $\mathcal{X}$ to $\mathcal{Y}$, described by a stochastic matrix C, whose elements C_{xy} give the probability to observe $y \in \mathcal{Y}$ when

the input is $x \in \mathcal{X}$ (namely $P_{Y|X}$). Running C with input π induces a joint distribution on $\mathcal{X} \times \mathcal{Y}$ denoted by $\pi \triangleright C$.

In the g-leakage framework [4] an adversary is described by a set $\mathcal{W}$ of *guesses* (or *actions*) that it can make about the secret, and by a *gain function* $g(w, x)$ expressing the gain of selecting the guess w when the real secret is x. The prior g-vulnerability is the *expected gain* of an optimal guess, given a prior distribution on secrets:

$$V_g(\pi) \stackrel{\text{def}}{=} \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \pi_x \cdot g(w, x)\,. \tag{1}$$

In the posterior case, the adversary observes the output of the system which allows to improve its guess. Its expected gain is given by the posterior g-vulnerability, according to

$$V_g(\pi, C) \stackrel{\text{def}}{=} \sum_{y \in \mathcal{Y}} \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \pi_x \cdot C_{xy} \cdot g(w, x)\,. \tag{2}$$

Finally, the multiplicative[1] and additive g-leakage quantify how much a specific channel C *increases* the vulnerability of the system:

$$\mathcal{L}_g^{\mathrm{M}}(\pi, C) \stackrel{\text{def}}{=} \frac{V_g(\pi, C)}{V_g(\pi)}\,, \quad \mathcal{L}_g^{\mathrm{A}}(\pi, C) \stackrel{\text{def}}{=} V_g(\pi, C) - V_g(\pi)\,. \tag{3}$$

The choice of the gain function g allows to model a variety of different adversarial scenarios. The simplest case is the *identity* gain function, given by $\mathcal{W} = \mathcal{X}$, $g_{\mathrm{id}}(w, x) = 1$ iff $x = w$ and 0 otherwise. This gain function models an adversary that tries to guess the secret exactly in one try; $V_{g_{\mathrm{id}}}$ is the *Bayes-vulnerability*, which corresponds to the complement of the Bayes error (cfr. [4]).

However, the interest in g-vulnerability lies in the fact that many more adversarial scenarios can be captured by a proper choice of g. For instance, taking $\mathcal{W} = \mathcal{X}^k$ with $g(w, x) = 1$ iff $x \in w$ and 0 otherwise, models an adversary that tries to guess the secret correctly in k tries. Moreover, guessing the secret *approximately* can be easily expressed by constructing g from a metric d on $\mathcal{X}$; this is a standard approach in the area of *location privacy* [36, 37] where $g(w, x)$ is taken to be inversely proportional to the Euclidean distance between w and x. Several other gain functions are discussed in [4], while [3] shows that *any* vulnerability function satisfying basic axioms can be expressed as V_g for a properly constructed g.

The main focus of this paper is estimating the posterior g-vulnerability of the system from such samples. Note that, given $V_g(\pi, C)$, estimating $\mathcal{L}_g^{\mathrm{M}}(\pi, C)$ and $\mathcal{L}_g^{\mathrm{A}}(\pi, C)$ is straightforward, since $V_g(\pi)$ only depends on the prior (not on the system) and it can be either computed analytically or estimated from the samples.

2.2 Artificial Neural Networks

We provide a short review of the aspects of ANN that are relevant for this work. For further details, we refer to [5, 28, 29]. Neural networks are usually modeled as directed graphs with weights on the connections and nodes that forward information through "activation functions", often introducing non-linearity (such as sigmoids or soft-max). In particular, we consider an instance of learning known as *supervised learning*, where input samples are provided to the network model together with target labels (supervision). From the provided data and by means of iterative updates of the connection weights, the network learns how the data and respective labels are distributed. The training procedure, known as back-propagation, is an optimization process aimed at minimizing a loss function that quantifies the quality of the network's prediction w.r.t. the data.

Classification problems are a typical example of tasks for which supervised learning works well. Samples are provided together with target labels which represent the classes they belong to. A model can be trained using these samples and, later on, it can be used to predict the class of new samples.

According to the No Free Lunch theorem (NFL) [40], in general, it cannot be said that ANN are better than other ML methods. However, it is well known that the NFL can be broken by additional information on the data or the particular problem we want to tackle, and, nowadays, for most applications and available data, especially in multidimensional domains, ANN models outperform other methods therefore representing the state-of-the-art.

2.3 k-Nearest Neighbors

The k-NN algorithm is one of the simplest algorithms used to classify a new sample given a training set of samples labelled as belonging to specific classes. This algorithm assumes that the space of the features is equipped with a notion of distance. The basic idea is the following: every time we need to classify a new sample, we find the k samples whose features are closest to those of the new one (nearest neighbors). Once the k nearest neighbors are selected, a majority vote over their class labels is performed to decide which class should be assigned to the new sample. For further details, as well as for an extensive analysis of the topic, we refer to the chapters about k-NN in [29, 35].

3 LEARNING g-VULNERABILITY: STATISTICAL BOUNDS

This section introduces the mathematical problem of learning g-vulnerability, characterizing universal learnability in the present framework. We derive distribution-free bounds on the accuracy of the estimation, implying the estimator's statistical consistence.

3.1 Main definitions

We consider the problem of estimating the g-vulnerability from samples via ML models, and we show that the analysis of this estimation can be conducted in the general statistical framework of maximizing an expected functional using observed samples. The idea can be described using three components:

- A generator of random secrets $x \in \mathcal{X}$ with $|\mathcal{X}| < \infty$, drawn independently from a fixed but unknown distribution $P_X(x)$;
- a channel that returns an observable $y \in \mathcal{Y}$ with $|\mathcal{Y}| < \infty$ for every input x, according to a conditional distribution $P_{Y|X}(y|x)$, also fixed and unknown;
- a learning machine capable of implementing a set of rules $f \in \mathcal{H}$, where $\mathcal{H}$ denotes the class of functions $f : \mathcal{Y} \to \mathcal{W}$ and $\mathcal{W}$ is the finite set of guesses.

Moreover let us note that

$$g : \mathcal{W} \times \mathcal{X} \to [a, b] \tag{4}$$

where a and b are finite real values, and $\mathcal{X}$ and $\mathcal{W}$ are finite sets. The problem of learning the g-vulnerability is that of choosing

[1] Originally the multiplicative version of g-leakage was defined as the log of the definition given here. In recent literature the log is not used anymore. Anyway, the two definitions are equivalent for system comparison, since log is monotonic.

the function $f : \mathcal{Y} \to \mathcal{W}$ which maximizes the functional $V(f)$, representing the expected gain, defined as:

$$V(f) \stackrel{\text{def}}{=} \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} g\big(f(y),x\big)P_{XY}(x,y). \tag{5}$$

Note that $f(y)$ corresponds to the "guess" w, for the given y, in (2). The maximum of $V(f)$ is the g-vulnerability, namely:

$$V_g \stackrel{\text{def}}{=} \max_{f\in\mathcal{H}} V(f). \tag{6}$$

3.2 Principle of the empirical g-vulnerability maximization

Since we are in the black box scenario, the joint probability distribution $P_{XY} \equiv \pi\triangleright C$ is unknown. We assume, however, the availability of m independent and identically distributed (i.i.d.) samples drawn according to P_{XY} that can be used as a training set $\mathcal{D}_m$ to solve the maximization of f over $\mathcal{H}$ and additionally n i.i.d. samples are available to be used as a validation[2] set $\mathcal{T}_n$ to estimate the average in (5). Let us denote these sets as: $\mathcal{D}_m \stackrel{\text{def}}{=} \big\{(x_1,y_1),\ldots,(x_m,y_m)\big\}$ and $\mathcal{T}_n \stackrel{\text{def}}{=} \big\{(x_{m+1},y_{m+1}),\ldots,(x_{m+n},y_{m+n})\big\}$, respectively.

In order to maximize the g-vulnerability functional (5) for an unknown probability measure P_{XY}, the following principle is usually applied. The expected g-vulnerability functional $V(f)$ is replaced by the *empirical g-vulnerability functional*:

$$\widehat{V}_n(f) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{(x,y)\in\mathcal{T}_n} g\big(f(y),x\big), \tag{7}$$

which is evaluated on $\mathcal{T}_n$ rather than P_{XY}. This estimator is clearly unbiased in the sense that $\mathbb{E}\big[\widehat{V}_n(f)\big] = V(f)$.

Let $f_m^\star$ denote the empirical optimal rule given by

$$f_m^\star \stackrel{\text{def}}{=} \arg\max_{f\in\mathcal{H}} \widehat{V}_m(f),\ \ \widehat{V}_m(f) \stackrel{\text{def}}{=} \frac{1}{m}\sum_{(x,y)\in\mathcal{D}_m} g\big(f(y),x\big), \tag{8}$$

which is evaluated on $\mathcal{D}_m$ rather than P_{XY}. The function $f_m^\star$ is the optimizer according to $\mathcal{D}_m$, namely the best way among the functions $f : \mathcal{Y} \to \mathcal{W}$ to approximate V_g by maximizing $\widehat{V}_m(f)$ over the class of functions $\mathcal{H}$. This principle is known in statistics as the Empirical Risk Maximization (ERM).

Intuitively, we would like $f_m^\star$ to give a good approximation of the g-vulnerability, in the sense that its expected gain

$$V(f_m^\star) \ = \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} g\big(f_m^\star(y),x\big)P_{XY}(x,y) \tag{9}$$

should be close to V_g. Note that the difference

$$V_g - V(f_m^\star) = \max_{f\in\mathcal{H}} V(f) - V(f_m^\star) \tag{10}$$

is always non negative and represents the gap by selecting a possible suboptimal function $f_m^\star$. Unfortunately, we are not able to compute $V(f_m^\star)$ either, since P_{XY} is unknown. In its place, we have to use its estimation $\widehat{V}_n(f_m^\star)$ Hence, the real estimation error is $|V_g - \widehat{V}_n(f_m^\star)|$. Note that:

$$|V_g - \widehat{V}_n(f_m^\star)| \leqslant (V_g - V(f_m^\star)) + |\widehat{V}_n(f_m^\star) - V(f_m^\star)|, \tag{11}$$

where the first term in the right end side represents the error induced by using the trained model $f_m^\star$ and the second represents the error induced by estimating the g-vulnerability over the n samples in the validation set.

By using basics principles from statistical learning theory, we study two main questions:

- When does the estimator $\widehat{V}_n(f_m^\star)$ work? What are the conditions for its statistical consistency?
- How well does $\widehat{V}_n(f_m^\star)$ approximate V_g? In other words, how fast does the sequence of largest empirical g-leakage values converge to the largest g-leakage function? This is related to the so called rate of generalization of a learning machine that implements the ERM principle.

3.3 Bounds on the estimation accuracy

In the following we use the notation $\sigma_f^2 = Var(g(f(Y),X))$, where $Var(Z)$ stands for the variance of the random variable Z. Namely, σ_f^2 is the variance of the gain for a given function f. We also use $\mathbb{P}$ to represent the probability induced by sampling the training and validation sets over the distribution P_{XY}.

Next proposition, proved in Appendix B.1, states that we can probabilistically delimit the two parts of the bound in (11) in terms of the sizes m and n of the training and validation sets.

Proposition 3.1 (Uniform deviations). *For all $\varepsilon > 0$,*

$$\mathbb{P}\left(\left|\widehat{V}_n(f_m^\star) - V(f_m^\star)\right| \geqslant \varepsilon\right) \leqslant 2\mathbb{E}\,\exp\left(-\frac{n\,\varepsilon^2}{2\,\sigma^2_{f_m^\star} + 2\,(b-a)\varepsilon/3}\right), \tag{12}$$

where the expectation is taken w.r.t. the random training set, and

$$\mathbb{P}\left(V_g - V(f_m^\star) \geqslant \varepsilon\right) \leqslant 2\sum_{f\in\mathcal{H}} \exp\left(-\frac{m\,\varepsilon^2}{8\sigma_f^2 + 4(b-a)\varepsilon/3}\right). \tag{13}$$

Inequality (12) shows that the estimation error due to the use of a validation set in $\widehat{V}_n(f_m^\star)$ instead of the true expected gain $V(f_m^\star)$ vanishes with the number of validation samples. On the other hand, expression (13) implies 'learnability' of an optimal f, i.e., the suboptimality of $f_m^\star$ learned using the training set $\widehat{V}_m(f_m^\star)$ vanishes with the number of training samples.

On the other hand the bound in eq. (13) depends on the underlying distribution and the structural properties of the class $\mathcal{H}$ through the variance. However, it does not explicitly depend on the optimization algorithm (e.g., stochastic gradient descent) used to learn the function $f_m^\star$ from the training set, which just comes from assuming the worst-case upper bound over all optimization procedures. The selection of a "good" subset of candidate functions, having a high probability of containing an almost optimal classifier, is a very active area of research in ML [16], and hopefully there will be some result available soon, which together with our results will provide a practical method to estimate the bound on the errors. In appendix F we discuss heuristics to choose a "good" model, together with a new set of experiments showing the impact of different architectures.

[2] We call $\mathcal{T}_n$ validation set rather than test set, since we use it to estimate the g-vulnerability with the learned $f_m^\star$, rather than to measure the quality of $f_m^\star$.

Theorem 3.2. *The averaged estimation error of the g-vulnerability can be bounded as follows:*

$$\mathbb{E}\big|V_g - \widehat{V}_n(f_m^\star)\big| \leqslant V_g - \mathbb{E}\big[V(f_m^\star)\big] + \mathbb{E}\big|V(f_m^\star) - \widehat{V}_n(f_m^\star)\big|, \quad (14)$$

where the expectations are understood over all possible training and validation sets drawn according to P_{XY}. Furthermore, let $\sigma^2 = max_{f\in\mathcal{H}} Var\big(g(f(Y),X)\big)$, then :

$$\mathbb{E}\big|V(f_m^\star) - \widehat{V}_n(f_m^\star)\big| \leqslant \frac{4\eta}{n}\exp\left(-\frac{n\sigma^2}{2\eta}\right) + \sqrt{\frac{2\sigma^2\eta\pi}{n}}\,\mathrm{erf}\left(\sigma^2\Big/\sqrt{\frac{2\sigma^2\eta\pi}{n}}\right), \quad (15)$$

where $\eta = (1 + (b-a)/3)$ for $\sigma^2 \leqslant \varepsilon$, and, otherwise,

$$V_g - \mathbb{E}\big[V(f_m^\star)\big] \leqslant |\mathcal{H}|\frac{8(1+\eta)}{m}\exp\left(-\frac{m\sigma^2}{4(1+\eta)}\right) + |\mathcal{H}|\sqrt{\frac{4\sigma^2(1+\eta)\pi}{m}}\,\mathrm{erf}\left(\sigma^2\Big/\sqrt{\frac{4\sigma^2(1+\eta)\pi}{m}}\right), \quad (16)$$

with $\mathrm{erf}(\theta) \stackrel{def}{=} \frac{2}{\sqrt{\pi}}\int_0^\theta \exp(-\mu^2)d\mu$.

Interestingly, the term $V_g - \mathbb{E}\big[V(f_m^\star)\big]$ is the average error induced when estimating the function $f_m^\star$ using m samples from the training set while $\mathbb{E}\big|V(f_m^\star) - \widehat{V}_n(f_m^\star)\big|$ indicates the average error incurred when estimating the g-vulnerability using n samples from the validation set. Clearly, in eq. (15), the scaling of these bounds with the number of samples are very different which can be made evident by using the order notation:

$$\sup_{P_{XY}} \mathbb{E}\big|V(f_m^\star) - \widehat{V}_n(f_m^\star)\big| \in O\left(\frac{1}{\sqrt{n}}\right), \quad (17)$$

$$\sup_{P_{XY}} \big\{V_g - \mathbb{E}\big[V(f_m^\star)\big]\big\} \in O\left(\frac{|\mathcal{H}|}{\sqrt{m}}\right). \quad (18)$$

These bounds indicate that the error in (17) vanishes much faster than the error in (18) and thus, the size of the training set, in general, should be kept larger than the size of the validation set, i.e., $n \ll m$.

3.4 Sample complexity

We now study how large the validation set should be in order to get a good estimation. For $\varepsilon, \delta > 0$, we define the sample complexity as the set of smallest integers $M(\varepsilon,\delta)$ and $N(\varepsilon,\delta)$ sufficient to guarantee that the gap between the true g-vulnerability and the estimated $\widehat{V}_n(f_m^\star)$ is at most ε with at least $1-\delta$ probability:

Definition 3.3. For $\varepsilon, \delta > 0$, let all pairs $\big(M(\varepsilon,\delta), N(\varepsilon,\delta)\big)$ be the set of smallest (m,n) sizes of training and validation sets such that:

$$\sup_{P_{XY}} \mathbb{P}\Big[|V_g - \widehat{V}_n(f_m^\star)| > \varepsilon\Big] \leqslant \delta. \quad (19)$$

Next result says that we can bound the sample complexity in terms of $\varepsilon, \delta, \sigma$, and $|b-a|$ (see Appendix B.3 for the proof).

Corollary 3.4. *The sample complexity of the ERM algorithm g-vulnerability is bounded from above by the set of values satisfying:*

$$M(\varepsilon,\delta) \leqslant \frac{8\,\sigma^2 + 4\,(b-a)\varepsilon/3}{\varepsilon^2}\ln\left(\frac{2\,|\mathcal{H}|}{\delta-\Delta}\right), \quad (20)$$

$$N(\varepsilon,\delta) \leqslant \frac{2\,\sigma^2 + 2\,(b-a)\varepsilon/3}{\varepsilon^2}\ln\left(\frac{2}{\Delta}\right), \quad (21)$$

for all Δ such that $0 < \Delta < \delta$.

The theoretical results of this section are very general and cannot take full advantage of the specific properties of a particular model or data distribution. In particular, it is important to emphasize that the upper bound in (11) is independent of the learned function $f_m^\star$ because $|\widehat{V}_n(f_m^\star) - V(f_m^\star)| \leqslant \max_{f\in\mathcal{H}}|\widehat{V}_n(f) - V(f)|$ and because of (39), and thus these bounds are independent of the specific algorithm and training sets in used to solve the optimization in (8). Furthermore, the f maximizing $|V(f) - \widehat{V}_n(f)|$ in those in-equations is not necessarily what the algorithm would choose. Hence the bounds given in Theorem 3.2 and Corollary 3.4 in general are not tight. However, these theoretical bounds provide a worst-case measure from which learnability holds for all data sets.

In the next section, we will propose an approach for selecting $f_m^\star$ and estimating V_g. The experiments in Section 5 suggest that our method usually estimates V_g much more accurately than what is indicated by Theorem 3.2.

4 FROM g-VULNERABILITY TO BAYES VULNERABILITY VIA PRE-PROCESSING

This is the core section of the paper, which describes our approach to select the $f_m^\star$ to estimate V_g.

In principle one could train a neural network to learn $f_m^\star$ by using $-\widehat{V}_m(f)$ as the loss function, and minimizing it over the m training samples (cfr. Equation 8). However, this would require $\widehat{V}_m(f)$ to be a differentiable function of the weights of the neural network, so that its gradient can be computed during the back-propagation. Now, the problem is that the g component of $\widehat{V}_m(f)$ is essentially a non-differentiable function, so it would need to be approximated by a suitable differentiable (surrogate) function, (e.g., as it is the case of the Bayes error via the cross-entropy). Finding an adequate differentiable function to replace each possible g may be a challenging task in practice. If this surrogate does not preserve the original dynamic of the gradient of g with respect to f, the learned f will be far from being optimal.

In order to circumvent this issue, we propose a different approach, which presents two main advantages:

(1) it reduces the problem of learning $f_m^\star$ to a standard classification problem, therefore it does not require a different loss function to be implemented for each adversarial scenario;
(2) it can be implemented by using *any* universally consistent learning algorithm (i.e., any ML algorithm approximating the ideal Bayes classifier).

The reduction described in the above list (item 1) is based on the idea that, in the g-leakage framework, the adversary's goal is not to directly infer the actual secret x, but rather to select the optimal guess w about the secret. As a consequence, the training of the ML classifier to produce $f_m^\star$ should not be done on pairs of

type (x, y), but rather of type (w, y), expressing the fact that the best guess, in the particular run which produced y, is w. This shift from (x, y) to (w, y) is via a *pre-processing* and we propose two distinct and systematic ways to perform this transformation, called *data* and *channel pre-processing*, respectively. The two methods are illustrated in the following sections.

We remind that, according to section 3, we restrict, wlog, to non-negative g's. If g takes negative values, then it can be shifted by adding $-\min_{w,x} g(w, x)$, without consequences for the g-leakage value (cfr. [2, 4]). Furthermore we assume that there exists at least a pair (x, w) such that $\pi_x \cdot g(w, x) > 0$. Otherwise V_g would be 0 and the problem of estimating it will be trivial.

4.1 Data pre-processing

The data pre-processing technique is completely black-box in the sense that it does not need access to the channel. We only assume the availability of a set of pairs of type (x, y), sampled according to $\pi \triangleright C$, the input-output distribution of the channel. This set could be provided by a third party, for example. We divide the set in $\mathcal{D}_m$ (training) and $\mathcal{T}_n$ (validation), containing m and n pairs, respectively.

For the sake of simplicity, to describe this technique we assume that g takes only integer values, in addition to being non-negative. The construction for the general case is discussed in Appendix C.3.

Algorithm 1: Algorithm for data pre-processing

Input: $\mathcal{D}_m$; *Output*: $\mathcal{D}'_{m'}$;
1. $\mathcal{D}'_{m'} := \varnothing$;
2. For each x, y, let u_{xy} be the number of copies of (x, y) in $\mathcal{D}_m$;
3. For each x, y, w, add $u_{xy} \cdot g(w, x)$ copies of (w, y) to $\mathcal{D}'_{m'}$.

The idea behind the data pre-processing technique is that the effect of the gain function can be represented in the transformed dataset by amplifying the impact of the guesses in proportion to their reward. For example, consider a pair (x, y) in $\mathcal{D}_m$, and assume that the reward for the guess w is $g(w, x) = 5$, while for another guess w' is $g(w', x) = 1$. Then in the transformed dataset $\mathcal{D}'_{m'}$ this pair will contribute with 5 copies of (w, y) and only 1 copy of (w', y). The transformation is described in Algorithm 1. Note that in general it causes an expansion of the original dataset.

Estimation of V_g. Given $\mathcal{D}_m$, we construct the set $\mathcal{D}'_{m'}$ of pairs (w, y) according to Algorithm 1. Then, we use $\mathcal{D}'_{m'}$ to train a classifier $f^\star_{m'}$, using an algorithm that approximates the ideal Bayes classifier. As proved below, $f^\star_{m'}$ gives the same mapping $\mathcal{Y} \to \mathcal{W}$ as the optimal empirical rule $f^\star_m$ on $\mathcal{D}_m$ (cfr. subsection 3.2). Finally, we use $f^\star_m$ and $\mathcal{T}_n$ to compute the estimation of $V_g(\pi, C)$ as in (7), with f replaced by $f^\star_m$.

Correctness. We first need some notation. For each (w, y), define:

$$U(w, y) \stackrel{\text{def}}{=} \sum_x \pi_x \cdot C_{xy} \cdot g(w, x)\,, \tag{22}$$

which represents the "ideal" proportion of copies of (w, y) that $\mathcal{D}'_{m'}$ should contain. From $U(w, y)$ we can now derive the ideal joint distribution on $\mathcal{W} \times \mathcal{Y}$ and the marginal on $\mathcal{W}$:

$$P_{WY}(w, y) \stackrel{\text{def}}{=} \frac{U(w, y)}{\alpha}, \quad \text{where} \quad \alpha \stackrel{\text{def}}{=} \sum_{y,w} U(w, y)\,, \tag{23}$$

(note that $\alpha > 0$ because of the assumption on π and g),

$$\xi_w \stackrel{\text{def}}{=} \sum_y P_{WY}(w, y). \tag{24}$$

The channel of the conditional probabilities of y given w is:

$$E_{wy} \stackrel{\text{def}}{=} \frac{P_{WY}(w, y)}{\xi_w}\,. \tag{25}$$

Note that $P_{WY} = \xi \triangleright E$. By construction, it is clear that the $\mathcal{D}'_{m'}$ generated by Algorithm 1 could have been generated, with the same probability, by sampling $\xi \triangleright E$. The following theorem, whose proof is in Appendix C.1, establishes that the g-vulnerability of $\pi \triangleright C$ is equivalent to the Bayes vulnerability of $\xi \triangleright E$, and hence it is correct to estimate $f^\star_m$ as an empirical Bayes classifier $f^\star_{m'}$ trained on $\mathcal{D}'_{m'}$.

Theorem 4.1 (Correctness of data pre-processing). *Given a prior π, a channel C, and a gain function g, we have*

$$V_g(\pi, C) \;= \alpha \cdot V_{g_{\text{id}}}(\xi, E)\,,$$

where α, ξ and E are those defined in (23), (24) *and* (25)*, respectively, and g_{id} is the identity function (cfr. section 2), i.e., the gain function corresponding to the Bayesian adversary.*

Estimation error. To reason about the error we need to consider the optimal empirical classifiers. Assuming that we can perfectly match the $U(w, y)$ above with the u_{xy} of Algorithm 1, we can repeat the same reasoning as above, thus obtaining $\widehat{V}_m(f) = \alpha \cdot \widehat{V}_{m'}(f)$, where $V_m(f)$ is the empirical functional defined in (8), and $\widehat{V}_{m'}(f)$ is the corresponding empirical functional for g_{id} evaluated in $\mathcal{D}_{m'}$:

$$\widehat{V}_{m'}(f) \stackrel{\text{def}}{=} \frac{1}{m'} \sum_{(w,y)\in\mathcal{D}_{m'}} g_{\text{id}}\big(f(y), w\big) \tag{26}$$

$f^\star_{m'}$ is the maximizer of this functional, i.e. $f^\star_{m'} \stackrel{\text{def}}{=} \arg\max_{f\in\mathcal{H}} \widehat{V}_{m'}(f)$. Therefore we have:

$$f^\star_m = \arg\max_{f\in\mathcal{H}} \widehat{V}_m(f) = \arg\max_{f\in\mathcal{H}} (\alpha \cdot \widehat{V}_{m'}(f)) = \arg\max_{f\in\mathcal{H}} \widehat{V}_{m'}(f) = f^\star_{m'}.$$

A bound on the estimation error of this method can therefore be obtained by using the theory developed in previous section, applied to the Bayes classification problem. Remember that the estimation error is $f^\star_m$ is $|V_g - \widehat{V}_n(f^\star_m)|$. With respect to the estimation error of the corresponding Bayes classifier, we have a magnification of a factor α as shown by the following formula, where $\widehat{V}_{n'}$ represents the empirical functional for the Bayes classifier:

$$|V_g - \widehat{V}_n(f^\star_m)| = |\alpha \cdot V_{g_{\text{id}}} - \alpha \cdot \widehat{V}_{n'}(f^\star_{m'})| = \alpha \cdot |V_{g_{\text{id}}} - \widehat{V}_{n'}(f^\star_{m'})|.$$

However, the normalized estimation error (cfr. section 5) remains the same because both numerator and denominator are magnified by a factor α.

Concerning the probability that the error is above a certain threshold ε, we have the same bound as those for the Bayes classifier in Proposition 3.1 and the other results of previous section, where ε is replaced by $\alpha\varepsilon$, m, n by m', n', σ^2 by $\alpha^2\sigma^2$, and $|b - a| = 1$

(because it's a Bayes classifier). It may sounds a bit surprising that the error for the estimation of the g-vulnerability is not much worse than for the estimation of the Bayes error, but we recall that we are essentially solving the same problem, only every quantity is magnified by a factor α. Also, we are assuming that we can match perfectly U_{xy} by u_{xy}. When g ranges in a large domain this may not be possible, and we should rather resort to the channel pre-processing method described in the next section.

4.2 Channel pre-processing

For this technique we assume *black-box access* to the system, i.e., that we can execute the system while controlling each input, and collect the corresponding output.

The core idea behind this technique is to transform the input of C into entries of type w, and to ensure that the distribution on the w's reflects the corresponding rewards expressed by g.

More formally, let us define a distribution τ on $\mathcal{W}$ as follows:

$$\tau_w \stackrel{\text{def}}{=} \frac{\sum_x \pi_x \cdot g(w,x)}{\beta} \quad \text{where} \quad \beta \stackrel{\text{def}}{=} \sum_{x,w} \pi_x \cdot g(w,x)\,, \quad (27)$$

(note that β is strictly positive because of the assumptions on g and π), and let us define the following matrix R from $\mathcal{W}$ to $\mathcal{X}$:

$$R_{wx} \stackrel{\text{def}}{=} \frac{1}{\beta} \cdot \frac{1}{\tau_w} \cdot \pi_x \cdot g(w,x)\,. \quad (28)$$

It is easy to check that R is a stochastic matrix, hence the composition RC is a channel. It is important to emphasize the following:

Remark In the above definitions, β, τ and R depend solely on g and π, and not on C.

The above property is crucial to our goals, because in the black-box approach we are not supposed to rely on the knowledge of C's internals. We now illustrate how we can estimate V_g using the pre-processed channel RC.

Estimation of V_g. Given RC and τ, we build a set $\mathcal{D}''_{m''}$ consisting of pairs of type (w,y) sampled from $\tau \triangleright RC$. We also construct a set $\mathcal{T}_n$ of pairs of type (x,y) sampled from $\pi \triangleright C$. Then, we use $\mathcal{D}''_{m''}$ to train a classifier $f^\star_m$, using an algorithm that approximates the ideal Bayes classifier. Finally, we use $f^\star_m$ and $\mathcal{T}_n$ to compute the estimation of $V_g(\pi, C)$ as in (7), with f replaced by $f^\star_m$.

Alternatively, we could estimate $V_g(\pi, C)$ by computing the empirical Bayes error of $f^\star_m$ on a validation set $\mathcal{T}_n$ of type (w,y) sampled from $\tau \triangleright RC$, but the estimation would be less precise. Intuitively, this is because RC is more "noisy" than C.

Correctness. The correctness of the channel pre-processing method is given by the following theorem, which shows that we can learn $f^\star_m$ by training a Bayesian classifier on a set sampled from $\tau \triangleright RC$.

Theorem 4.2 (Correctness of channel pre-processing). *Given a prior π and a gain function g, we have that, for any channel C:*

$$V_g(\pi, C) \;= \beta \cdot V_{g_{\mathrm{id}}}(\tau, RC) \quad \textit{for all channels } C.$$

where β, τ and R are those defined in (27) *and* (28).

Interestingly, a result similar to Theorem 4.2 is also given in [6], although the context is completely different from ours: the focus of [6], indeed, is to study how the leakage of C on X may induce also a leakage of other sensitive information Z that has nothing to do with C (in the sense that is not information manipulated by C). We intend to explore this connection in the context of a possible extension of our approach to this more general scenario.

Estimation error. Concerning the estimation error, the results are essentially the same as in previous section (with α replaced by β). As for the bound on the probability of error, the results are worse, because the original variance σ^2 is magnified by the channel pre-processing, which introduces a further factor of randomness in the sampling of training data in 12, which means that in practice this bound is more difficult to estimate.

4.3 Pros and cons of the two methods

The fundamental advantage of data pre-processing is that it allows to estimate V_g from just samples of the system, without even black-box access. In contrast to channel pre-processing, however, this method is particularly sensitive to the values of the gain function g. Large gain values will increase the size of $\mathcal{D}'_{m'}$, with consequent increase of the computational cost for estimating the g-vulnerability. Moreover, if g takes real values then we need to apply the technique described in Appendix C.3, which can lead to a large increase of the dataset as well. In contrast, the channel pre-processing method has the advantage of controlling the size of the training set, but it can be applied only when it is possible to interact with the channel by providing input and collecting output. Finally, from the precision point of view, we expect the estimation based on data pre-processing to be more accurate when g consists of small integers, because the channel pre-processing introduces some extra noise in the channel.

5 EVALUATION

In this section we evaluate our approach to the estimation of g-vulnerability. We consider four different scenarios:

(1) $\mathcal{X}$ is a set of (synthetic) numeric data, the channel C consists of *geometric noise*, and g is the *multiple guesses* gain function, representing an adversary that is allowed to make several attempts to discover the secret.
(2) $\mathcal{X}$ is a set of locations from the Gowalla dataset [1], C is the optimal noise of Shokri et al. [37], and g is one of the functions used to evaluate the privacy loss in [37], namely a function anti-monotonic on the distance, representing the idea that the more the adversary's guess is close to the target (i.e., the real location), the more he gains.
(3) $\mathcal{X}$ is the Cleveland heart disease dataset [24], C is a *differentially private* (DP) mechanism [25, 26], and g assigns higher values to worse heart conditions, modeling an adversary that aims at discovering whether a patient is at risk (for instance, to deny his application for health insurance).
(4) $\mathcal{X}$ is a set of passwords of 128 bits and C is a *password checker* that leaks the time before the check fails, but mitigates the timing attacks by applying some random delay and the bucketing technique (see, for example, [31]). The function g represents the part of the password under attack.

For each scenario, we proceed in the following way:

- We consider 3 different samples sizes for the training sets that are used to train the ML models and learn the $\mathcal{Y} \to \mathcal{W}$ remapping. This is to evaluate how the precision of the estimate depends on the amount of data available, and on its relation with the size of $|\mathcal{Y}|$.
- In order to evaluate the variance of the precision, for each training size we create 5 different training sets, and
- for each trained model we estimate the g-vulnerability using 50 different validation sets.

5.1 Representation of the results and metrics

We graphically represent the results of the experiment as box plots, using one box for each size. More precisely, given a specific size, let $\widehat{V}_n^{ij}$ be the g-vulnerability estimation on the j-th validation set computed with a model trained over the i-th training set (where $i \in \{1, \ldots, 5\}$ and $j \in \{1, \ldots, 50\}$). Let V_g be the real g-vulnerability of the system. We define the *normalized estimation error* δ_{ij} and the mean value $\overline{\delta}$ of the δ_{ij}'s as follows:

$$\delta_{ij} \stackrel{\text{def}}{=} \frac{|\widehat{V}_n^{ij} - V_g|}{V_g}, \quad \text{with} \quad \overline{\delta} \stackrel{\text{def}}{=} \frac{1}{250} \sum_{i=1}^{5} \sum_{j=1}^{50} \delta_{ij}\,. \tag{29}$$

In the graphs, the δ_{ij}'s are reported next to the box corresponding to the size, and $\overline{\delta}$ is the black horizontal line inside the box.

Thanks to the normalization the δ_{ij}'s allow to compare results among different scenario and different levels of (real) g-vulnerability. Also, we argue that the percentage of the error is more meaningful than the absolute value. The interested reader, however, can find in Appendix H also plots showing the estimations of the g-vulnerability and their distance from the real value.

We also consider the following typical measures of precision:

$$\text{dispersion} \stackrel{\text{def}}{=} \sqrt{\frac{1}{250} \sum_{i=1}^{5} \sum_{j=1}^{50} (\delta_{ij} - \overline{\delta})^2}\,, \tag{30}$$

$$\text{total error} \stackrel{\text{def}}{=} \sqrt{\frac{1}{250} \sum_{i=1}^{5} \sum_{j=1}^{50} \delta_{ij}^2}\,. \tag{31}$$

The dispersion is an average measure of how far the normalized estimation errors are from their mean value when using same-size training and validation sets. On the other hand, the total error is an average measure of the normalized estimation error, when using same-size training and validation sets.

In order to make a fair comparison between the two pre-processing methods, intuitively we need to use training and validation sets of "equivalent size". For the validation part, since the "best" f function has been already found and therefore we do not need any pre-processing anymore, "equivalent size" just means same size. But what does it mean, in the case of training sets built with different pre-processing methods? Assume that we have a set of data $\mathcal{D}_m$ coming from a third party collector (recall that m represents the size of the set), and let $\mathcal{D}'_{m'}$ be the result of the data pre-processing on $\mathcal{D}_m$. Now, let $\mathcal{D}''_{m''}$ be the dataset obtained drawing samples according to the channel pre-processing method. Should we impose $m'' = m$ or $m'' = m'$? We argue that the right choice is the first one, because the amount of "real" data collected is m. Indeed, $\mathcal{D}'_{m'}$ is generated synthetically from $\mathcal{D}_m$ and cannot contain more information about C than $\mathcal{D}_m$, despite its larger size.

5.2 Learning algorithms

We consider two ML algorithms in the experiments: k-Nearest Neighbors (k-NN) and Artificial Neural Networks (ANN). We have made however a slight modification of k-NN algorithm, due to the following reason: recall that, depending on the particular gain function, the data pre-processing method might create many instances where a certain observable y is repeated multiple times in pair with different w's. For the k-NN algorithm, a very common choice is to consider a number of neighbors which is equivalent to natural logarithm of the total number of training samples. In particular, when the data pre-processing is applied, this means that $k = \log(m')$ nearest neighbors will be considered for the classification decision. Since $\log(m')$ grows slowly with respect to m', it might happen that k-NN fails to find the subset of neighbors from which the best remapping can be learned. To amend this problem, we modify the k-NN algorithm in the following way: instead of looking for neighbors among all the m' samples, we only consider a subset of $l \leqslant m'$ samples, where each value y only appears once. After the $\log(l)$ neighbors have been detected among the l samples, we select w according to a majority vote over the m' tuples (w, y) created through the remapping.

The distance on which the notion of neighbor is based depends on the experiments. We have considered the standard distance among numbers in the first and fourth experiments, the Euclidean distance in the second one, and the Manhattan distance in the third one, which is a stand choice for DP.

Concerning the ANN models, their specifics are in Appendix D. Note that, for the sake of fairness, we use the same architecture for both pre-processing methods, although we adapt number of epochs and batch size to the particular dataset we are dealing with.

5.3 Frequentist approach

In the experiments, we will compare our method with the frequentist one. This approach has been proposed originally in [9] for estimating mutual information, and extended successively also to min-entropy leakage [18]. Although not considered in the literature, the extension to the case of g-vulnerability is straightforward. The method consists in estimating the probabilities that constitute the channel matrix C, and then calculating analytically the g-vulnerability on C. The precise definition is in Appendix E.

In [14] it was observed that in the case of the Bayes error the frequentist approach performs poorly when the size of the observable domain $|\mathcal{Y}|$ is large with respect to the available data. We want to study whether this is the case also for g-vulnerability.

For the experiment on the multiple guesses the comparison is illustrated in the next section. For the other experiments, because of lack of space, we have reported it in the Appendix H.

5.4 Experiment 1: multiple guesses

We consider a system in which the secrets $\mathcal{X}$ are the integers between 0 and 9, and the observables $\mathcal{Y}$ are the integers between 0 and 15999. Hence $|\mathcal{X}| = 10$ and $|\mathcal{Y}| = 16\text{K}$. The rows of the channel

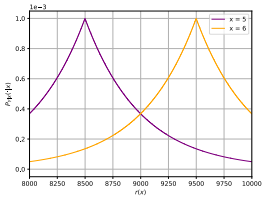


Figure 1: The channel of Experiment 1. The two curves represent the distributions $P_{Y|X}(\cdot|x)$ for two adjacent secrets: $x = 5$ and $x = 6$.

C are geometric distributions centered on the corresponding secret:

$$C_{xy} = P_{Y|X}(y|x) = \lambda \exp(-\nu|r(x) - y|), \tag{32}$$

where:

- ν is a parameter that determines how concentrated around $y = x$ the distribution is. In this experiment we set $\nu = 0.002$;
- r is an auxiliary function that reports $\mathcal{X}$ to the same scale of $\mathcal{Y}$, and centers $\mathcal{X}$ on $\mathcal{Y}$. Here $r(x) = 1000\,x + 3499.5$;
- $\lambda = {e^{\nu}-1}/{(e^{\nu}+1)}$ is a normalization factor

Figure 1 illustrates the shape of C_{xy}, showing the distributions $P_{Y|X}(\cdot|x)$ for two adjacent secrets $x = 5$ and $x = 6$. We consider an adversary that can make two attempts to discover the secret (two-tries adversary), and we define the corresponding gain function as follows. A guess $w \in \mathcal{W}$ is one of all the possible combinations of 2 different secrets from $\mathcal{X}$, i.e., $w = \{x_0, x_1\}$ with $x_0, x_1 \in \mathcal{X}$ and $x_0 \neq x_1$. Therefore $|\mathcal{W}| = \binom{10}{2} = 45$. The gain function g is then

$$g(w, x) = \begin{cases} 1 & \text{if } x \in w \\ 0 & \text{otherwise}. \end{cases} \tag{33}$$

For this experiment we consider a uniform prior distribution π on $\mathcal{X}$. The true g-vulnerability for these particular ν and π, results to be $V_g = 0.892$. As training sets sizes we consider 10K, 30K and 50K, and 50K for the validation sets.

5.4.1 Data pre-processing. (cfr. Section 4.1). The plot in Figure 3 shows the performances of the k-NN and ANN models in terms of normalized estimation error, while Figure 2 shows the comparison with the frequentist approach. As we can see, the precision of the frequentist method is much lower, thus confirming that the trend observed in [14] for the Bayes vulnerability holds also for g-vulnerability. It is worth noting that, in this experiment, the pre-processing of each sample (x, y) creates 9 samples (matching y with each possible $w \in \mathcal{W}$ such that $w = \{x, x'\}$ with $x' \neq x$). This means that the sample size of the pre-processed sets is 9 times the size of the original ones. For functions g representing more than 2 tries this pre-processing method may create training sets too large. In the next section we will see that the alternative channel pre-processing method can be a good compromise.

5.4.2 Channel pre-processing. (cfr. Section 4.2) The results for hannel pre-processing are reported in Figure 4. As we can see, the estimation is worse than with data pre-processing, especially for the k-NN algorithm. This was to be expected, since the random

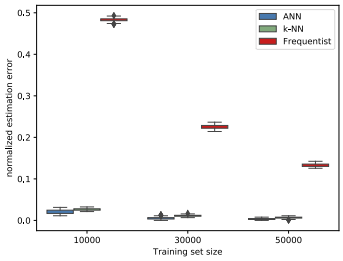


Figure 2: Multiple guesses scenario, comparison between the frequentist and the ML estimations with data pre-processing.

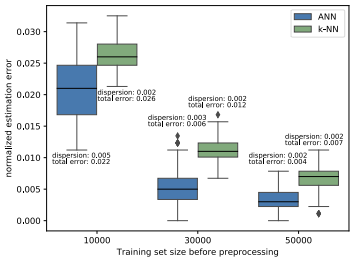


Figure 3: Multiple guesses scenario, magnification of the part of Figure 2 on the k-NN and ANN estimations.

sampling to match the effect of g introduces a further level of confusion, as explained in Section 4.2. Nevertheless, these results are still much better than the frequentist case, so it is a good alternative method to apply when the use of data pre-processing would generate validation sets that are too large, which could be the case when the matrix representing g contains large numbers with a small common divider. Additional plots are provided in Appendix H.

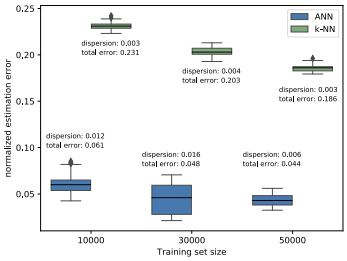


Figure 4: Multiple guesses scenario, k-NN and ANN estimation with channel pre-processing

5.5 Experiment 2: location privacy

In this section we estimate the g-vulnerability of a typical system for location privacy protection. We use data from the open Gowalla dataset [1], which contains the coordinates of users' check-ins. In particular, we consider a square region in San Francisco, USA, centered in (latitude, longitude) = (37.755, −122.440), and with 5Km long sides. In this area Gowalla contains 35162 check-ins.

We discretize the region in 400 cells of 250m long side, and we assume that the adversary's goal is to discover the cell of a check-in. The frequency of the Gowalla check-ins per cell is represented by the heat-map in Figure 5. From these frequencies we can directly estimate the distribution representing the prior of the secrets [27].

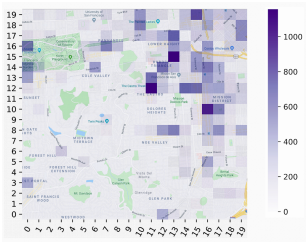


Figure 5: Heat-map representing the Gowalla check-ins distribution in the area of interest; the density of check-ins in each cell is reported in the color bar on the side

The channel C that we consider here is the optimal obfuscation mechanism proposed in [37] to protect location privacy under a utility constraint. We recall that the framework of [37] assumes two loss functions, one for utility and one for privacy. The utility loss of a mechanism, for a certain prior, is defined as the expected utility loss of the noisy data generated according to the prior and the mechanism. The privacy loss is defined in a similar way, except that we allow the attacker to "remap" the noisy data so to maximize the privacy loss. For our experiment, we use the Euclidean distance as loss function for the utility, and the g function defined in the next paragraph as loss function for the privacy. For further details on the construction of the optimal mechanism we refer to [37].

We define $\mathcal{X}, \mathcal{Y}$ and $\mathcal{W}$ to be the set of the cells. Hence $|\mathcal{X}| = |\mathcal{Y}| = |\mathcal{W}| = 400$. We consider a g function representing the precision of the guess in terms of Euclidean distance: the closer the guess is to the real location, the higher is the attacker's gain. Specifically, our g is illustrated in Figure 6, where the central cell represents the real location x. For a generic "guess" cell w, the number written in w represent $g(w, x)$. [3]

In this experiment we consider training set sizes of 100, 1k and 10K samples respectively. After applying the data pre-processing transformation, the size of the resulting datasets is approximately 18 times that of the original one. This was to be expected, since the sum of the values of g in Figure 6 is 20. Note that this sum and the increase factor in the dataset do not necessarily coincide, because the latter is also influenced by the prior and by the mechanism.

[3]Formally, g is defined as $g(w, x) = \lfloor(\gamma \exp(-\alpha d(w, x)/l))\rfloor$, where $\gamma = 4$ is the maximal gain, $\alpha = 0.95$ is a normalization coefficient to control the skewness of the exponential, d is the euclidean distance and $l = 250$ is the length of the cells' side. The symbol $\lfloor\cdot\rfloor$ in this context represents the rounding to the closest integer operation.

Figure 6: The "diamond" shape created by the gain function around the real secret; the values represent the gains assigned to each guessed cell w when x is the central cell.

Figure 7 and Figure 8 show the performance of k-NN and ANN for both data and channel pre-processing. As expected, the data pre-processing method is more precise than the channel pre-processing one, although only slightly. The ANN model is also slightly better than the k-NN in most of the cases.

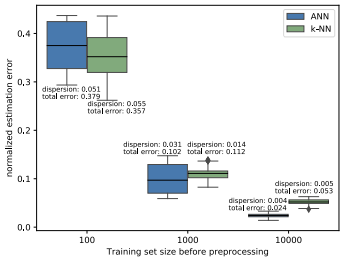


Figure 7: Location privacy scenario, k-NN and ANN estimation with data pre-processing

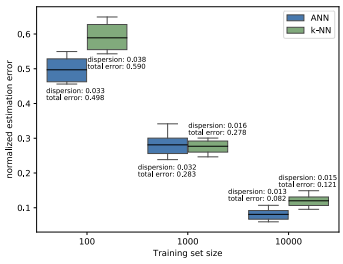


Figure 8: Location privacy scenario, k-NN and ANN estimation with channel pre-processing

5.6 Experiment 3: differential privacy

In this section we consider a popular application of DP: individual data protection in medical datasets from which we wish to extract some statistics via counting queries. It is well known that the release of exact information from the database, even if it is only the result of statistical computation on the aggregated data, can leak sensitive information about the individuals. The solution proposed by DP is

to obfuscate the released information with carefully crafted noise that obeys certain properties. The goal is to make it difficult to detect whether a certain individual is in the database or not. In other words, two adjacent datasets (i.e., datasets that differ only for the presence of one individual) should have almost the same likelihood to produce a certain observable result.

In our experiment, we consider the Cleveland heart disease dataset [24] which consist of 303 records of patients with a medical heart condition. Each condition is labeled by an integer number indicating the severity: from 0, representing a healthy patient, to 4, representing a patient whose life is at risk.

We assume that the hospital publishes the histogram of the patients' records, i.e., the number of occurrences for each label. To protect the patients' privacy, the hospital sanitizes the histogram by adding geometric noise (a typical DP mechanism) to each label's count. More precisely, if the count of a label is z_1, the probability that the corresponding published number is z_2 is defined by the distribution in (32), where x and y are replaced by z_1 and z_2 respectively, and r is 1. Note that z_1, the real count, is an integer between 0 and 303, while its noisy version z_2 ranges on all integers. As for the value of ν, in this experiment we set it to 1.

The secrets space $\mathcal{X}$ is set to be a set of two elements: the full dataset, and the dataset with one record less. These are adjacent in the sense of DP, and, as customary in DP, we assume that the record on which the two databases differ is the target of the adversary. The observables space $\mathcal{Y}$ is the set of the 5-tuples produces by the noisy counts of the 5 labels. $\mathcal{W}$ is set to be the same as $\mathcal{X}$.

We assume that the adversary is especially interested in finding out whether the patient has a serious condition. The function g reflects this preference by assigning higher value to higher labels. Specifically, we set $\mathcal{W} = \mathcal{X}$ and

$$g(w,x) = \begin{cases} 0, & \text{if } w \neq x \\ 1, & \text{if } w = x \wedge x \in \{0,1,2\} \\ 2, & \text{if } w = x \wedge x \in \{3,4\}, \end{cases} \tag{34}$$

For the estimation, we consider 10K, 30K and 50K samples for the training sets, and 50K samples for the validation set. For the

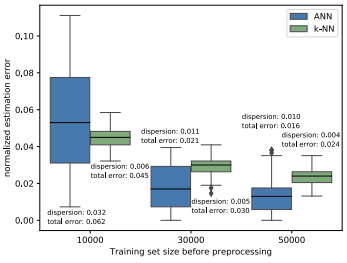


Figure 9: Differential privacy scenario, k-NN and ANN estimation with data pre-processing

experiments with k-NN we choose the Manhattan distance, which is typical for DP[4]. In the case of data pre-processing the size of

[4]The Manhattan distance on histograms corresponds to the total variation distance on the distributions resulting from the normalization of these histograms.

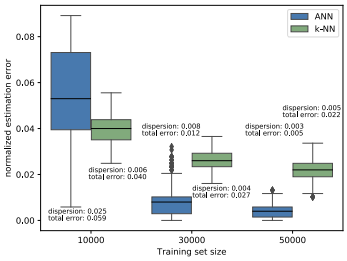


Figure 10: Differential privacy scenario, k-NN and ANN estimation with channel pre-processing

the transformed training set $\mathcal{D}'_{m'}$ is about 1.2 times the original $\mathcal{D}_m$. The performances of the data and channel pre-processing are shown in Figures 9 and Figure 10 respectively. Surprisingly, in this case the data pre-processing method outperforms the channel pre-processing one, although only slightly. Additional plots, including the results for the frequentist approach, can be found in Appendix H.

5.7 Experiment 4: password checker

In this experiment we consider a password checker, namely a program that tests whether a given string corresponds to the password stored in the system. We assume that string and password are sequences of 128 bits, an that the program is "leaky", in the sense that it checks the two sequences bit by bit and it stops checking as soon as it finds a mismatch, reporting failure. It is well known that this opens the way to a timing attack (a kind of side-channel attack), so we assume that the system tries to mitigate the threat by adding some random delay, sampled from a Laplace distribution and then bucketing the reported time in 128 bins corresponding to the positions in the sequence (or equivalently, by sampling the delay from a Geometric distribution, cfr. Equation 32). Hence the channel C is a $2^{128} \times 128$ stochastic matrix.

The typical attacker is an interactive one, which figures out larger and larger prefixes of the password by testing each bit at a time. We assume that the attacker has already figured out the first 6 bits of the sequence and it is trying to figure out the 7-th. Thus the prior π is distributed (uniformly, we assume) only on the sequences formed by the known 6-bits prefix and all the possible remaining 122 bits, while the g function assigns 1 to the sequences whose 7-th bit agrees with the stored password, and 0 otherwise. Thus g is a *partition gain function* [4], and its particularity is that for such kind of functions data pre-processing and channel pre-processing coincide. This is because $g(w,x)$ is either 0 or 1, so in both cases we generate exactly one pair (w,y) for each pair (x,y) for which $g(w,x) = 1$. Note that in this case the data pre-processing transformation does not increase the training set, and the channel pre-processing transformation does not introduce any additional noise. The RC matrix (cfr. Section 4.1) is a 2×128 stochastic matrix. The experiments are done with training sets of 10K, 30K and 50K samples. The results are reported in Figure 11. We note that the estimation error is quite small, especially in the ANN case. This is because the learning problem is particularly simple since, by

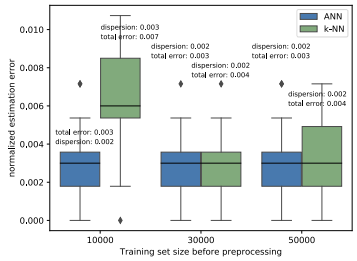


Figure 11: Password checker scenario, k-NN and ANN estimation with data and channel pre-processing

considering the g-leakage and the preprocessing, we have managed to reduce the problem to learning a function of type $\mathcal{Y} \to \mathcal{W}$, rather than $\mathcal{Y} \to \mathcal{X}$, and there is a huge difference in size between $\mathcal{W}$ and $\mathcal{X}$ (the first is 2 and the latter is 2^{128}). Also the frequentist approach does quite well (cfr. Appendix H) , and this is because $\mathcal{Y}$ is small. With a finer bucketing (on top of the Laplace delay), or no bucketing at all, we expect that the difference between the accuracy of the frequentist and of the ML estimation would be much larger.

5.8 Discussion

In almost all the experiments, our method gives much better result than the frequentist approach (see Figure 2 and the other plots in the appendix Appendix H). The exception of the second experiment can be explained by the fact that the observable space is not very large, which is a scenario where the frequentist approach can be successful because the available data is enough to estimate the real distribution. In general, with the frequentist approach there is no real learning, therefore, if $|\mathcal{Y}|$ is large and the training set contains few samples, we cannot make a good guess with the observables never seen before [14]. In ML, on the contrary, we can still make an informed guess, as ML models are able to generalize from samples, especially when the Bayes error is small.

We observe that ANN outperforms the k-NN in all experiments. This is because usually ANN models are better at generalizing, and hence provide better classifiers. In particular, k-NN are not very good when the distributions are not smooth with respect to the metric with respect to which the neighbor relation is evaluated [14].

The data pre-processing method gives better results, than the channel pre-processing in all experiments except the third one (DP), in which the difference is very small. The main advantage of using the channel pre-processing method is when the gain function is such that the data pre-processing would generate a set too large, as explained in Section 4.3.

The experiments show that our method is not too sensitive to the size of $|\mathcal{Y}|$. On the other hand, the size of $|\mathcal{X}|$ is important, because the ML classifiers are in general more precise (approximate better the ideal Bayes classifier) when the number of classes are small. This affects the estimation error of both the Bayes vulnerability and the g-vulnerability. However, for the latter there is also the additional problem of the magnification due to the g. To better understand this point, consider a modification of the last experiment, and assume that the password checker is not leaky, i.e., the observables are only *fail* or *success*. A pair $(x, success)$ would have a negligible probability of appearing in the training set, hence our method, most likely, would estimate the vulnerability to be 0. This is fine if we are trying to estimate the Bayes vulnerability, which is also negligible. But the g-vulnerability may not be negligible, in particular if we consider a g that gives an enormous gain for the success case. If we can ensure that all the pairs (x, y) are represented in the training set in proportion to their probability in P_{XY}, then the above "magnification" in g-vulnerability is not a problem, because our method will ensure the also the pairs (w, y) would be magnified (with respect to the the pairs (w, y)) in the same proportion.

6 CONCLUSION AND FUTURE WORK

We have proposed an approach to estimate the g-vulnerability of a system under the black-box assumption, using machine learning. The basic idea is to reduce the problem to learn the Bayes classifier on a set of pre-processed training data, and we have proposed two techniques for this transformation, with different advantages and disadvantages. We have then evaluated our approach on various scenarios, showing favorable results. We have compared our approach to the frequentist one, showing that the performances are similar on small observable domains, while ours performs better on large ones. This is in line with what already observed in [14] for the estimation of the Bayes error.

As future work, we plan to test our framework on more real-life scenarios such as the web fingerprinting attacks [13, 15] and the AES cryptographic algorithm [22]. We also would like to consider the more general case, often considered in Information-flow security, of channels that have both “high” and “low” inputs, where the first are the secrets and the latter are data visible to, or even controlled by, the adversary. Finally, a more ambitious goal is to use our approach to minimize the g-vulnerability of complex systems, using a GAN based approach, along the lines of [33].

ACKNOWLEDGMENTS

This research was supported by DATAIA “Programme d’Investissement d’Avenir” (ANR-17-CONV-0003). It was also supported by the ANR project REPAS, and by the Inria/DRI project LOGIS. The work of Catuscia Palamidessi was supported by the project HYPATIA, funded by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation program, grant agreement n. 835294. This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 797805.

REFERENCES

[1] 2011. Gowalla dataset. https://snap.stanford.edu/data/loc-Gowalla.html.
[2] Mário S. Alvim, Konstantinos Chatzikokolakis, Annabelle McIver, Carroll Morgan, Catuscia Palamidessi, and Geoffrey Smith. 2014. Additive and Multiplicative Notions of Leakage, and Their Capacities. In *IEEE 27th Computer Security Foundations Symposium, CSF 2014, Vienna, Austria, 19-22 July, 2014*. IEEE, 308–322. https://doi.org/10.1109/CSF.2014.29
[3] Mário S. Alvim, Konstantinos Chatzikokolakis, Annabelle McIver, Carroll Morgan, Catuscia Palamidessi, and Geoffrey Smith. 2016. Axioms for Information Leakage. In *Proceedings of the 29th IEEE Computer Security Foundations Symposium (CSF)*. 77–92. https://doi.org/10.1109/CSF.2016.13

[4] Mário S. Alvim, Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Geoffrey Smith. 2012. Measuring Information Leakage Using Generalized Gain Functions. In *25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27, 2012*, Stephen Chong (Ed.). IEEE Computer Society, 265–279. http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6265867

[5] Christopher M. Bishop. 2007. *Pattern recognition and machine learning, 5th Edition.* Springer. I–XX, 1–738 pages. http://www.worldcat.org/oclc/71008143

[6] Nicolás E. Bordenabe and Geoffrey Smith. 2016. Correlated Secrets in Quantitative Information Flow. In *CSF*. IEEE Computer Society, 93–104. http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7518122

[7] Michele Boreale. 2009. Quantifying information leakage in process calculi. *Inf. Comput* 207, 6 (2009), 699–725.

[8] S. Boucheron, G. Lugosi, and P. Massart. 2013. *Concentration Inequalities: A Nonasymptotic Theory of Independence.* OUP Oxford.

[9] Konstantinos Chatzikokolakis, Tom Chothia, and Apratim Guha. 2010. Statistical Measurement of Information Leakage. In *TACAS (Lecture Notes in Computer Science, Vol. 6015)*. Springer, 390–404.

[10] Konstantinos Chatzikokolakis, Natasha Fernandes, and Catuscia Palamidessi. 2019. Comparing systems: max-case refinement orders and application to differential privacy. In *Proceedings of the 32nd IEEE Computer Security Foundations Symposium*. Hoboken, United States, 442–457. https://doi.org/10.1109/CSF.2019.00037

[11] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. 2008. Anonymity protocols as noisy channels. *Inf. Comput* 206, 2-4 (2008), 378–401.

[12] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. 2008. On the Bayes risk in information-hiding protocols. *Journal of Computer Security* 16, 5 (2008), 531–571. https://doi.org/10.3233/JCS-2008-0333

[13] Giovanni Cherubin. 2017. Bayes, not Naïve: Security Bounds on Website Fingerprinting Defenses. *PoPETs* 2017, 4 (2017), 215–231.

[14] Giovanni Cherubin, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. 2019. F-BLEAU: Fast Black-box Leakage Estimation. *IEEE Symposium on Security and Privacy* abs/1902.01350 (2019). http://arxiv.org/abs/1902.01350

[15] Giovanni Cherubin, Jamie Hayes, and Marc Juárez. 2017. Website Fingerprinting Defenses at the Application Layer. *PoPETs* 2017, 2 (2017), 186–203. https://doi.org/10.1515/popets-2017-0023

[16] Lénaïc Chizat and Francis Bach. 2020. Implicit Bias of Gradient Descent for Wide Two-layer Neural Networks Trained with the Logistic Loss. In *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria] (Proceedings of Machine Learning Research, Vol. 125)*, Jacob D. Abernethy and Shivani Agarwal (Eds.). PMLR, 1305–1338. http://proceedings.mlr.press/v125/chizat20a.html

[17] Tom Chothia and Apratim Guha. 2011. A Statistical Test for Information Leaks Using Continuous Mutual Information. In *CSF*. IEEE Computer Society, 177–190. http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5991608

[18] Tom Chothia, Yusuke Kawamoto, and Chris Novakovic. 2013. A tool for estimating information leakage. In *International Conference on Computer Aided Verification (CAV)*. Springer, 690–695.

[19] Tom Chothia, Yusuke Kawamoto, and Chris Novakovic. 2014. LeakWatch: Estimating Information Leakage from Java Programs. In *ESORICS (2) (Lecture Notes in Computer Science, Vol. 8713)*, Miroslaw Kutylowski and Jaideep Vaidya (Eds.). Springer, 219–236.

[20] David Clark, Sebastian Hunt, and Pasquale Malacaria. 2001. Quantitative Analysis of the Leakage of Confidential Data. *Electr. Notes Theor. Comput. Sci* 59, 3 (2001), 238–251.

[21] George Cybenko. 1992. Approximation by superpositions of a sigmoidal function. *MCSS* 5, 4 (1992), 455.

[22] Eloi de Chérisey, Sylvain Guilley, Olivier Rioul, and Pablo Piantanida. 2019. Best Information is Most Successful - Mutual Information and Success Rate in Side-Channel Analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019, 2 (2019), 49–79. https://doi.org/10.13154/tches.v2019.i2.49-79

[23] Luc Devroye, László Györfi, and Gábor Lugosi. 1996. *Vapnik-Chervonenkis Theory.* Springer New York, New York, NY, 187–213. https://doi.org/10.1007/978-1-4612-0711-5_12

[24] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository (Heart Disease Data Set). https://archive.ics.uci.edu/ml/datasets/heart+Disease

[25] Cynthia Dwork. 2006. Differential Privacy. In *33rd International Colloquium on Automata, Languages and Programming (ICALP 2006) (Lecture Notes in Computer Science, Vol. 4052)*, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.). Springer, 1–12. http://dx.doi.org/10.1007/11787006_1

[26] Cynthia Dwork, Frank Mcsherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *In Proceedings of the Third Theory of Cryptography Conference (TCC) (Lecture Notes in Computer Science, Vol. 3876)*, Shai Halevi and Tal Rabin (Eds.). Springer, 265–284.

[27] Ehab ElSalamouny and Catuscia Palamidessi. 2020. Full Convergence of the Iterative Bayesian Update and Applications to Mechanisms for Privacy Protection. arXiv:1909.02961 [cs.CR] To appear in the proceedings of EuroS&P.

[28] Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. 2016. *Deep Learning.* MIT Press. 1–775 pages. http://www.deeplearningbook.org/

[29] T. Hastie, R. Tibshirani, and J. Friedman. 2001. *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer-Verlag.

[30] Boris Köpf and David A. Basin. 2007. An information-theoretic model for adaptive side-channel attacks. In *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson (Eds.). ACM, 286–296.

[31] Boris Köpf and Markus Dürmuth. 2009. A Provably Secure and Efficient Countermeasure against Timing Attacks. In *Proceedings of the 2009 22nd IEEE Computer Security Foundations Symposium (CSF '09)*. IEEE Computer Society, USA, 324–335. https://doi.org/10.1109/CSF.2009.21

[32] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[33] Marco Romanelli, Catuscia Palamidessi, and Konstantinos Chatzikokolakis. 2020. Generating Optimal Privacy-Protection Mechanisms via Machine Learning, In Proceedings of the IEEE International Symposium on Computer Security Foundations (CSF). *CoRR.* arXiv:1904.01059 http://arxiv.org/abs/1904.01059

[34] S. Y. Sekeh, B. Oselio, and A. O. Hero. 2020. Learning to Bound the Multi-Class Bayes Error. *IEEE Transactions on Signal Processing* 68 (2020), 3793–3807.

[35] Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding machine learning : from theory to algorithms.* http://www.worldcat.org/search?qt=worldcat_org_all&q=9781107057135

[36] Reza Shokri, George Theodorakopoulos, Jean-Yves Le Boudec, and Jean-Pierre Hubaux. 2011. Quantifying Location Privacy. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 247–262. http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5955408;http://www.computer.org/csdl/proceedings/sp/2011/4402/00/index.html

[37] Reza Shokri, George Theodorakopoulos, Carmela Troncoso, Jean-Pierre Hubaux, and Jean-Yves Le Boudec. 2012. Protecting location privacy: optimal strategy against localization attacks. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, Ting Yu, George Danezis, and Virgil D. Gligor (Eds.). ACM, 617–627. http://dl.acm.org/citation.cfm?id=2382196

[38] Geoffrey Smith. 2009. On the Foundations of Quantitative Information Flow. In *Foundations of Software Science and Computational Structures, 12th International Conference, FOSSACS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings (Lecture Notes in Computer Science, Vol. 5504)*, Luca de Alfaro (Ed.). Springer, 288–302.

[39] Kagan Tumer and Joydeep Ghosh. 2003. Bayes Error Rate Estimation Using Classifier Ensembles. *International Journal of Smart Engineering System Design* 5, 2 (2003), 95–109. https://doi.org/10.1080/10255810305042 arXiv:https://doi.org/10.1080/10255810305042

[40] David H. Wolpert. 1996. The Lack of A Priori Distinctions Between Learning Algorithms. *Neural Computation* 8, 7 (1996), 1341–1390.

A AUXILIARY RESULTS

Proposition A.1 (Bernstein's inequality [8]). *Let $Z_1, \ldots, Z_n \sim Z$ be i.i.d. random variables such that $Z \in [a, b]$ almost surely and let $S_n = \frac{1}{n}\sum_{i=1}^{n} Z_i - \mathbb{E}[Z]$ and $v = Var(Z)$ the variance of Z. Then, for any $\varepsilon > 0$, we have:*

$$\mathbb{P}[S_n \geqslant \varepsilon] \leqslant \exp\left(-\frac{n\,\varepsilon^2}{2\,v + 2(b-a)\varepsilon/3}\right). \tag{35}$$

Compared to the Hoeffding's inequality, it is easy to check that, for regimes where ε is small, Bernstein's inequality offers tighter bounds for $v \ll (b-a)^2$.

Lemma A.2. *Let $\sigma^2 = Var(Z)$ and let Z be a real-valued random variable such that for all $0 < t \leqslant \sigma^2$,*

$$\mathbb{P}(Z \geqslant t) \leqslant 2q \exp\left(-\frac{t^2}{r^2}\right). \tag{36}$$

Then,

$$\int_0^{\sigma^2} \mathbb{P}(Z \geqslant t)dt \leqslant qr\sqrt{\pi}\,\mathrm{erf}\left(\frac{\sigma^2}{r}\right), \tag{37}$$

where, for large x,

$$\operatorname{erf}(x) \approx 1 - \frac{\exp(-x^2)}{x\sqrt{\pi}} + O\left(x^{-1}\exp(-x^2)\right). \tag{38}$$

Proof.

$$\int_0^{\sigma^2} \mathbb{P}(Z \geqslant t)dt \leqslant \int_0^{\sigma^2} 2q \exp\left(-\frac{t^2}{r^2}\right) dt$$
$$= qr\sqrt{\pi}\operatorname{erf}\left(\frac{\sigma^2}{r}\right),$$

and eq. (38) follows from the Taylor's expansion of the erf function. □

B PROOFS FOR THE STATISTICAL BOUNDS

The following lemma is a simple adaption of the uniform deviations of relative frequencies from probabilities theorems in [23].

Lemma B.1. *The following inequality holds:*

$$V_g - V(f_m^\star) \leqslant 2\max_{f\in\mathcal{H}} \left|\widehat{V}_m(f) - V(f)\right|. \tag{39}$$

Proof.

$$\begin{aligned}
V_g - V(f_m^\star) &= V(f^\star) - \widehat{V}_m(f_m^\star) + \widehat{V}_m(f_m^\star) - V(f_m^\star) \\
&\leqslant V(f^\star) - \widehat{V}_m(f_m^\star) + \left|\widehat{V}_m(f_m^\star) - V(f_m^\star)\right| && (40)\\
&\leqslant V(f^\star) - \widehat{V}_m(f^\star) + \left|\widehat{V}_m(f_m^\star) - V(f_m^\star)\right| && (41)\\
&\leqslant \left|V(f^\star) - \widehat{V}_m(f^\star)\right| + \left|\widehat{V}_m(f_m^\star) - V(f_m^\star)\right| && (42)\\
&\leqslant \max_{f\in\mathcal{H}}\left|\widehat{V}_m(f) - V(f)\right| + \max_{f\in\mathcal{H}}\left|\widehat{V}_m(f) - V(f)\right| && (43)\\
&\leqslant 2\max_{f\in\mathcal{H}}\left|\widehat{V}_m(f) - V(f)\right|. && (44)
\end{aligned}$$

□

B.1 Proof of Proposition 3.1

Proposition 3.1 (Uniform deviations). *For all $\varepsilon > 0$,*

$$\mathbb{P}\left(\left|\widehat{V}_n(f_m^\star) - V(f_m^\star)\right| \geqslant \varepsilon\right) \leqslant 2\mathbb{E}\exp\left(-\frac{n\,\varepsilon^2}{2\,\sigma^2_{f_m^\star} + {}^{2\,(b-a)\varepsilon}/_{3}}\right), \tag{12}$$

where the expectation is taken w.r.t. the random training set, and

$$\mathbb{P}\left(V_g - V(f_m^\star) \geqslant \varepsilon\right) \leqslant 2\sum_{f\in\mathcal{H}}\exp\left(-\frac{m\,\varepsilon^2}{8\sigma_f^2 + {}^{4(b-a)\varepsilon}/_{3}}\right). \tag{13}$$

Proof. We first prove (12). We have:

$$\begin{aligned}
&\mathbb{P}\left(\left|\widehat{V}_n(f_m^\star) - V(f_m^\star)\right| \geqslant \varepsilon\right) \\
&= \mathbb{E}_{\mathcal{D}_m\sim P_{XY}^m}\mathbb{P}\left(\left|\widehat{V}_n(f_m^\star) - V(f_m^\star)\right| \geqslant \varepsilon \mid \mathcal{D}_m\right) && (45)\\
&\leqslant 2\mathbb{E}_{\mathcal{D}_m\sim P_{XY}^m}\exp\left(-\frac{n\,\varepsilon^2}{2\sigma^2_{f_m^\star} + {}^{2(b-a)\varepsilon}/_{3}}\right), && (46)
\end{aligned}$$

where (45) follows from the definition of $\mathbb{P}$: we consider the expectation of the probability over all training sets $\mathcal{D}_m$ sampled from P_{XY}, and then for each $\mathcal{D}_m$ we take the probability over all possible validation sets $\mathcal{T}_n$ sampled again from P_{XY}. Consider the series of (X_i, Y_i)'s sampled from P_{XY} that constitute the validation set, and define the random variables $Z_i = g(f_m^\star(Y_i), X_i)$. Note that $\mathcal{D}_m$ determines $f_m^\star$, hence, for a given $\mathcal{D}_m$ the Z_i are i.i.d.. The inequality (46) then follows by applying Proposition A.1. Indeed, since the Z_i's are i.i.d., they all have the same expectation and the same variance, hence $S_n = \frac{1}{n}\sum_{i=1}^n Z_i - \mathbb{E}[Z] = \widehat{V}_n(f_m^\star) - V(f_m^\star)$, and $v = Var(Z) = \sigma^2_{f_m^\star}$. The factor 2 in front of the exp is because we consider the absolute value of S_n.

As for the bound (13), we have:

$$\begin{aligned}
\mathbb{P}\left(V_g - V(f_m^\star) \geqslant \varepsilon\right) &\leqslant \mathbb{P}\left(\max_{f\in\mathcal{H}}\left|\widehat{V}_m(f) - V(f)\right| \geqslant \varepsilon/2\right) && (47)\\
&= \mathbb{P}\left(\bigcup_{f\in\mathcal{H}}\left\{\left|\widehat{V}_m(f) - V(f)\right| \geqslant \varepsilon/2\right\}\right) && (48)\\
&\leqslant \sum_{f\in\mathcal{H}}\mathbb{P}\left(\left|\widehat{V}_m(f) - V(f)\right| \geqslant \varepsilon/2\right) && (49)\\
&\leqslant \sum_{f\in\mathcal{H}} 2\exp\left(-\frac{m\,\varepsilon^2}{8\sigma_f^2 + {}^{4(b-a)\varepsilon}/_{3}}\right), && (50)
\end{aligned}$$

where (47) follows from Lemma B.1, steps (48) and (49) are standard, and (50) follows from the same reasoning that we have applied to prove (12). Here we do not take the expectation on the training sets because in each term of the summation the f is fixed. □

B.2 Proof of Theorem 3.2

Theorem 3.2. *The averaged estimation error of the g-vulnerability can be bounded as follows:*

$$\mathbb{E}\left|V_g - \widehat{V}_n(f_m^\star)\right| \leqslant V_g - \mathbb{E}\left[V(f_m^\star)\right] + \mathbb{E}\left|V(f_m^\star) - \widehat{V}_n(f_m^\star)\right|, \tag{14}$$

where the expectations are understood over all possible training and validation sets drawn according to P_{XY}. Furthermore, let $\sigma^2 = max_{f\in\mathcal{H}} Var\big(g(f(Y), X)\big)$, then :

$$\begin{aligned}
\mathbb{E}\left|V(f_m^\star) - \widehat{V}_n(f_m^\star)\right| \leqslant{}& \frac{4\eta}{n}\exp\left(-\frac{n\sigma^2}{2\eta}\right) \\
&+ \sqrt{\frac{2\sigma^2\eta\pi}{n}}\operatorname{erf}\left({}^{\sigma^2}\Big/\sqrt{\frac{2\sigma^2\eta\pi}{n}}\right), && (15)
\end{aligned}$$

where $\eta = (1 + {}^{(b-a)}/_{3})$ for $\sigma^2 \leqslant \varepsilon$, and, otherwise,

$$\begin{aligned}
V_g - \mathbb{E}\left[V(f_m^\star)\right] \leqslant{}& |\mathcal{H}|\frac{8(1+\eta)}{m}\exp\left(-\frac{m\sigma^2}{4(1+\eta)}\right) + \\
&|\mathcal{H}|\sqrt{\frac{4\sigma^2(1+\eta)\pi}{m}}\operatorname{erf}\left({}^{\sigma^2}\Big/\sqrt{\frac{4\sigma^2(1+\eta)\pi}{m}}\right), && (16)
\end{aligned}$$

with $\operatorname{erf}(\theta) \overset{def}{=} \frac{2}{\sqrt{\pi}}\int_0^\theta \exp(-\mu^2)d\mu$.

Proof. Observe that

$$\begin{aligned}
\mathbb{E}\left|V_g - \widehat{V}_n(f_m^\star)\right| &= \mathbb{E}\left|V_g - V(f_m^\star) + V(f_m^\star) - \widehat{V}_n(f_m^\star)\right| \\
&\leqslant V_g - \mathbb{E}\left[V(f_m^\star)\right] + \mathbb{E}\left|V(f_m^\star) - \widehat{V}_n(f_m^\star)\right|,
\end{aligned}$$

which follows from the triangular inequality.

First, let us call σ^2 the worst case variance defined above which, according to Popoviciu's inequality is upper-bounded by $(b-a)^2/4$. Second, let us consider that one main advantage of deriving bounds from Bernstein's inequality is that it allows allow the upper-bound in eq. (35) grows as $\exp(-n\varepsilon)$ instead of $\exp(-n\varepsilon^2)$ if $v \leqslant \varepsilon$. Moreover eq. (46) is upper-bounded by $2\exp\left(-n\,\varepsilon^2/2\sigma^2+2(b-a)\varepsilon/3\right)$. This said we are going to consider two cases:

i) $\boldsymbol{\sigma^2 \leqslant \varepsilon}$:

$$\mathbb{E}|V(f_m^\star) - \widehat{V}_n(f_m^\star)| \leqslant \int_{\sigma^2}^{\infty} 2\exp\left(-\frac{n\varepsilon^2}{2\sigma^2 + 2(b-a)\varepsilon/3}\right) d\varepsilon \quad (51)$$

$$\leqslant \int_{\sigma^2}^{\infty} 2\exp\left(-\frac{n\varepsilon}{2 + 2(b-a)/3}\right) d\varepsilon \quad (52)$$

$$= \frac{4(1+(b-a)/3)}{n}\exp\left(-\frac{n\sigma^2}{2(1+(b-a)/3)}\right), \quad (53)$$

and then,

$$V_g - \mathbb{E}\big[V(f_m^\star)\big] \leqslant \int_{\sigma^2}^{\infty} \sum_{f\in\mathcal{H}} 2\exp\left(-\frac{m\,\varepsilon^2}{8\sigma_f^2 + 4(b-a)\varepsilon/3}\right) d\varepsilon \quad (54)$$

$$\leqslant \sum_{f\in\mathcal{H}} \int_{\sigma^2}^{\infty} 2\exp\left(-\frac{m\,\varepsilon^2}{8\sigma^2 + 4(b-a)\varepsilon/3}\right) d\varepsilon \quad (55)$$

$$\leqslant \sum_{f\in\mathcal{H}} \int_{\sigma^2}^{\infty} 2\exp\left(-\frac{m\,\varepsilon}{8 + 4(b-a)/3}\right) d\varepsilon \quad (56)$$

$$\leqslant |\mathcal{H}|\frac{8(2+(b-a)/3)}{m}\exp\left(-\frac{3m\sigma^2}{4(6+(b-a))}\right), \quad (57)$$

ii) $\boldsymbol{\sigma^2 > \varepsilon}$:

$$\mathbb{E}|V(f_m^\star) - \widehat{V}_n(f_m^\star)| \leqslant \int_{0}^{\sigma^2} 2\exp\left(-\frac{n\varepsilon^2}{2\sigma^2 + 2(b-a)\varepsilon/3}\right) d\varepsilon \quad (58)$$

$$\leqslant \int_{0}^{\sigma^2} 2\exp\left(-\frac{n\varepsilon^2}{2\sigma^2 + 2(b-a)\sigma^2/3}\right) d\varepsilon \quad (59)$$

$$= \sqrt{\frac{2\sigma^2(1+(b-a)/3)}{n}}\sqrt{\pi}\,\mathrm{erf}\left(\frac{\sigma^2}{\sqrt{\frac{2\sigma^2(1+(b-a)/3)}{n}}}\right), \quad (60)$$

considering $r = \sqrt{\frac{2\sigma^2(1+(b-a)/3)}{n}}$, $q = 1$ and applying lemma A.2. And finally,

$$V_g - \mathbb{E}\big[V(f_m^\star)\big] \leqslant \sum_{f\in\mathcal{H}} \int_{0}^{\sigma^2} 2\exp\left(-\frac{m\,\varepsilon^2}{8\sigma_f^2 + 4(b-a)\varepsilon/3}\right) d\varepsilon \quad (61)$$

$$\leqslant 2|\mathcal{H}| \int_{0}^{\sigma^2} \exp\left(-\frac{m\,\varepsilon^2}{8\sigma^2 + 4(b-a)\varepsilon/3}\right) d\varepsilon \quad (62)$$

$$\leqslant 2|\mathcal{H}| \int_{0}^{\sigma^2} \exp\left(-\frac{m\,\varepsilon^2}{8\sigma^2 + 4(b-a)\sigma^2/3}\right) d\varepsilon \quad (63)$$

$$= |\mathcal{H}|\sqrt{\frac{8\sigma^2 + 4\sigma^2(b-a)/3}{m}}\sqrt{\pi}\,\mathrm{erf}\left(\frac{\sigma^2}{\sqrt{\frac{8\sigma^2+4\sigma^2(b-a)/3}{m}}}\right), \quad (64)$$

according to lemma A.2 with $r = \sqrt{\frac{8\sigma^2+4\sigma^2(b-a)/3}{m}}$ and $q = |\mathcal{H}|$.

□

B.3 Proof of Corollary 3.4

Corollary 3.4. *The sample complexity of the ERM algorithm g-vulnerability is bounded from above by the set of values satisfying:*

$$M(\varepsilon,\delta) \leqslant \frac{8\,\sigma^2 + 4\,(b-a)\varepsilon/3}{\varepsilon^2}\ln\left(\frac{2\,|\mathcal{H}|}{\delta-\Delta}\right), \quad (20)$$

$$N(\varepsilon,\delta) \leqslant \frac{2\,\sigma^2 + 2\,(b-a)\varepsilon/3}{\varepsilon^2}\ln\left(\frac{2}{\Delta}\right), \quad (21)$$

for all Δ *such that* $0 < \Delta < \delta$.

Proof. We first notice that

$$\mathbb{P}\left(|V_g - \widehat{V}_n(f_m^\star)| \geqslant \varepsilon\right) \leqslant \mathbb{P}\left(V_g - V(f_m^\star) \geqslant \varepsilon\right) + \mathbb{P}\left(|V(f_m^\star) - \widehat{V}_n(f_m^\star)| \geqslant \varepsilon\right), \quad (65)$$

and thus from (12), (13) in Proposition 3.1, we have:

$$\mathbb{P}\left(|V_g - \widehat{V}_n(f_m^\star)| \geqslant \varepsilon\right) \leqslant 2|\mathcal{H}|\exp\left(-\frac{m\,\varepsilon^2}{8\,\sigma^2 + 4\,(b-a)\varepsilon/3}\right) + 2\exp\left(-\frac{n\,\varepsilon^2}{2\sigma^2 + 2\,(b-a)\varepsilon/3}\right). \quad (66)$$

Let us require:

$$2\,|\mathcal{H}|\,\exp\left(-\frac{m\,\varepsilon^2}{8\,\sigma^2 + 4\,(b-a)\varepsilon/3}\right) \leqslant (\delta-\Delta), \quad (67)$$

$$2\,\exp\left(-\frac{n\,\varepsilon^2}{2\,\sigma^2 + 2\,(b-a)\varepsilon/3}\right) \leqslant \Delta, \quad (68)$$

which satisfies the desired condition:

$$\mathbb{P}\left(|V_g - \widehat{V}_n(f_m^\star)| \geqslant \varepsilon\right) \leqslant \delta, \quad (69)$$

for any $0 < \Delta < \delta$. Finally, from the previous inequality we can derive lower bounds on n and m:

$$m \geqslant \frac{8\,\sigma^2 + 4\,(b-a)\varepsilon/3}{\varepsilon^2}\ln\left(\frac{2\,|\mathcal{H}|}{\delta-\Delta}\right), \quad (70)$$

$$n \geqslant \frac{2\,\sigma^2 + 2\,(b-a)\varepsilon/3}{\varepsilon^2}\ln\left(\frac{2}{\Delta}\right), \quad (71)$$

which by definition of sample complexity shows the corollary. □

C PRE-PROCESSING

C.1 Data pre-processing

Theorem 4.1 (Correctness of data pre-processing). *Given a prior* π*, a channel* C*, and a gain function* g*, we have*

$$V_g(\pi, C) \;= \alpha \cdot V_{g_{\mathrm{id}}}(\xi, E)\,,$$

where α*,* ξ *and* E *are those defined in* (23), (24) *and* (25)*, respectively, and* g_{id} *is the identity function (cfr. section 2), i.e., the gain function corresponding to the Bayesian adversary.*

Proof.

$$V_{g_{\text{id}}}(\xi, E) = \sum_y \max_w \sum_{w'} \xi w' \cdot E_{w'y} \cdot g_{\text{id}}(w, w') = \tag{72}$$

$$\sum_y \max_w (\xi_w E_{wy}) = \sum_y \max_w P_{WY}(w, y) = \sum_y \max_w \frac{U(w, y)}{\alpha} \tag{73}$$

$$= \frac{1}{\alpha} \cdot \sum_y \max_w \sum_x \pi_x \cdot C_{xy} \cdot g(w, x) = 1/\alpha \cdot V_g(\pi, C) \tag{74}$$

□

C.2 Channel pre-processing

Theorem 4.2 (Correctness of channel pre-processing). *Given a prior π and a gain function g, we have that, for any channel C:*

$$V_g(\pi, C) \;= \beta \cdot V_{g_{\text{id}}}(\tau, RC) \quad \textit{for all channels } C.$$

where β, τ and R are those defined in (27) *and* (28).

Proof. In this proof we use a notation that highlights the structure of the preprocessing. We will denote by G be the matrix form of g, i.e., $G_{wx} = g(w, x)$, and by Ψ^π the square matrix with π in its diagonal and 0 elsewhere. We have that $\beta = \|G\Psi^\pi\|_1 = \sum_{w,x} G_{wx}\pi_x$, which is strictly positive because of the assumptions on g and π. Furthermore, we have

$$\tau^T \;=\; \beta^{-1} G\Psi^\pi \mathbf{1}\,, \qquad R \;=\; \beta^{-1}(\Psi^\tau)^{-1} G\Psi^\pi\,,$$

where $\mathbf{1}$ is the vector of 1s and τ^T represents the transposition of vector τ. Note that $(\Psi^\tau)^{-1}$ is a diagonal matrix with entries τ_w^{-1} in its diagonal. If $\tau_w = 0$ then the row $R_{w,\cdot}$ is not properly defined; but its choice does not affect $V_{g_{\text{id}}}(\tau, RC)$ since the corresponding prior is 0; so we can choose $R_{w,\cdot}$ arbitrarily (or equivalently remove the action w, it can never be optimal since it gives 0 gain). It is easy to check that τ is a proper distribution and R is a proper channel:

$$\begin{aligned} \textstyle\sum_w \tau_w \;&=\; \textstyle\sum_w \beta^{-1} \sum_x G_{wx}\pi_x \;&=\; \beta^{-1}\beta \;&=\; 1\,, \\ \textstyle\sum_x R_{w,x} \;&=\; \textstyle\sum_x \frac{1}{\tau_w}\beta^{-1} G_{wx}\pi_x \;&=\; \frac{\tau_w}{\tau_w} \;&=\; 1\,. \end{aligned}$$

Moreover, it holds that:

$$\beta\Psi^\tau R \;=\; \beta\Psi^\tau \beta^{-1}\Psi^{\tau-1} G\Psi^\pi \;=\; G\Psi^\pi\,.$$

The main result follows from the trace-based formulation of posterior g-vulnerability [4], since for any channel C and strategy S, the above equation directly yields

$$\begin{aligned} V_g(\pi, C) \;=\; \max_S \mathbf{tr}(G\Psi^\pi CS) \;&=\; \beta \cdot \max_S \mathbf{tr}(\Psi^\tau RCS) \\ &=\; \beta \cdot V_{g_{\text{id}}}(\tau, RC)\,, \end{aligned}$$

where $\mathbf{tr}(\cdot)$ is the matrix trace. □

C.3 Data pre-processing when g is not integer

Approximating g so that it only takes values $\in \mathbb{Q}_{\geqslant 0}$ allows us to represent each gain as a quotient of two integers, namely

$$\text{Numerator}(G_{w,x}) / \text{Denominator}\,(G_{w,x}).$$

Let us also define

$$K \stackrel{\text{def}}{=} \text{lcm}_{wx}(\text{Denominator}(G_{w,x})), \tag{75}$$

where $\text{lcm}(\cdot)$ is the least common multiple. Multiplying G by K gives the integer version of the gain matrix that can replace the original one. It is clear that the calculation of the least common multiplier, as well as the increase in the amount of data produced during the dataset building using a gain matrix forced to be integer, might constitute a relevant computational burden.

D ANN MODELS

We list here the specifics for the ANNs models used in the experiments. All the models are simple feed-forward networks whose layers are fully connected. The activation functions for the hidden neurons are rectifier linear functions, while the output layer has softmax activation function.

The loss function minimized during the training is the cross entropy, a popular choice in classification problems. The remapping $\mathcal{Y} \to \mathcal{W}$ can be in fact considered as a classification problem such that, given an observable, a model learns to make the best guess.

For each experiments, the models have been tuned by cross-validating them using one randomly chosen training sets among the available ones choosing among the largest in terms of samples. The specs are listed experiment by experiment in table 2.

E FREQUENTIST APPROACH DESCRIPTION

In the frequentist approach the elements of the channel, namely the conditional probabilities $P_{Y|X}(y|x)$, are estimated directly in the following way: the empirical prior probability of x, $\widehat{\pi}_x$, is computed by counting the number of occurrences of x in the training set and dividing the result by the total number of elements. Analogously, the empirical joint probability $\widehat{P}_{XY}(x, y)$ is computed by counting the number of occurrences of the pair (x, y) and dividing the result by the total number of elements in the set. The estimation $\widehat{C}_{xy}$ of C_{xy} is then defined as

$$\widehat{C}_{xy} = \frac{\widehat{P}_{XY}(x, y)}{\widehat{\pi}(x)}. \tag{76}$$

In order to have a fair comparison with our approach, which takes advantage of the fact that we have several training sets and validation sets at our disposal, while preserving at the same time the spirit of the frequentist approach, we proceed as follows: Let us consider a training set $\mathcal{D}_m$, that we will use to learn the best remapping $\mathcal{Y} \to \mathcal{W}$, and a validation set $\mathcal{T}_n$ which is then used to actually estimate the g-vulnerability. We first compute $\widehat{\pi}$ using $\mathcal{D}_m$. For each y in $\mathcal{Y}$ and for each $x \in \mathcal{X}$, the empirical probability $\widehat{P}_{X|Y}$ is computed using $\mathcal{D}_m$ as well. In particular, $\widehat{P}_{X|Y}(x|y)$ is given by the number of times x appears in pair with y divided by the number of occurrences of y. In case a certain y is in $\mathcal{T}_n$ but not in $\mathcal{D}_m$, it is assigned the secret $x' = \arg\max_{x \in \mathcal{X}} \widehat{\pi}$ so that $\widehat{P}_{X|Y}(x'|y) = 1$ and $\widehat{P}_{X|Y}(x) = 0, \forall x \neq x'$. It is now possible to find the best mapping for each y defined as $w(y) = \arg\max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \widehat{P}_{X|Y}(x|y) g(w, x)$. Now we compute the empirical joint distribution for each (x, y) in $\mathcal{T}_n$, namely $\widehat{Q}_{XY}$, as the number of occurrences of (x, y) divided by the total number of samples in $\mathcal{T}_n$. We now estimate the g-vulnerability on the validation samples according to:

$$\widehat{V}_n = \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} \widehat{Q}_{XY}(x, y) g(w(y), x). \tag{77}$$

		Hyper-parameters				
Experiment	**Pre-processing**	learning rate	hidden layers	epochs	hidden units per layer	batch size
Multiple guesses	Data	10^{-3}	3	700	[100, 100, 100]	1000
	Channel	10^{-3}	3	500	[100, 100, 100]	1000
Location Priv.	Data	10^{-3}	3	1000	[500, 500, 500]	200, 500, 1000
	Channel	10^{-3}	3	200, 500, 1000	[500, 500, 500]	20, 200, 500
Diff. Priv.	Data	10^{-3}	3	500	[100, 100, 100]	200
	Channel	10^{-3}	3	500	[100, 100, 100]	200
Psw SCA	-	10^{-3}	3	700	[100, 100, 100]	1000

Table 2: Table with the hyper-parameters setting for each one of the experiments above. When multiple values are provided for the parameters of an experiment it is to be intended that each value corresponds to a specific size of the training set (sorted from the smallest to the largest number of samples).

F ANN: MODEL SELECTION AND IMPACT ON THE ESTIMATION

In this section we are going to:

- briefly summarize the widely known background of the model selection problem from the machine learning standpoint;
- show, through a new set of experiments, how this problem affects the leakage estimation;
- propose a heuristic which can help the practitioner in the choice of the model on the same line as classical machine learning techniques.

The problem of model selection in machine learning is still an open one and, although a state-of-the art algorithm does not exists, heuristics can be proposed to lead the practitioner in the hard task of choosing a model over others. First, let us underline that the choice of a specific model in the context of machine learning, and specifically neural networks (and deep learning), must go through the hyper-parameter optimization procedure. In fact, if nowadays neural nets and especially deep models represent the state-of-the-art solutions to most of the automatic decision problem, it is also true that, with respect to other simpler methods, they introduce the need for hyper-parameters optimization. Some techniques, such as grid and random search as well as Bayesian optimization have been suggested during the years, especially when not so many parameters need to be tuned. Two aspects must be considered:

(1) the hyper-parameter optimization relies on try and error strategy,
(2) the results are highly dependent on the data distribution and how well the samples represent such distribution.

In particular, if we consider the typical classification problem framework in neural nets (which we build on to create our framework) we expect the network to reproduce in output the distribution $P_{class|input}$ from the observed data. In this context, the practitioner should be careful to avoid two main problems which might affect the models, namely under-fitting and over-fitting. Both problems undermine the generalization capabilities of the models: the former occurs when the model is too simple to correctly represent the distribution; the latter occurs when the model is over-complicated, especially for the given amount of samples, and it fits the training data "too well" but this does not translates into good performances on other samples drawn from the same distribution.

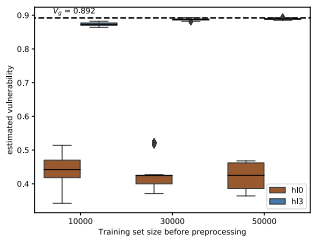
In order to understand how these problems impact our framework, we propose an analysis of the first experiment presented in the paper, considering different networks models and focusing on the different choice of number of hidden layers. In fig. 12 we compare a model with no hidden layers, *hl0*, and the three hidden layers model presented in the paper, *hl3*. Using neural networks without hidden layers is not common. Indeed, theoretical results (cfr. [21]) state that the simplest universal approximator can be modeled as a one hidden layered neural network. Although this holds in theory, in practice it is well known that this might require layers with too many neurons, and therefore, multiple hidden layers architectures have been gaining ground in real world applications.

Therefore, we do not expect too much from the network with no hidden layers and, indeed, the results represented in fig. 12a and fig. 12b show that the estimation capabilities of this shallow model are very far from the performance obtained with the three layered model. The model with no hidden layer is too simple to reproduce the input data distribution and therefore it does not generalize well in the problem of predicting the best (i.e. most probable) w when a certain y is input.

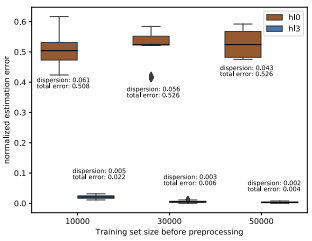
Let us now focus on the results in fig. 13, where we compare the results of three different models with one, two and three hidden layers (respectively *hl1*, *hl2*, and *hl3*). With this specific experiment we want to:

- analyze the behavior of different models when we change the number of hidden layer (maintaining the number of hidden neurons per layer fixed to 100) and we consider different sizes for the learning set;
- introduce a possible heuristic to guide practitioners in the model selection task.

As we can see, observing the box-plots in fig. 13a and fig. 13b from left to right, when the amount of samples is relatively small, the shallow model, i.e. *hl1* performs slightly better than the other two models. This is because, since the samples provided are not enough to accurately describe the distribution, a shallow model would be prone to under-fitting the training data, producing what seems to be a more general decision. However, as the number of samples increases, and consequently, we have a better representation of the distribution through the the data samples, a deeper model is able



(a) Estimated vulnerability.



(b) Normalized estimation error.

Figure 12: Multiple guesses scenario, comparison between a model with no hidden layers (*hl0*) and the three hidden layered model present in the paper (*hl3*).

to reproduce the data distribution with increased accuracy. This results in better performances when trying to predict the best w for a given input y. As one can see the three hidden layered model keeps improving when the amount of samples available for the training increases. The improvement is limited for the two hidden layers model while for the shallowest model, i.e. *hl1*, there is not meaningful improvement at all.

Therefore, provided the availability of enough samples, a good heuristic for the practitioner would be to pick the model that maximizes the leakage, which represents the strongest adversary. In practice, this boils down to trying several models increasing their complexity as long as an increased complexity translates into a higher leakage estimation. This also holds for models with different architecture: if switching from dense to convolutional layers results in a higher leakage estimation, then the convolutional model should be preferred. Even though this is still an open problem for the machine learning field, and we do not provide any no guarantees that, eventually, the optimal model is going to be retrieved, this can be considered a valid empirical approach. Indeed, the principle of no free lunch in machine learning tells that no model can guarantee convergence on its final sample performance. Therefore when a finite amount of sample is available, the practitioner should evaluate several model and stick to a heuristic, as the one we suggest, in order to select the best model.

In order to strengthen this point and also address the comment on the use of different architecture and their impact on the estimation, we produce yet another experimental result. Inspired by [34], we consider the problem of estimating the Bayes error rate (BER, aka Bayes risk or Bayes leakage) using the MNIST dataset (a benchmark dataset for ML classification problems). In [34], the authors use an empirical method to estimate bounds for the BER and, in order to do so, they need to perform dimensionality reduction (they try both principal component analysis, or PCA, and auto-encoding via an auto-encoder neural network). Indeed MNIST, being a 28px $\times$ 28px images dataset, contains high dimensionality samples and all the three applied methods (bounds estimation, k-NN and random forest) benefit from dimensionality reduction. Given that the samples distribution for MNIST is unknown, in [34] the authors aim at checking whether the three estimations confirm each other. And indeed they do. However, it is well known that in many image classification tasks, the state-of-the-art is represented by deep learning and, for instance convolutional neural nets. While in [34] the authors study how the addition of new layers to the model affects the bound estimation, we focus on the comparison between two network models, a dense and a convolutional one.

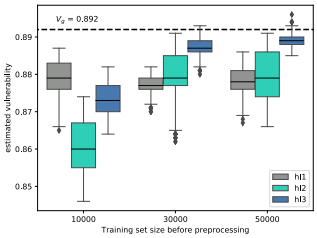
In particular, the dense network model we consider has one hidden layer with 128 hidden units and ReLu activation function. Such an architecture corresponds to a model with 101700 training parameters (connection weights and bias weights). In order to reduce the tendency to over-fit the training data, we consider improving the model with two dropout layers, one before the hidden layer and one after, with dropout rate of 20% and 50% respectively.

The convolutional neural network we consider is based on the one proposed in [32], which goes by the name of LeNet. In particular, the implementation of this net only requires 38552 training parameters, almost one third of those required by the dense model which is shallower but has many hidden neurons. This convolutional network consists of a couple of 2D convolutional layers with 3×3 kernel, alternated with a couple of average pool layer. We then have two dense layers with 60 and 42 hidden neurons respectively and both preceded by dropout layers with dropout rate of 40% and 30% respectively.

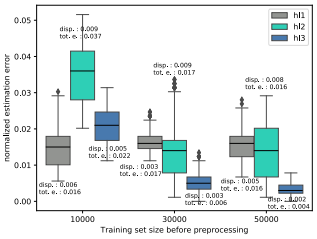
Both the dense and the LeNet model have a soft max final layer with 10 nodes, one for each class. We train both models on the MNIST 50000 training samples. We split the remaining 10000 samples set into 10 subsets, each with 1000 samples. We use the trained models to estimate the BER on these 10 subsets and we obtain the results represented in fig. 14, where the estimated vulnerability is the estimated BER. The average values, represented by the black horizontal lines within the box-plots, are directly comparable to the results in [34].

We notice that:

- considering the aforementioned paper, the dense network's performances are comparable to those of the random forest and k-NN algorithms and the LeNet's performances are comparable to those of the convolutional net considered by the authors;
- the LeNet model's estimate is lower, i.e. the modeled Bayesian adversary is stronger than in the case of the dense net;



(a) Estimated vulnerability.



(b) Normalized estimation error.

Figure 13: Multiple guesses scenario, comparison between a model with one hidden layers (*hl1*), a model with two hidden layers (*hl2*), and the three hidden layered model presented in the paper (*hl3*).

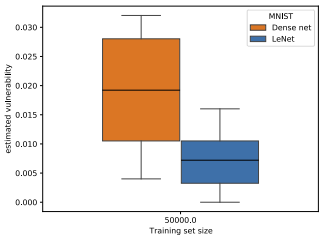


Figure 14: MNIST experiment: dense network vs. LeNet model estimation of the Bayes risk. A smaller risk corresponds to a higher leakage and a more powerful adversary able to take more advantage of the information revealed by the data.

- according to the previously proposed heuristics, given that we can only estimate the Bayes leakage from samples for the MNIST case, the results of the LeNet model are of higher interest, given that it is bigger than the leakage estimate through the bound;
- it is therefore important, when only relying on data for leakage estimation, to compare different models and always look for the state-of-the-art to design the adversary which exploits the system's leakage.

G MAJORITY VOTE

In this section we show an alternative procedure to perform leakage estimation. Given a set of models, instead of using each one of them to obtain an estimate, we derive a new model from them and we use this new model to estimate the leakage. According to [39], we obtain the new model by taking a majority vote on the predictions of each model in the model set, i.e. when each one of the models receives the input it outputs a class. The class eventually assigned to the observable is the class most frequently predicted by the model ensemble (or a random one in case of ties, since we are considering the simplest way to aggregate models).

We consider the first experiment proposed in our paper, in particular the 5 models obtained by training on the 5 i.i.d. training sets of size 10K samples.

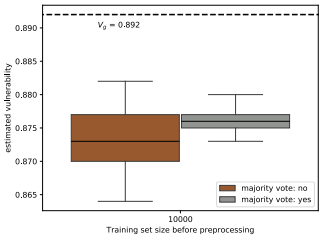
As we can see in fig. 15, the box-plot corresponding to the estimation not based on the majority vote is the same already reported (cfr. section 5.4). Compared to it, the one based on majority vote shows a higher (and therefore more precise) average estimation and a lower dispersion.

However, it is important to notice that, when in this case we consider a majority vote ensemble, since we are exploiting the "expertise" of many models trained on i.i.d. training samples of size 10K, we are actually exploiting the knowledge coming from 50K samples. We therefore analyze the case in which, according to what has already been done in section 5.4, the same 50K samples, obtained by merging the 5 datasets with 10K samples each, are used to train a single model. The results are showed in fig. 16. In this case, it is clear that having multiple weak classifiers and taking the majority vote according to the simple procedure described above, gives worse results in terms of leakage estimation performances than using all the samples to train a strong classifier.

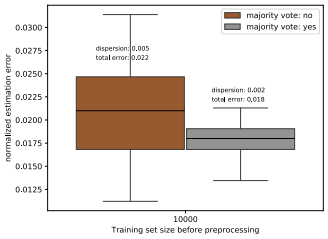
H SUPPLEMENTARY PLOTS

In the next pages, we show supplementary plots for each experiment presented in section 5. In particular, we have included the plots representing the comparison between the estimated vulnerability and the real one, and the plots showing the comparison between the frequentist approach and ours.

Figure 17 is related to the multiple guesses scenario, Figure 19 is related to the location privacy one, Figure 20 is related to the differential privacy experiment, and Figure 18 to the password checker one.

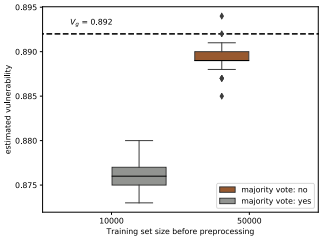


(a) Estimated vulnerability.

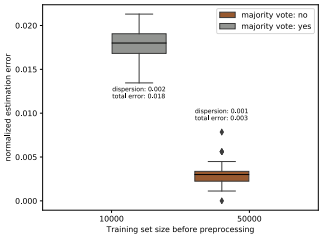


(b) Normalized estimation error.

Figure 15: Comparison between our leakage estimate and the one obtained with a majority vote based model. In both cases each model is trained on 10K samples.

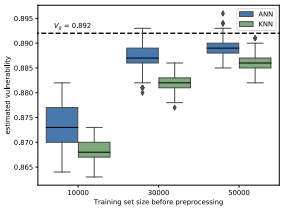


(a) Estimated vulnerability.

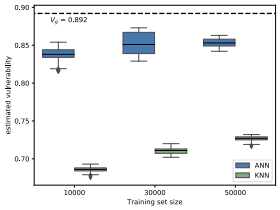


(b) Normalized estimation error.

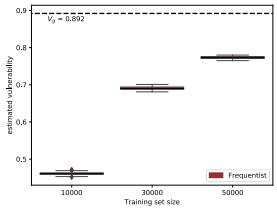
Figure 16: Comparison between the majority vote model and a model trained on all the samples available to each model involved in the majority vote.



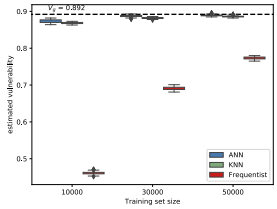
(a) Vulnerability estimation for ANN and k-NN with data pre-processing.



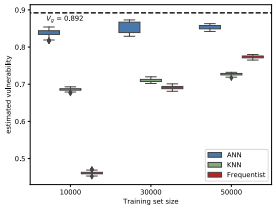
(b) Vulnerability estimation for ANN and k-NN with channel pre-processing.



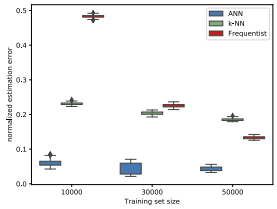
(c) Vulnerability estimation for the frequentist approach.



(d) Vulnerability estimation for ANN and k-NN with data pre-processing, and the frequentist approach.

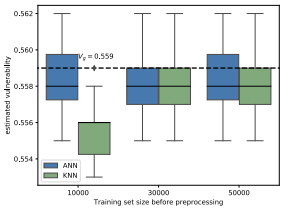


(e) Vulnerability estimation for ANN and k-NN with channel pre-processing, and the frequentist approach.

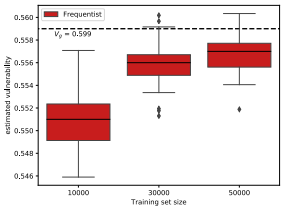


(f) Normalized estimation error for ANN and k-NN with channel pre-processing, and the frequentist approach.

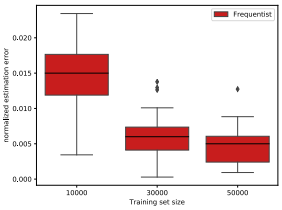
Figure 17: Supplementary plots for the multiple-guesses experiment.



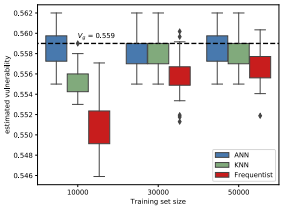
(a) Vulnerability estimation for ANN and k-NN with data and channel pre-processing.



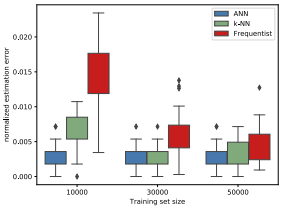
(b) Vulnerability estimation for the frequentist approach.



(c) Normalized estimation error for the frequentist approach.

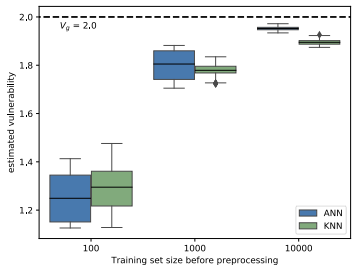


(d) Vulnerability estimation for ANN and k-NN with data and channel pre-processing, and the frequentist approach.

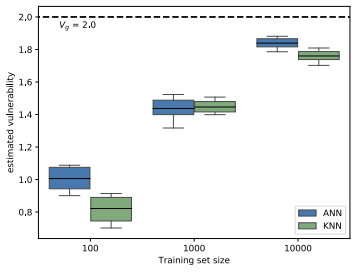


(e) Normalized estimation error for ANN and k-NN with data and channel pre-processing, and the frequentist approach.

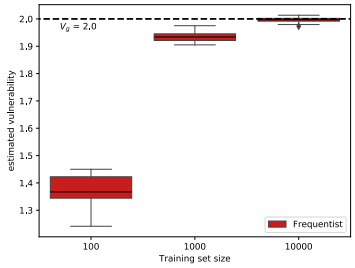
Figure 18: Supplementary plots for the password-checker experiment.



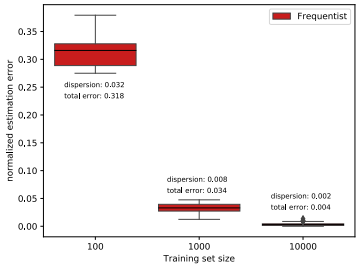
(a) Vulnerability estimation for ANN and k-NN with data pre-processing.



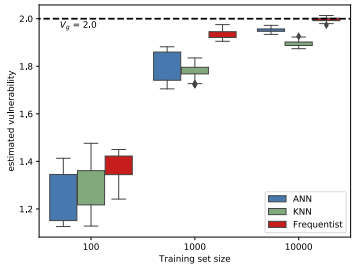
(b) Vulnerability estimation for ANN and k-NN with channel pre-processing.



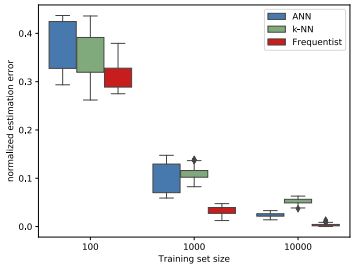
(c) Vulnerability estimation for the frequentist approach.



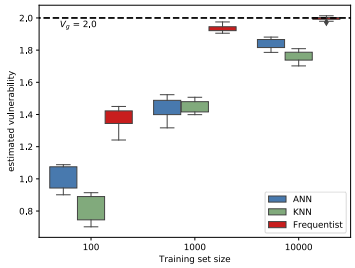
(d) Normalized estimation error for the frequentist approach.



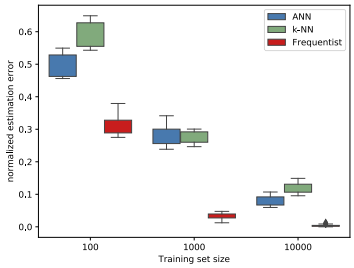
(e) Vulnerability estimation for ANN and k-NN with data pre-processing, and the frequentist approach.



(f) Normalized estimation error for ANN and k-NN with data pre-processing, and the frequentist approach.

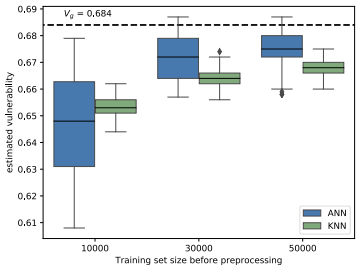


(g) Vulnerability estimation for ANN and k-NN with channel pre-processing, and the frequentist approach.

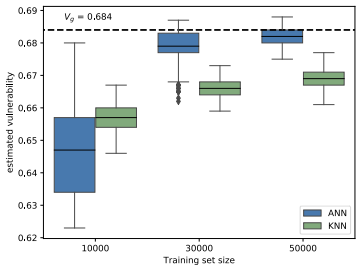


(h) Normalized estimation error for ANN and k-NN with channel pre-processing, and the frequentist approach.

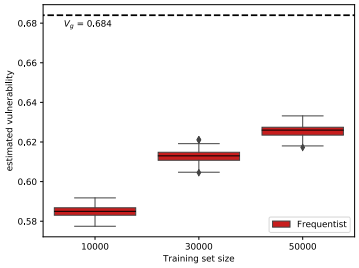
Figure 19: Supplementary plots for the location-privacy experiment.



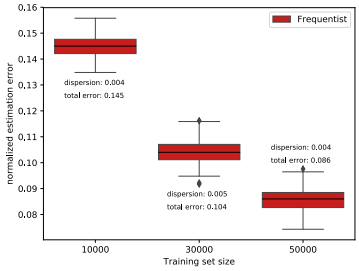
(a) Vulnerability estimation for ANN and k-NN with data pre-processing.



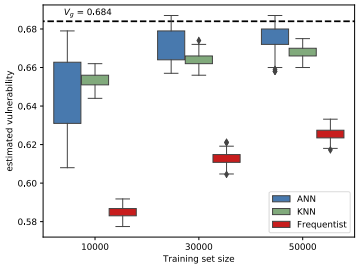
(b) Vulnerability estimation for ANN and k-NN with channel pre-processing.



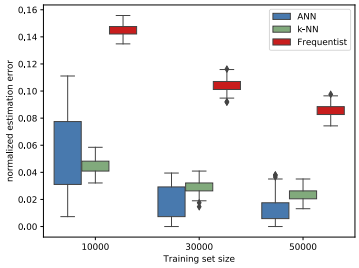
(c) Vulnerability estimation for the frequentist approach.



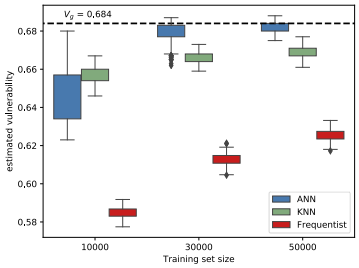
(d) Normalized estimation error for the frequentist approach.



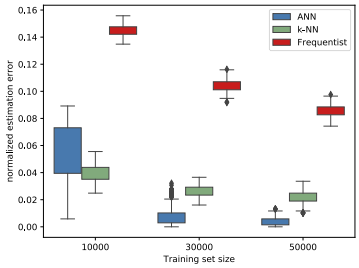
(e) Vulnerability estimation for ANN and k-NN with data pre-processing, and the frequentist approach.



(f) Normalized estimation error for ANN and k-NN with data pre-processing, and the frequentist approach.



(g) Vulnerability estimation for ANN and k-NN with channel pre-processing, and the frequentist approach.



(h) Normalized estimation error for ANN and k-NN with channel pre-processing, and the frequentist approach.

Figure 20: Supplementary plots for the differential-privacy experiment.