Black-box Security

Measuring Black-box Information Leakage via Machine Learning

by

Giovanni Cherubin

Submitted for the degree of

Doctor of Philosophy in Information Security and Machine Learning

at

Royal Holloway, University of London

Declaration of Authorship

I, Giovanni Cherubin, hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Giovanni Cherubin
13 October 2018

Abstract

Determining how much information about a secret is leaked by a system is one of the most fundamental questions in security and privacy. It gives roots to several fields, such as Cryptography and side channel studies, and it has countless applications, ranging from network traffic analysis attacks to program analysis.

In this manuscript, we wish to measure the leakage (or *security*) of a system, considered as a black-box: we assume no knowledge of its internals, and we base our estimates on examples of secret inputs and respective outputs. We refer to this practice as *Black-box security*, which can be used whenever the system cannot be modelled formally (e.g., because its internals are too complex). Black-box security methods have been historically based on classical Statistics ideas, which although caused strong limitations: they required observing at least one example for each input-output combination, which does not scale to large real-world systems (e.g., they need several millions examples for a 10 bits input and 10 bits output), nor to those with continuous output.

We here introduce new principles for Black-box security estimation, which originate from the Machine Learning (ML) theory. They are based on the following observation: measuring the leakage of a system is equivalent to estimating the error of an ML rule from a particular class: the *universally consistent* rules. This gives access to several new Black-box security estimators, which scale to large real-world systems, requiring fewer examples than previous methods. This also allows bringing from the ML literature: impossibility results, and the idea of features to improve an estimator's convergence.

We apply these techniques to real-world problems, such as i) user location data, obfuscated with location-privacy mechanisms, and ii) for measuring the security of defences against a major traffic analysis attack, Webpage Fingerprinting (WF). Notably, the latter constitutes, to the best of our knowledge, the first security estimation method for generic WF defences, after roughly 15 years since this attack's introduction. We also suggest several extensions of the framework (e.g., continuous secret input space, and more general classes of adversaries), some of which inspired by recent advances in the ML theory (Conformal Prediction), and we envision future applications for our methods (e.g., Membership Inference attacks, and generic ML-based attacks).

Acknowledgements

I wish to express my deepest gratitude to Alex Gammerman, my supervisor through my MSc and doctoral studies. He gave me wise support, courage and trust, and he opened for me the gates to the most beautiful ideas in learning theory. His friendly advice accompanied me throughout these years, and for that I thank him sincerely. I am truly grateful to Kenny Paterson, my adviser, who followed my steps closely, and guided me through the various security aspects of my research. He also gave me invaluable lessons on academic writing and on how to communicate my thoughts.

I discussed this work with several people, who influenced me profoundly.
I thank Catuscia Palamidessi and Kostas Chatzikokolakis, who hosted me during short-but-intense visits at École Polytechnique during the last years of my PhD, and introduced me to Quantitative Information Flow research. Our collaboration has been incredibly fruitful and enjoyable, and I look forward to learning more and more from them. I thank Marco Stronati, a good friend, who never shied away from listening and giving me advice, and thanks to whom I met Catuscia and Kostas. I thank Tom Ristenpart, for hosting me at Cornell Tech to work on Membership Inference attacks; from him, I learned more on how attacks are formalised in Cryptography. I express my gratitude to HP Labs in Bristol, where Adrian Baldwin and Jonathan Griffin hosted me for a fantastic internship; from them, I learned to juggle between academic and industrial research, and the true value of rest time for productivity and creativity. Furthermore, I want to sincerely thank my other collaborators: Marc Juarez and Jamie Hayes, with whom I had one of the most fun and synergistic paper writing experiences, and Ilia Nouretdinov, who helped me a lot in my first steps within Conformal Prediction. I thank Volodya Vovk, with whom I had some of the most exciting conversations on learning, ranging from its Philosophy to its computational foundations, and many people from RHUL and beyond, for the numerous chats on this manuscript and related topics: Martin Albrecht, Lorenzo Cavallaro, Ben Curtis, Jason Crampton, Alex Davidson, Jean Paul Degabriele, Naomi Farley, Gregory Fenn, Adrià Gascón, Paul Grubbs, Tor-

ben Hansen, Jonathan Hoyland, Roberto Jordaney, Alexey Koloydenko, David Korczynski, Marie-Sarah Lacharite, Zhiyuan Luo, Thalia May, Feargus Pendlebury, Rachel Player, Eamonn Postlethwaite, Dusan Repel, Claudio Rizzo, Sam Scott, Pip Thornton, Paolo Toccaceli, Thyla van der Merwe, Fernando Virdia, and Joanne Woodage. I also want to thank the rest of the Centre for Doctoral Training (CDT) cohort I was part of: Simon Bell, Andreas Haggman, Suleman Ibrahim, Ela Lee, Carlton Shepherd; it has been a fantastic journey together.

I am grateful to the EPSRC Centre for Doctoral Training (CDT) in Cyber Security, coordinated by Carlos Cid with Claire Hudson, for generously supporting my travels and studentship. It allowed me to attend a great deal of conferences in a large number of countries: without this, many of my collaborators and publications may not be such.

I thank my friend Joseph O'Brien, and landlord during half of my doctoral studies; he tirelessly proofread my early writings, and gave me moral strength. And I thank the people who kept alive my insatiable curiosity and quest for Truth: Ricciarda De Caria, Ledo Stefanini, Lorenzo Grespan, and the late Sergio Paleari – a genius forgotten by society. I finally thank my parents and brother, who strongly and lovingly believed in me, and Elisa, who patiently supported me through these years, and gave me love I cannot describe. *To her, I dedicate this thesis.*

Contents

Luck—or lack thereof—only
exists
in the small sample

Part I

Leakage estimation via Machine Learning

1 Introduction

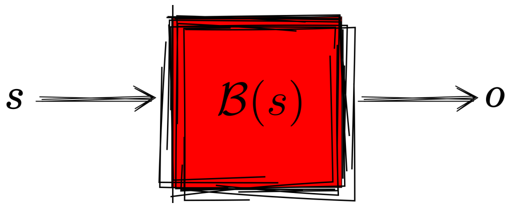

Figure 1.1: We consider a system as a black-box, taking a secret input $s \in \mathcal{S}$ and returning an output $o \in \mathcal{O}$ accordingly. Our goal is to measure how much o leaks about the input s.

Measuring how much a system leaks about its secret inputs is at the foundations of Information Security: from Cryptography to side channel analysis, knowing the leakage of a system is essential for designing systems that minimise the amount of information an adversary learns from their outputs.

Informally, by *security* of a system we mean various related concepts:

- how much information the system reveals about its secret inputs in its outputs; *outputs* here have a generic connotation: they represent any measurement that one can make to the system (e.g., running time, power consumption, actual output values);
- how hard it is for an adversary to predict the secret corresponding to a system's output.

Throughout this manuscript, we will generally use the terms "leakage" and "security" interchangeably. A technical difference exists between them – a leakage measure is also a type of security measure (chapter 4); however, they will represent for us the same concept in informal parlance.

Black-box security Whenever we can describe formally the internals of a system, we may be able to determine its security analytically (although possibly facing computational barriers). Unfortunately, most real-world systems are too complex to describe in a closed-form. Alternatively, one could add noise to a system's output, and achieve bounds on its security, as with Differential Privacy mechanisms (Dwork, 2006); however, i) determining the noise and the corresponding security level for some systems is often tedious, and ii) these methods usually do not account for the security of the system before the noise is applied: the system may already exhibit no leakage, and adding noise may just affect its utility.

In this manuscript, we seek to establish new principles for measuring the leakage of a system about its secret inputs; we work under the *black-box* assumption: the system's internals are unknown, and we estimate its leakage by only looking at examples of secret inputs and respective outputs. The basic idea is as follows: we query the system many times, for our choice of inputs, and observe the respective outputs; then we wish to estimate the system's leakage from these input-output examples, for some definition of leakage. We make no assumption on the system itself: the relation between inputs and outputs is ruled by an unknown probability distribution; however, we require that the system (and this probability distribution) does not change between queries. We will look for estimators that only require a few examples to attain good estimates.

Quantitative Information Flow The problem of leakage estimation has been the central subject of Quantitative Information Flow (QIF) research. In the context of Black-box security, the QIF community introduced several estimation tools, originating from Information Theory and classical Statistics principles (Chatzikokolakis et al., 2010; Chothia et al., 2013, 2014). These tools are based on the idea of estimating the probability distribution induced by a system by counting the relative frequencies of its inputs and outputs; once the distribution is estimated, one can compute the desired leakage measure analytically. We refer to this method as the *frequentist* approach. Note that the same approach has been used in other areas of security under different names (e.g., Cai et al. (2014c)).

Unfortunately, the frequentist approach is strongly limited: whenever the output space is large, it requires too many input-output examples from the system to provide accurate estimates, which makes it impractical for several real-world problems. As a further consequence, this approach cannot be used when the system's output takes continuous values (e.g., time side channels).

ML for Black-box security estimation The main observation of our work is that Black-box security can be formulated as a Machine Learning (ML) problem. This has several theoretical and empirical consequences for the foundations of

Black-box security. Perhaps the most interesting of them has a practical nature: the leakage of a system can be measured by using a class of asymptotically optimal ML techniques (*universally consistent* rules). This fact alone makes it feasible to measure the security of systems that were too large or complex to evaluate until now. Indeed, differently from the frequentist approach, these techniques can exploit regularities in the output space (e.g., notions of distance) to achieve convergence within a small number of input-output examples.

Security measures In this manuscript, we are interested in estimating security measures that are defined in terms of the *Bayes risk* – the ideal smallest error an adversary commits – and the *random guessing error* – a baseline indicating the optimal error against perfectly secure systems (chapter 4).

In particular, we introduce the *Bayes security* measure (β), as a generalisation of the cryptographic concept of advantage. We prove bounds on β with respect to the probability distribution on the secret inputs of a system (chapter 4), and we use it to measure the security of Webpage Fingerprinting (WF) defences (chapter 9).

Estimating λ-security and (λ, Φ)-security We introduce a distinction between two approaches for estimating a generic security measure λ (such as β): the *direct* approach (Part II), and *through features* (Part III). The former estimates λ "directly" on input-output data, and it gives raise to the notion of λ-security. The latter, which takes inspiration from ML, applies a transformation Φ (the *features*) to the system's output before estimation, and it produces (λ, Φ)-security; this is a weaker notion than λ-security, and we will use it whenever a direct estimate of λ does not converge within the amount of available data.

Applications We expect black-box security to benefit from ML techniques whenever there exist some "regularity" in a system's outputs. This is the case for several real-world applications (e.g., side channels and privacy); on the other hand, when there is no such regularity, we expect them to behave similarly to the *status quo* (frequentist approach).

We evaluate numerous applications of our methods. In Part II, we compare ML techniques with the frequentist approach, both on synthetic and real-world data (e.g., a location privacy dataset, and a time side channel attack to e-Passports). In Part III, we measure (β, Φ)-security of WF defences; to our knowledge, this is also the first method for provably measuring the security of generic WF defences. In section 9.3 we introduce a WF defence, ALPaCA, whose security we conjecture can be measured with the stronger notion β-security.

Extensions We devote the last part of this manuscript to extensions of our framework (Part IV). Notably, we suggest how to extend our results to systems with infinite input spaces, we propose an ML-inspired method for tackling a more general class of adversaries (*many-observations* adversaries), and we conjecture that our results are also applicable to virtually any ML-based attack. Finally, we introduce an adversary based on Conformal Prediction (Vovk et al., 2005), who can output a set of candidate predictions Γ for the secret input, and guarantees on the probability that Γ contains the correct prediction.

On Black-box security The presented approach shifts the frequentist paradigm that reigned in Black-box security until now, by giving it new foundations in the ML theory. The resulting methods perform well empirically, making it possible to evaluate security in a much wider range of real-world applications than before.

Unfortunately, the generality and wide applicability of Black-box security come at a price: when a system has continuous outputs, the guarantees one obtains with Black-box security are asymptotic in the number of examples; similarly, as a consequence of the so called No Free Lunch theorem in learning, in the small sample there exists no classifier that outperforms all the others across all problems (distributions). Nevertheless, we may claim we *know* something in the long run. Hence the sentence at the beginning of this manuscript.

Code In Cherubin et al. (2019) we proposed a tool, F-BLEAU (Fast Black-box Leakage Estimation AUtomated), to estimate the security of a system from examples of its inputs and outputs. The tool is available as Open Source.[1] Whilst we believe the major contributions of this manuscript are theoretical, we wish for it to serve also as a guide for analysts to measure the security of real-world systems. We will therefore include: i) code examples, with the commands one should run to use the estimators we introduce, and ii) links to datasets and resources for replicating our experiments.

[1] https://github.com/gchers/fbleau.

Reading this Manuscript

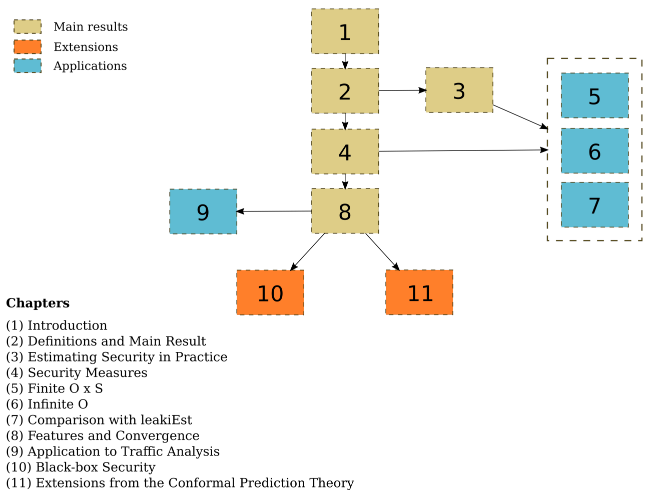
Main results
Extensions
Applications
1
2
3
4
5
6
7
8
9
10
11
Chapters
(1) Introduction
(2) Definitions and Main Result
(3) Estimating Security in Practice
(4) Security Measures
(5) Finite O x S
(6) Infinite O
(7) Comparison with leakiEst
(8) Features and Convergence
(9) Application to Traffic Analysis
(10) Black-box Security
(11) Extensions from the Conformal Prediction Theory

Figure 1.2: Chapters diagram.

The core of the ideas in this manuscript (Figure 1.2) are in chapters: 2, 4, 8, with extensions in chapter 10 and chapter 11. A security analyst wishing to apply our results should also consult chapter 3.

The remaining chapters (5, 6, 7, and 9) are applications of our methods to synthetic and real-world data.

Contributions

We list the main contributions of this manuscript.

- We show that the information leakage (security) of a system can be estimated in a black-box manner by using universally consistent (UC) ML rules (chapter 2). This gives access to a wide set of theoretical results (e.g., section 2.6) and practical estimators to measure security (chapter 3). This appeared in Cherubin (2015, 2017); Cherubin et al. (2019), but it is formalised in a more general form in this manuscript. From ML, we also import the idea of "features", and define a security notion, (λ, Φ)-security, based on it (chapter 8).

- We introduce a new security measure, Bayes security (β), and prove its consistency w.r.t. the choice of a system's prior probabilities (chapter 4).

- We compare extensively our methods with the *status quo* (the frequentist approach), and apply them to synthetic data and to several real-world problems, such as: i) location privacy, ii) side channel to an exponentiation algorithm, iii) time side channel in e-Passports, iv) evaluation of WF defences (Part II and Part III). We also propose a new WF defence, for which we conjecture one can measure the security more accurately (chapter 9).

- We suggest several extensions of the main framework, such as: i) continuous secret spaces, ii) application to many-observations adversaries, iii) extension to Conformal Prediction (CP) adversaries (Part IV). We also suggest our methods are applicable to almost any ML-based adversary (section 10.4).

List of papers

Cherubin (2015) Technical report containing most results of (Cherubin, 2017); see next entry for details. (**technical report**)

Cherubin (2017) Formalises Webpage Fingerprinting (WF) as a Machine Learning (ML) classification task, and shows that we can use a lower-bound of the NN classifier by Cover and Hart (1967), and a UC rule (k_n-NN) to measure the security of WF defences. It is to our knowledge the first generic method to measure security against WF. It is also the source of our main observation: the error of UC learning rules can be used as a security estimator.

Cherubin et al. (2017) Introduces two server-side application-layer WF defences; both offer state-of-the-art security whilst being easy to implement in practice. One of them, ALPaCA, is discussed in this manuscript –because of potentially interesting theoretical guarantees–, and it is currently being deployed in practice.

Cherubin et al. (2019) Extends the intuition of Cherubin (2017) and brings it into the Quantitative Information Flow (QIF) theory. It states formally the connection between ML and Black-box security, thoroughly analyses nearest neighbour rules both on synthetic and real-world examples, and highlights their strengths and weaknesses.

Chatzikokolakis et al. (2018) Analyses β, the security metric introduced in Cherubin (2017), proves some results (e.g., prior-consistency), and compares it with other metrics. (**in preparation**)

Further work

The author also published the following papers during his doctoral studies.

Cherubin et al. (2015) Proposes a clustering algorithm, Conformal Clustering, based on Conformal Predictors (CPs), and applies it to network traffic generated by bots. CPs are wrappers around ML techniques, providing them with validity guarantees, and they will be used in Part IV to extend our framework.

Cherubin and Nouretdinov (2016) Applies CPs as a basis for an alternative to the (standard) List Viterbi algorithm for decoding a Hidden Markov Model sequence. Outperforms the standard algorithm when its strong assumptions on data are violated.

Cherubin (2018) Considers a majority vote strategy for ensembling CPs' predictions, and formulates validity guarantees for ensembles of correlated and uncorrelated CPs.

Cherubin et al. (2018) Uses exchangeability martingales to perform feature selection in anomaly detection problems. In this manuscript, we suggest future applications of exchangeability martingales in the context of Black-box security (Part IV).

2 Definitions and Main Result

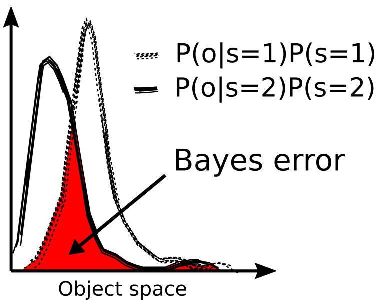

Figure 2.1: The optimal error for the problem we consider is that of the *Bayes adversary*. The Bayes adversary predicts for an object the most likely secret according to $P(o|s)P(s)$; his expected error (the Bayes error) is represented by the red area.

In this chapter, we define a class of attacks where an adversary aims to predict the secret input of a system given its output, and we indicate the "Bayes adversary" as the idealised optimal adversary. On the basis of this adversary's error (the *Bayes error*), we will derive several security measures in chapter 4.

We then focus on the goal of estimating the Bayes error of a system (and, thus, its security) in a black-box manner, and we show that: i) the attack we consider can be framed as an ML classification problem, and ii) the security of a system can be measured by computing the error of asymptotically optimal ML rules. This observation allows us to provide a new foundation for the theory of black-box security estimation on the basis of ML; this results in new theoretical and practical tools to measure the security of a system. In the next parts of this manuscript, we will show these methods: i) make it practical to evaluate a much wider range of systems than before, and ii) they often outperform the *status quo*.

2.1 Threat model

A black-box $\mathcal{B} : \mathcal{S} \mapsto \mathcal{O}$ is a possibly randomised algorithm receiving secret inputs s from a space of secrets $\mathcal{S}$, and returning objects (or, equivalently, *outputs*) accordingly; we call example a pair of object and secret (o, s). The secrets' space $\mathcal{S}$ is finite, although the results presented here are generalisable to continuous secrets (section 10.1); we will consider alternatively two cases for the object space $\mathcal{O}$: discrete and continuous, which will be specified where appropriate. We say a black-box is *sampled* for a secret s, meaning that algorithm $\mathcal{B}$ is run with input s, and returns output $o = \mathcal{B}(s)$. Unless otherwise stated, secret inputs are chosen according to a set of prior probabilities (or *priors*) π , with $\pi(s) = P(s)$.

We call the tuple $(\pi, \mathcal{B})$ a *system*. A system induces on the example space $\mathcal{O} \times \mathcal{S}$ a joint probability distribution $\mu(o, s)$; we say that someone "queries" the system to indicate that they sample an example from $\mu(o, s)$. We assume this distribution does not change over time; i.e., $(\pi, \mathcal{B})$ stays the same across different queries. Observe that this is equivalent to assuming that examples (o_i, s_i) are independently sampled from the same distribution $\mu(o, s)$ (i.e., they are i.i.d.). We will suggest a test to verify this assumption on real data (section 11.4), and some implications of its violation for our purposes (subsection 9.2.3).

The framework we propose captures attacks of the following kind – although, as we will see later, our results generalise to much stronger adversaries. Consider an adversary whose goal is to predict the secret s corresponding to an observation o for a given system $(\pi, \mathcal{B})$. The adversary operates in two phases: *training*, and *attack*. In the training phase, the adversary is given oracle access to the system, and queries it n times for his choice of secrets to obtain the respective outputs: $\{(o_1, s_1), ..., (o_n, s_n)\}$; this constitutes his training data. Then, in the attack phase, a secret s is sampled according to priors π, and the adversary is asked to predict s given the corresponding black-box output $o = \mathcal{B}(s)$.[1]

More precisely, the adversary selects a function $g : \mathcal{O} \mapsto \mathcal{S}$ on the basis of the training data, which predicts a secret given an object; we call g a *classifier*. In the attack phase, he is evaluated with the expected risk over μ for a new object o:

$$R^g := \mathbb{E}(\ell(s, g(o)))\,;$$

here $\ell : \mathcal{S} \times \mathcal{S} \mapsto \mathbb{R}_{\geq 0}$ is a loss function, whose definition is problem-related. Throughout this document, we will limit our discussion to the 0-1 loss function, $\ell := I(s \neq g(o))$, where I is the indicator function taking value 1 when $s \neq$

[1]Observe that, in the scenario we described, the adversary receives only one observation o to predict s; this captures a wide range of attacks. In section 10.2, we will also consider an extension where he can observe several outputs all belonging to the same secret.

$g(o)$, 0 otherwise. For this loss function, that the expected risk coincides with the classifier's expected probability of error:

$$R^g = \mathbb{E}(I(s \neq g(o))) = P(s \neq g(o))\,;$$

for short, we will refer to R^g as the error probability or simply *error*. The 0-1 loss is meaningful for a great majority of real-world attacks, when $\mathcal{S}$ is finite; in chapter 10 we suggest extensions.

Remark As we will see in the next section, our framework and results generalise to a much knowledgeable adversary: an adversary who knows the true underlying distribution μ (or, equivalently, who knows the internals of the system $(\pi, \mathcal{B})$), and aims at predicting s given o. Note that, differently from the one we described in this section, such an adversary does not need any training data.

2.2 Idealised adversaries

We now define two idealised adversaries, whose probabilities of error are the basis for constructing security measures in chapter 4. They are the Bayes and the random guessing adversaries.

2.2.1 Bayes adversary

Suppose one knows the true distribution $\mu(o, s)$; then the Bayes classifier minimises their expected risk:

Definition 2.1 (Bayes classifier)**.** *Let $\mu(s \mid o)$, with $s \in \mathcal{S}$ and $o \in \mathcal{O}$, indicate the conditional* (a posteriori) *probability distribution of secret s given black-box output o. The Bayes classifier $g^* : \mathcal{O} \mapsto \mathcal{S}$ predicts, for a new object o:*

$$g^*(o) := \operatorname*{argmax}_{s \in \mathcal{S}} \mu(s \mid o) = \operatorname*{argmax}_{s \in \mathcal{S}} \mu(o, s)\,.$$

The Bayes classifier minimises the error among all classifiers:

Theorem 2.2 (Optimality of the Bayes classifier (e.g., Devroye et al. (2013))). *For any classifier $g : \mathcal{O} \mapsto \mathcal{S}$:*

$$P(g^*(o) \neq s) \leq P(g(o) \neq s)\,.[2]$$

We call *Bayes adversary*[3] the optimal idealised adversary who knows the true distribution $\mu(o, s)$ induced by a system, and uses the Bayes classifier to make predictions. We call the error of this adversary, R^*, the *Bayes error* (or *Bayes risk*).

R^* is obtained analytically from the a posteriori conditional distribution $\mu(s \mid o)$ as follows. Upon observing a new object o, g^* makes a conditional error:

$$r^*(o) := 1 - \max_{s \in \mathcal{S}} \mu(s \mid o)\,.$$

By taking the expectation of $r^*(o)$ over the distribution on $\mathcal{O}$ we obtain:

$$R^* := \mathbb{E}(r^*) = 1 - \sum_{o \in \mathcal{O}} \max_{s \in \mathcal{S}} \mu(o, s)\,,$$

supposing $\mathcal{O}$ is finite; when it is infinite, we appropriately replace the summation with an integral.

Remark 1 Under our threat model (section 2.1), the Bayes risk is the smallest error achievable by any adversary, even with unbounded computational resources.

Remark 2 Because the real distribution on $\mathcal{O} \times \mathcal{S}$ is usually not known for real-world systems, it is also not possible to determine R^* analytically. However, methods exist for estimating the Bayes risk; this will be the focus of the next chapters.

2.2.2 Random guessing

It is useful to define a "baseline" to consider when measuring the Bayes risk for a system; to this end, we construct the *random guessing* adversary as follows.

[2]In our definitions we implicitly used the notion of 0-1 loss. However, the Bayes classifier and its optimality result can be generalised to other loss functions.

[3]Whilst "Bayesian adversary" would be grammatically a more appropriate choice, such name would recall "Bayesian statistics", which uses very different assumptions and techniques from the ones we consider. Because Bayes classifier and Bayes risk commonly indicate respectively the classifier used by the optimal adversary in our context and his error, we choose this name.

The random guessing adversary knows the secrets' priors π of a system $(\pi, \mathcal{B})$, but he is not given access to its black-box $\mathcal{B}$.[4] Under these circumstances, the best this adversary can do is to always output the secret with the largest prior. This is what we refer to as random guessing (implicit: "according to priors").

Definition 2.3 (Random guessing error). *Consider a system $(\pi, \mathcal{B})$ with priors π, $\pi(s) = P(s)$ for $s \in \mathcal{S}$. The random guessing error R^π is the error committed by an adversary who always outputs the most likely secret,*

$$R^\pi := 1 - \max_{s \in \mathcal{S}} \pi(s)\,.$$

2.3 Problem definition

The main focus of this manuscript is to estimate the security of a system in a black-box manner. We will measure security according to security measures λ that can be expressed as a function of the Bayes risk and random guessing error (chapter 4); therefore, to estimate the security of a system will be for us equivalent to estimating its associated Bayes risk. We state this problem as follows.

Problem 2.4 (*Direct* black-box security estimation). *Given a dataset of examples, $\{(o_1, s_1), ..., (o_n, s_n)\}$, obtained by sampling n times a system $(\pi, \mathcal{B})$, we wish to compute an estimate $\hat{R}^*$ of the true Bayes risk R^* of the system.*

We will then compute the value of a desired security measure λ on the basis of $\hat{R}^*$.

Moreover, it is desirable:

- that the estimates require a small number n of examples to converge to R^*;
- to have several options on the guarantees that the estimates provide (e.g., convergence from above or below);
- to also have conservative estimates (e.g., proper lower bounds) of R^*.

2.4 Learning rules

Informally, an ML rule, (or, simply, *learning rule*) is an algorithm that selects a classifier, from a family of classifiers, according to *training* data $Z_{train} = \{(o_1, s_1), ..., (o_n, s_n)\}$, with the goal of minimising the expected error on a new object.

[4]Observe this is equivalent to considering a system $(\pi, \mathcal{B})$ where the output of $\mathcal{B}$ leaks nothing about its secret input.

For our purposes, a learning rule is a sequence of functions $\{g_n, n \geq 1\}$; with a slight terminology overload, we call $g_n : \mathcal{O} \times \{\mathcal{O} \times \mathcal{S}\}^n \mapsto \mathcal{S}$ a classifier, and will write $g_n(o; Z_{train})$ or simply $g_n(o)$ to indicate that we use the classifier selected by the rule according to Z_{train} to predict the secret $\hat{s}$ for an object o.

We evaluate the performances of a rule with its expected risk:

$$R^{g_n} := \mathbb{E}(\ell(s, g_n(o))) = P(s \neq g_n(o)) \,.$$

In the next section, we will show that a particular class of *learning rules* can be used to estimate the Bayes risk, and therefore a system's security.

2.5 UC learning rules are leakage estimators

We can finally state a central observation for this manuscript. To estimate the security of a system is equivalent to estimating the error of a rule from a particular class of rules: *universally consistent* (UC). This observation allows to bring a vast amount of results from the ML theory into the problem we formulated; notably, this is a paradigm shift from the standard statistical approaches (subsection 3.1.1) that have been used so far to measure the leakage of black-boxes.

Consider a distribution $\mu(o, s)$, and a learning rule g_n selecting a classifier using n training examples sampled from μ. Intuitively, we would like its expected error for a new example (o, s) to approximate better and better the Bayes risk of the corresponding system as the training set increases in size. The following definitions capture this intuition.

We first define consistency with respect to a particular distribution:

Definition 2.5 (Consistent Learning Rule). *Let μ be a distribution on $\mathcal{O} \times \mathcal{S}$ and let g_n be a learning rule, which is trained from n examples sampled from μ. The rule g_n is consistent if its expected error satisfies $R^{g_n} \to R^*$ as $n \to \infty$.*[5]

The next definition generalises to all distributions:

Definition 2.6 (Universally consistent Learning Rule). *A learning rule is universally consistent if it is consistent for any distribution μ on $\mathcal{O} \times \mathcal{S}$.*

[5]We will not make explicit the mode of convergence specified by the notation $R^{g_n} \to R^*$ (e.g., weak or strong), because, for most well-behaved rules, consistency and strong consistency are equivalent (Devroye et al., 2013).

By this definition, the error of a classifier selected according to a UC rule is an estimate of the Bayes risk. In the next chapter we will describe several UC rules, and explain into details how to estimate R^* in practice from a dataset.

2.6 Impossibility results

UC rules give asymptotic guarantees, and one may wonder whether it is possible to have optimality guarantees in the finite sample. Unfortunately, the following two theorems show that: i) if $\mathcal{O}$ is continuous, then no learning rule can assure to converge with a certain rate, and ii) in general, no rule is optimal for all learning problems (i.e., distributions μ).

Theorem 2.7 (No convergence rate (Antos et al., 1999))**.** *If $\mathcal{O}$ is infinite, and under no further assumption on the distribution, for any UC rule there exists a distribution $\mu(o,s)$ for which the error of such rule converges to R^* arbitrarily slowly.*

This theorem states that, for any learning algorithm, one can find a distribution for which the algorithm performs arbitrarily bad. One may then wonder if there is a learning algorithm that converges faster than the others on average, among the possible choices of distributions μ on $\mathcal{O} \times \mathcal{S}$.

The No Free Lunch (NFL) theorem gives a negative answer. We provide here a simplified version.

Theorem 2.8 (No Free Lunch (Wolpert, 2002))**.** *Consider two learning rules, f_n and g_n, and a training set of examples $\{(o_1, s_1), ..., (o_n, s_n)\}$ sampled from a joint distribution μ on $\mathcal{O} \times \mathcal{S}$. Then, there is always a distribution μ such that $\mathbb{E}_\mu R^{f_n} < \mathbb{E}_\mu R^{g_n}$, and vice versa, there is always a distribution μ' such that $\mathbb{E}_{\mu'} R^{f_n} < \mathbb{E}_{\mu'} R^{g_n}$.*

Important Remark

These theorems imply that, in practical applications of Black-box security, one should **always** evaluate many estimators (learning rules), and then consider the one that converged faster (e.g., take the smallest error estimate).

3 Estimating Security in Practice

This chapter describes UC rules and how to use them to estimate R^*, in practice, from a dataset $Z = \{(o_1, s_1), ..., (o_n, s_n)\}$ of examples sampled from a system.

We first provide examples of UC rules, that we will evaluate in the next parts of this manuscript. Then, we focus on various options to estimate the error of a UC rule using a dataset of examples.

3.1 UC rules

Method	Guarantee	Space $\mathcal{O}$	Assumptions
Frequentist	$\to R^*$	finite	
NN	$\to R^*$	finite	
k_n-NN	$\to R^*$	infinite	
$SVM - rbf$	$\to R^*$	infinite	$\mathcal{O}$ compact subset of $\mathbb{R}^m$
NN Bound	$\leq R^*$	infinite	$(d, \mathcal{O})$ separable

Table 3.1: Estimates' guarantees as $n \to \infty$. Guarantees on infinite spaces also hold for finite space (potentially under weaker assumptions). Separability is sometimes required on a metric space $(d, \mathcal{O})$, where d is a distance metric.

We describe various UC rules and a lower bound estimate of R^*. Table 3.1 summarises them, together with their assumptions and the guarantees they provide.

3.1.1 Frequentist approach

The frequentist approach (also *look-up table*) was formalised by Chatzikokolakis et al. (2010) in the context of QIF, and it is the *de facto* standard for black-box security estimation. It also appeared in different forms in other security domains; for example, it was used by Cai et al. (2014c); Wang et al. (2014) to measure the security of Webpage Fingerprinting defences (chapter 9).

This approach is only applicable when $\mathcal{O}$ is finite. It consists in estimating the underlying joint and prior distributions by counting the frequencies of secrets and objects in the training data $Z = \{(o_1, s_1), ..., (o_n, s_n)\}$, $n \geq 1$:

$$\mu(o,s) \approx \hat{\mu}(o,s) := \frac{|\{i = 1, ..., n \mid o_i = o, s_i = s\}|}{n}$$
$$\pi \approx \hat{\pi}(s) := \frac{|\{i = 1, ..., n \mid s_i = s\}|}{n} .$$

The frequentist classifier works as follows:

$$Freq(o) = \begin{cases} \operatorname{argmax}_s \hat{\mu}(o,s) & \text{if } (o, \cdot) \in Z \\ \operatorname{argmax}_s \hat{\pi}(s) & \text{otherwise} . \end{cases}$$

This classifier bases its classification for o on: i) the estimated joint distribution, if o was observed in the training data, ii) the estimated prior distribution otherwise.

Consider a finite example space $\mathcal{O} \times \mathcal{S}$. Provided with enough examples, the frequentist approach always converges: clearly, $\hat{\mu} \to \mu$ as $n \to \infty$, because events' frequencies converge to their probabilities by the Law of Large Numbers.

However, this approach has a fundamental issue. Given a training set, the frequentist classifier can tell something meaningful (i.e., better than random guessing) for an object o, only as long as o appeared in the training set; but, for very large systems (e.g., large object space), the probability of observing an example for each object within the training set becomes small, and the frequentist classifier approaches random guessing. We study this matter further in subsection 5.2.4 and in Appendix 1. An additional related issue of the frequentist approach is that, whenever the objects come from nosy measurements, it has a tendency to strongly underestimate the Bayes risk; we discuss this in subsection 9.1.4.

Code

Run[a]:

```
$ fbleau frequentist <training file> <validation file>
```

[a]This, and the following code snippets in this chapter, show how to estimate the Bayes risk (and, therefore, the security) of a system using F-BLEAU (Cherubin et al., 2019). They require sampling examples (o_i, s_i) from the system, and splitting them into two `.csv` files, respectively containing training and validation datasets. The chosen estimator (e.g., frequentist) will be trained on the training set, and its error estimated on the validation set (see the validation estimate in section 3.2 for details). More information on the file format on the GitHub page: `https://github.com/gchers/fbleau`.

3.1.2 NN

Nearest Neighbour (NN) is one of the simplest classifiers: given a training set and a new object o, it predicts the secret of its closest training observation (*nearest neighbour*), with respect to some distance metric.[1] It can be defined for an infinite object space, where although it does not guarantee UC.

We introduce a novel formulation of NN, which can be seen as an extension of the frequentist approach, that takes into account *ties* (i.e., neighbours that are equally close to the object to predict o); the corresponding rule guarantees UC when $\mathcal{O}$ is finite (Cherubin et al., 2019).

Consider a training set $\{(o_1, s_1), ..., (o_n, s_n)\}$, an object o, and a distance metric $d : \mathcal{O} \times \mathcal{O} \mapsto \mathbb{R}_{\geq 0}$. The NN classifier predicts a secret for o by taking a majority vote over the set of secrets whose objects have the smallest distance to o. Formally, let $I_{min}(o) = \{i \mid d(o, o_i) = \min_{j=1...n} d(o, o_j)\}$ and define:

$$NN(o) = s_{h(o)} \quad \text{where } h(o) = \operatorname*{argmax}_{i \in I_{min}(o)} |\{j \in I_{min}(o) \mid s_j = s_i\}| \,.$$

We show that NN is universally consistent for finite $\mathcal{O} \times \mathcal{S}$:

Theorem 3.1 (Universal consistency of NN)**.** *Consider a distribution on $\mathcal{O} \times \mathcal{S}$, where $\mathcal{O}$ and $\mathcal{S}$ are finite. Let R_n^{NN} be the expected error of the NN classifier for a new observation o. As the number of training examples $n \to \infty$:*

$$R_n^{NN} \to R^*.$$

Sketch Proof. For an observation o that appears in the training set, the NN classifier is equivalent to the frequentist approach (i.e., they have identical output for the same input). For a finite space $\mathcal{O} \times \mathcal{S}$, as $n \to \infty$, the probability that the training set contains all $o \in \mathcal{O}$ approaches 1. Therefore, the NN rule is asymptotically equivalent to the frequentist approach, and its error also converges to R^*. □

Code

Run:

```
$ fbleau --knn=1 <training file> <validation file>
```

[1]The guarantees we formulate in this chapter for various nearest neighbour rules are independent of the distance metric.

3.1.3 k_n-NN

NN is not UC for infinite $\mathcal{O}$. However, we can achieve universal consistency in this case with the k-NN classifier, an extension of NN, for appropriate choices of its parameter k.

The k-NN classifier takes a majority vote among the secrets of its k neighbours Breaking ties in the k-NN definition requires more care than with NN. In literature, this is generally done via strategies that add randomness or arbitrariness to the choice (e.g., if two neighbours have the same distance, select the one with the smallest index in the training data) (Devroye et al., 2013); however, while these strategies make for an easier theoretical treatment, they do not necessarily perform well when there are many ties (e.g., finite $\mathcal{O}$). We propose a novel tie breaking strategy, which takes into account ties whilst giving more importance to the closest neighbours. In early experiments, we observed this strategy had a faster convergence than standard approaches (Cherubin et al., 2019).

Consider a training set $\{(o_1, s_1), ..., (o_n, s_n)\}$, a new object o, and some metric $d : \mathcal{O} \times \mathcal{O} \mapsto \mathbb{R}_{\geq 0}$. Let $o_{(i)}$ denote the i-th closest object to o, and $s_{(i)}$ its respective secret. If ties do not occur after the k-th neighbour (i.e., if $d(o, o_{(k)}) < d(o, o_{(k+1)})$), then k-NN outputs the most frequent among the secrets of the first k neighbours:

$$k\text{-}NN(o) = \underset{s \in \mathcal{S}}{\operatorname{argmax}} \left| \{ i = 1, ..., k \mid s_{(i)} = s \} \right| .$$

If ties exist after the k-th neighbour, that is, for some $k' \leq k < k''$:

$$d(o, o_{(k'-1)}) < d(o, o_{(k')}) = ... = d(o, o_{(k)}) = ... = d(o, o_{(k'')}) < d(o, o_{(k''+1)}) ,$$

we proceed as follows. Let $\hat{s}$ be the most frequent secret among $\left\{ s_{(k')}, ..., s_{(k'')} \right\}$; k-NN predicts the most frequent secret in the following multiset, truncated at the tail to have size k:

$$s_{(1)}, s_{(2)}, ..., s_{(k'-1)}, \hat{s}, \hat{s} ..., \hat{s} .$$

We now define k_n-NN, a UC learning rule that selects a k-NN classifier for a training set of n examples by choosing k as a function of n.

Definition 3.2 (k_n-NN rule)**.** *Given a training set of n examples, the k_n-NN rule selects a k-NN classifier, where k is chosen such that $k_n \to \infty$ and $k_n / n \to 0$ as $n \to \infty$.*

Stone proved that the k_n-NN rule is UC:

Theorem 3.3 (k_n-NN is UC (Stone, 1977)). *For all distributions the expected error of the k_n-NN rule converges to R^* as $n \to \infty$.*

This holds for any distance metric. In our experiments, we will use the Euclidean distance, and we will evaluate two k_n-NN rules, $k_n = \log(n)$ (where log is the natural log) and $k_n = \log_{10} n$.

In the case of finite output spaces, ties may occur. Theorem 3.3 holds for a k-NN formulation that splits ties in a simplistic way: if two objects o_i and o_j are equidistant to o, it declares o_i closer iff $i < j$. We shall now prove that our tie-splitting method is also UC for finite output spaces, by showing that our formulation of k-NN converges to the frequentist approach, which is UC in finite spaces.

We first state a lemma. Consider i.i.d. objects $\{o_1, ..., o_n\}$, and an object o from the same distribution. Reminder: the support of a distribution μ, $\text{supp}(\mu)$, is the set of o s.t. $\mu(B_\delta(o)) > 0$ for all $\delta > 0$, where $B_\delta(o)$ is the δ-ball centered in o.

Lemma 3.4 (Devroye et al. (2013)). *If $o \in \text{supp}(\mu)$ and $\lim_{n\to\infty} k_n/n = 0$, then $d(o, o_{(k)}) \to 0$ with probability one.*

We can now prove that our tie-splitting method is UC for finite $\mathcal{O}$.

Proof. Let us indicate with *maj A* the function returning the most frequent element of the multiset A, (e.g., $maj\{1,2,2,3\} = 2$). By the tie-splitting procedure described above, the k-NN prediction when ties occur is the majority vote:

$$k\text{-}NN(o) = maj\{s_{(1)}, s_{(2)}, ..., s_{(k'-1)}, \hat{s}, \hat{s}..., \hat{s}\}\,, \tag{3.1}$$

where the set is truncated at the tail to have size k; in the expression, $\hat{s}$ is determined as: $\hat{s} = maj\{s_{(k')}, ..., s_{(k'')}\}$, where $o_{(k')}$ and $o_{(k'')}$ are respectively the first and the last neighbours for which $d(o, o_{(k')}) = d(o, o_{(k)}) = d(o, o_{(k'')})$.

If we let $k/n \to 0$, then $d(o, o_{(k)}) \to 0$ by Lemma 3.4. Furthermore, $k' = 1$, and the k-NN prediction (Equation 3.1) reduces to $k\text{-}NN(o) = maj\{\hat{s}, \hat{s}..., \hat{s}\} = \hat{s}$. Finally:

$$\begin{aligned}
\hat{s} &= maj\{s_{(k')}, ..., s_{(k'')}\} \\
&= maj\{s_{(1)}, ..., s_{(k'')}\} \\
&= maj\{s_i \mid d(o, o_i) = d(o, o_{(k)}) = 0, i = 1, ..., n\} \\
&= maj\{s_i \mid o_i = o, i = 1, ..., n\} \\
&= \operatorname*{argmax}_s |\{i = 1, ..., n \mid o_i = o, s_i = s\}| = Freq(o)\,.
\end{aligned}$$

□

Code

F-BLEAU currently implements k_n-NN for two choices of k: $k_n = \log(n)$ and $k_n = \log_{10}(n)$. Run:

```
$ fbleau log <training file> <validation file>
or
$ fbleau log10 <training file> <validation file>
```

3.1.4 SVM

Support Vector Machine (SVM) classifiers work by (possibly) mapping objects into a Hilbert space (the so-called *feature space*), and then determining a hyperplane that separates them under some constraints.

Consider a binary classification problem, $|\mathcal{S}| = 2$. Select a function $\mathsf{k} : \mathcal{O} \times \mathcal{O} \mapsto \mathcal{R}$, such that there exists a Hilbert space H (feature space) and a mapping $\Phi : \mathcal{O} \mapsto H$ (*feature mapping*) with:

$$\mathsf{k}(x,y) = \langle \Phi(x), \Phi(y) \rangle \quad \forall x,y \in \mathcal{O}\,;$$

k is called a *kernel*; select a parameter $c > 0$. SVM classification is obtained by:

$$SVM(o) = \operatorname{sign}(\langle w, \Phi(\cdot) \rangle + b)$$

where the weights w and b are obtained as a solution to the following quadratic optimisation problem:

$$\begin{array}{lll} \text{minimise} & \langle w,w \rangle + c\sum_{i=1}^{n} \xi_i & \text{for } w,b,\xi \\ \text{subject to} & s_i(\langle w, \Phi(o_i) \rangle + b) \geq 1 - \xi_i, & i = 1,...,n\,, \\ & \xi_i \geq 0 & i = 1,...,n\,. \end{array}$$

Steinwart (2002) proved that SVM is UC for certain kernels k when c is chosen according to a set of functions $c = c_n$ in the number of training examples n. In particular, Steinwart showed that SVM is UC for a popular kernel, Gaussian RBF, $\mathsf{k}(x,y) := \exp(-\gamma||x-y||^2)$, for fixed $\gamma > 0$:

Theorem 3.5 (SVM-RBF universal consistency (Steinwart, 2002)). *Let $|\mathcal{S}| = 2$, consider a compact $\mathcal{O} \subset \mathbb{R}^q$, and a Gaussian RBF kernel* k. *For all $n \geq 1$ select $c_n = n^{\beta-1}$ for some $0 < \beta < \frac{1}{q}$. Then SVM with kernel k and sequence $\{c_n\}$ is UC.*

Unfortunately, achieving UC in a multiclass setting ($|\mathcal{S}| > 2$) with SVM is harder, and naïve extensions from binary SVM to multiclass (e.g., one-vs-rest (Friedman et al., 2001)) yield to inconsistent rules. Tewari and Bartlett (2007) proposed principles to determine UC of multiclass SVM methods based on the equivalence between classification calibration and UC; they showed that the method by Lee et al. (2004) is UC, and claimed that both the approaches by Weston and Watkins (1998) and Crammer and Singer (2001) were not UC. The latter claim was refused by Glasmachers (2010), because of a logic fallacy in the inconsistency argument. Glasmachers (2010) also claimed that the method by Crammer and Singer (2001) was UC, but the proof was found wrong, with an *Erratum* note on the paper itself; the note also claims the method by Crammer and Singer (2001) is *not* UC.

Code

SVM is not implemented in F-BLEAU. However, one can use `scikit-learn`'s implementation[a][b]. Select parameter `C`$=c_n = n^{\beta-1}$, as a function of the number of training examples n and a chosen β. Then, for training data `Z`, and test objects `Otest`$=\{o_1, o_2, ...\}$ and secrets `Stest` $= \{s_1, s_2, ...\}$:

```python
from sklearn.svm import SVC
svm = SVC(C=C, kernel='rbf')
svm.train(Ztrain)
estimate = 1 - svm.score(Otest, Stest)
```

[a] http://scikit-learn.org.

[b] Note that one should only use this for binary classification, as the multiclass methods implemented in the library (one-vs-rest and Crammer and Singer (2001)) are not UC.

3.1.5 Neural Networks

Literature abounds with further examples of UC rules.

A notable class is feed-forward Neural Networks (NNet). An NNet with one hidden layer and step activation function σ is UC when the layer's number of neurons k is chosen as a function of the size of training data n such that $\mathsf{k} \to \infty$ and $\frac{\mathsf{k}\log(n)}{n} \to 0$ as $n \to \infty$ (Faragó and Lugosi, 1993).

Other examples include histogram rules (e.g., Devroye et al. (2013)). Of course, this chapter does not aim to be a comprehensive overview of all UC method, but rather to illustrate a set of techniques that one can use to measure security.

3.1.6 NN bound by Cover and Hart (1967)

A very interesting result for our goals is a lower bound on R^* due to Cover and Hart (1967) coming from the error of the NN classifier. This bound was first used to estimate the leakage of a black-box system by Cherubin (2017).

Theorem 3.6 (NN Bound). *Let R_n^{NN} be the expected error of the NN classifier given n training examples. As $n \to \infty$, the following inequality holds:*[2]

$$\frac{|\mathcal{S}|-1}{|\mathcal{S}|}\left(1-\sqrt{1-\frac{|\mathcal{S}|}{|\mathcal{S}|-1}R_n^{NN}}\right) \leq R^* .$$

We derive this from the inequality by Cover and Hart (1967) in Appendix 2. This lower bound is as tight as possible, in the sense that there exist distributions for which equality is reached. However, in practice we observed it tends to be largely conservative (e.g., section 9.2). Nevertheless, this characteristic becomes very useful in Information Security contexts, where an analyst may need to have a stronger confidence of security (e.g., an overestimate of the adversary's error).

Code

Run:

```
$ fbleau nn-bound <training file> <validation file>
```

3.1.7 A remark on convergence rates

Because no rate-of-convergence results can be proved under the sole i.i.d. assumption (section 2.6), it is possible that for certain distributions a non-UC rule outperforms UC rules for a finite sample size. This is fairly trivial to show: we could craft a particularly malicious distribution of data such that a non-UC rule converges quickly, but for which some UC rules (e.g., k_n-NN, SVM) perform poorly.

[2]An implementation note: $\frac{L}{L-1}\hat{R}^{NN}$ is usually smaller or equal to 1. However, because of noise in experiments, this quantity may take larger values than 1, making the value under square root negative. In practice, we use $1-\min\left(\frac{L}{L-1}\hat{R}^{NN},1\right)$ under square root to avoid this.

This fact does not change our recommendation: one should always try many estimator, and use the one converging faster. Nonetheless, if an analyst: i) suspects none of the UC methods will converge, or ii) needs to make a very conservative choice for security requirements, then they should use conservative estimates (e.g., NN lower bound).

3.2 Error estimates

The consistency of a UC rule g_n is formulated on its expected error when given a training set of n examples:

$$R_n = \mathbb{E}(\ell(s, g_n(o))) = \int_{\mathcal{O}\times\mathcal{S}} \ell(s, g_n(o))\mu(o,s)\,\mathrm{d}o\,\mathrm{d}s\,, \tag{3.2}$$

where ℓ is the 0-1 loss in this manuscript, unless otherwise specified.

Using this expression to determine the *true* R_n requires knowing the distribution μ. While we will be able to compute Equation 3.2 in synthetic experiments (section 5.2), in practical applications we need to estimate R_n from a dataset.

This section describes estimates of R_n that can be computed from a dataset Z. Note that, whilst this topic has a wide literature, we will focus on the methods with useful properties for our purposes.

3.2.1 Validation estimate

Ideally, one should estimate a rule's error on data generated independently from the training set (i.e., in a separate experiment). However, this is often not possible. A common alternative is to split the collected data into two, training Z_{train} and validation set Z_{val}; then the *validation error* of a learning rule g_n trained on Z_{train} is given by its average error on the validation set:

$$\hat{R}_n := \frac{1}{n} \sum_{(o_i,s_i)\in Z_{val}} I(g_n(o_i) \neq s_i)\,.$$

This method has low computational requirements, as opposed to other methods we will describe. However, its estimate is usually biased with respect to the particular (Z_{train}, Z_{val}) split (e.g., Devroye et al. (2013)). A simple extension of this is to estimate the error multiple times for different training-validation set splits, and then to average their estimates.

Code

Note All the code snippets illustrated so far used the validation estimate.

3.2.2 k-fold Cross Validation estimate

A popular error estimation method is k-fold Cross Validation (CV). Given a dataset Z, and for a chosen integer $k > 1$, we randomly split Z into k parts (*folds*). Then, for each fold, we train a classifier on the remaining $k - 1$ folds and estimate the error on the current fold. The final estimate is obtained by averaging the estimates.

A typical choice for k is 5 or 10, which was empirically shown to give a good bias/variance trade-off.

A special case of this estimate is Leave One Out CV (LOOCV), obtained for $k = n$, which although has a high variance.

3.2.3 Resubstitution estimate

The resubstitution estimate (or *training set error* or *apparent error*) is obtained by both training a rule and estimating its error on the same dataset Z. This estimate is optimistically biased (i.e., it converges to the real R_n from below). Interestingly, it was used to obtain a lower bound of the Bayes risk with the k_n-NN rule (e.g., Fukunaga and Hummels (1987)); note that this lower bound estimate, differently from the NN bound, guarantees asymptotic convergence to R^*.

4 Security Measures

Until a decade ago, the most popular measure of leakage in information flow theories was Shannon mutual information (MI). However, in his seminal paper, Smith (2009) showed that MI is not appropriate to represent a realistic attacker; he therefore suggested to use notions based on the Bayes risk.[1]

In this chapter, we describe two security measures that can be derived from the Bayes risk, ME and Multiplicative Leakage, and we introduce a new one, the Bayes Security measure (β). The need for many security measures is that there is likely no "appropriate" measure for all security applications (Alvim et al., 2012); furthermore, different measures provide different guarantees.

Remark 1 Note that in this text we use interchangeably the terms *leakage* and *security* measure, although technically they have a subtle difference: a security measure $\lambda \in \Lambda \subseteq \mathbb{R}_{\geq 0}$ is a generic numeric indicator of a system's security; a leakage measure is a security measure that takes higher values to indicate higher vulnerability to an attack (i.e., to indicate the system leaks more information). We will specify whether a security measure is a leakage measure whenever necessary.

Remark 2 Although there is a relation between MI and Bayes risk (Santhi and Vardy, 2006), the corresponding threat models are very different: MI corresponds to an attacker who can try infinitely many times to guess the secret; the Bayes risk represents an adversary who has only one try at his disposal (Smith, 2009). Consequently, measures that are based on the Bayes risk (e.g., Renyi's min-entropy (ME)) and MI can give very different results. For instance, Smith (2009) shows two systems that have almost the same MI, but one has an ME several orders of magnitude larger than the other one; conversely, there are examples of two systems such that ME is 0 for both, while the MI is 0 in one case and strictly positive (several bits) in the other one.

[1]Precisely, Smith derives notions on the basis of the *vulnerability*, the complement of the Bayes risk.

4.1 Multiplicative Leakage

Multiplicative Leakage (Braun et al., 2009) is defined as:

$$\mathrm{M} = \frac{1 - R^*}{1 - R^\pi}\,.$$

This leakage measure offers a fundamental guarantee: if computed for uniform priors (i.e., $R^\pi = 1 - 1/|\mathcal{S}|$), it upper-bounds the value of the Multiplicative Leakage computed for any other set of priors (Braun et al., 2009).

More formally, it satisfies prior-consistency, which we define as follows:

Definition 4.1 (Prior-consistency)**.** *Consider a black-box $\mathcal{B}$, and let $\lambda(\pi, \mathcal{B})$ be a security measure computed for the system $(\pi, \mathcal{B})$, for some choice of priors π. Also, let v be the uniform priors, $v(s) := P(s) = {}^1/_{|S|}, \forall s \in \mathcal{S}$. We say that λ is prior-consistent if:*

$$\begin{aligned} \lambda(v, \mathcal{B}) &= \min_\pi \lambda(\pi, \mathcal{B}) \quad \textit{if smaller } \lambda \textit{ means "more secure"} \\ &\textit{or} \\ \lambda(v, \mathcal{B}) &= \max_\pi \lambda(\pi, \mathcal{B}) \quad \textit{if larger } \lambda \textit{ means "more secure"}\,. \end{aligned}$$

In other words, it is prior-consistent if, when computed for uniform priors, it lower-bounds (or upper-bounds, depending on the formulation of λ) the security measure.

Remark A prior-consistent security measure allows a security analyst to neglect prior probabilities when measuring the leakage of a system while having guarantees for all priors. Of course, the analyst may also compute the measure with the true priors whenever they are known, which would result in tighter bounds.

4.2 Min-entropy

Min-entropy leakage is defined as follows:

$$\mathrm{ME} := -\log_2(1 - R^\pi) + \log_2(1 - R^*)\,.$$

We will use it to compare our methods with leakiEst, a well-known black-box leakage estimation tool, in chapter 7.

4.3 Bayes security measure (β)

A more recent measure, which we refer to as Bayes Security measure (β), has the characteristic of being bounded between $[0, 1]$, where 1 means perfect security (i.e., zero leakage) (Cherubin, 2017).

It derives from the well-known concept of advantage (Adv) in Cryptography, which informally measures how better than random guessing an adversary can do. In Cryptography, advantage is defined for a two classes scenario ($|\mathcal{S}| = 2$), and assuming uniform priors over the secrets.

In Cherubin (2017), we generalised this notion to $|\mathcal{S}| \geq 2$ and (possibly) non-uniform priors. By considering $1 - \mathsf{Adv}$, the resulting security measure is:

$$\beta := \frac{R^*}{R^\pi} .$$

We now show that β is prior-consistent for $|\mathcal{S}| = 2$. Note that, even if the case $|\mathcal{S}| = 2$ corresponds to the measure $1 - \mathsf{Adv}$, which was widely studied in Cryptography, its prior-consistency was unknown to the best of our knowledge.

Theorem 4.2 (β prior-consistency for $|\mathcal{S}| = 2$ (Chatzikokolakis et al., 2018))**.** *Consider a system $(\pi, \mathcal{B})$ on $\mathcal{O} \times \mathcal{S}$, with $|\mathcal{S}| = 2$, and let $R^*(\pi)$ and $R^\pi(\pi)$ be respectively the Bayes risk and random guessing error computed for a choice of priors π, and let $\beta(\pi) = {}^{R^*(\pi)}/{}_{R^\pi(\pi)}$. Let v be the uniform priors. Then:*

$$\beta(v) \leq \beta(\pi) \quad \forall \pi .$$

Proof. Observe that $R^*(\pi)$ is concave. Let $\mathcal{S} = \{s_1, s_2\}$ and, without loss of generality, $p = \max_{s \in \mathcal{S}} \pi(s) = s_1$, and let $c = 2(1-p)$. Notice that $c \in [0, 1]$, hence c and $1-c$ are convex coefficients, and note that $\pi = cv + (1-c)\pi^*$, where π^* is the point distribution on s_1, i.e., $\pi^*(s_1) = 1$ and $\pi^*(s_2) = 0$. Then:

$$\begin{aligned} R^*(\pi) &= R^*(cv + (1-c)\pi^*) \\ &\geq cR^*(v) + (1-c)R^*(\pi^*) \\ &= cR^*(v) \\ &= 2(1-p)R(v) ; \end{aligned}$$

the second step comes from the concavity of R^*, the third one because $R^*(\pi^*) = 0$. Therefore:

$$\frac{R^*(\pi)}{R^\pi(\pi)} = \frac{R^*(pi)}{(1-p)} \geq \frac{R^*(v)}{(1/2)} = \frac{R^*(v)}{R^\pi(v)} .$$

□

Unfortunately, this result does not generalise to $|\mathcal{S}| > 2$. We show this by providing a counterexample only for the case $|\mathcal{S}| = 3$, and we leave a more extensive treatment to future research. Consider the following conditional distribution $\mu(o \mid s)$ on the example space $\mathcal{O} \times \mathcal{S} = \{1, 2, 3\} \times \{1, 2, 3\}$:

$$\begin{aligned}\mu(o \mid s = 1) &= (0.2, 0.3, 0.5)\\ \mu(o \mid s = 2) &= (0.5, 0.3, 0.2)\\ \mu(o \mid s = 2) &= (0.3, 0.2, 0.5) .\end{aligned}$$

Then consider two choices of priors: $\pi = (0.1, 0.45, 0.45)$, and uniform priors v. Then, by simple calculations: $R^*(\pi) = 0.415$ and $R^*(v) = 0.567$, and therefore $\beta(v) = 0.840 > \beta(\pi) = 0.755$, thereby disproving prior-consistency for $|\mathcal{S}| = 3$.

We claim this generalises to larger secret spaces (i.e., β is not prior-consistent for $|\mathcal{S}| > 2$), but we do not include a proof, as of minor interest for this manuscript.

What's next

One can estimate the security of a system from black-box examples either by working directly on the objects $o \in \mathcal{O}$ or on a transformation $\Phi(o)$ of them (*feature* mapping). In Part II we treat the former, in Part III the latter.

Part II

Direct Estimation

5 Finite $\mathcal{O} \times \mathcal{S}$

The case of a finite example space $\mathcal{O} \times \mathcal{S}$ is the standard setting of Quantitative Information Flow (QIF) research (Smith, 2009).

Whilst originally QIF assumed an analyst knows the true distribution $\mu(o, s)$ induced by a system $(\pi, \mathcal{B})$, and they can therefore compute R^* analytically, a case was made for black-box estimation (e.g., Chatzikokolakis et al. (2010)): in the finite case, black-box methods are useful when i) a closed-form expression of the system is too complex to derive, or even ii) when $\mu(o, s)$ is known, but the space $\mathcal{O} \times \mathcal{S}$ is too large and it is computationally infeasible to determine R^* analytically.

To date, the major black-box estimation methods proposed in QIF are based on the frequentist approach (subsection 3.1.1). In this chapter, we compare this approach with UC rules on both synthetic and real-world data. We focus our discussion on nearest neighbour UC rules (NN and k_n-NN), which in our formulation can be seen as an extension of the frequentist approach (section 3.1).

5.1 Notation

It is useful to introduce the following notation from the QIF literature. We define a *channel matrix* $\mathcal{C}_{s,o}$, describing the conditional *a posteriori* distribution over the objects given a secret, $\mathcal{C}_{s,o} := P(o|s)$. Note that the tuple $(\pi, \mathcal{C}_{s,o})$, for some priors π, is sufficient to describe a system's underlying distribution $\mu(o, s)$ (and, therefore, the system itself); indeed, given $(\pi, \mathcal{C}_{s,o})$, the corresponding distribution is $\mu(o, s) = \mathcal{C}_{s,o}\pi(s)$.

5.2 Synthetic systems

We evaluate the frequentist and nearest neighbour estimates on discrete synthetic systems defined for various distributions on the channel matrix.

Name	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{O}\rvert$	R^*
Random 100-100	100	100	0.979
Geometric 1.0 100x10K	100	10K	~ 0
Geometric 0.1 100x10K	100	10K	0.007
Geometric 0.01 100x10K	100	10K	0.600
Geometric 0.5 100x10K	100	10K	~ 0
Geometric 0.5 1Kx100K	1K	100K	~ 0
Geometric 0.5 10Kx1M	10K	1M	~ 0
Geometric 0.2 100x1K	100	1K	0.364
Geometric 0.02 100x10K	100	10K	0.364
Geometric 0.002 100x100K	100	100K	0.364

Table 5.1: Synthetic systems we consider, and their real Bayes risk R^*.

5.2.1 Experimental setting

We sample n examples from a system's distribution, and then compute the estimate on the whole object space as in Equation 3.2; this is possible because $\mathcal{O}$ is finite and μ is known.

We estimate the Bayes risk with the following estimators: frequentist – the *status quo* –, NN, and k_n-NN (for the choices $k_n = \log(n)$ and $k_n = \log_{10}(n)$), as we described in chapter 3.

Since for synthetic data we also know the real Bayes risk, we can measure how many examples are required for the convergence of each estimate. We do this as follows: let R_n^g be an estimate of R^*, trained on a dataset of n examples. We say the estimate δ-converged to R^* after n examples if its relative change from R^* is smaller than δ:

$$\frac{\left|R_n^g - R^*\right|}{R^*} < \delta\,.$$

While the relative change has the advantage of taking into account the magnitude of the compared values, it is not defined when the denominator is 0; therefore, when $R^* \approx 0$ (Table 5.1), we verify convergence with the absolute change:

$$\left|R_n^g - R^*\right| < \delta\,.$$

Table 5.1 summarises the systems we evaluate in our experiments; we describe them into details in what follows. We assume uniform priors for all the systems.

5.2.2 Geometric systems

We first consider systems generated by Geometric noise functions, which are one of the typical mechanisms used to implement Differential Privacy (Dwork, 2006). We consider different parameters to illustrate the effect of their variation on the convergence of nearest neighbour methods and the frequentist one.

System description

Let $\mathcal{S} = \{1, 2, \ldots, w\}$ and $\mathcal{O} = \{1, 2, \ldots, w'\}$, associated with the standard notion of distance. Two numbers $s, s' \in \mathcal{S}$ are called *adjacent* if $s = s' + 1$ or $s' = s + 1$.

Let ν be a real non-negative number and consider a function $t : \mathcal{S} \mapsto \mathcal{O}$. The channel matrix of the Geometric system is:

$$\mathcal{C}_{s,o} = P(o \mid s) = \alpha \exp(-\nu \,|\, t(s) - o \,|)\,,$$

where α is a normalising factor. Note that the privacy level is defined by ν/Δ_t, where Δ_t is the sensitivity of t:

$$\Delta_t = \max_{s_1 \sim s_2 \in \mathcal{S}} (t(s_1) - t(s_2))\,,$$

and $s_1 \sim s_2$ means s_1 and s_2 are adjacent. Now let $t(s) = s \cdot w'/w$. We define

$$\alpha = \begin{cases} e^{\nu}/(e^{\nu}+1) & \text{if } o = 1 \text{ or } o = w' \\ (e^{\nu}-1)/(e^{\nu}+1) & \text{otherwise}\,, \end{cases}$$

so to truncate the distribution at its boundaries.

We will now consider the following three parameters:

- the privacy level ν/Δ_t, which here is equal to $\nu|\mathcal{S}|/|\mathcal{O}|$,
- the size of the secret space $|\mathcal{S}|$, and
- the ratio $|\mathcal{O}|/|\mathcal{S}|$;

we vary each of these parameters one at the time, to isolate their effect on the convergence rate.

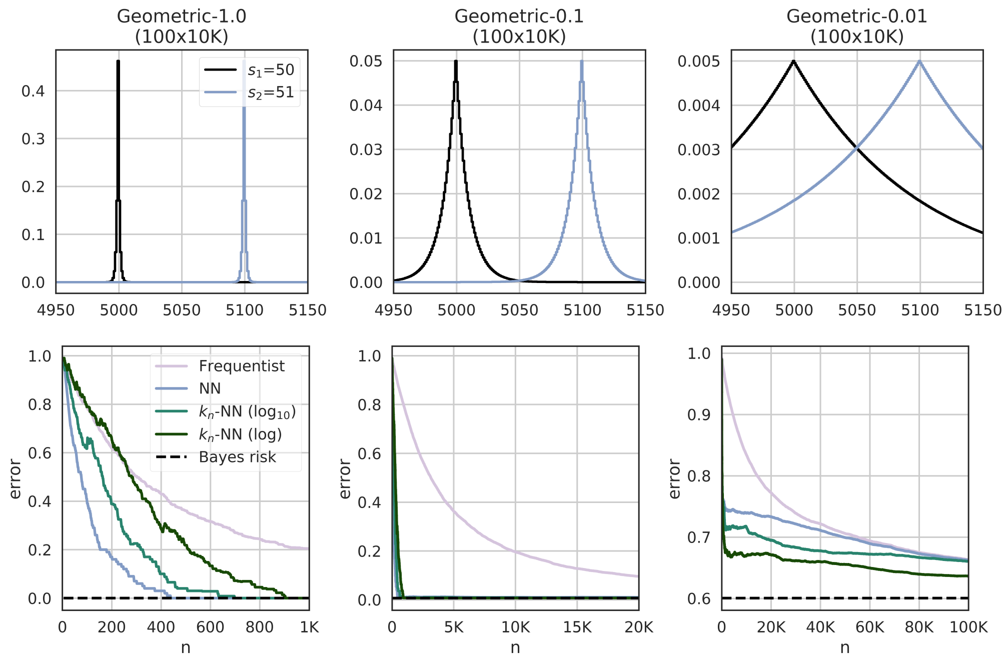

Figure 5.1: Estimates' convergence for Geometric systems when varying their privacy level. The respective distributions are shown in the top figure for two adjacent secrets $s_1 \sim s_2$.

				k_n-NN	
System	δ	Freq.	NN	log$_{10}$	log
Geometric	0.1	1 989	**262**	391	674
100x10K	0.01	19 823	**420**	628	894
ν = 1.0	0.001	198 057	**434**	693	899
Geometric	0.1	18 105	**264**	391	668
100x10K	0.01	127 201	**434**	628	894
ν = 0.1	0.001	X	X	10 727	**900**
Geometric	0.1	105 448	103 352	99 847	**34 238**
100x10K					
ν = 0.01					

Table 5.2: Convergence of the estimates when varying ν.

Variation of the privacy level

We fix $|\mathcal{S}| = 100$, $|\mathcal{O}| = 10$K, and we consider three cases: $\nu = 1.0$, $\nu = 0.1$, and $\nu = 0.01$. The results for the estimation of the Bayes risk and the convergence rate are illustrated in Figure 5.1 and Table 5.2 respectively. In the tables, results are reported for δ convergence levels $\{0.1, 0.01, 0.001\}$; an "X" means a particular estimate did not converge within 500K examples, a missing row for a certain δ means no estimate converged.

The results indicate that the nearest neighbour methods have a much faster convergence than the standard frequentist approach, particularly when dealing with larger systems. The reason is that Geometric systems have a regular behaviour with respect to the Euclidean metric, which can be exploited by NN and k_n-NN to make good predictions for unseen objects.

Variation of the input size

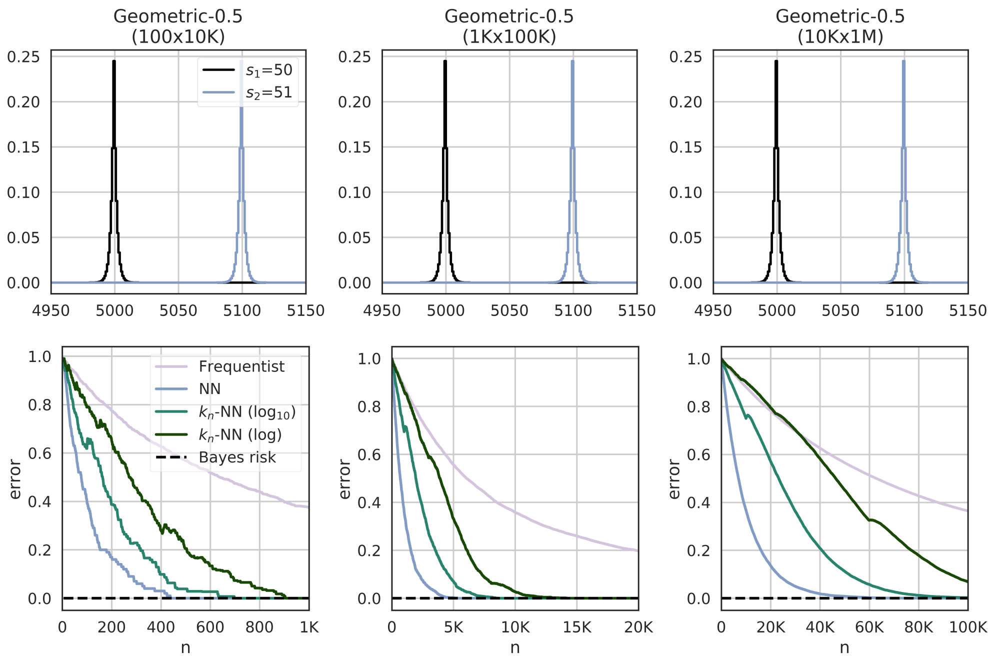

Figure 5.2: Estimates' convergence for Geometric systems when varying the number of secrets. The respective distributions are shown in the top figure for two adjacent secrets $s_1 \sim s_2$.

Here we fix $\nu = 0.5$, $|\mathcal{O}|/|\mathcal{S}| = 100$, and we consider three cases $|\mathcal{S}| = 100$, $|\mathcal{S}| = 1\text{K}$, and $|\mathcal{S}| = 10\text{K}$. The results are in Figure 5.2 and Table 5.3. They confirm what was logical to expect, namely that if we scale the number of inputs of a factor c and all the other parameters remain the same, then the results (the number of

				k_n-NN	
System	δ	Freq.	NN	$\log_{10}$	log
Geometric	0.1	3 926	**264**	391	678
100x10K	0.01	38 181	**434**	628	894
ν = 0.5	0.001	371 823	**434**	693	899
Geometric	0.1	39 461	**2 191**	4 570	7 287
1Kx100K	0.01	403 135	**4 083**	7 018	11 337
ν = 0.5	0.001	X	**5 329**	8 427	14 133
Geometric	0.1	X	**22 929**	51 705	92 740
10Kx1M	0.01	X	**46 712**	82 211	X
ν = 0.5	0.001	X	**66 610**	X	X

Table 5.3: Convergence of the estimates when varying $|\mathcal{S}|$.

examples necessary to get the same estimation) are scaled by the same factor c, for all the methods.

Variation of the ratio $|\mathcal{O}|/|\mathcal{S}|$

				k_n-NN	
System	δ	Freq.	NN	$\log_{10}$	log
Geometric	0.1	8 674	8 702	7 103	**2 500**
100x1K	0.01	**51 689**	60 791	60 791	60 791
ν = 0.2	0.001	**180 659**	**180 659**	**180 659**	**180 659**
Geometric	0.1	85 907	85 639	70 998	**11 192**
100x10K					
ν = 0.02					
Geometric	0.1	X	X	413 969	**2 962**
100x100K					
ν = 0.002					

Table 5.4: Convergence of the estimates when varying $|\mathcal{O}|/|\mathcal{S}|$.

Now we fix $|\mathcal{S}| = 100$, $\nu|\mathcal{O}|/|\mathcal{S}| = 2$, and we consider three cases $|\mathcal{O}|/|\mathcal{S}| = 10$, $|\mathcal{O}|/|\mathcal{S}| = 100$, and $|\mathcal{O}|/|\mathcal{S}| = 1K$. (Note that as a consequence also ν has to vary: we have to set ν to 0.2, 0.02, and 0.002, respectively.) Results in Figure 5.3 and Table 5.4 show how the nearest neighbour methods become much better than the frequentist approach as $|\mathcal{O}|/|\mathcal{S}|$ increases. This is because the larger is the object space,

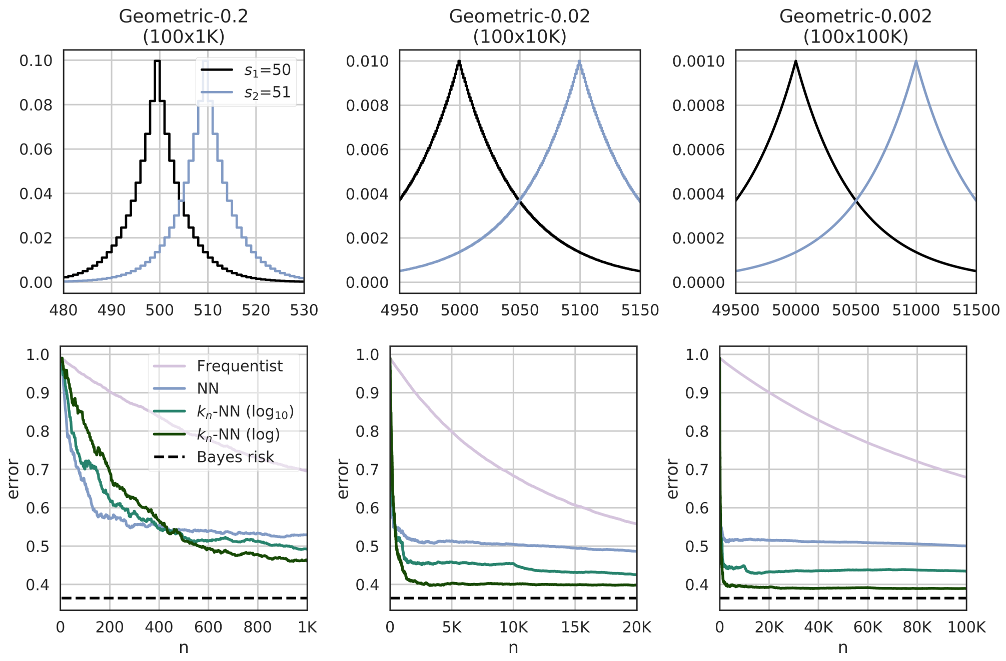

Figure 5.3: Estimates' convergence for Geometric systems when varying the ratio $|\mathcal{O}|/|\mathcal{S}|$. The respective distributions are shown in the top figure for two adjacent secrets $s_1 \sim s_2$.

the larger is the number of unseen objects at the moment of classification, and the more the frequentist approach has to rely on random guessing. The nearest neighbour methods are not largely affected because they can rely on the proximity to outputs already classified.

5.2.3 Spiky system: making nearest neighbour rules fail

Nearest neighbour rules can take advantage of the metric on the object space to improve their convergence considerably. However, there are systems for which the frequentist outperforms NN and k_n-NN. While this does not come as a surprise, given that an "optimal" learning rule does not exist (section 2.6), investigating the form of such systems is important to understand when these methods fail.

System description We construct an example of such systems, which we call the Spiky system. Consider an observation space constituted of q consecutive integer numbers $\mathcal{O} = \{0, ..., q-1\}$, for some even positive integer q, and secrets' space $|\mathcal{S}| = 2$. Assume that $\mathcal{O}$ is a ring with the operations $+$ and $-$ defined as the sum

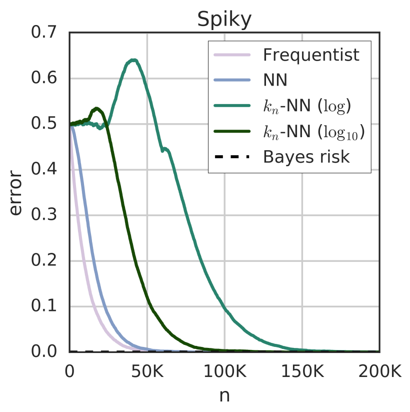

Figure 5.4: Estimates' convergence for a Spiky system (2x10K).

and the difference modulo q, respectively, and consider the distance on $\mathcal{O}$ defined as: $d(i,j) = |i - j|$. Note that $(\mathcal{O}, d)$ is a "circular" structure, i.e., $d(q-1, 0) = 1$. The Spiky system has uniform prior, and channel:

$$\mathcal{C}_{s,o} = \begin{bmatrix} 2/q & 0 & 2/q & \dots & 0 \\ 0 & 2/q & 0 & \dots & 2/q \end{bmatrix}.$$

This system is crafted so that most neighbours of an observable are more likely to be associated with the wrong secret. This means that NN and k_n-NN rules will tend to predict the wrong secret, until enough examples are available.

Discussion We conducted experiments for a Spiky system of size $|\mathcal{O}| = 10K$. Results in Figure 5.4 confirm our hypothesis: nearest neighbour rules are misled for this system.

Interestingly, while the NN estimate keeps decreasing as the number of examples n increases, there is a certain range of n's where the k_n-NN estimates become worse than random guessing. Intuitively, this is because when n becomes larger than $|\mathcal{O}|$ all elements in $\mathcal{O}$ tend to be covered by the examples. For every $i \in \mathcal{O}$ there are two neighbours, $i-1$ and $i+1$, that belong to the class opposite to the one of i, so if k is not too small with respect to n, it is likely that in the multiset of the k closest neighbours of i, the number of $i-1$'s and $i+1$'s exceeds the number of i's, which means that i will be misclassified. As n increases, however, the ratio between k and the number of i's in the examples tends to decrease (because $k/n \to 0$ as $n \to \infty$), hence at some point we will have enough i's to win the majority vote in the k neighbours (i's are considered before than $i-1$'s and $i+1$'s, by nearest neighbour definition) so i will not be misclassified any more.

Concerning the comparison between the NN and frequentist estimates, we can do it analytically. We start by computing the expected error of the NN method on the spiky system in terms of the number of training examples n. Let T^n be a training set of examples of size n. Given a new object i, let us consider the NN estimate $r_n(i)$ of the conditional Bayes risk r^* for i (i.e., the expected classification error for i). This corresponds to the probability that the element o closest to i that appears in the training set is at odd distance from i (i.e., $d(i,o) = 2\ell + 1$, for some natural number ℓ). Namely it is the probability that:

- i is not in training data but either $i+1$ or $i-1$ are, or
- $i, i \pm 1, i \pm 2$ are not in training data but either $i+3$ or $i-3$ are, or
- …etc.

Hence we have:

$$\begin{aligned} r(i) = P(d(i,o) = 2l+1) &= \\ &= P(i \notin T^n, i+1 \in T^n) + P(i \notin T^n, i-1 \in T^n) + \dots \\ &= 2 \cdot \sum_{\ell=0}^{q/4-1} a^{4\ell+1}(1-a), \end{aligned}$$

where $a = (1 - 1/q)^n$ is the probability that an element $e \in \mathcal{O}$ does not occur in any of the n examples of the training set. (Thus $a^{4\ell+1}$ represents the probability that none of the elements $i, i \pm 1, i \pm 2, i \pm 2s$, with $\ell = 2s$, appear in the training set, and $1-a$ represents the probability that the element $2s+1$ (resp. $2s-1$) appears in the training set.) By using the result of the geometric series

$$\sum_{t=0}^{m} a^t = \frac{1-a^{m+1}}{1-a},$$

we obtain:

$$r_n(i) = 2a\frac{1-a^q}{(1+a^2)(1+a)}.$$

Since we assume that the distribution on $\mathcal{O}$ is uniform, we have $R_n^{NN} = r_n(i)$.

We want to study how the error estimate depends on the relative size of the training set with respect to the size of $\mathcal{O}$. Hence, let $x = {}^{n}/_{q}$. Then we have $a = (1 - {}^{1}/_{q})^{qx}$, which, for large q, becomes $a \approx e^{-x}$. Therefore:

$$R_x^{NN} \approx 2e^{-x} \frac{1 - e^{-qx}}{(1 + e^{-2x})(1 + e^{-x})}.$$

It is easy to see that $R_x^{NN} \to {}^{1}/_{2}$ for $x \to 0$, and $R_x^{NN} \to 0$ for $x \to \infty$, as expected.

Consider now the frequentist estimate R_x^{Freq}. In this case, given an element $i \in \mathcal{O}$, the classification is done correctly if i appears in the training set. Otherwise, we do random guessing, which gives a correct or wrong classification with equal probability. Only the latter case contributes to the probability of error, hence the error estimate is half the probability expectation that i does not belong to the training set:

$$R_x^{Freq} = \frac{1}{2}(1 - \frac{1}{q})^n \approx \frac{1}{2}e^{-x}.$$

Therefore, R_x^{NN} is always above R_x^{Freq}.

5.2.4 Random system

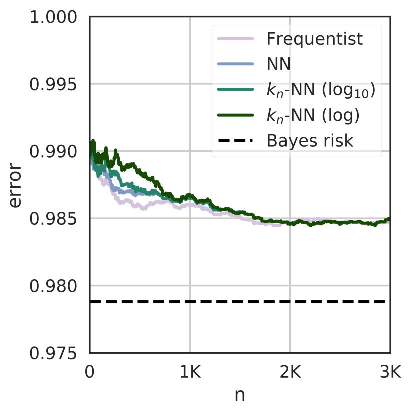

Figure 5.5: Estimates' convergence for a Random system (100 × 100).

In the previous sections, we have seen cases when our methods greatly outperform the frequentist approach, and a contrived system example for which they fail. We now consider a system generated randomly to evaluate their performances for an "average" system.

			k_n-NN	
δ	Freq.	NN	$\log_{10}$	log
0.01	**77**	134	197	495
0.001	668 124	668 124	668 124	668 124

Table 5.5: Random: examples required for δ-convergence.

System description The channel matrix of a Random system is produced by drawing its elements from the uniform distribution, $\mathcal{C}_{s,o} \xleftarrow{\$} Uni(0,1)$, and then normalising its rows appropriately so that $\sum_{o \in \mathcal{O}} P(o|s) =$ for all $s \in \mathcal{S}$.

Evaluation We consider a Random system with $|\mathcal{S}| = |\mathcal{O}| = 100$, and count the number of examples required for δ-convergence. Table 5.5 reports the results.

The frequentist estimate is slightly better than NN and k_n-NN for $\delta = 0.01$. However, for stricter convergence requirements ($\delta = 0.001$), all the methods require the same (large) number of examples. Figure 5.5 show that indeed the methods begin to converge similarly already after 1K examples.

5.2.5 Comparison with the frequentist approach

Results showed that nearest neighbour estimates require significantly fewer examples than the frequentist approach when dealing with large systems that have a notion of distance on their output; however, they are generally equivalent to the frequentist approach in the case of small systems.

To better understand why this is the case, we obtain a crude approximation of the frequentist Bayes risk estimate.

$$R_n^{Freq} \approx R^* \left(1 - \left(1 - \frac{1}{|\mathcal{O}|}\right)^n\right) + R^\pi \left(1 - \frac{1}{|\mathcal{O}|}\right)^n .$$

This approximation, derived and studied in Appendix 1, makes the very strong assumption that all objects $o \in \mathcal{O}$ are equally likely to be sampled from $\mu(o,s)$, i.e.: $P(o) = \frac{1}{|\mathcal{O}|}$. However, it is enough to give us an insight on the performance of frequentist approach: $\left(1 - \frac{1}{|\mathcal{O}|}\right)^n$ is the probability that some object does not appear within a training set of size n. This probability weighs the value of the frequentist estimate between the optimal R^*, used when the object appears in the training data, and random guessing R^π. This estimate converges to the Bayes risk asymptotically. However, it shows that the principal factor influencing its convergence rate is the probability of observing an object – and, therefore, the number of objects that have non-negligible probability of being observed, which for real-world systems is closely related to the size of the object space itself.

5.3 Application to time side channel in finite field exponentiation

We use F-BLEAU to measure the leakage in the running time of the *square-and-multiply* exponentiation algorithm in the finite field $\mathbb{F}_{2^w}$; exponentiation in $\mathbb{F}_{2^w}$ is relevant, for example, for the implementation of the ElGamal cryptosystem.

We consider a hardware equivalent implementation of the algorithm computing m^s in $\mathbb{F}_{2^w}$, which can be thought of as the decryption of a message m under key s. We focus our analysis on the simplified scenario of a *one-observation* adversary, who makes exactly *one* measurement of the algorithm's execution time o, and aims to predict the corresponding secret key s.

A similar analysis was done by Backes and Köpf (2008) by using a leakage estimation method based on the frequentist approach. Their analysis also extended to a *many-observations adversary*, that is, an adversary who can run the algorithm many times for the same secret s, thereby obtaining q timing observations $(o_1, ..., o_q)$ from which to predict s. We introduce an extension of the NN classifier to this setting in section 10.2.

5.3.1 Side channel description

Square-and-multiply is a fast algorithm for computing m^s in the finite field $\mathbb{F}_{2^w}$, where w here represents the bit size of the operands m and s. It works by performing a series of multiplications according to the binary representation of the exponent s, and its running time is correlated to the number of 1's in s. This fact was noticed by Kocher (1996), who suggested side channel attacks to the RSA cryptosystem based on time measurements.

5.3.2 Message blinding

We assume the system implements *message blinding*, a technique which hides to an adversary the value m for which m^s is computed. Blinding was suggested as a method for thwarting time side channels (Kocher, 1996), which works as follows.

Operands' size	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{O}\rvert$
4 bits	2^4	34
6 bits	2^6	123
8 bits	2^8	233
10 bits	2^{10}	371
12 bits	2^{12}	541

Table 5.6: Size of secret and output space for the side channel to finite field exponentiation.

Consider, for instance, decryption for the RSA cryptosystem: $m^d(modN)$, for some decryption key d. The cryptographic hardware first computes $m \cdot r^e$, where e is the encryption key and r is some random value; then it computes $(mr^e)^d$, and it returns the decrypted message after dividing the result by r.

Message blinding has the advantage of hiding information to an adversary; however, it was shown that it is not enough for preventing time side channels (e.g., Backes and Köpf (2008)).

5.3.3 Implementation and results

We consider a `Gezel` implementation of finite field exponentiation. `Gezel`[1] is a description language for clocked hardware, equipped with a simulation environment whose executions preserve the corresponding circuit's timing information. We measure the number of clock cycles as a timing information o, which means that our observations take finite values. The guarantees of `Gezel` mean that the number of clock cycles we measure reflects the corresponding circuit implementation (Köpf and Basin, 2006).

We compare the performances of the frequentist and nearest neighbour approaches in terms of the number of black-box examples required for convergence. For each bit size $w \in \{4, 6, .., 12\}$, and for all the values $(m_i, s_i) \in \{0, ..., 2^w - 1\}^2$, we run the exponentiation algorithm to compute m^s, and we measure its execution time o_i. We estimate the Bayes risk by training a classifier on an increasing dataset of increasing examples, and by computing its error on a validation set. We set the size of the validation set to $\min(0.2 \cdot 2^{2w}, 250\,000)$.

Results in Figure 5.6 show that, while for small bit sizes the frequentist approach outperforms nearest neighbour rules, as w increases the frequentist approach re-

[1]`http://rijndael.ece.vt.edu/gezel2`.

quires a larger number of examples. Nevertheless, in these experiments we did not notice a substantial advantage in nearest neighbour rules, even though the output space is equipped with a notion of metric. Table 5.6 helps interpreting this result: for larger bit sizes w of the exponentiation operands, the number of possible output values (i.e., clock cycles) only increase minimally; note that the size of the output space $\mathcal{O}$ here is given by the number of individual values that clock cycles took. This confirms that, as noticed in our previous experiments, nearest neighbour and frequentist estimates tend to perform similarly for systems with a small output space.

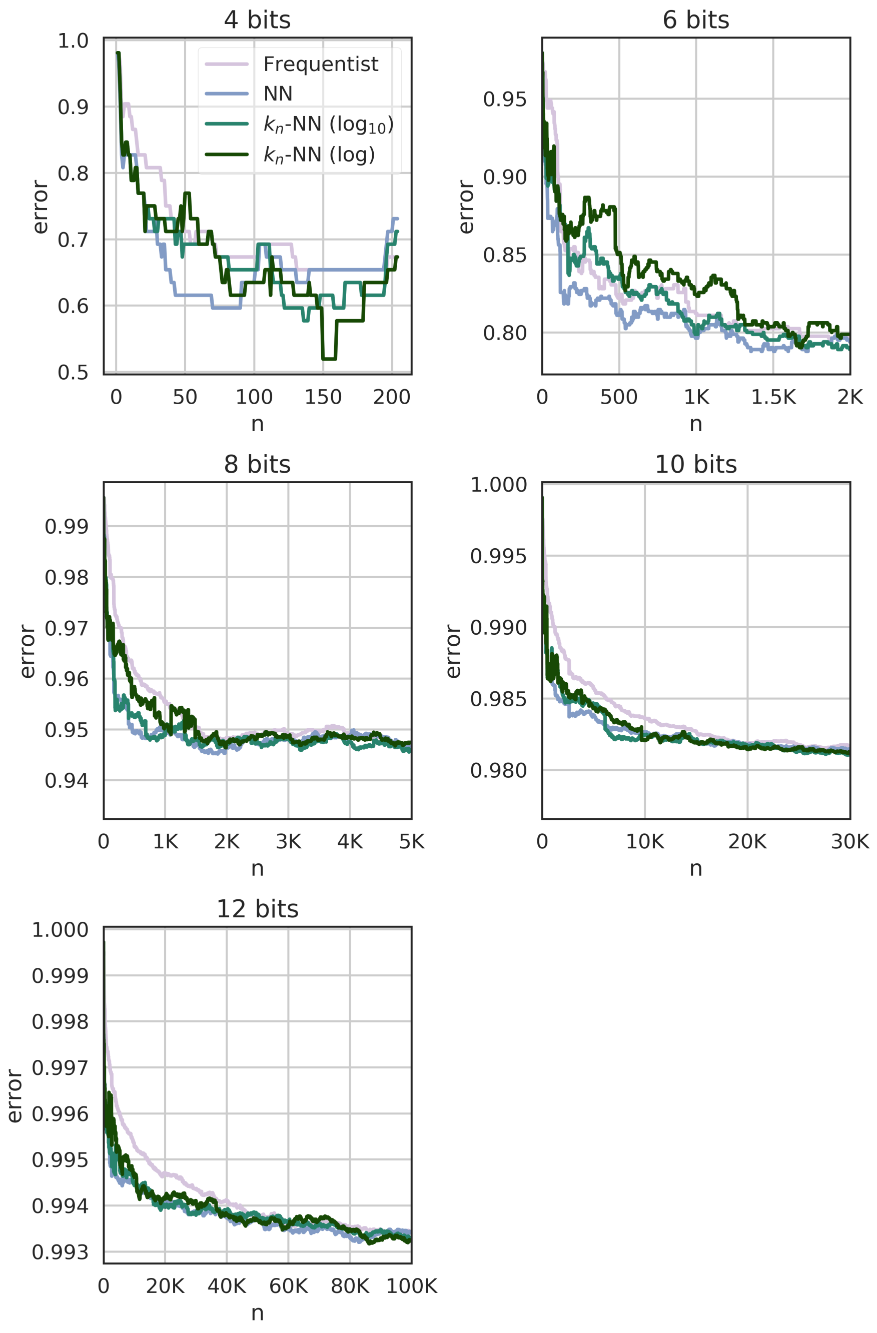

Figure 5.6: Convergence of the estimates for the time side channel attack to the exponentiation algorithm as the bit size of the operands increases.

6 Infinite $\mathcal{O}$

In this chapter, we consider an application where the object space is continuous. Note that, in this case, the frequentist approach cannot be used (nor defined formally). NN does not guarantee UC for continuous $\mathcal{O}$; however, we will include it in this evaluation, as it can be used even in this case.

We evaluate a dataset of users' locations, `Gowalla`, protected under three privacy mechanisms; two of them output finite values, in which case we will do a comparison with the frequentist approach, one has a continuous output. We consider further applications with continuous $\mathcal{O}$ in chapter 7, chapter 9, and section 10.4.

6.1 Application to location privacy

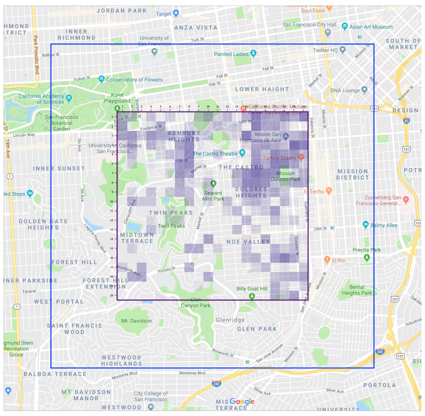

Figure 6.1: Area of San Francisco considered for the experiments. The input locations corresponds to the inner square, the output locations to the outer one. The coloured cells represent the distribution of the `Gowalla` checkins.

We show that our techniques can be successfully applied to estimate the degree of protection provided by mechanisms such as those used in location privacy. Since our goal is to evaluate their precision, we consider basic mechanisms for which the Bayes risk can also be computed analytically, so that we can use it for comparison. Of course, the intended applications of our methods are mechanisms or situations where the Bayes risk *cannot* be computed directly, either because this is too complicated, or because of the presence of unknown factors. Examples abound; for instance, the availability of additional information, like the presence of points of interest (e.g., shops, churches), or geographical characteristics of the area (e.g., roads, lakes) can affect the Bayes risk in ways that are impossible to evaluate formally.

We will consider the planar Laplacian and the planar Geometric, that are the typical mechanisms used to obtain geo-indistinguishability (Andrés et al., 2013), and one of the optimal mechanisms proposed by Oya et al. (2017) as a refinement of the optimal mechanism by Shokri et al. (2012). In particular, we will use the mechanism that achieves an optimal trade-off between privacy (measured as residual entropy) and utility loss (measured as expected distance between the true location and the obfuscated one). The construction of such a mechanism relies on an algorithm that was independently proposed by Blahut and by Arimoto to solve the problem of achieving an optimal trade-off between the minimisation of the distortion rate and the minimisation of the mutual information (Cover and Thomas, 2006). From now on, we shall refer to this as the Blahut-Arimoto mechanism. Note that the Laplacian is a continuous mechanism, i.e., it outputs obfuscated locations on the continuous plane. The other two are discrete.

In these experiments, we estimate the error of UC rules on a validation set (section 3.2), for an increasing number n of training examples.

6.1.1 The Gowalla dataset

We will consider real location data from the `Gowalla` dataset (Gow; Cho et al., 2011), which contains users' checkins and their location in terms of latitude and longitude. We use data from a squared area in San Francisco, centred in the point of latlon coordinates (37.755, -122.440), and extending for 1.5 Km in each direction. This input area corresponds to the inner (purple) square in Figure 6.1. We discretise the input using a grid of 20×20 cells of size 150×150 Sq m; the secret space $\mathcal{S}$ of the system consists therefore of 400 locations. The prior distribution on the secrets is derived from the `Gowalla` checkins, and it is represented in Figure 6.1 by the different colour intensities on the cells of the input grid.

The output area is represented in Figure 6.1 by the outer (blue) square. It spawns 1050 m (7 cells) more than the input square on every side. The reason we consider a larger area for the output is that the planar Laplace and the planar Geometric

Mechanism	ν	R^*	Utility
Blahut-Arimoto	2	0.760	334.611
	4	0.571	160.839
	8	0.428	96.2724
Geometric	2	0.657	288.372
	4	0.456	144.233
	8	0.308	96.0195
Laplacian	2	0.657	288.66
	4	0.456	144.232
	8	0.308	96.212

Table 6.1: True Bayes risk and utility for `Gowalla` dataset defended under various location privacy mechanisms.

naturally expand outside the input square.[1] Since the planar Laplacian is continuous, its output domain $\mathcal{O}$ is constituted by all the points of the outer square. As for the planar Geometric and the Blahut-Arimoto mechanisms, which are discrete, we divide the output square in a grid of 350×340 cells of size 15×15 Sq m. The size of $\mathcal{O}$ for these mechanisms is therefore $340 \times 340 = 115,600$ cells.

6.1.2 Defences

The *planar Geometric* mechanism has channel matrix $C_{s,o}$, representing the conditional probability to report the location o when the true location is s:

$$C_{s,o} = \alpha \exp\left(-\frac{\ln \nu}{100} d(s,o)\right),$$

where ν is a parameter controlling the level of noise, α is a normalisation factor, and $d(s,o)$ is the Euclidean distance between s and o.

The conditional probability of the *planar Laplacian* is defined by the same equation, except that o belongs to a continuous domain, and the equation defines a probability density function.

As for the *Blahut-Arimoto*, it is obtained as the result of an iterative algorithm, whose definition can be found in Cover and Thomas (2006).

[1] In fact, these functions distribute the probability on the infinite plane, but on locations very distant from the origin the probability becomes negligible.

6.1.3 Results

We evaluated the estimation's convergence to the Bayes risk as a function of the number of training examples n, and for different values of the level of noise: $\nu = \{2, 4, 8\}$. Table 6.1 reports the true Bayes risk for the `Gowalla` dataset, defended under these mechanisms for the various choices of parameters.

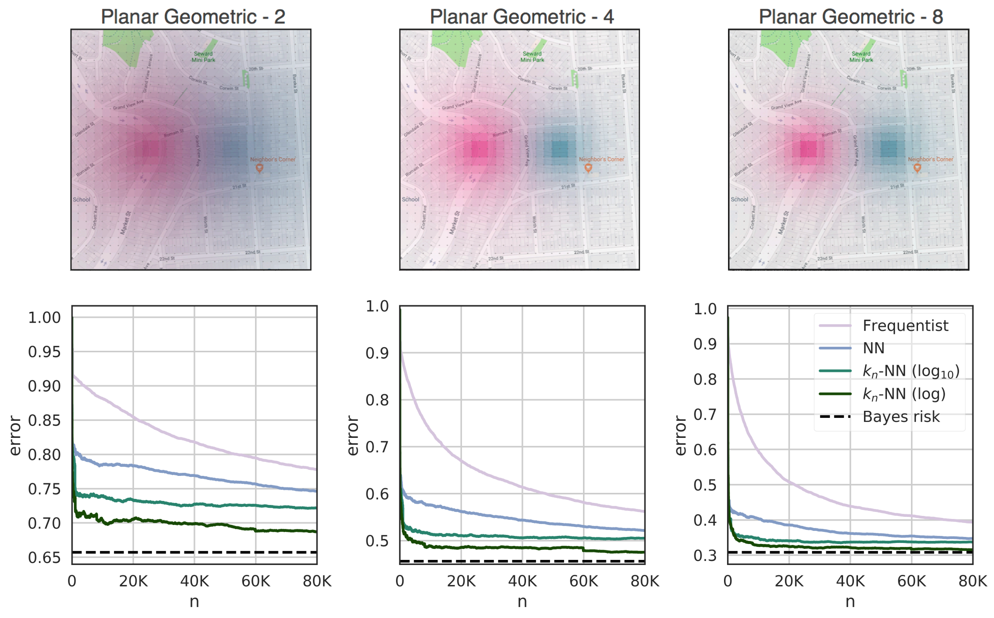

Figure 6.2: Estimates' convergence rate for the planar Geometric defence applied to the `Gowalla` dataset, for $\nu = 2$, $\nu = 4$ and $\nu = 8$, respectively. On top of each graph is the distribution of the geometric noise for two adjacent secrets.

				k_n-NN	
ν	δ	Freq.	NN	log 10	log
2	0.1	X	X	26 809	**1 102**
	0.05	X	X	X	**54 914**
4	0.1	X	X	35 942	**2 820**
	0.05	X	X	X	**45 032**
8	0.1	X	X	13 236	**5 249**
	0.05	X	X	X	**19 948**

Table 6.2: Convergence for the Planar Geometric for various ν.

The results for the geometric noise are shown in Figure 6.2. We observe that convergence is faster when ν is higher (i.e., with less noise and therefore lower Bayes risk); this is in line with the results for the synthetic systems of chapter 5. In all cases, the nearest neighbour methods outperform the frequentist one; this was expected given the large output space. Table 6.2 shows the number of examples required to achieve δ-convergence to the Bayes risk. The symbol "X" means that we did not achieve the required level of approximation within 80K examples.

The corresponding results for the Laplacian noise are shown in Figure 6.2 and Table 6.3. In this case we did not evaluate the frequentist approach, which cannot be used in the continuous case. Results indicate that δ-convergence is reached within a small number of examples by the k_n-NN estimator, for $k_n = \log(n)$.

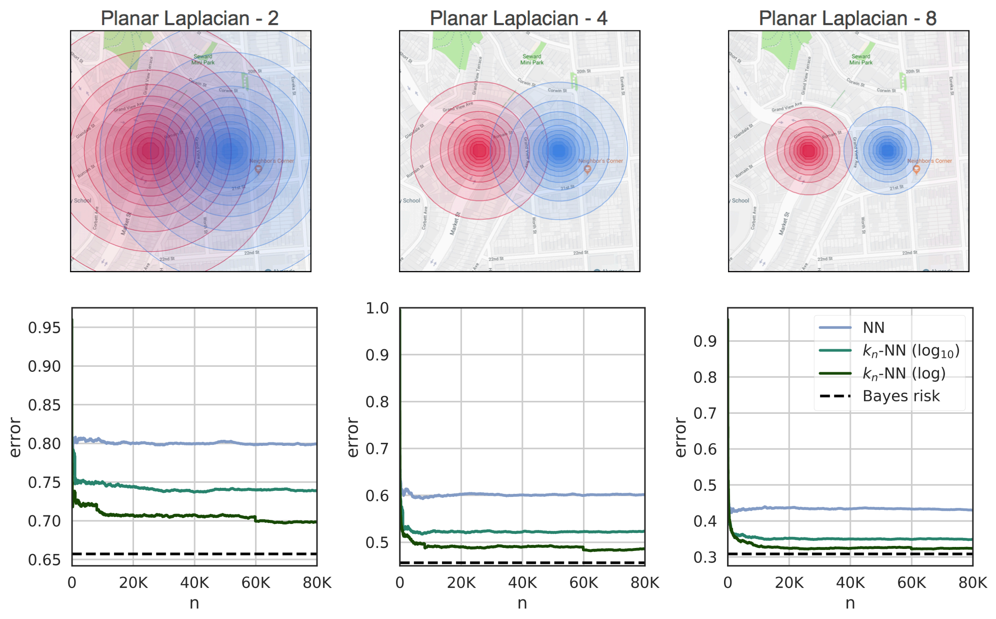

Figure 6.3: Estimates' convergence speed for the planar Laplacian defence applied to the `Gowalla` dataset, for $\nu = 2$, $\nu = 4$ and $\nu = 8$, respectively. On top of each graph is the distribution of the geometric noise for two adjacent secrets.

The case of the Blahut-Arimoto mechanism is quite different: surprisingly, the output probability concentrates on a small number of locations. For instance, in the case $\nu = 2$, with 100K sampled pairs we obtained only 19 different output locations (which reduced to 14 after we mapped them on the 20×20 grid). Thanks to the small number of actual outputs, all the methods converge very fast. The results are shown in Figure 6.4 and in Table 6.4.

				k_n-NN	
ν	δ	Freq.	NN	log 10	log
2	0.1	N/A	X	X	**259**
4	0.1	N/A	X	X	**4 008**
8	0.1	N/A	X	X	**6 135**
	0.05	N/A	X	X	**19 961**

Table 6.3: Convergence for the Planar Laplacian for various ν.

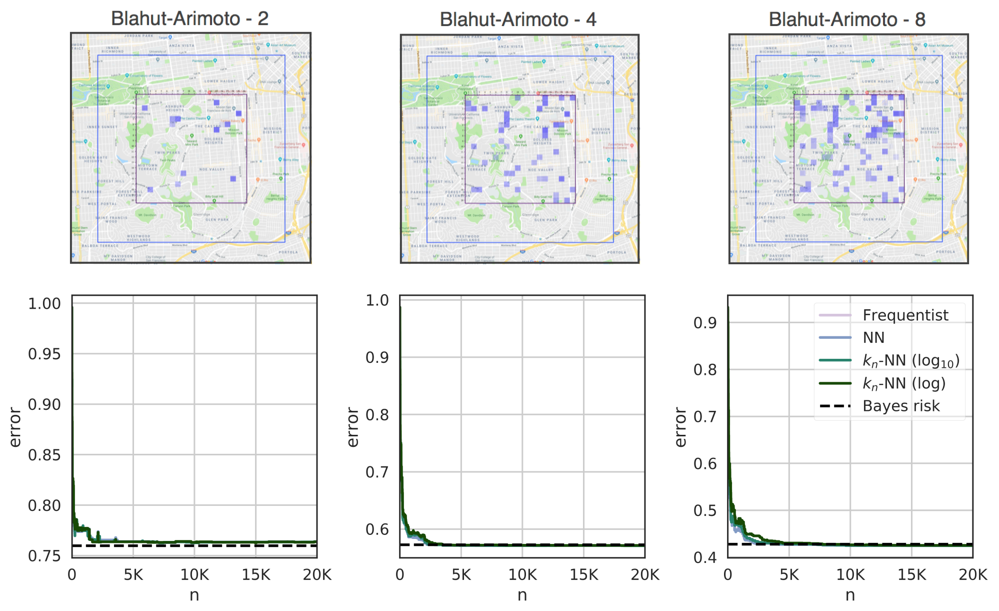

Figure 6.4: Estimates' convergence speed for the planar Laplacian defence applied to the `Gowalla` dataset, for $\nu = 2$, $\nu = 4$ and $\nu = 8$, respectively. On top of each graph it is represented the distribution of the output probability as produced by the mechanism. All the outputs with non-null probability turn out to be inside the input square. The Blahut-Arimoto noise for two adjacent input cells distributes in on the outputs with non-null probability in a way similar to the laplacians. The outputs are originally points on the 340×340 output grid, but here are mapped on the coarser 20×20 grid for the sake of visual clarity.

				k_n-NN	
ν	δ	Freq.	NN	log 10	log
2	0.1	**37**	**37**	**37**	**37**
	0.05	**135**	**135**	**135**	**135**
	0.01	1 671	1 664	**1 408**	**1 408**
	0.005	6 179	5 724	**1 671**	**1 671**
	0.001	X	X	X	X
4	0.1	**220**	**220**	**220**	257
	0.05	503	**502**	509	703
	0.01	**2 029**	**2 029**	2 055	2 404
	0.005	2 197	**2 055**	2 280	2 658
	0.001	X	**2 404**	2 830	3 481
8	0.1	**345**	398	553	1 285
	0.05	1 285	**1 211**	1 343	1 679
	0.01	2 104	**2 017**	2 495	4 190
	0.005	**2 231**	**2 231**	3 433	6 121
	0.001	**3 881**	**3 881**	6 079	7 724

Table 6.4: Convergence for the Blahut-Arimoto for various ν.

7 Comparison with leakiEst

In the previous chapters, we have seen that nearest neighbour methods tend to substantially outperform the frequentist approach whenever there is a notion of distance on the output space, and in particular when the output space is large. We will call F-BLEAU (Fast Black-box Leakage Estimation AUtomated) the tool we introduced in Cherubin et al. (2019), which runs several nearest neighbour estimators and the frequentist approach, and then selects the smallest estimate.

LeakWatch (Chothia et al., 2014) and leakiEst (Chothia et al., 2013) are the state-of-the-art Black-box security measurement tools. Both are based on the frequentist approach, and they should therefore inherit the drawbacks we have seen in the previous chapters. leakiEst can be considered as an evolution of LeakWatch: they both compute Shannon mutual information (MI) and Min-entropy leakage (ME) on the finite output case, but leakiEst also computes MI in the infinite output case, under some continuity conditions. leakiEst runs two evaluations: i) evidence / no evidence of leakage (*zero-leakage* test), and ii) leakage estimation. These are accompanied by confidence indications, and it is possible that leakiEst reports no evidence of leakage, and yet a non-zero leakage estimation.

In this chapter, we compare leakiEst with F-BLEAU.[1] In particular, we wish to verify whether the advantage of nearest neighbour techniques with respect to the frequentist approach, that we observed in chapter 5 and chapter 6, translates into an advantage also with respect to leakiEst. We perform this comparison for a time side channel on the RFID on European passports, and on the `Gowalla` examples that we considered in the previous chapter.

7.1 Time side channel on e-Passports' RFID

Chothia and Smirnov (2010) discovered a side channel attack in the way the protocols of various European countries exchanged message some years ago (the proto-

[1]Note that, while F-BLEAU can also compute the frequentist estimate, we limit it to nearest neighbour methods for this discussion.

Passport	leakiEst: Evidence of leak? (MI)	F-BLEAU: R^*
British	yes (0.053)	0.383
German	**no** (0.152)	0.490
Greek	**no** (0.034)	0.462
Irish	yes (0.421)	0.350

Table 7.1: Estimated leakage of European passports.

cols have been corrected since then). The problem was that, upon receiving a message, the e-Passport would first check the Message Authentication Code (MAC), and only *afterwards* verify the nonce – so to assert the message was not replayed. Therefore an attacker, who had previously intercepted a valid message from a legitimate session, could replay the message and detect a difference between the response time of the victim's passport and any other passport; this could be used to track the victim. To avoid such attack, Chothia et al. (2013) proposed to add padding to the response time, and they used leakiEst to measure the effectiveness of such defence.

We compared F-BLEAU and leakiEst on the data with time padding applied, available on the leakiEst webpage.[2] The secret space $\mathcal{S}$ contains answers to the binary question: "is this the same passport?"; the dataset is balanced ($R^\pi = 0.5$).

On continuous data, leakiEst only deals with MI, which is not directly comparable to leakage measures derivable from R^*. However, we can base our comparison on leakiEst's zero-leakage test: indeed, MI is 0 if and only if $R^* = R^\pi$.

For F-BLEAU, we randomly split the data into training (75%) and validation set, and then estimated R^* on the latter; we repeated this for 100 different random initialisation seeds, and averaged the estimates. Table 7.1 reports the results: there are two cases where leakiEst did not find enough evidence of leakage, whereas F-BLEAU indicated the leakage was non-negligible: in the case of the German passport, a Bayes error of 0.49 corresponds to a probability of 0.51 to detect the victim's passport, and for the Greek passport a Bayes error of 0.46 corresponds to a probability of 0.54. This suggests an inaccuracy of the zero-leakage test by leakiEst, possibly due to the small size of the available data.

[2] `www.cs.bham.ac.uk/research/projects/infotools/leakiest/examples/epassports.php`.

Mechanism	ν	leakiEst: Conf? (ME)	F-BLEAU: ME	True ME
B.-Arimoto	2	no* (1.481)	1.479	1.501
	4	no* (2.305)	2.310	2.304
	8	no* (2.738)	2.746	2.738
Geometric	2	**no** (2.585)	1.862	1.988
	4	**no** (2.859)	2.591	2.638
	8	**no** (3.105)	2.983	2.996
Mechanism	ν	leakiEst: Conf? (MI)	F-BLEAU: ME	True ME
Laplacian	2	**no** (1.150)	1.802	1.987
	4	**no** (1.911)	2.550	2.631
	8	**no** (2.401)	2.970	3.003

Table 7.2: Estimated leakage of privacy mechanisms on `Gowalla` data. The leakiEst ME estimate is often much farther from its true value than F-BLEAU's estimate.

7.2 Gowalla dataset

We now compare F-BLEAU with leakiEst on the location privacy mechanisms examined in section 6.1: Blahut-Arimoto, planar Geometric, and planar Laplacian. For the first two mechanisms we compare the estimated values of ME. For the latter this is not possible because the Laplacian is continuous, where leakiEst can only estimate MI.

We run F-BLEAU and leakiEst on the defended datasets, comprising of $n =$ 100K examples. The results are reported in Table 7.2, where "Conf?" indicates whether leakiEst considers having achieved the intended level of confidence, or not. The values between parentheses indicate the leakage estimate that leakiEst reports. On the planar Geometric leakiEst reports "Too small sample size", and indeed its estimate of ME is quite distant from the true ME. F-BLEAU, on the contrary, provides a quite tight bound (recall that F-BLEAU provides a lower bound of the true ME). The situation is similar for the planar Laplacian.

On the Blahut-Arimoto, the situation is more interesting: because of the small number of actual outputs, F-BLEAU and the frequentist approach perform equally well (see also section 6.1), hence we were expecting a similar outcome from leakiEst. This was not the case: for Blahut-Arimoto, leakiEst still reports "Too small sample size". However, we think this is because leakiEst takes into account the number of outputs declared, instead of the actual number generated with the examples. Indeed, its ME estimate is close to ours. Hence this problem should be easy to fix simply by inferring the output space size from the examples (this is the meaning of the “*” in the leakiEst column in Table 7.2).

Part III

Estimation through Features

8 Features and Convergence

So far we estimated the security of a system with learning rules trained and tested *directly* on the example space $\mathcal{O} \times \mathcal{S}$. However, with the goal of speeding up the convergence rate of a rule (i.e., reducing the number of examples required for its convergence), it is common practice in ML to apply a transformation $\Phi : \mathcal{O} \mapsto \mathcal{O}'$ to the objects before passing them to the rule. We call Φ *features*.

In this chapter, we use features to define the notion of (λ, Φ)-security. This notion, although weaker than λ-security, will allow us to tackle problems where the direct estimation approach does not converge.

8.1 Features

Features are high level descriptions of objects, which should help predicting their corresponding secrets. They represent measurements that one can make of reality (e.g., the colour of an object, its sizes, its shape).

More formally, a feature is an algorithm of the form $\phi : \mathcal{O} \mapsto \mathcal{O}^{\phi}$, which extracts from the original object $o \in \mathcal{O}$, containing all the information available, a new object $\phi(o) \in \mathcal{O}^{\phi}$. For instance, if o is a pixel matrix representing an image, and the corresponding secret s indicates whether this image represents a sea or a mountain, ϕ could be the algorithm returning the number of blue pixels in o.

In practice, it is convenient to define a set of features, $\Phi = \{\phi_1, \phi_2, ...\}$. With a slight abuse of notation, we will use $\Phi : \mathcal{O} \mapsto \mathcal{O}'$ to refer to the algorithm that runs each feature on an object o and concatenates the corresponding outputs:

$$o' = \Phi(o) := (\phi_1(o), \phi_2(o), ...) .$$

We call Φ features (or *feature set* or *feature mapping*), and $\mathcal{O}'$ the *feature space*.

8.2 A brief remark

One may wonder why we make features explicit in our formalisation, rather than including them into the learning rule. Indeed, for any learning rule g_n and features Φ, there exists a learning rule $g_n^{\Phi}(o)$ which first performs the mapping $o' = \Phi(o)$ and then runs g_n on input o'. Furthermore, it may be argued that some classifiers, such as neural networks, already embed a feature transformation.

We do this for two reasons. Firstly, we believe that features are such a fundamental aspect of learning that one could base learning theory on them rather than on classifiers; this aspect has been marginalised by most statistical learning literature. Secondly, in the context of Black-box security, it will be convenient to define security measures on the basis of a feature set whenever convergence is not reached via direct estimation. We will illustrate an example of this in chapter 9.

8.3 Problem definition

In section 2.3, we defined the *direct* estimation problem, i.e., estimating the Bayes risk (and, therefore, a security measure λ) directly on examples from $\mathcal{O} \times \mathcal{S}$. We now redefine this problem by taking into account features.

Problem 8.1 (Black-box estimation through features)**.** *Consider a dataset of examples* $\{(o_1, s_1), ..., (o_n, s_n)\}$*, obtained by sampling n times a system* $(\pi, \mathcal{B})$*. We wish to select a feature set* Φ*, and to compute an estimate* $\hat{R^*_\Phi}$ *of the true Bayes risk* R^*_Φ *of the system* $(\pi, \Phi \circ \mathcal{B})$*. Here* $\Phi \circ \mathcal{B}$ *indicates the composition of* $\mathcal{B}$ *and* Φ*, which for input* $o \in \mathcal{O}$ *runs algorithm* $\mathcal{B}$ *and then maps its output into the feature space:* $(\Phi \circ \mathcal{B})(o) := \Phi(\mathcal{B}(o))$*.*

For a certain set of features Φ, and a security measure λ derived from an estimate $\hat{R^*_\Phi}$, we declare a system is (λ, Φ)-secure. As we see in the remainder of this chapter, (λ, Φ)-security is a weaker notion than λ-security, as it only bounds the security against adversaries that use a set of features Φ (or any "less effective" one); if a better set of features is ever discovered, the actual security of the system may be worse. However, this notion is sometimes necessary in real-world applications (e.g., chapter 9).

8.4 Results and discussion

We first state an important but fairly intuitive result: the Bayes risk cannot be improved by a transformation of the object space (i.e., features).

Theorem 8.2 (Optimality of original space (e.g., Devroye et al. (2013))). *Consider a system $(\pi, \mathcal{B})$ inducing a distribution μ on $\mathcal{O} \times \mathcal{S}$. Let R^* be the Bayes risk on $\mathcal{O} \times \mathcal{S}$, and R^*_Φ the Bayes risk on the feature space $\mathcal{O}' \times \mathcal{S} = \Phi(\mathcal{O}) \times \mathcal{S}$. Then:*

$$R^*_\Phi \geq R^* .$$

We do not give a formal proof of this; intuitively, it holds because, for any transformation Φ, $\Phi(o)$ will contain at most as much information as the original object $o \in \mathcal{O}$.

From Theorem 8.2 we should conclude that, theoretically speaking, one should never work in a feature space. However, this is not true in practice: empirically, a UC rule may achieve a much smaller error after feature transformation. This apparently counterintuitive result has to do with the asymptotic guarantees of UC: whilst in the infinite sample a UC rule on the original space $\mathcal{O}$ will converge to R^* (i.e., the smallest error among all feature mappings), in finite sample conditions, the feature mapping may give the rule a faster convergence.[1]

Faster convergence in the feature space may be due to various reasons, which are not easy to categorise. Without aiming for completeness, we give two: i) the feature mapping may discard noisy features, whose utility is only marginal, ii) a learning rule generally performs better for small-dimensional objects (e.g., because of the Curse of Dimensionality).

Remark 1 For the problem of Black-box security, we suggest the analyst should first attempt measuring convergence of a direct estimate. If such estimate does not work, they may select a set of features Φ, and achieve the weaker (λ, Φ)-security notion. It is hard to provide more general guidelines on what problems require the latter approach; however, in real-world situations, it seems that human analysts are fairly good at discerning when this is the case. This supports our conjecture that features, and not classifiers, are at the basis of learning.

Remark 2 A fairly common research pattern in Black-box security applications is to look for feature sets that are *complete*; that is, such that the feature mapping Φ is a bijection (e.g., Cai et al. (2014c)). Whilst in theory this assures $R^*_\Phi = R^*$, in prac-

[1]Clearly, because of the theorem we just stated, the error of a UC rule in the feature space Φ does not necessarily converge to the Bayes risk. However, we refer here to the possibility that, for a limited number of examples, it produces a "closer" estimate to R^* than when working in the original space.

tice there are distributions for which an estimate based on Φ does not converge in the finite sample, whilst a non-complete feature set Φ' makes it converge.

8.5 Features and attack's computational complexity

Many guarantees in Cryptography are based on the computational time complexity of an attack. In Black-box security and QIF, this is usually not taken into account, and security is usually measured with classical definitions of information (e.g., Shannon entropy). Smith (2009) suggested future work may try to include computational aspects in their definitions.

In Cherubin (2017), we proposed a direction for moving from (λ, Φ)-security to λ-security (i.e., to achieve independence from features, even when the direct estimation approach does not converge); along the process, we also introduced a notion of computational complexity for the attacks we consider.

We suggested that, whenever a direct estimate does not converge, one should look for (and estimate security with) an *efficient* feature set Φ^*, which we define as follows. Let R^* be the true Bayes error associated with a system $(\pi, \mathcal{B})$ (i.e., the Bayes risk on full information). An efficient feature set $\Phi^* : \mathcal{O} \mapsto \mathcal{O}'$ is an algorithm for which:

1. $R^*_{\Phi^*} = R^* + \delta$, for some negligible $\delta \geq 0$;
2. the algorithm Φ^* is computationally efficient (e.g., time, memory);
3. an estimate on the feature space $\mathcal{O}' \times \mathcal{S}$ converges within a small number of examples.

Note that, in this case, (λ, Φ^*)-security would be equivalent to λ-security, because of the first requirement.

We believe this could open a new line of research both in Black-box security and ML, which could draw from randomness notions such as Kolmogorov or Levin complexity (Levin, 1973; Ming and Vitányi, 1997; Vovk et al., 2015) to describe the problem of finding efficient feature sets.

9 Application to Traffic Analysis

Traffic Analysis refers to a wide range of attacks, where a passive adversary infers something about the content of encrypted network traffic from its observable characteristics (e.g., packets' size, time and direction). We apply the methods we introduced so far to measure the security of defences against Webpage Fingerprinting[1] (WF), a major class of Traffic Analysis. In WF, an adversary aims to predict which webpage a victim visits, by only looking at the encrypted network traffic she produces.

This chapter develops as follows. In section 9.1 we introduce WF attacks, and in section 9.2 we apply the NN bound (Theorem 3.6) to measure the (β, Φ)-security of WF defences, where β is the Bayes security measure we introduced in chapter 4, and (β, Φ) denotes estimation of β through features (chapter 8). This application, which appeared in Cherubin (2017), was the first method that could provably measure the security of any WF defence in a black-box manner – and indeed the first use of the main idea in this manuscript. In section 9.3, we introduce a practical WF defence (ALPaCA), which appeared in Cherubin et al. (2017); this defence will allow us to make some deeper remarks on features in section 9.4, where we conjecture on the possibility of estimating its security directly (i.e., not through features). In chapter 10, we will suggest the possibility of applying the same approach to other traffic analysis attacks.

9.1 Webpage Fingerprinting

We presently provide an informal description of WF attacks, which we will formalise in the next section.

[1]Most literature in the past 15 years referred to this problem as Website Fingerprinting. While many of them acknowledged such terminology is incorrect, the common use prevailed so far. However, given the raise of studies which do proper *Website* Fingerprinting (i.e., where the adversary aims to predict the website and not the webpage a victim visits), there is a real need to start using the proper terminology. In this work, we will commit to the "Webpage Fingerprinting" term, here abbreviated to WF, as opposed to Website Fingerprinting (WsF).

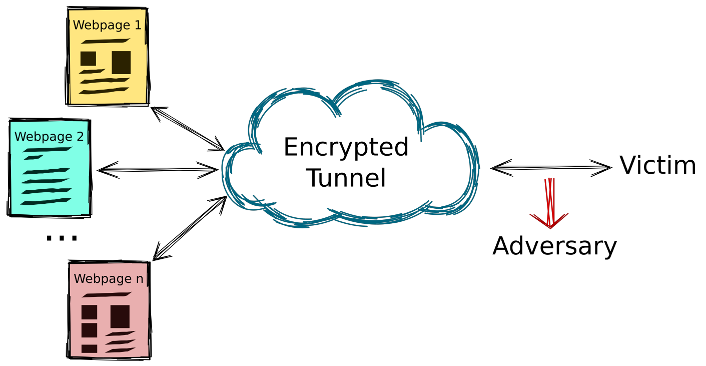

Figure 9.1: In the Closed World scenario of WF, the victim visits one page among those monitored by the adversary, which the adversary has to guess.

9.1.1 Background

In WF attacks, a local passive adversary observes the encrypted network traffic generated by a victim, while she loads a webpage via an *encrypted tunnel*. An encrypted tunnel represents a communication channel between the victim and the web server such that:

1. all the traffic is encrypted;
2. an observer who is local to the victim (e.g., her ISP) knows her IP address, but not that of the web server.

Examples of encrypted tunnels are VPNs, SSH tunnels, and anonymity networks (e.g. Tor). However, variants of WF have been used against encrypted wireless networks; for example, British broadcaster BBC allegedly used similar techniques to detect if a user without TV licence was using BBC's online service *iPlayer* (Foster, 2016). The natural candidate for the attacks and defences described in this chapter is the Tor network, which anonymises the traffic coming from a victim by re-routing it via a number of relays (usually 3); understanding how the Tor network works is not required to follow this section, and we refer the interested reader to the seminal paper on Tor by Syverson et al. (2004).

To avoid trivial solutions, we assume that the adversary cannot decrypt the packets. However, when the victim uses a WF defence mechanism (e.g., adds noise to her network traffic), we will assume the adversary knows what kind of mechanism she uses.

In WF, the adversary selects a set of webpages (henceforth called *monitored* pages), and his goal is to predict which one of them the victim visits, if any. De-

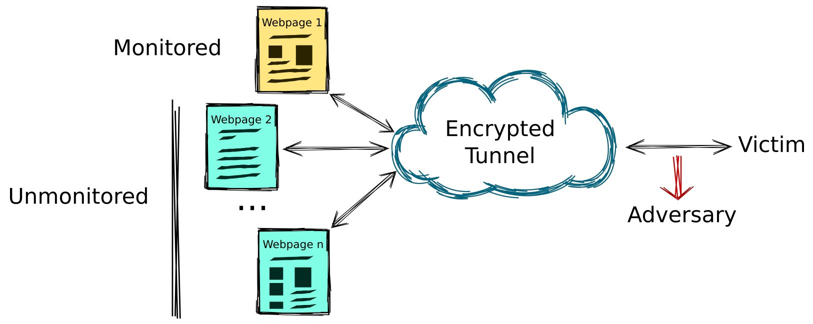

Figure 9.2: In One VS All, a special case of the Open World scenario, the WF adversary only monitors one page, and the victim can visit a much larger set of *unmonitored* pages.

pending on the choice of monitored webpages, and on the assumptions on which webpages the victim may visit, we formulate the following scenarios.

Closed World The victim is only allowed to visit one of the monitored pages (Figure 9.1). This scenario is a simplification of real-world attacks, but because it allows an adversary to perform the attack in nearly-perfect conditions, it is generally used for evaluating the security of defences.

Open World This scenario resembles more closely a real-world setting, where the victim can visit both monitored and unmonitored pages. It is therefore typically used for evaluating attacks, and their scalability to the real world. In addition to the Closed World scenario, we will evaluate defences under the One VS All scenario, a special case of Open World, which indicates the ability of a defence to protect an individual page (Figure 9.2). In the One VS All scenario, the adversary only monitors one webpage, and the victim can either visit that page, or one of the *unmonitored* pages; for this reason, this scenario can be seen as an extreme case of the Open World scenario.

A WF attack has two phases: *training* and *attack*. In the training phase, the adversary generates defended network traffic on his computer, by loading monitored pages several times, and applying the same defence mechanism as the target victim; then, he extracts features from this traffic, and trains an ML classifier on them. This phase can be performed offline. Features are high level descriptions of data (network traffic, in this case), as we introduced in chapter 8. In this section, we will show experimentally that, in general, we can only measure the security of WF defences through features (i.e., not directly), thereby obtaining (λ, Φ)-security for some security measure λ; for this reason, we will argue that features are the major next step for future WF research (section 9.2). In the attack phase, the adversary collects defended network traffic coming from the victim. He extracts features from the new traffic, and uses the ML classifier to predict which webpage the victim loaded.

We shall now describe the major WF attacks and defences, and previous directions in provable evaluation of defences

9.1.2 Major defences

WF defences can be divided into *deterministic*, which deterministically transform the characteristics of traffic, and *probabilistic* (also called elsewhere *random*), where randomness is involved in the morphing process. We will consider six defences for evaluation: two deterministic (BuFLO and Tamaraw), four probabilistic (Randomised Pipelining, Decoy Pages, CS-BuFLO, WTF-PAD).

Randomised Pipelining (RP) randomises the order of HTTP requests in the HTTP pipeline. This defence is embedded by default in the Tor browser. Since in experiments we will defend network traffic coming from the Tor browser, we will implicitly assume RP is underlying to any of the defences we shall now present; consequently, "No Defence" will refer to plain Tor traffic (i.e., only defended using RP) (Perry, 2011).

BuFLO For parameters (d, ρ, τ), BuFLO sends packets of fixed size d, with frequency ρ, for at least τ time. If more time is required by the page load, packets of size d are sent with frequency ρ for the time needed (Dyer et al., 2012).

Tamaraw Similarly to BuFLO, it sends fixed-size packets at fixed intervals. It considers distinct intervals between packets with respect to their direction: outgoing packets are sent every ρ_{out} seconds, incoming packets are sent every ρ_{in} seconds, and $\rho_{out} > \rho_{in}$. It further pads the number of packets sent in both directions by multiples of a padding parameter (Cai et al., 2014c).

Decoy Pages Loads a randomly chosen page in the background together with the requested page. The traffic introduced by the background page should confuse features the adversary. It is a probabilistic defence, and it is arguably the easiest to implement among the defences presented here (Panchenko et al., 2011).

CS-BuFLO is a modification of BuFLO, which reduces overheads and eases implementation. As with BuFLO, it sends packets of size d with frequency ρ. However, the value of ρ is dynamically adapted to the network bandwidth, and events like client's `onLoad` (i.e., the browser finished loading a page) and `channel_idle` are used to determine the end of communication. Padding is also inserted at the end of communication, up to a certain transmission size (Cai et al., 2014a).

WTF-PAD is based on Adaptive Padding (AP) (Shmatikov and Wang, 2006). At each endpoint (client, server), the defence sends real or dummy packets according to the state (idle, burst, gap) of an internal state machine. This allows adaptively morphing traffic to avoid large overheads (Juarez et al., 2016). This defence is relatively easy to implement, although it is currently not well understood how to optimally determine the distributions that regulate transitions of the internal state machines; in experiments we used the distributions by Pulls (2016a).

Designing a WF defence requires: i) understanding of how WF attacks work and what they target, and ii) engineering the defence on the top of other network protocols. In the past, probabilistic defences attracted the attention of researchers because they are often easier to implement, and they tend to introduce less overheads in terms of time and bandwidth. However, before our work there was no generic method for evaluating their security. More in general, there seems to be a pattern in literature where WF defences that are secure by construction are impractical to engineer, while the security of defences that are easier to engineer cannot be provably evaluated. With this work, we hope to fill this gap, by allowing to measure the security of defences in a black-box setting.

9.1.3 Major attacks

In the context of WF, where attacks are usually based on ML methods, we will estimate security measures conservatively, by using the NN bound by Cover and Hart (1967) (section 3.1). To verify the convergence of our estimates in this setting, we will compare them with the error of previous WF attacks, and assert they are a lower bound in practice.

Rank	Feature Description
1	# incoming packets
2	# outgoing packets (fraction of tot # packets)
3	# incoming packets (fraction of tot # packets)
4	Standard deviation of outgoing packet ordering list
5	# outgoing packets
6	Sum of items in the alternative concentration feature list
7	Average of outgoing packet ordering list
8	Sum of incoming, outgoing, and total # of packets
9	Sum of alternative # packets per second
10	Tot # packets
11-18	Packet concentration and ordering feature list
19	# incoming packets stats in first 30 packets
20	# outgoing packets stats in first 30 packets

Table 9.1: Best 20 features according to the feature analysis by Hayes and Danezis (Hayes and Danezis, 2016). Each feature is extracted from the network trace corresponding to a webpage load, and features are concatenated to form objects o.

A WF attack is composed of two choices: a feature set and an ML classifier. In our evaluation we will consider 5 attacks: LL, VNG++, k-NN, CUMUL, and k-FP; we omit here their description (i.e., the enumeration of their feature set and classifier), which is not central to our evaluation, and we address the interested reader to Appendix 3.

In Table 9.1, we report the 20 most important features in the attacks up to 2017, according to a study by Hayes and Danezis (2016); in this work, they analysed features used for previous attacks, and proposed an attack, k-FP, based on them.

9.1.4 Previous directions in provable evaluation of WF defences

Cai et al. (2014c) Proposed a method to compute a lower bound of error for WF adversaries. Let us call *packet sequence* a sequence specifying size, time, and direction of each packet within the network trace corresponding to a webpage load; note that, under the threat model we will formulate, this constitutes the only information available to an adversary when the victim loads a webpage. Cai et al. (2014c) considered an idealised "lookup-table" adversary, who knows exactly what network traffic each webpage can produce. This adversary creates a look-up table $T(p) = \{w_1, w_2, ...\}$, which associates each packet sequence p of network traffic to the set of webpages w_i that can produce that traffic (Figure 9.3). The packet sequences of two pages are considered to be the same if all their packets have the same size, direction and time. The error of such an adversary is computed by counting the collisions (i.e., how many pages produce the same traffic), and it is the smallest error achievable by any WF adversary.

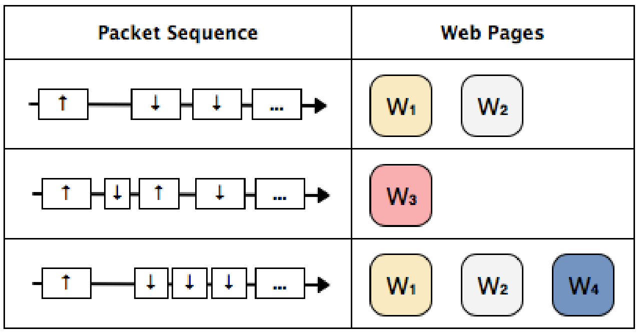

Figure 9.3: The idealised lookup-table adversary proposed by Cai et al. knows exactly what network traffic each webpage ($w_1, w_2, ...$) may produce, and only commits an error when many pages exhibit identical traffic.

Observe that this methodology is equivalent to the frequentist approach we described in subsection 3.1.1, and it has analogous drawbacks. The smallest change in the quantities observed by the adversary (e.g., time, total bandwidth), which is likely to happen in network communications because of noise, leads similar traffic to be misclassified by the adversary. Furthermore, the accuracy of such an adversary can only be estimated for a defence if: i) the defence is deterministic, and ii) we know how it is constructed internally. The second requirement is of particular interest: if we were to blindly apply this method to some defence, the results would largely underestimate both probabilistic and deterministic defences, because of noise in network data. Indeed, Cai et al. only applied this method to defences for which, by design, only some characteristics of the traffic (e.g. the total transmission size) could change; even in this case, however, they had to apply the method to partial information (e.g., they excluded timing information), as it would have otherwise returned unreasonable bounds.

Wang and Goldberg (2015) Proposed an extension of the method by Cai et al. for probabilistic defences. This method attempts to approximate the probability distribution of network traffic defended by a defence, and to derive from it the smallest error achievable. However, to estimate such distribution i) they use a look-up table strategy, which is highly susceptible to noise, and ii) they compute it by running the defence a finite number of times. Because of these issues, the estimated distribution is not guaranteed to approximate the real distribution, and the method does not necessarily guarantee valid bounds.

9.2 Measuring the security of WF defences

In this section, we give a formal description of WF, we show how the NN bound described in section 3.1 can be used to lower bound an adversary's error, and we evaluate our findings on real-world data.

9.2.1 WF formulation

$\mathcal{W}$ is a set containing all possible webpages. An adversary selects a subset of such pages, $\mathcal{W}_m$, which we call monitored webpages. We define a set of unmonitored webpages, $\mathcal{W}_u$, such that $\mathcal{W}_m \cap \mathcal{W}_u = \emptyset$ and $\mathcal{W}_m \cup \mathcal{W}_u \subseteq \mathcal{W}$. The victim can visit any page in $\mathcal{W}_m \cup \mathcal{W}_u$. It is natural to assume that $|\mathcal{W}_m| > 0$, and $|\mathcal{W}_m \cup \mathcal{W}_u| > 1$. We talk about *Closed World* scenario when $\mathcal{W}_u = \emptyset$. *Open World* scenario is when $|\mathcal{W}_u| > 0$. We also define the *One VS All* scenario, a subclass of Open World, where the adversary only monitors one webpage: $|\mathcal{W}_m| = 1$ and $|\mathcal{W}_u| > 0$.

A packet sequence $p \in \mathcal{P}$ is a finite list of arrival time τ_j, size σ_j, and direction $\delta_j \in \{\uparrow, \downarrow\}$ of packets:

$$p = (\tau_1, \sigma_1, \delta_1), (\tau_2, \sigma_2, \delta_2), \ldots,$$

with, $\tau_1 = 0$ and $\tau_{j+1} \geq \tau_j$, $\sigma_j \in (0, MTU]$, where MTU (maximum transmission unit) is 1500. As with previous work, we assume that a WF adversary, when observing the encrypted network traffic corresponding to a page load, gets a packet sequence as the only information to make his prediction (e.g., Cai et al. (2014c)).

A label (or *secret*) $s \in \mathcal{S}$ is associated with a packet sequence p according to the webpage $w \in \mathcal{W}$ that produced it. Specifically,

$$s(w) = \begin{cases} w & \text{if } w \in \mathcal{W}_m \\ \odot & \text{otherwise}\,; \end{cases}$$

in other words, symbol $\odot$ represents a page that is not monitored.

A WF defence is a (possibly randomised) algorithm $D : \mathcal{P} \mapsto \mathcal{P}$ that takes as input a packet sequence, and returns a defended packet sequence. Note that D also includes the trivial "no-defence" algorithm, which simply outputs its input.

A WF adversary is a pair $\mathcal{A} = (\Phi, \mathrm{T})$ of a feature set Φ and an ML training algorithm T. A feature set is a list of algorithms:

$$\Phi = (\phi_1, \phi_2, ..., \phi_Q)\,,$$

with $\phi_q : \mathcal{P} \mapsto \mathbb{R}^{d_q}$ and $d_q > 0$ for $q = 1, 2, ..., Q$; each ϕ_q gets as input a packet sequence, and returns a vector of d_q real values. With a slight abuse of notation, we will use Φ to refer to the algorithm:

$$\Phi : \mathcal{P} \mapsto \mathcal{O} \quad ,$$

which runs each ϕ_q on an input packet sequence p, and concatenates the resulting vectors of values:

$$\Phi(p) = (\phi_1(p), ..., \phi_Q(p)) \quad ;$$

We call $\mathcal{O} = \mathbb{R}^d$ the object space, where $d = \sum_{q=1}^{Q} d_q$, and we call $x = \Phi(p)$ an object. An ML training algorithm $\mathrm{T} : (\mathcal{O} \times \mathcal{S})^* \mapsto \mathcal{G}$ is an algorithm that accepts an arbitrary number of pairs of objects and respective labels, and returns a classifier $g \in \mathcal{G}$ from the collection $\mathcal{G} = \{g \mid g : \mathcal{O} \mapsto \mathcal{S}\}$. A training algorithm T is class-specific, in the sense that it returns classifiers of a specific class (e.g., logistic regression, SVM).

9.2.2 WF attacks

We shall now describe a WF attack. Consider the case where a victim uses defence D to protect her traffic. In the training phase, a WF adversary $\mathcal{A} = (\Phi, \mathrm{T})$ generates n pairs of packet sequences, corresponding to his choice of webpages with $s_i \in \mathcal{W}_m$, defended using defence D:

$$(p'_i, s_i)_{i=1}^{n} = (D(p_i), s_i)_{i=1}^{n}\,.$$

The adversary extracts objects from the packet sequences, obtaining a training set:

$$Z_{train} = (o_i, s_i)_{i=1}^{n} = (\Phi(p'_i), s_i)_{i=1}^{n}\,.$$

Then, he trains a classifier on the training set: $g = \mathrm{T}(Z_{train})$.

In the attack phase, the victim visits a webpage with label $s_{n+1} \in \mathcal{S}$, using defence D, which produces a defended packet sequence p'_{n+1}. The adversary eavesdrops this packet sequence, extracts a test object $o_{n+1} = \Phi(p'_{n+1})$, and outputs a prediction $\hat{s}_{n+1} = f(o_{n+1})$.

The adversary is evaluated in terms of the probability that classifier g misclassifies the test object x_{n+1}:

$$R^{\mathcal{A}} = P(g(o_{n+1}) \neq s_{n+1})\,.$$

In practice, this probability is estimated using the empirical error $\hat{R}^{\mathcal{A}}$, by averaging the error committed by the adversary over a validation set of n_{val} pairs $Z_{val} = \{(o_{n+1}, s_{n+1}), ..., (o_{n+n_{val}}, s_{n+n_{val}})\}$ as follows (section 3.2):

$$\hat{R}^{\mathcal{A}} = \frac{1}{n_{val}} \sum_{i=n+1}^{n+n_{val}} I(g(o_i) \neq s_i)\,,$$

where I is the indicator function.

9.2.3 Assumptions

We will make the standard i.i.d. assumption on $(o_1, s_1), ..., (o_n, s_n), (o_{n+1}, s_{n+1})$; that is, examples are independently sampled from some (unknown) distribution. In other words, we expect that the traffic produced by each webpage follows the same distribution. Violations of this assumption (e.g., the adversary trains on a different version of the webpage) would generally be disadvantageous for an adversary (Juárez et al., 2014). In section 11.4, we discuss a method to verify the i.i.d. assumption. We make no further assumption on the underlying distribution.

9.2.4 Application of the NN bound

We use the NN bound $\hat{R}^*$ introduced in subsection 3.1.6 to bound a WF adversary, and then define (β, Φ)-security on its basis.

From chapter 8, we can show that the error of a WF adversary using feature set Φ is asymptotically lower-bounded by the NN bound estimate computed for the same features.

Theorem 9.1 (Lower bound of WF adversary's error (Cherubin, 2017))**.** *Fix a feature set Φ. Sample $Z = \{(o_1, s_1), ..., (o_{n+1}, s_{n+1})\}$ i.i.d. from a distribution on $\mathcal{O} \times \mathcal{S}$, where o_i's take values from a separable metric space on $\mathcal{O}$; let $Z_{train} = \{(o_1, s_1), ..., (o_n, s_n)\}$.*

For any training algorithm $\mathrm{T} \in \mathcal{T}$, where $\mathcal{T} = \{\mathrm{T} \mid \mathrm{T} : (\mathcal{O} \times \mathcal{S})^* \mapsto \mathcal{G}\}$ selecting classifiers from $\mathcal{G} = \{g \mid g : \mathcal{O} \mapsto \mathcal{S}\}$, consider an attack $\mathcal{A} = (\Phi, \mathrm{T})$, and let $R_n^{\mathcal{A}}$ be *the probability that the selected classifier, $g = \mathrm{T}(Z_{train})$, misclassifies a test object o_{n+1}; let R_n^{NN} be the expected error of the NN classifier trained on Z_{train} at predicting o_{n+1}, and let $\hat{R}_n^*$ be the NN bound computed from R_n^{NN} as in Theorem 3.6. Then, as $n \to \infty$:*

$$\hat{R}_n^* \leq R^* \leq R_n^{\mathcal{A}} \quad ,$$

where R^* is the error of the Bayes classifier, which has perfect knowledge of the prior probabilities and underlying distribution from which Z was sampled.

Proof. The second inequality, $R^* \leq R_n^{\mathcal{A}}$, follows trivially from the fact that the Bayes classifier commits the smallest error among the collection of classifiers $\mathcal{G} = \{g \mid \mathcal{O} \mapsto \mathcal{S}\}$, and any training algorithm $\mathrm{T} \in \mathcal{T}$ returns a model $g \in \mathcal{G}$.

The first inequality, $\hat{R}_n^* \leq R^*$, holds because of Theorem 3.6. □

This guarantee is independent of the NN's distance metric. In section 9.2.6, we compare different metrics, and select Euclidean for the security evaluation.

In practice, we need to compute the NN bound on a finite dataset. In subsection 9.2.6, we show empirically that the lower bound estimate, computed for a certain set of features, is always smaller than the respective attack error.

9.2.5 Methodology

Experiments in this section aim at i) verifying the validity of the NN bound estimate $\hat{R}^*$ empirically, ii) measuring the evolution of feature sets in previous WF attacks, and iii) measuring the (β, Φ)-security of WF defences. We conducted experiments for the most influential attacks and defences on the dataset by Wang et al. (2014). Henceforth, we refer to this dataset as `WCN+`.

Dataset The `WCN+` dataset is a collection of network traces produced by visiting, over Tor, 100 unique webpages 90 times each. Because in Tor traffic packets have a fixed size, attacks using size of packets as a feature may be penalised.

Background noise The security bounds of a WF defence should be determined under optimal conditions for the adversary. Any noise in the data (e.g., background webpages or BitTorrent clients) would produce an overestimate of the defence's security. For this reason, we did not add further noise to the dataset.

Distribution of the labels in data   We designed the experiments so that training and validation sets would have a uniform distribution of labels. This prevents an unbalance towards webpages that are easier to defend, which would cause security overestimates.

Scenarios   We performed experiments in the Closed World and One VS All scenarios. The former is of common use when evaluating defences, and it indicates the ability of an adversary to distinguish many pages in nearly-perfect conditions. One VS All is an extreme case of Open World scenario, in which an adversary only targets one webpage. This scenario highlights how well a defence protects unique pages. We remark that our method is applicable to any other case of Open World.

Resources

The `WCN+` dataset can be found at:

`https://cs.uwaterloo.ca/~t55wang`.

The code of the major WF attacks and defences is openly available; it is usually written in `Python` or `C`. We acknowledge the use of the following:

- T. Wang: Decoy Pages, BuFLO, Tamaraw, k-NN[a]
- K. P. Dyer: VNG++, LL[b]
- J. Hayes: k-FP[c]
- M. Juarez: WTF-PAD.[d]

We created a `Python` framework to abstract the different APIs employed by researchers, with the goal of unifying the way experiments are carried out in WF research. We implemented CS-BuFLO, and the CUMUL attack, and adapted other researchers' code to the API. We created routines to compute error bounds and $(\beta,\Phi)-$*security* of WF defences as they are described in this manuscript. This code and links to the data can be found at:

`https://github.com/gchers/wfes`.

[a]`https://cs.uwaterloo.ca/~t55wang/wf.html`.
[b]`https://github.com/kpdyer/website-fingerprinting`.
[c]`https://github.com/jhayes14/k-FP`.
[d]`https://github.com/wtfpad/wtfpad`.

Distance	$\hat{R}^*$ (%)	Feature Set
Euclidean	5.8	k-NN
Standardised Euclidean	6.0	CUMUL
City Block	6.6	k-NN
Levenshtein	12.2	packets' sizes

Table 9.2: Comparison of the NN lower bound $\hat{R}^*$ obtained using different distance metrics. Experiments were performed against No Defence (RP), on the WCN+ dataset. In this table, the name of an attack refers to its feature set.

9.2.6 Evaluation through features

This section justifies empirically the use of the NN bound and of the respective (β, Φ)-security measure. Specifically, it shows that: i) the NN lower bound estimate $\hat{R}^*$ is indeed a lower bound for the error of the major attacks, and ii) recent WF attacks did not manage to improve their feature sets significantly (if at all), which suggests it is safe to use a security measure that depends on a feature set.

What Distance Metric The lower bound estimate $\hat{R}^*$ depends on a distance metric. Whilst its guarantee holds regardless this choice, different metrics may perform better on our data.

We experimented with four distance metrics: Euclidean, Standardised Euclidean, City Block and Levenshtein. The first three distances were applied to feature sets; the estimate corresponding to the best performing feature set is shown. We applied Levenshtein distance directly to packet sequences as follows. Given a packet sequence $((\tau_j, \sigma_j, \delta_j))$, we created a binary string containing 0 when $\delta_j = \uparrow$, and 1 when $\delta_j = \downarrow$; this makes sense in the context of Tor traffic, where packets' sizes are the same; by doing so we excluded timing information. We then computed the NN-based bound on the binary strings using Levenshtein as a distance metric.

Results in Table 9.2 indicate the distance metric not impact heavily on the bound. Levenshtein distance, however, performed poorly; this may be due to the fact that it does not consider timing information. In the following experiments we use Euclidean distance.

Empirical validity of the bound Under the sole i.i.d. assumption, it is not possible to prove convergence rates of a Bayes error estimate (section 2.6). We studied the convergence of $\hat{R}^*$ in finite sample conditions experimentally, for an

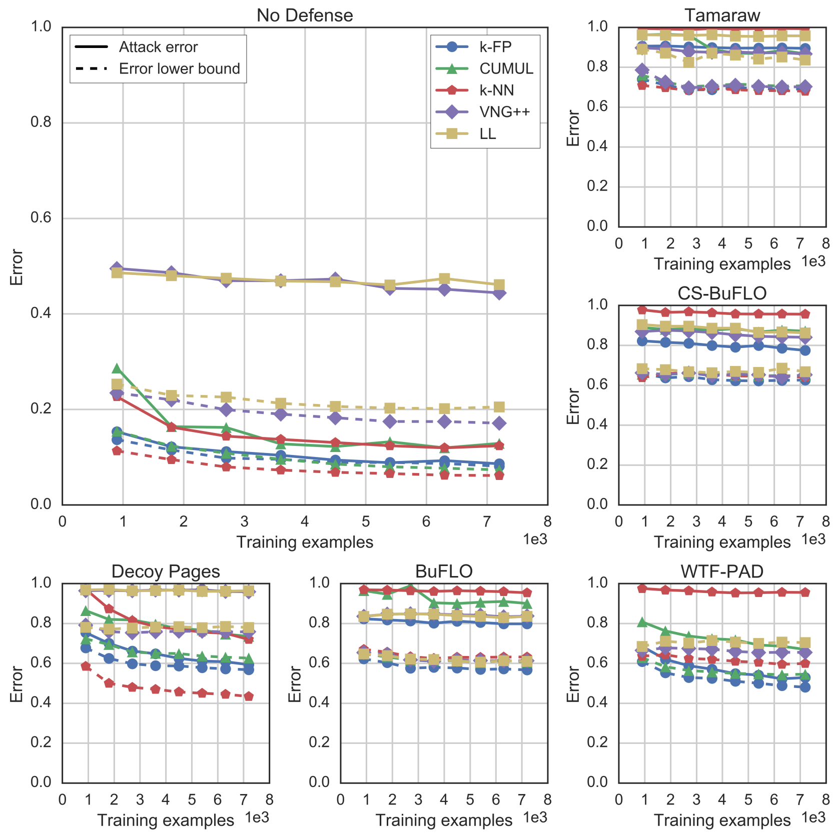

Figure 9.4: Validity of the lower bound $\hat{R}^*$ on the `WCN+` dataset (Closed World scenario), with respect to an increasing number of training examples. The lower bound estimates $\hat{R}^*$ (dashed lines) are smaller than the error of an attack (continuous lines) performed using the same feature set (same colour).

increasing number of training examples. We used the following heuristics to verify the estimates' convergence:

- verify that $\hat{R}^*$ computed for a feature set is lower than the error of an attack performed with the same feature set;
- visually inspect that the decreasing trend of the estimate becomes slower.

We performed these experiments in a Closed World scenario on the `WCN+` dataset, which is composed of $n_{tot} = 9000$ network traces from 100 distinct pages. We kept a fixed validation set of size $n_{val} = 0.2 \times n_{tot}$, and iteratively increased the number of traces of the training set: $n = \{0.1 \times n_{tot}, ..., 0.8 \times n_{tot}\}$. To show that the bound is valid for adversaries predicting new data, we computed the bound only on the training set, and compared it with the attack error on the validation set. More specifically, for an adversary $\mathcal{A} = (\Phi, \mathrm{T})$, we computed the attack error $R^{\mathcal{A}}$ by training the attack's classifier on the training set, and predicting the validation set. We computed on the training set the lower bound $\hat{R}^*$, for the same feature set Φ, using 5-folds cross validation (CV) to reduce the variance.

Figure 9.4 presents the results of these experiments. Each colour represents a set of features; continuous lines indicate the error of an attack, and dashed lines error bounds. The figure shows that, for any size of the training set, the bound estimate computed for a feature set is a lower bound of the error committed by an attack using the same features. Furthermore, visually, the decreasing trend of the estimates seems to converge towards an asymptote, with some exceptions; e.g., the NN bound for defence Decoy Pages, computed for the feature set of the "k-NN" attack by Wang et al. (2014), seems to have not reached convergence.

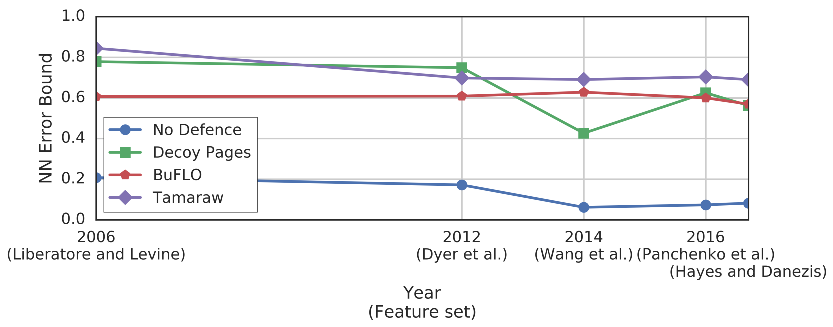

Figure 9.5: NN lower bound of R^* when using the feature sets of the major attacks up to 2017, against the most influential WF defences. Feature sets have not improved substantially since the attack by Wang et al. (2014). Bounds were computed on the `WCN+` dataset in a Closed World scenario.

Evolution of feature sets The Bayes error also indicates the smallest error achievable with a certain feature set. We use $\hat{R}^*$ to compare the feature sets of major WF attacks, and address the question: did WF attacks' feature sets improve?

We computed the estimates in a Closed World scenario, on the full `WCN+` dataset, using 5-folds CV. Figure 9.5 shows the evolution of feature sets in the major attacks from 2006 to 2016. We observe that, against most defences, the feature set used by the 2014 attack k-NN (Wang et al., 2014) performs nearly as well as the one employed by the current state-of-the-art attack, k-FP (Hayes and Danezis, 2016).

These results support the following hypothesis: improving feature sets is becoming harder, and we might reach a limit for their improvement soon. This strengthens the value of a security measure depending on a feature set.

9.2.7 Privacy evaluation of WF defences

We measured (β, Φ)-security and overheads of the defences, under Closed World and One VS All scenarios; for the estimates, we used the entire `WCN+` dataset

We computed the packet overhead Oh_D^b for each packet sequence p and a defence D as:

$$Oh_D^b = \left(\frac{|D(p)|}{|p|} - 1 \right) \times 100\,,$$

where $|p|$ indicates the number of packets in packet sequence p. We computed the time overhead Oh_D^t by dividing the difference $t_\ell - t_1$ for packets in packet sequence p by the same computed for packets in p'. We report the median overhead among packet sequences for both Oh^b and Oh^t.

For the Closed World scenario, we considered all the dataset, extracted the features, and computed $\hat{R}^*$ (Theorem 3.6) using 5-folds CV. We then determined (β, Φ)-security. Random guessing error in this scenario ($|\mathcal{S}| = 100$) was $R^\pi = 0.99$.

In the One VS All case, for each potential target $w \in \mathcal{W}$, we took all 90 examples from webpage w, and then randomly selected 90 examples from other pages, which we labelled $\odot$. Finally, we computed the bounds in this binary classification setting, where random guessing error was 0.5. Since the method produced a bound for each page, indicating its fingerprintability, we averaged these results. In this scenario, we observed a high variance, which is due to the heterogeneity of the pages. We also considered taking the minimum of the bounds as an alternative to averaging. Unfortunately, to date no defence is able to achieve a minimum bound better than 2.9% (Tamaraw). Future work may consider alternative methods to combine error bounds in this scenario.

Table 9.3 shows the results in both scenarios. Results indicate that the best defence for the Closed World scenario is Tamaraw. However, we notice that BuFLO and Decoy Pages outperform Tamaraw in the One VS All scenario. This suggests that Decoy Pages, a very simple defence to implement, could be a safe choice for defending individual pages from WF attacks. Results also confirm that Tor Browser's default defence, RP, is ineffective, as it was suggested by previous work (Cai et al., 2012; Wang and Goldberg, 2013; Juárez et al., 2014).

9.2.8 Direct evaluation

We now attempt to achieve β-security, rather than (β, Φ)-security, by computing the estimates directly on packet sequences (i.e., without feature extraction).

	Closed World		One VS All		Overhead	
Defence	$\hat{R}^*$ (%)	(β, Φ)	$\hat{R}^*$ (%)	(β, Φ)	Bw (%)	T (%)
No Defence	6.2 ± 0.3	(0.06, W+)	2.5 ± 1.7	(0.05, W+)	0.0	0.0
Decoy Pages	42.6 ± 0.6	(0.43, W+)	**14.7 ± 5.5**	**(0.29, W+)**	134.4	59.0
BuFLO	56.9 ± 1.0	(0.58, HD)	**14.4 ± 13.7**	**(0.29, HD)**	110.1	79.1
Tamaraw	**69.0 ± 0.9**	**(0.70, W+)**	12.7 ± 5.8	(0.25, W+)	257.6	341.4
CS-BuFLO	61.9 ± 0.9	(0.63, HD)	8.1 ± 3.5	(0.16, HD)	67.2	575.6
WTF-PAD	48.6 ± 1.2	(0.49, HD)	9.0 ± 3.8	(0.18, P+)	247.0	0.0

Table 9.3: Bayes lower bounds $\hat{R}^*$ and (β, Φ)-security of WF defences on the WCN+ dataset. A lower bound $\hat{R}^*$ is the smallest error an adversary achieves in a specific scenario (Closed World, One VS All). (β, Φ)-security indicates how far a defence is from perfection (achieved for $\beta = 1$). In bold, the most (β, Φ)-secure defences. We use an attack authors' initials to indicate the attack's feature set Φ (i.e., W+: Wang et al. (2014), P+: Panchenko et al. (2016), HD: Hayes and Danezis (2016)).

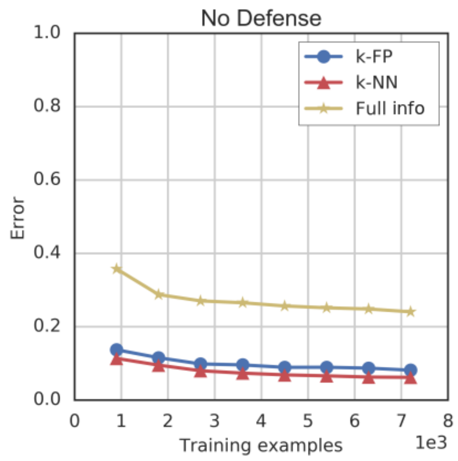

Figure 9.6: Comparison of lower bounds $\hat{R}^*$ computed using full information (i.e., in the packet sequence space) and those computed using the feature sets of the best attacks (k-NN, k-FP). Bounds computed on full information do not converge rapidly enough, and are invalidated by those using feature sets.

We concatenate time and size of each packet sequence to form a vector, and pad such vector to the same length. We then use Euclidean distance in this space to estimate the NN bound.

Unfortunately, the estimates obtained on full information are worse than those achieved using feature sets (Figure 9.6). This result is not too surprising: in WF attacks, classifiers tend to converge more rapidly in the feature space than in the original space. This shows that Black-box security, when applied to generic WF

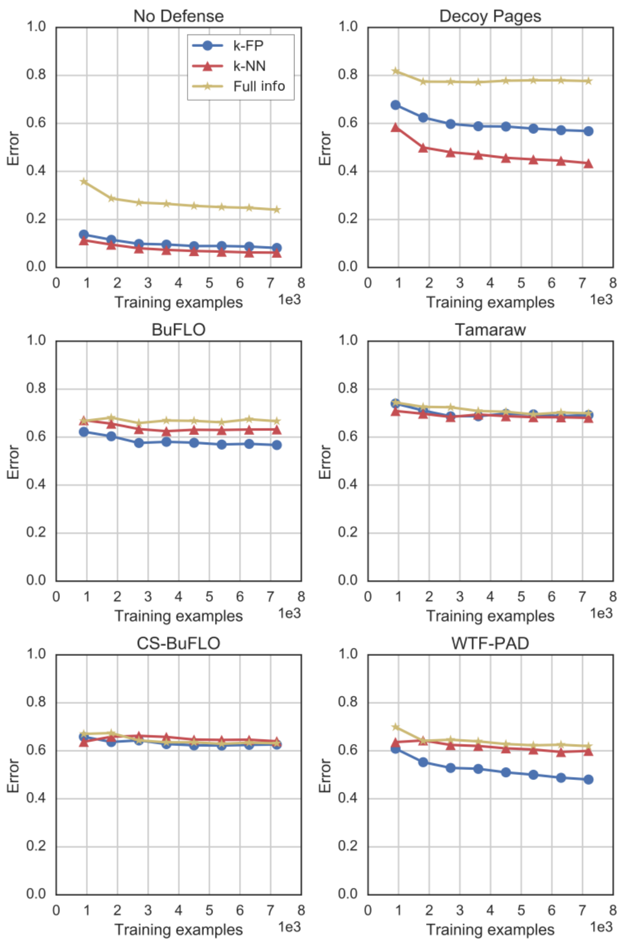

Figure 9.7: Comparison of lower bounds $\hat{R}^*$ computed using full information (i.e., packet sequences) and those computed using the feature sets from the best attacks (k-NN, k-FP) on the `WCN+` dataset. Results show that bounds using full information typically do not converge quickly enough.

defences, should rely on feature sets. However, we notice that, for some defences (Tamaraw and CS-BuFLO), the bounds computed on full information are close to the error of the best classifiers (Figure 9.7). This suggests their bounds may be valid independently of the features.

Future research may also find a better performing distance metric, which might favour a quicker convergence of the Bayes error estimate directly computed in the packet sequence space.

Defence	$\hat{R}^*$ (%)	Cai et al. (%)	Cai et al. - full (%)
BuFLO	57	53	19
Tamaraw	69	91	11

Table 9.4: Comparison of the bound for deterministic defences suggested by Cai et al. and the lower bound estimate $\hat{R}^*$. The last column indicates the bound by Cai et al. on full information, which indicates its susceptibility to noise.

9.2.9 Comparison with previous evaluation

We computed the bound based on the frequentist approach that was proposed by Cai et al. (2014c), on the full `WCN+` dataset. As discussed in subsection 9.1.4, this method is only applicable to deterministic defences, and it needs to be adapted to the defence.

We estimated the bound by Cai et al. for defences BuFLO and Tamaraw. With the goal of reducing noise, their method requires limiting the information available to an adversary: against BuFLO, an adversary can only observe the transmission size; against Tamaraw, the number of incoming and outgoing packets.[2]

Table 9.4 compares the bound by Cai et al. with $\hat{R}^*$. Results do not exhibit any particular trend: their bound is larger than $\hat{R}^*$ for Tamaraw, but smaller for BuFLO. We notice, however, that for Tamaraw their error bound is higher than the error of the best performing attack (86.6%, VNG++) (Figure 9.4). This may be due to noise in the data.

To better grasp the impact of noise on their bound, we estimated it using all the information available to a real-world adversary (Table 9.4). Interestingly, in this case the error bound for Tamaraw was smaller than the one for BuFLO. This indicates that the lookup-table strategy is strongly affected by noise, which suggests that it may not be reliable for evaluating security for traffic analysis.

Discussion

We presented the first method for measuring the security of generic WF defences. Unfortunately, we could only measure the weaker notion of security, (β, Φ)-security (i.e., estimated through features). In the next section, we introduce ALPaCA, a defence for which we conjecture we can measure β-security directly.

[2]The reason for this comes from the design of these defence, which provably hide the remaining information. However, this means evaluation by Cai et al. (2014c) relies on describing formally the internals of a defence, which is often not possible.

9.3 Application layer Webpage Fingerprinting defences

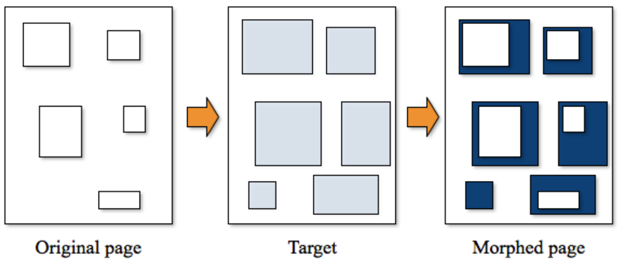

Figure 9.8: Graphical representation of ALPaCA. Once a target page is defined, by using one of the two variants P-ALPaCA and D-ALPaCA , ALPaCA pads the contents of the original page as defined by the target, generating new padding objects if needed. The original and morphed page will look identical to a user.

In this section we introduce ALPaCA, a WF defence targeted specifically at `.onion` sites, websites that are hosted over the Tor network (Cherubin et al., 2017).[3] In section 9.4, we conjecture ALPaCA's security may be evaluated without the need for feature extraction (i.e., direct estimation approach).

In the past, WF defences were designed:

1. at the **network layer**: they would operate directly on network packets, by morphing them (e.g., padding or splitting their content across multiple packets) or delaying them;

2. from the **client-side**: the defence would run on the client's computer, possibly in cooperation with the server.

However, these defences were usually impractical (or extremely difficult) to deploy, for the following reasons: 1) a defence working at the network layer generally requires major changes to the TCP/IP stack or in the anonymisation software used (e.g., the Tor client); 2) optimal defences (i.e., defences that by construction leak the least information whilst consuming the least bandwidth) require prior knowledge of the webpages they were defending.

We propose a defence mechanism called ALPaCA (*Application Layer Padding Concerns Adversaries*), which: i) operates at the application layer, and ii) is run

[3] In the same work, we also introduced LLaMA, a client-side defence that should serve as a drop-in replacement until ALPaCA is deployed. In this manuscript we focus on ALPaCA, because of its interesting properties (section 9.4), and we refer the interested reader to Cherubin et al. (2017) for details on LLaMA.

by the web server. Being server-side, ALPaCA has complete knowledge of the webpages it should defend, and it can decide the amount of padding accordingly. Furthermore, being implemented at the application layer makes it easier to deploy.

As a real life motivating example, we were contacted by SecureDrop, an organisation that provides `.onion` services for the anonymous communication between journalists and whistleblowers. They are concerned that sources wishing to use their service can be de-anonymised through WF. As a consequence, they are interested in using a server-side WF defence. We have included a SecureDrop website in all the datasets used for the evaluation of defences.

WF attacks are possible because different webpages serve different content. High level features, such as the number of requests the browser makes to download a page, the order of these requests, and the size of each response, induce distinctive low level features observed in the network traffic (Dyer et al., 2012; Panchenko et al., 2011). For instance, the number of requests sent by the browser corresponds to the number of objects embedded in the page such as images, scripts, stylesheets, and so on. For this reason, we believe that adding the padding to the actual contents of the page is a more natural strategy for hiding traffic features than sending dummy packets: if the defence successfully conceals high-level features, the low-level features will follow.

9.3.1 Background and assumptions

We define threat model, assumptions, and illustrate related research.

Threat model We consider the problem of WF as described in section 9.1. However, this time we assume the victim visits an `.onion` site, rather than a common website on the Internet. Observe that `.onion` sites have several characteristics that make them particularly vulnerable to WF attacks, even in an Open World scenario: their overall number is largely inferior to that of common websites, and they are more static (often they have no active content at all) (Kwon et al., 2015). Furthermore, considering that they tend to host sensitive content, WF attacks to `.onion` sites constitute a substantial privacy concern.

We will construct ALPaCA at the server-side. Luo et al. (2011) argued that a WF defence should be implemented at the client-side, because web servers have no incentive to offer such a service. However, we believe `.onion` site operators are aware of the privacy concerns that Tor clients have, and they would make the necessary (minor) modifications in the server to implement our defence.

For the design of ALPaCA, we will assume there is no dynamic content. This includes content generated at the client-side (e.g., AJAX) as well as the server-side (e.g., a PHP script polling a database). This assumption simplifies the design of the server-side defence: ALPaCA requires to know the size of the web resources being loaded, and it is hard to estimate the size of dynamic content a priori.

To assume that no JavaScript will run in the browser is not as unrealistic as it may seem. The Tor Browser's[4] security slider allows users to select different levels of security, disabling partially or totally JavaScript. Furthermore, privacy-aware websites (e.g., SecureDrop) already recommend their clients to disable JavaScript to prevent attacks such as cross-site scripting. It is reasonable to think that clients who wish to protect themselves against WF will first disable JavaScript to prevent these and other attacks.

Intuition Most WF defences in the literature are based on *link-padding*. The traffic morphing approach attempts to transform the traffic of a page to resemble that of another page (Wright et al., 2009; Lu et al., 2010; Panchenko et al., 2011), or to generalise groups of traffic traces into anonymity sets (Wang et al., 2014; Cai et al., 2014b). The main downside of this type of defences is that they require a large database of traffic traces that would be costly to maintain (Juárez et al., 2014). We refer the reader to section 9.1 for a list of the major WF defences.

Link-padding aims to conceal patterns in web traffic by adding varying amounts of dummy messages and delays in flows. It has been used for traffic morphing to cause confusion in the classifier by disguising the target page fingerprint as that of another page (Wright et al., 2009). However, as Dyer et al. (2012) noticed, traffic morphing techniques produce high bandwidth overheads as some packets must be buffered for a long period. The strategy of ALPaCA is different from traffic morphing, in that page contents are not disguised as other pages', but rather the content is modified to become less fingerprintable. The intuition behind ALPaCA is to make each resource look like an "average" resource, according to the distribution of resources in the world of pages. This approach reduces the overheads with respect to morphing, as resizing an object to an average size will tend to require less amount of padding than to an object of a specific page. We have also

[4]The recommended way to visit `.onion` sites is via the Tor Browser, a modified version of Firefox which channels all its traffic through the Tor network, and in addition prevents common deanonymisation attacks (e.g., Browser Fingerprinting).

experimented with morphing the contents in a page to make it look like another page.[5] This can be seen as the application-level counterpart of *traffic morphing*.

9.3.2 ALPaCA

ALPaCA is a server-side defence that pads the contents of a webpage and creates new content with the objective of concealing distinctive features at the network level. We demonstrate that this strategy is not only effective, but also practical to deploy. We have implemented and evaluated ALPaCA as a script that periodically runs on a server hosting an `.onion` site.

We first show that it is possible to pad the most popular types of webpage objects (e.g., images, HTML) to a desired size, without altering how they look to a user. We then propose two variants of server-side defences, referred to as P-ALPaCA and D-ALPaCA . For a given page to defend, both ALPaCA variants choose a suitable list of sizes T, the *target*. A target specifies the number and size of the objects of the morphed page; P-ALPaCA and D-ALPaCA differ in how they select such a target. Then, the objects of the original page are padded to match the sizes defined in T. If T contains more elements than the page's objects, then new objects ("padding objects") are created and referenced from the morphed HTML page. Figure 9.8 gives a high level overview of this process.

Padding an object to a target size

We now describe how we can pad most object types. It is important to note that an adversary looking at encrypted packets cannot: i) distinguish the type of objects that are being downloaded, nor ii) infer how much padding was added to such objects or whether they were padded at all. By padding an object directly on the server, we can control how large it will look like at the network level. While this control is not complete (because of compression in the HTTP protocol), experiments show that this discrepancy does not largely affect the security.

Table 9.5 shows the types of objects that we can pad up to a desired size, and their frequency within the `.onion` site world. To pad text objects (e.g., HTML and CCS) we can add the desired amount of random data into a comment. To pad binary objects (e.g., images), it is normally sufficient to append random data to the end of the file; in fact, the file structure allows programs to recognise the end of the file even after this operation. We verified that binary files would not be

[5]Details of this are in Cherubin et al. (2017).

Table 9.5: Padding the most frequent objects in `.onion` sites up to a desired size. "Frequency" indicates how frequent the object type was among the `.onion` sites we examined. "N.O." stands for "not observed". We assume JavaScript is disabled, although it is possible to morph JS files as shown.

Content type	**Morphing**	**Frequency**
PNG, ICO, JPG, GIF, BMP	Append random bytes to the file.	51%
HTML	Insert random data within a comment "<!–", "–>".	13%
CSS	Insert random data within a comment "/*" "*/".	12%
JS	Insert random data within a comment "/*" "*/".	13%
MP3	Append random bytes to the file.	N.O.
AVI	Append random bytes to the file.	N.O.

corrupted after such padding, and we refer to Cherubin et al. (2017) for details. We suspect that many other types of objects can be morphed analogously, by appending random bytes or by inserting data in comments or unused sections of the type structure. We remark, however, that in experiments we did not remove from webpages the objects that we could not morph.

Morphing a page to a target T

We introduce the basic ALPaCA morphing algorithm, which morphs the contents of a page to match the sizes defined by a target T. The target is selected differently by the two versions of ALPaCA, as presented later, and it defines the size of the objects that the morphed page should have.

The algorithm keeps two lists: M, containing the morphed objects $\mathcal{o}$, and P, which keeps track of the sizes in T that have not been used for morphing an object; both lists M and P are initially empty. The algorithm sequentially considers the objects of the original page from the smallest to the largest; for object $\mathcal{o}$, it seeks the smallest size $t \in T$ which $\mathcal{o}$ can be padded (i.e., for which $size(\mathcal{o}) \leq t$). Once it has found such a t, it removes all the elements of T smaller than t, and pads $\not{o}$ to size t; the elements removed from T at this stage (except t) are put into P. After all the original objects have been morphed, the sizes remaining in T are appended to P. New "padding objects" (objects containing random bytes) are generated according to the sizes in P. We make sure that padding objects will be downloaded by a browser, but will not be shown, by inserting a reference to them in the HTML page as if they were hidden images.[6] Finally, the HTML page itself is padded to a target size by the defence. In Appendix 4 we illustrate the algorithm.

P-ALPaCA Probabilistic-ALPaCA (P-ALPaCA) generates a target by randomly sampling from a distribution that represents real-world `.onion` sites. Specifically, it has access to three probability distributions, D_n, D_h and D_s, defined respectively on the number of objects a page has, the size of the HTML page and the size of each of its objects. The defence samples a target T using these distributions, and then morphs the original page as shown in Algorithm 2 in Appendix 4.

We estimated D_n, D_h and D_s using Kernel Density Estimation (KDE) from $5,295$ unique `.onion` websites we crawled. In Appendix 4.2 we show the resulting distributions D_n, D_h and D_s, and provide details on how we estimated them.

The defence first samples the number of objects n for the morphed page according to D_n. Then, it samples the size of the morphed HTML from D_h, and n sizes from D_s which constitute a target T. Finally, it attempts to morph the original page to T; if morphing fails, the procedure is repeated.

Because sampling from the distributions can (with low probability) produce very large targets T, we introduced a parameter *max_bandwidth* to P-ALPaCA . Before morphing, the defence checks that the total page size is within this parameter: $\sum_{t \in T} t \leq max\_bandwidth$. If not, the sampling procedure is repeated.

D-ALPaCA We propose a second variant, Deterministic-ALPaCA (D-ALPaCA), which decides deterministically by how much a page's objects should be padded. The defence is inspired by Tamaraw (Cai et al., 2014c), which pads the number of packets in a network trace to a multiple of a padding parameter L. D-ALPaCA has the advantage of introducing less overheads than P-ALPaCA , but experimental results suggest this defence is slightly less effective against a WF adversary.

[6]To add an invisible object called "rnd.png" to an HTML page we insert `<img src="rnd.png" style="visibility:hidden">`. The browser will consider this a PNG file and it will download it, but it will not attempt to show it. The file, thus, needs not to respect the PNG format, and it can just contain random bytes.

D-ALPaCA accepts as input three parameters: λ, σ and max_s, where max_s should be a multiple of σ. It pads the number of objects of a page to the next multiple of λ, and the size of each object to the next multiple of σ. Then, if the target number of objects is larger than the original number of objects, it creates padding objects of size sampled uniformly at random from $\{\sigma, 2\sigma, ..., max\_s\}$. Details of D-ALPaCA are in Appendix 4.

9.3.3 Experimental design

In this manuscript, we only include a limited evaluation of ALPaCA, under a Closed World scenario. We refer to the original paper for a more thorough analysis, and for details on the data collection (Cherubin et al., 2017).

Data We evaluate ALPaCA via a live implementation on the Tor network. We first crawl over a significant fraction of the total Tor `.onion` sites, obtained from *Ahmia*[7], the most popular search engine for onion services, and we used them to estimate the distributions for P-ALPaCA . Note that Ahmia maintains a blacklist of illegal `.onion` sites, which we excluded from our crawls.

We collected 40 network traffic loads (*instances*) for a set of chosen 100 `.onion` sites. We applied P-ALPaCA and D-ALPaCA , and evaluated them against state-of-the-art WF attacks. Note that, in this work, we did not estimate (β, Φ)-security as introduced in section 9.2, although that is a possible future application.

	Latency		Volume	
	%	Avg. (s)	%	Avg. (KB)
Undefended	–	3.99	–	175
P-ALPaCA	52.6	6.09	86.2	326
D-ALPaCA (2, 500, 5000)	66.3	6.63	3.66	182
D-ALPaCA (2, 5000, 5000)	56.1	6.22	9.84	193
D-ALPaCA (5, 2500, 5000)	61.7	6.44	15.1	202
D-ALPaCA (10, 5000, 5000)	41.7	5.65	44	254

Table 9.7: P-ALPaCA & D-ALPaCA latency and bandwidth overheads.

[7] https://ahmia.fi.

Overheads We evaluated D-ALPaCA under four different parameter choices: (λ, σ, max_s): $(2, 500, 5000)$, $(2, 5000, 5000)$, $(5, 2500, 5000)$, $(10, 5000, 5000)$. As for P-ALPaCA , its distributions were estimated as described in Appendix 4.2.

In Table 9.7, we see that average latencies are approximately 40-60% greater in the defended traces than in the undefended ones. The extra time that the user will spend loading the pages is between two and three seconds.

	k-NN (%)	k-FP (%)	CUMUL (%)
Undefended	45.6	69.6	55.6
P-ALPaCA	0.2	9.5	15.6
D-ALPaCA (2, 500, 5000)	9.5	22.7	27.0
D-ALPaCA (2, 5000, 5000)	12.5	34.4	40.0
D-ALPaCA (5, 2500, 5000)	5.8	22.3	30
D-ALPaCA (10, 5000, 5000)	7.2	22.9	33.0
Decoy (Panchenko et al., 2011)	4.9	11.2	X
Tamaraw (Cai et al., 2014c)	6.8	14.0	X
BuFLO (Dyer et al., 2012)	5.3	13.3	X

Table 9.8: Closed world comparison of P-ALPaCA and D-ALPaCA with other defences. CUMUL attack depends on packet lengths and so defences that only operate on time information cannot be applied.

Closed World classification We performed a Closed World WF attack on P-ALPaCA defended, D-ALPaCA defended and undefended `.onion` sites. We use the attacks CUMUL, k-FP, and k-NN for evaluation (section 9.1); the number of neighbours used for k-FP and k-NN is fixed at 2.[8]

Table 9.8 shows the Closed World classification results of undefended `.onion` sites against `.onion` sites with each instance uniquely defended using P-ALPaCA or D-ALPaCA . WF attacks are ineffective under both defences, and in fact P-ALPaCA improves upon *Tamaraw* and *BuFLO*. D-ALPaCA does slightly worse than the P-ALPaCA in terms of defending `.onion` sites, but as can it be seen from Table 9.7, it has real advantages in terms of limiting bandwidth overheads. For example, D-ALPaCA with parameters $(2, 500, 5000)$, reduced k-FP accuracy from 69.6% to 22.7%, compared to the P-ALPaCA which reduced attack accuracy to

[8]We use Tobias Pulls' implementation of the k-NN WF attack (Pulls, 2016b).

10%. But, D-ALPaCA $(2, 500, 5000)$ required 23.6 times less bandwidth than P-ALPaCA to achieve these results. Results also show no notable difference in attack accuracy when changing parameters; however, unsurprisingly, smaller parameters led to smaller bandwidth overheads.

9.3.4 Limitations

ALPaCA is easy to deploy, and it showed high a high degree of security. We discuss its limitations.

Defence Fingerprinting An adversary could carry on what we call a Defence Fingerprinting attack: given the traffic between the victim and a web server, he may predict whether the web server is running ALPaCA or not. This is possible because ALPaCA-defended `.onion` sites may look different at the network level from those that do not. This attack may be a concern if the `.onion` sites implementing ALPaCA are a small number (e.g., because of a slow rate of adoption).

Other leakage An aspect which was mostly neglected by previous WF literature is that there may be more leakage than the one exhibited by network traffic. For instance, an `.onion` site may take longer to reply to the client than another site. While this leakage is negligible for most WF scenarios, cases may exist where an adversary can exploit this.

In particular, we suspect there could be an active WF attack, where the adversary, who tries to distinguish for example between two `.onion` sites, purposely slows down one of the two (e.g., with a DoS-like attack).

Resources

The source code and datasets of both ALPaCA and LLaMA are publicly available on GitHub.[a] The original code is also available on an `.onion` site[b], which is protected using our defence ALPaCA.
We are now developing ALPaCA as a library, `libalpaca`, with the intent of deploying it as a module for web servers such as Apache and NGINX:

`https://github.com/camelids/libalpaca`.

[a] `http://github.com/camelids`.
[b] `http://3tmaadslguc72xc2.onion`.

9.4 Extension: Measuring λ-security for ALPaCA

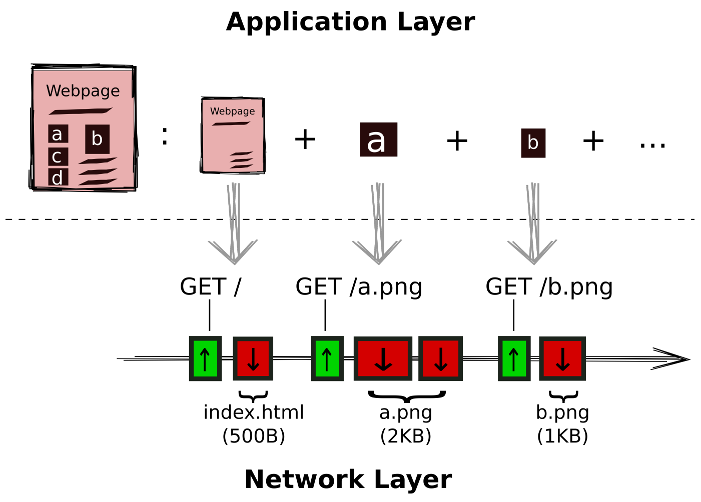

Figure 9.9: How a page load translates into network traffic. The network traffic generated when loading the page mostly depends on the size (and number) of its objects.

We conjecture it is possible to estimate the black-box security of ALPaCA via the direct method (chapter 2), therefore obtaining λ-security, rather than through features (obtaining (λ, Φ)-security).

This is based on the following hypothesis: because ALPaCA morphs a webpage's objects directly, we can assume that the generated network traffic (and, therefore, any corresponding feature), depends directly on them (Figure 9.9). In other words, the random variable $\mathbf{p} \in \mathcal{P}$, which defines the packet sequences generated by some webpage W, is entirely described by the size of the objects of W themselves, $W = (\sigma_1, ..., \sigma_q)$. Clearly, this requires assuming that other sources of randomness or leakage (e.g., a website's latency) are negligible; this hypothesis was supported by our experiments on ALPaCA in the previous section, where the indistinguishability of webpages at the application layer (i.e., page's objects) resulted in indistinguishability at the network layer.

For this reason, we can estimate the security of ALPaCA directly on these objects W, rather than on packet sequences p or features $\Phi(p)$; observe that this would not be possible for any WF defence. But this fact is important: whilst packet sequences p carry a lot of noise due to network-dependent events, a webpage's objects W do not. For this reason, we conjecture that a *direct* security estimate on W would guarantee convergence. We leave to future research to verify this conjecture empirically.

Part IV

Extensions, Future Work, and Conclusions

10 Black-box Security

This chapter suggests several extensions of the Black-box security framework we described so far, and it highlights future applications.

10.1 Infinite secret space $|\mathcal{S}|$

Throughout this manuscript, we assumed a finite $\mathcal{S}$. A natural extension of our framework is to consider the case when both $\mathcal{O}$ and $\mathcal{S}$ are infinite.

10.1.1 Formulation

In ML terms, this corresponds to the problem of *regression*. Consider an example space $\mathcal{O} \times \mathcal{S}$, with infinite $\mathcal{O}$ and $\mathcal{S}$, and define a joint distribution $\mu(o, s)$. In regression, one aims to find a function $g : \mathcal{O} \mapsto \mathcal{S}$ that minimises the expected loss $\mathbb{E}(\ell(s, g(o)))$ for a new example (o, s) sampled from $\mu(o, s)$; we call g a *regressor*.

In the classification setting we described in chapter 2, we selected ℓ to be the 0-1 loss function; this loss function is clearly not applicable to regression. We will consider two loss functions:

$$\begin{aligned} (L_1) \quad & \ell(s, \hat{s}) := |s - \hat{s}| \\ (L_2) \quad & \ell(s, \hat{s}) := (s - \hat{s})^2 \end{aligned}$$

10.1.2 UC

Literature contains several UC results in the case of regression. We report one, based on the k_n-NN regression rule.

Consider a training set $(o_1, s_1), ..., (o_n, s_n)$, a test object o with secret s, and let $(o_{(i)}, s_{(i)})$ denote the i-th nearest neighbour of o according to some distance metric d. The k-NN regressor g outputs:

$$g(o) := \frac{1}{k} \sum_{i=1}^{k} s_{(i)} \,.$$

We define the k_n-NN regression rule as the rule using the k-NN regressor and choosing $k = k_n$ such that $k_n/n \to 0$ and $k_n \to \infty$ as $n \to \infty$. Then Stone (1977), as a corollary of the same theorem that proved UC of the k_n-NN rule (section 3.1), showed that the k_n-NN regression rule is UC both for the L_1 and L_2 loss functions.[1]

Note that lower bounds similar to the one by Cover and Hart (1967), which we described in section 3.1, exist for regression (Cover, 1968).

10.1.3 Applications

Several applications exist for this extended framework. For example, in our experiments on the `Gowalla` dataset (section 6.1), the adversary aimed to guess users' geographical coordinates, and was penalised with the 0-1 loss function. A better model is to use a "less rigid" loss notion (e.g., L_1 or L_2), and therefore employ a regression rule to estimate the Bayes risk; this would penalise less a prediction that is closer to the real location, which represents better real-world threat models.

We leave the application of regression to real-world Black-box security problems to future research.

Remark A rule that is UC for the L_2 loss function is also consistent in classification (0-1 loss), after an appropriate thresholding. However, convergence in a regression setting requires more examples than classification, which suggests regression is a harder problem than classification (Devroye et al., 2013).

10.2 Extension to many-observations adversaries

In the problem we considered, an adversary receives one test object o, and has to predict the corresponding secret s; this is called a *one-observation* adversary.

[1]The UC results established by Stone (1977) apply to a more general definition of loss function; we refer the interested reader to the reference for details.

We now consider the more general scenario of a *many-observations* adversary. This adversary is given, *for the same* secret s, a vector of q objects $(o_1, ..., o_q)$. As before, his goal is to predict s, although this time he has more information.

We define the Bayes risk in this context, introduce a novel estimator based on the nearest neighbor classifier, and then suggest real-world applications.

10.2.1 Bayes Risk

Let us consider a system $(\pi, \mathcal{B})$, inducing a joint distribution μ on $\mathcal{O} \times \mathcal{S}$, and for simplicity suppose $\mathcal{O} \times \mathcal{S}$ is finite. The Bayes risk of a many-observations adversary, upon making q observations, is:

$$R^*_q := \sum_{(o_1,...,o_q) \in \mathcal{O}^q} 1 - \max_s \prod_{i=1}^{q} \mu(o_i, s)\,;$$

the summation is appropriately replaced by an integral when $\mathcal{O}$ is infinite.

10.2.2 Frequentist approach

Intuitively, the frequentist approach works as before (subsection 3.1.1), by replacing the distribution μ with an estimate $\hat{\mu}$, and π with an estimate $\hat{\pi}$. Formally, suppose it is trained on training data $Z = \{(o_1, s_1), ..., (o_n, s_n)\}$; then, its prediction for an observation of q objects is:

$$Freq(o_1, ..., o_q) = \begin{cases} \operatorname{argmax}_s \prod_{i=1}^{q} \hat{\mu}(o_i, s) & \text{if } (o_1, ..., o_q) \text{ in training set} \\ \operatorname{argmax}_s \hat{\pi}(s) & \text{otherwise}\,. \end{cases}$$

Now, if all $(o_1, ..., o_m)$ were observed in the training data, this is optimal: as we noticed in subsection 3.1.1, $\hat{\mu} \to \mu$ and $\hat{\pi} \to \pi$ as $n \to n$ because of the Law of Large Numbers.

However, similarly to what we observed for the frequentist approach in the one-observation scenario, if some o_i is missing, the frequentist cannot say as much, and indeed his error is equivalent to random guessing for that object.

10.2.3 Nearest Neighbour for many-observations adversary

We can extend the frequentist approach by using the nearest neighbour idea even in this scenario. Consider a training set $Z = \{(o_1, s_1), ..., (o_n, s_n)\}$, a new object o_i, and let $N(o_i, Z)$ be the secret of the nearest neighbour of o_i within Z. Then we define the following classifier:

$$NN((o_1, ..., o_q)) = \underset{s}{\operatorname{argmax}} \prod_{i=1}^{q} \hat{\mu}(N(o_i, Z), s)\,;$$

in this case, if some o_i did not appear in the training set, its prediction is replaced by that of its nearest neighbour.

We claim this classifier is UC in the many-observations setting (i.e., it converges to R^*_q for a chosen $q > 0$). This is easy to see: as $n \to \infty$, the nearest neighbour of each test object o_i approaches o_i itself, making NN equivalent to the frequentist approach, which is UC.

We leave the empirical evaluation of this method to future research; however, we suspect this NN method to have similar advantages over the frequentist approach to those we observed in chapter 5.

10.2.4 Applications

This extended scenario has a wide applicability. For example, it was considered by Backes and Köpf (2008) for the side channel attack to the exponentiation algorithm we described in chapter 5; there, an adversary could make the hardware compute the algorithm q times for the same key s, thereby obtaining q timing measurements from which to predict the key. The estimation approach used by Backes and Köpf (2008) was based on the frequentist principle.

We suggest this scenario could be also used to extend from Webpage Fingerprinting (section 9.1), where an adversary guesses the webpage the victim is viewing, to Website Fingerprinting where he wants to predict the website – possibly made of multiple webpages. In this case, the adversary would observe q webpage loads, assuming they all belong to the same website s, and try to predict s.

10.3 Further extensions

Extension to various loss functions In this manuscript we considered two alternative loss functions to the 0-1 loss. However, sometimes one may want to define custom loss functions; for example, we may want to capture a greater risk for an adversary in predicting a certain secret rather than another one, i.e.:

$$\ell(s, \hat{s}) := I(s \neq \hat{s})w_{\hat{s}}\,,$$

for some set of weights w_s, $s \in \mathcal{S}$.

The extension to such loss functions is straightforward for frequentist approaches. We leave the extension of our techniques to this setting as future research.

Confidence levels Under certain assumptions on the distribution, frequentist estimates can be associated with confidence intervals (e.g., Chothia et al. (2013)).

One could obtain confidence levels also from UC rules, by estimating their error using, for example, bootstrap techniques (Friedman et al., 2001). We leave the empirical evaluation of these and other techniques as future work.

We remark, however, that under our weak assumptions it is not possible to determine convergence rates for UC rules, nor valid confidence interval. In chapter 11, we show that Conformal Predictors (CP) can immediately guarantee predictions' validity, although by shifting the learning problem.

Other security measures In this manuscript, we focused on security measures that are computed as a function of the Bayes risk and random guessing error (chapter 4).

However, the security and QIF communities have been also interested in other leakage measures, such as Mutual Information (MI), which can be estimated by using a frequentist approach. Methods exist for estimating MI by exploiting the nearest neighbour principle (e.g., Kraskov et al. (2004)). We leave to future research the analysis of relevant literature, and the application of these methods to Black-box security.

10.4 Future applications

Applications for our methods abound. For example, many traffic analysis problems have a similar form of WF attacks (chapter 9), and we suspect they can be tackled with our framework to produce either λ-security or (λ, Φ)-security guarantees, for some desired security measure λ.

We now focus on a future application example, Membership Inference attacks, and then suggest a stronger generalisation of our methods' applicability.

10.4.1 Membership Inference

Membership Inference attacks were recently introduced by Shokri et al. (2017). They are carried out against an ML classifier $f : \mathcal{X} \mapsto [0,1]^L$, which outputs, for a test object $x \in \mathcal{X}$, a distribution over the L labels (*confidence vector*); for example, the i-th value in the confidence vector indicates how much f "is confident" that i is the correct prediction for its input.

Membership Inference Suppose a company trains a classifier f on a *private* dataset D, but it gives users oracle access to f. This setting is known as Machine Learning as a Service (MLaaS), where for instance the company allows users to query the model to receive predictions.

In this setting, the goal of the adversary with oracle access to f is to decide whether some test object x was in the training dataset D or not; that is, for an object x, the adversary should predict the answer to the binary question "Is $x \in D$?".

Intuitively, this attack works because f may be overly confident on the predictions for objects $x \in D$ (e.g., express very high confidence for its label). One may think this attack is only possible because of overfitting; however, it was shown that even when no overfitting occurs Membership Inference attacks are sometimes feasible (e.g., Yeom et al. (2018)).

Measuring β-security We suggest that our framework can be used for measuring the security of a classifier f against Membership Inference attacks. Note that, to the best of our knowledge, there are currently no methods for measuring the security of a generic classifier for these attacks, other than running state-of-the-art attacks against it.

One can proceed as follows. Create a dataset of examples $\{(o_1, s_1), ..., (o_n, s_n)\}$, where $o_i := f(x_i)$ is the confidence vector associated by the classifier to some object x_i, and $s_i := I(x_i \in D)$ tells whether x_i was in the training data or not. Observe that we do not specify from what distribution should come the x_i's that do not belong to D; in Shokri et al. (2017), they were sampled from a dataset "similar" to D, although it is difficult to provide guidelines on this. As for the priors $\pi(s)$, we can select them as uniform, and then use a prior-consistent security measure, such as β or Min-entropy leakage (chapter 4).[2]

Then, we can simply estimate the Bayes risk R^* on the dataset as shown in chapter 2, and therefore compute an appropriate security measure. Note that in this case, because the example space $\mathcal{O} \times \mathcal{S} = [0,1]^L \times \{0,1\}$ may be large, one may want to opt for a conservative estimate of R^* (e.g., NN bound, subsection 3.1.6). We leave the empirical evaluation of this idea to future research.

10.4.2 Generic ML-based attacks

In this manuscript, we have seen several applications of our methods, ranging from traffic analysis to inference attacks.

We can state something even more general on their applicability: whenever in a security attack the adversary makes use of an ML classifier (or regressor), and is evaluated with respect to his prediction risk, we can use the methods we introduced in this manuscript to measure security against this attack.

This may appear a trivial conclusion: researchers have been constructing attacks on the basis of ML for long, and our methods are also based on ML techniques. However, we claim our contributions for this kind of attacks are twofold: i) we provided a theoretical basis to claim (and verify) the optimality of our methods in this context (i.e., now we *know* why ML should be used in these security problems), and ii) we give a conservative estimate (NN bound, subsection 3.1.6) which can be used to lower-bound the error (and the security) for any existing attack.

[2]Note that the Bayes security measure β is prior-consistent for $|\mathcal{S}| = 2$, which is the case for Membership Inference attacks.

11 Extensions from the Conformal Prediction Theory

This section uses tools from the seminal work by Vovk et al. (2005), namely Conformal Predictors (CP) and exchangeability martingales, to extend our framework.

11.1 Background

We describe CPs, which can be used for prediction, and exchangeability martingales for testing the i.i.d. assumption on data.

11.1.1 Conformal Predictors

Conformal Predictors (CP)[1] are wrappers around ML classifiers (here called *nonconformity measures*[2]), equipping them with the validity property: for a test object $o \in \mathcal{O}$ with label $s \in \mathcal{S}$, and for a chosen significance level $\varepsilon \in [0, 1]$, a CP predicts a set $\Gamma^{\varepsilon} \subseteq \mathcal{S}$ of candidate labels (*prediction set*); such prediction set is conservatively valid, in that it guarantees that $Pr(y \notin \Gamma^{\varepsilon}) \leq \varepsilon$. In order to evaluate the tightness of a CP's predictions, an *efficiency* criterion (e.g., average size of Γ^{ε}) is adopted in applications (Vovk et al., 2005, 2016).

Internally, a CP uses the nonconformity measure to perform a randomness test for a test object o, given a training set of examples $\{(o_1, s_1), ..., (o_n, s_n)\}$, which outputs a p-value for each possible label $\hat{s} \in \mathcal{S}$ for the hypothesis: "$(o, \hat{s})$ belongs to the same distribution as the training examples". The p-value associated with each candidate label $\hat{s}$ is then thresholded by ε to decide whether to accept the hypothesis, and therefore to include $\hat{s}$ in the prediction set Γ^{ε}.

[1]This description of CP is taken from Cherubin (2018).
[2]More precisely, a nonconformity measure is a scoring function, such as an ML classifier with probabilistic output.

Validity Formally, a CP, $C^{A,\varepsilon} : \mathcal{O} \times (\mathcal{O}, \mathcal{S})^n \mapsto \mathcal{P}(\mathcal{S})$, with nonconformity measure A and significance level $\varepsilon \in [0, 1]$, is an algorithm taking as input a training set of examples $(o_i, s_i)_{i=1}^n$ and a new object $o \in \mathcal{O}$, and returning a prediction set $\Gamma^\varepsilon \subseteq \mathcal{S}$ of candidate labels for o (Vovk et al., 2005). We describe the CP algorithm in Algorithm 1. Whenever possible, we will leave the nonconformity measure A implicit, and refer to a CP simply with C^ε.

The following result holds for a CP.

Theorem 11.1 (CP validity (Vovk et al., 2005)). *Let C^ε be a CP, for some significance level $\varepsilon \in [0, 1]$. Let*

$$\Gamma^\varepsilon := C^\varepsilon(o, ((o_1, s_1), ..., (o_n, s_n)))$$

be the prediction set associated with a test object o with true label s, given $n > 0$ training examples $(o_i, s_i)_{i=1}^n$. Then C^ε is conservatively ε-valid, in that it guarantees:

$$Pr(s \notin \Gamma^\varepsilon) \leq \varepsilon \,.$$

The validity defined in this theorem is a conservative validity. There exists an alternative formulation of the CP algorithm, *smoothed* CP[3], which gives exact validity, i.e., $Pr(s \notin \Gamma^\varepsilon) = \varepsilon$.

Nonconformity measures A nonconformity measure is a function $A : \mathcal{O} \times \mathcal{S} \times \{\mathcal{O} \times \mathcal{S}\}^* \mapsto \mathbb{R}_{\geq 0}$, which can be thought of as *scorer*: it takes as input a bag of examples and a new example, and it associates with the new example a score (the *nonconformity score*); this score represents how "strange" the new example looks like with respect to the other examples, and it takes a higher value the less this object conforms to them. Nonconformity measures can be constructed from virtually any ML classifier or regressor, and the validity guarantees of CP holds regardless this choice.

A simple example is a nonconformity measure based on the k-NN classifier, which returns the sum of the distances of the k nearest neighbours to the new example that have its same label. Formally, consider a multiset of examples $Z = \{(o_1, s_1), ..., (o_n, s_n)\}$, and a new example (o_{n+1}, s_{n+1}). Then let $Z_s = \{(o_i, s_i) \in Z \mid s_i = s_{n+1}\}$ be the multiset of examples with the same label as the new example,

[3]CPs considered here are formally known as *deterministic* CPs.

and indicate with $(o_{(i)}, s_{(i)})$ the i-th closest example to (o_{n+1}, s_{n+1}) within Z_s. Then the k-NN nonconformity measure for a chosen distance metric d is:

$$A = \sum_{i=1}^{k} d(o_{(i)}, o_{n+1}) \,.$$

Efficiency A CP gives immediately what is generally sought for in standard ML applications: a guarantee on the errors. This however comes at a price: its predictions may be uncertain ($|\Gamma^\varepsilon| > 1$) or even empty ($|\Gamma^\varepsilon| = 0$); in other words, a CP's prediction may be not *efficient*. (Nonetheless, we argue that a CP gives more information to the analyst: it tells how (im)precise its predictions are, whilst in standard ML one does not necessarily know the errors committed.)

This constitutes a shift in the standard problem of learning; the new goal is to produce efficient (or tight) predictions. Several efficiency measures have been proposed in the past (Vovk et al., 2016); the most intuitive one is the average size of a prediction set, $|\Gamma^\varepsilon|$, for a chosen significance level ε.

In order to achieve efficient predictions one should evaluate several nonconformity measures, and select the one achieving the best performances. However, this is problem-dependent, and we suspect one can prove the non-existence of optimal nonconformity scorers, similarly to what the NFL theorems proved for learning rules (section 2.6).

Algorithm 1 Deterministic CP for computing a p-value

```
function $C^{A,\varepsilon}(o, (z_1, ..., z_n))$
    Initialize $\Gamma^\varepsilon$ to the empty set
    for $\hat{s} \in \mathcal{S}$ do
        Set temporarily $z_{n+1} = (o, \hat{s})$
        for $i = 1, ..., n+1$ do
            $\alpha_i \leftarrow A(z_i, (z_1, ..., z_{n+1}) \setminus z_i)$
        end for
        $p_{\hat{s}} \leftarrow \frac{\#\{i | \alpha_i \geq \alpha_{n+1}\}}{n+1}$
        if $p_{\hat{s}} > \varepsilon$ then
            Add $\hat{s}$ to $\Gamma^\varepsilon$
        end if
    end for
    return $\Gamma^\varepsilon$
end function
```

11.1.2 Exchangeability martingales

Exchangeability martingales are tools introduced by Vovk et al. (2003) that allow testing whether a sequence of objects $v_1, ..., v_n$ is i.i.d. according to some (unknown) probability distribution.[4]

Intuition An exchangeability martingale is defined for a CP and a *betting strategy* $b : [0,1] \times [0,1]^* \mapsto [0,\infty)$ as follows (Vovk et al., 2003). We first use a CP in "online" mode (i.e., the prediction for $v_n + 1$ is obtained by training on $v_1, ..., v_n$) to generate p-values $p_1, ..., p_n$.[5] Then we compute martingale values, $M_1, ..., M_n$, one for each p-value, by using the betting function:

$$M_t = \prod_{i=1}^{t} b_i(p_i) \quad t = 1, 2, ... ;$$

we assume $M_0 = 1$.

Then we have the following guarantee. Consider the following statements:

(S1) Sequence $v_1, ..., v_n$ is i.i.d..

(S2) P-values $p_1, ..., p_n$ generated from $v_1, ..., v_n$ are i.i.d. uniformly in $[0,1]$.

(S3) For all $\vartheta \geq 1$, $P(\exists t : M_t \geq \vartheta) \leq \frac{1}{\vartheta}$.

The exchangeability martingale test is based on the fact that:

$$S1 \implies S2 \implies S3,$$

where the first implication is due to CP's properties (e.g., Vovk et al. (2005)), and the second one comes from Ville (1939). In other words, if the sequence is indeed i.i.d., then its martingales should never take a "large" value.

The exchangeability martingales i.i.d. test is defined for a threshold ϑ, which is usually set to 20 or 100, corresponding to significance levels of 0.05 and 0.01: if the martingale value M_t exceeds ϑ for some t then we should reject the hypothesis that the sequence $v_1, ..., v_n$ is i.i.d..

[4]More formally, they verify that the sequence of objects is exchangeable (i.e., their joint distribution is the same for any permutation of the sequence). However, to test for exchangeability is equivalent to testing for i.i.d..

[5]P-values are the intermediate values of Algorithm 1 $p_{\hat{s}}$. More details on p-value generation for exchangeability martingales can be found in Vovk et al. (2003).

Theoretical background and practical recommendations on the betting function are found in Vovk et al. (2003); Fedorova et al. (2012).

11.2 Conformal Prediction adversaries

We introduce a kind of attacks in Black-box security that we call *many-predictions* attacks. As in the original formulation (chapter 2), the adversary receives an object o obtained by sampling a system $(\pi, \mathcal{B})$, with secret s. Differently from before, however, we ask the adversary to output a set of predictions $\Gamma \subseteq \mathcal{S}$, and we define the prediction error to be 1 if $s \notin \Gamma$, 0 otherwise.

A trivial adversary exists, who outputs $\Gamma = \mathcal{S}$ for every input; clearly, however, his predictions are not "useful". We will therefore be interested in the *efficiency* of his predictions (informally, how tight Γ is), which can be measured for example with the average $|\Gamma|$ (subsection 11.1.1).

Under this threat model, CP methods allow us to disregard the number of errors the adversary commits, and to only focus on his efficiency. We call *CP adversary* a many-predictions adversary who uses a CP for prediction. Trivially, this adversary inherits the validity property from CP (Theorem 11.1):

Theorem 11.2 (Validity of CP adversary)**.** *A CP adversary selects a nonconformity scorer, and, for a chosen significance level ε, uses the corresponding CP to make predictions. In particular, suppose he observes n examples, $\{(o_1, s_1), ..., (o_n, s_n)\}$, sampled from a system $(\pi, \mathcal{B})$, and has to make a set prediction Γ for a new object o, with true secret s. Then for any $n > 0$ his error probability is upper-bounded by ε:*

$$P(s \notin \Gamma) \leq \varepsilon\,.$$

CP adversaries shift the usual black-box attack setting. Indeed, they *always know* that, with probability at least $1 - \varepsilon$, the true secret s is contained in a CP's output. Therefore, their goal is to choose a nonconformity scorer A so to minimise the prediction's uncertainty (e.g., average $|\Gamma^\varepsilon|$).

We suspect CP adversaries will have several applications in security and privacy. For example, in the context of WF attacks, a CP adversary would produce a set of candidate webpages, containing the true webpage $\varepsilon \cdot 100\%$ of the times, and his goal would be to make predictions tight.

11.3 Zero leakage test

Tests for *zero leakage* assert whether a system leaks no information – or, equivalently, if it is perfectly secure (i.e., $R^* \approx R^\pi$), by only observing examples sampled by the system. These methods give less information to a security analyst than the estimation framework we considered so far. However, they can be run before the actual estimation: clearly, if the zero-leakage test does not find any leakage, then there should be no need for measuring the amount of leakage.

To the best of our knowledge, leakiEst is the only tool providing a zero leakage test in the context of Black-box security; its test is based on Kernel Density Estimation (KDE) (Chothia and Guha, 2011; Chothia et al., 2013), which we partially evaluated in chapter 7. Chothia et al. (2013) also compared this method with other well-known non-parametric zero-leakage tests, such as Kolmogorov-Smirnov, BWS, Anderson-Darling, CVM, and concluded that leakiEst tends to be more reliable for security purposes under small sample conditions.

We suggest that CP methods can be used as the basis for zero-leakage tests; this is a fairly natural application of CP, which has its roots in the problem of determining whether an object belongs to the same distribution as training objects.

The approach we suggest is as follows. For simplicity, suppose $\mathcal{S} = \{s_1, s_2\}$ and consider two datasets, D_1 and D_2, containing objects respectively from secrets s_1 and s_2. Compute p-values $p_1, p_2, ...$ for the concatenated datasets (D_1, D_2) in an on-line setting, and use the exchangeability martingales test (subsection 11.1.1) to verify whether the sequence $\{o_i \in D_1 \cup D_2\}$ is i.i.d.; this test will give answer to the hypothesis "D_1 and D_2 were generated from the same distribution", or equivalently whether the system leaks something or not. Future work may implement this idea, and evaluate it empirically.

11.4 Verifying i.i.d. assumption

A very simple application that follows immediately from exchangeability martingales is the following. Throughout this manuscript we assumed $(\pi, \mathcal{B})$ (and therefore $\mu(o,s)$) was fixed; this implies assuming that examples sampled from $\mu(o,s)$ are i.i.d. according to some (unknown) distribution.

In practice, one may want to verify this assumption (e.g., because they are not sure the system does not change overtime). We suggest one can use the exchangeability martingales test for this purpose.

12 Conclusions

Black-box security allows estimating the security (or leakage) of a system without knowing its internals, and by only looking at examples of inputs and corresponding outputs. Methods for black-box estimation have been traditionally based on classical Statistics principles. Unfortunately, they often did not scale to large real-world systems, and they posed restrictions on the type of systems they could handle (e.g., only systems with finite outputs).

This manuscript established new principles for measuring Black-box security. We based this on the following intuition: to measure the security of a system is equivalent to estimating the prediction error of a universally consistent Machine Learning (ML) rule; this allowed us to bring results and estimators from the ML theory, effectively giving new foundations to Black-box security in the ML theory.

We defined two strategies for estimating a generic security measure λ: the *direct* approach, and estimation *through features*, respectively giving rise to λ-security, and (λ, Φ)-security notions. The latter, a weaker notion, applies a transformation (*features*) to the system's outputs before estimation, and it should be used when the direct approach does not converge. We also proposed and studied a new measure, Bayes security (β), as a generalisation of a well-known measure in Cryptography.

We used our approaches on synthetic data and for real-world applications. For instance, we measured λ-security for: i) a location privacy dataset, ii) a side channel on an exponentiation algorithm, and iii) a time side channel against e-Passports. We also estimated (λ, Φ)-security for defences to Webpage Fingerprinting (WF) attacks, a major class of Traffic Analysis, and we introduced a defence, ALPaCA, for which we conjectured one can estimate security directly, thereby achieving λ-security guarantees.

In the final part of this manuscript, we proposed several extensions of the method, such as systems with continuous input space, new loss functions, and an extension to a more general class of adversaries (*many-observations* adversaries). We leave to future research their empirical evaluation. We also suggested that our approaches can tackle virtually any ML-based attack (e.g., traffic analysis, Membership Inference), for which they can provide security lower bounds. We further proposed extensions inspired by the Conformal Prediction (CP) theory, such as the notion of CP adversaries, and as a conjecture we suggested a *zero-leakage* test. In the future, we hope these tools can be developed further and applied to even more problems.

Most of our theoretical results where based on the large sample assumption; indeed, for any security estimator, there exists some (possibly very contrived) system for which it would need an arbitrarily large number of input-output examples to converge. But unfortunately this is an intrinsic problem of Black-box security estimation, which we cannot avoid under our weak assumptions, and it is good practice to always measure security with more than one estimator. Nonetheless, we reiterate that our methods provide a sound approach to measure the security of real-world systems, and that they perform well empirically – indeed, they often outperform the standard frequentist approach. We therefore hope in the future they will find a wide application in security.

Glossary

`.onion` site	A website that is hosted over the Tor network.
WF	Webpage Fingerprinting, a major class of traffic analysis attacks, where an adversary aims at predicting the webpage the victim visits by only looking at the encrypted network traffic she produces.
WsF	Website Fingerprinting, an extension of WF, where the adversary aims at visiting which website (and not the individual webpage) the victim is viewing.
Adversary	In this manuscript, an adversary has the goal of predicting the input of a system given its output.
Convergence	An estimator converged if its estimate is "close" – for some definition of closeness – to the true value of the quantity it is trying to estimate. Convergence is measured with respect to the number of training examples available; a *fast* convergence rate means an estimator requires few examples for convergence. When the true value is not known (the common case in real-world applications), we establish heuristics criteria to determine whether an estimator converged (e.g., chapter 9).
ML	Machine Learning studies learning and prediction algorithms whose functionalities are programmed significantly according to observed data.
QIF	Quantitative Information Flow (Smith, 2009) is the research field interested in defining or estimating the leakage of a channel (system).
Security analyst	Someone who is interested in computing or estimating the security of a system.

System	A (possibly randomised) algorithm, receiving inputs and returning outputs according to some underlying distribution; inputs are chosen according to a set of priors specified by the system itself.
Tor network	A network that guarantees unlinkability of source and destination of a user, thereby guaranteeing her anonymity.
UC	A Universally Consistent (UC) rule is an ML rule whose error converges to the Bayes risk; in other words, UC rules are asymptotically optimal in the size of their training data.

List of Symbols

Bayes risk (R^*) For some loss function $\ell : \mathcal{S} \times \mathcal{S} \mapsto \mathbb{R}_{\geq 0}$, it is the risk of the (ideal) optimal adversary at predicting the secret input s corresponding to an object $o = \mathcal{B}(s)$. We will usually assume 0-1-loss, for which R^* corresponds to the adversary's probability of error (chapter 2).

Bayes security measure (β) The Bayes security measure, defined as in chapter 4.

Black-box ($\mathcal{B}$) A possibly randomised algorithm $\mathcal{B} : \mathcal{S} \mapsto \mathcal{O}$, defining how a system associates inputs to outputs.

Classifier (g) An algorithm $g : \mathcal{O} \mapsto \mathcal{S}$, which predicts a secret $\hat{s} = g(o)$ for a system's output o.

Distance metric (d) A distance metric $d : \mathcal{O} \times \mathcal{O} \mapsto \mathbb{R}_{\geq 0}$.

Error (R^g) The probability of error of a classifier (or learning rule) g.

Generic security measure (λ) A generic security (or leakage) measure, which can be computed as a function of R^* and R^π (chapter 4).

Generic security measure through features ((λ, Φ)) A generic security (or leakage) measure, computed using estimation *through features*: for a chosen transformation $\Phi : \mathcal{O} \mapsto \mathcal{O}'$ (*features*), mapping the objects $o \in \mathcal{O}$ into the feature space $\mathcal{O}'$, and for the corresponding Bayes risk R^*_Φ computed on the transformed example space $\mathcal{O}' \times \mathcal{S}$, we compute λ as a function of R^*_Φ, and declare the system is (λ, Φ)-secure (chapter 8).

Joint distribution (μ) The joint probability distribution $\mu(o, s)$ induced by a system $(\pi, \mathcal{B})$ on the example space $\mathcal{O} \times \mathcal{S}$.

Learning rule (g_n) An ML rule (or simply *learning rule*); it associates a classifier to a training set of size n; with a slight abuse of notation, we write $g_n(o)$ to indicate the chosen classifier.

Objects' space ($\mathcal{O}$) Space of the system's outputs (or *objects*). We indicate an output with $o \in \mathcal{O}$. We call $\mathcal{O} \times \mathcal{S}$ the example space.

Priors (π) Set of prior probabilities defining the probability of s being sampled, $\pi(s) = P(s)$.

Random guessing error (R^π) Smallest risk an adversary commits at predicting the secret input s when only knowing the system's priors π.

Secrets' space ($\mathcal{S}$) Space of the system's inputs (or *secrets*). We indicate an input with $s \in \mathcal{S}$.

System ($(\pi, \mathcal{B})$) A system is a tuple of prior probabilities π and a black-box $\mathcal{B}$.

Bibliography

The gowalla dataset. URL https://snap.stanford.edu/data/loc-gowalla.html.

M. S. Alvim, K. Chatzikokolakis, C. Palamidessi, and G. Smith. Measuring information leakage using generalized gain functions. In *Proceedings of the 25th IEEE Computer Security Foundations Symposium (CSF)*, pages 265–279. IEEE, 2012. doi: http://doi.ieeecomputersociety.org/10.1109/CSF.2012.26. URL http://hal.inria.fr/hal-00734044/en.

M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: differential privacy for location-based systems. In *Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS 2013)*, pages 901–914. ACM, 2013. ISBN 978-1-4503-2477-9. doi: 10.1145/2508859.2516735. URL http://doi.acm.org/10.1145/2508859.2516735.

A. Antos, L. Devroye, and L. Gyorfi. Lower bounds for bayes error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(7):643–645, 1999. doi: 10.1109/34.777375.

M. Backes and B. Köpf. Formally bounding the side-channel leakage in unknown-message attacks. In *European Symposium on Research in Computer Security*, pages 517–532. Springer, 2008.

C. Braun, K. Chatzikokolakis, and C. Palamidessi. Quantitative notions of leakage for one-try attacks. *Electronic Notes in Theoretical Computer Science*, 249:75–91, 2009. doi: 10.1016/j.entcs.2009.07.085.

X. Cai, X. C. Zhang, B. Joshi, and R. Johnson. Touching from a Distance: Website Fingerprinting Attacks and Defenses. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 605–616. ACM, 2012. doi: 10.1145/2382196.2382260. URL http://doi.acm.org/10.1145/2382196.2382260.

X. Cai, R. Nithyanand, and R. Johnson. CS-BuFLO: A Congestion Sensitive Website Fingerprinting Defense. In *Workshop on Privacy in the Electronic Society (WPES)*, pages 121–130. ACM, 2014a. doi: 10.1145/2665943.2665949. URL http://doi.acm.org/10.1145/2665943.2665949.

X. Cai, R. Nithyanand, and R. Johnson. Glove: A Bespoke Website Fingerprinting Defense. In *Workshop on Privacy in the Electronic Society (WPES)*, pages 131–134. ACM, 2014b. doi: 10.1145/2665943.2665950.

X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg. A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses. In *ACM Conference on Computer and Communications Security (CCS)*, pages 227–238. ACM, 2014c. doi: 10.1145/2660267.2660362. URL http://doi.acm.org/10.1145/2660267.2660362.

K. Chatzikokolakis, T. Chothia, and A. Guha. Statistical measurement of information leakage. In J. Esparza and R. Majumdar, editors, *Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 6015 of *Lecture Notes in Computer Science*, pages 390–404. Springer, 2010. ISBN 978-3-642-12001-5. URL http://dx.doi.org/10.1007/978-3-642-12002-2.

K. Chatzikokolakis, G. Cherubin, and C. Palamidessi. The bayes security measure. Technical report, 2018.

G. Cherubin. A valid and universal framework to evaluate website fingerprinting defences. Technical report, Royal Holloway University of London, 2015.

G. Cherubin. Bayes, not naïve: Security bounds on website fingerprinting defenses. *Proceedings on Privacy Enhancing Technologies*, 4:215–231, 2017.

G. Cherubin. Majority vote ensembles of conformal predictors. *Machine Learning*, Aug 2018. ISSN 1573-0565. doi: 10.1007/s10994-018-5752-y. URL https://doi.org/10.1007/s10994-018-5752-y.

G. Cherubin and I. Nouretdinov. Hidden markov models with confidence. In *Conformal and Probabilistic Prediction with Applications - 5th International Symposium, COPA 2016, Madrid, Spain, April 20-22, 2016, Proceedings*, pages 128–144, 2016. doi: 10.1007/978-3-319-33395-3_10.

G. Cherubin, I. Nouretdinov, A. Gammerman, R. Jordaney, Z. Wang, D. Papini, and L. Cavallaro. Conformal clustering and its application to botnet traffic. In *Statistical Learning and Data Sciences - Third International Symposium, SLDS 2015, Egham, UK, April 20-23, 2015, Proceedings*, pages 313–322, 2015. doi: 10.1007/978-3-319-17091-6_26.

G. Cherubin, J. Hayes, and M. Juarez. Website fingerprinting defenses at the application layer. *Proceedings on Privacy Enhancing Technologies*, 2:165–182, 2017. doi: 10.1515/popets-2017-0023.

G. Cherubin, A. Baldwin, and J. Griffin. Exchangeability martingales for selecting features in anomaly detection. In *Proceedings of the Seventh Workshop on Conformal and Probabilistic Prediction and Applications*, volume 91 of *Proceedings of Machine Learning Research*, pages 157–170. PMLR, 2018. URL http://proceedings.mlr.press/v91/cherubin18a.html.

G. Cherubin, K. Chatzikokolakis, and C. Palamidessi. F-bleau: Fast black-box leakage estimation. In *IEEE Symposium on Security and Privacy (S&P)*, 2019.

E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: User movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 1082–1090, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0813-7. doi: 10.1145/2020408.2020579. URL http://doi.acm.org/10.1145/2020408.2020579.

T. Chothia and A. Guha. A statistical test for information leaks using continuous mutual information. In *Proceedings of the 24th IEEE Computer Security Foundations Symposium, CSF 2011, Cernay-la-Ville, France, 27-29 June, 2011*, pages 177–190. IEEE Computer Society, 2011. doi: 10.1109/CSF.2011.19. URL https://doi.org/10.1109/CSF.2011.19.

T. Chothia and V. Smirnov. A traceability attack against e-passports. In *FC10: Proceedings of the 14th International Conference on Financial Cryptography and Data Security 2010*. LNCS, Springer-Verlag, 2010.

T. Chothia, Y. Kawamoto, and C. Novakovic. A tool for estimating information leakage. In *International Conference on Computer Aided Verification (CAV)*, pages 690–695. Springer, 2013.

T. Chothia, Y. Kawamoto, and C. Novakovic. LeakWatch: Estimating information leakage from java programs. In *Proc. of ESORICS 2014 Part II*, pages 219–236, 2014. doi: 10.1007/978-3-319-11212-1_13.

T. Cover. Estimation by the nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(1):50–55, 1968.

T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967. doi: 10.1109/TIT.1967.1053964. URL `http://dx.doi.org/10.1109/TIT.1967.1053964`.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., second edition, 2006.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research*, 2(Dec):265–292, 2001.

L. Devroye, L. Györfi, and G. Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 2013.

C. Dwork. Differential privacy. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *33rd International Colloquium on Automata, Languages and Programming (ICALP 2006)*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006. ISBN 3-540-35907-9. URL `http://dx.doi.org/10.1007/11787006_1`.

K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton. Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail. In *IEEE Symposium on Security and Privacy (S&P)*, pages 332–346. IEEE, 2012. doi: 10.1109/SP.2012.28. URL `http://dx.doi.org/10.1109/SP.2012.28`.

A. Faragó and G. Lugosi. Strong universal consistency of neural network classifiers. *IEEE Transactions on Information Theory*, 39(4):1146–1151, 1993.

V. Fedorova, A. J. Gammerman, I. Nouretdinov, and V. Vovk. Plug-in martingales for testing exchangeability on-line. In *Proceedings of the 29th International Conference on Machine Learning, ICML*, 2012.

P. Foster. BBC to deploy detection vans to snoop on internet users, 2016. URL `https://www.telegraph.co.uk/news/2016/08/05/bbc-to-deploy-detection-vans-to-snoop-on-internet-users`.

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

K. Fukunaga and D. M. Hummels. Bayes error estimation using parzen and k-nn procedures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (5): 634–643, 1987.

T. Glasmachers. Universal consistency of multi-class support vector classification. In *Advances in Neural Information Processing Systems*, pages 739–747, 2010.

J. Hayes and G. Danezis. k-fingerprinting: a Robust Scalable Website Fingerprinting Technique. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 1187–1203. USENIX Association, 2016.

M. Juárez, S. Afroz, G. Acar, C. Díaz, and R. Greenstadt. A critical evaluation of website fingerprinting attacks. In *ACM Conference on Computer and Communications Security (CCS)*, pages 263–274. ACM, 2014. doi: 10.1145/2660267.2660368. URL `http://doi.acm.org/10.1145/2660267.2660368`.

M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright. Toward an Efficient Website Fingerprinting Defense. In *European Symposium on Research in Computer Security (ESORICS)*, pages 27–46. Springer, Springer, 2016. doi: 10.1007/978-3-319-45744-4_2.

P. C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Annual International Cryptology Conference*, pages 104–113. Springer, 1996.

B. Köpf and D. Basin. Timing-sensitive information flow analysis for synchronous systems. In *European Symposium on Research in Computer Security*, pages 243–262. Springer, 2006.

A. Kraskov, H. Stögbauer, and P. Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

A. Kwon, M. AlSabah, D. Lazar, M. Dacier, and S. Devadas. Circuit fingerprinting attacks: passive deanonymization of tor hidden services. In *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015.*, pages 287–302. USENIX Association, 2015. URL `https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/kwon`.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465):67–81, 2004.

L. A. Levin. Universal sequential search problems. *Problemy Peredachi Informatsii*, 9(3):115–116, 1973.

M. Liberatore and B. N. Levine. "Inferring the source of encrypted HTTP connections". In *ACM Conference on Computer and Communications Security (CCS)*, pages 255–263. ACM, 2006. doi: 10.1145/1180405.1180437. URL http://doi.acm.org/10.1145/1180405.1180437.

L. Lu, E. Chang, and M. Chan. Website Fingerprinting and Identification Using Ordered Feature Sequences. In *European Symposium on Research in Computer Security (ESORICS)*, pages 199–214. Springer, 2010. doi: 10.1007/978-3-642-15497-3_13.

X. Luo, P. Zhou, E. W. W. Chan, W. Lee, R. K. C. Chang, and R. Perdisci. HTTPOS: Sealing Information Leaks with Browser-side Obfuscation of Encrypted Flows. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011*. IEEE Computer Society, 2011. URL http://www.isoc.org/isoc/conferences/ndss/11/pdf/6_3.pdf.

L. Ming and P. Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer Heidelberg, 1997.

S. Oya, C. Troncoso, and F. Pérez-González. Back to the drawing board: Revisiting the design of optimal location privacy-preserving mechanisms. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1959–1972. ACM, 2017. ISBN 978-1-4503-4946-8. doi: 10.1145/3133956.3134004. URL http://doi.acm.org/10.1145/3133956.3134004.

A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society, WPES 2011, Chicago, IL, USA, October 17, 2011*, pages 103–114. ACM, 2011. doi: 10.1145/2046556.2046570. URL http://doi.acm.org/10.1145/2046556.2046570.

A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel. Website fingerprinting at internet scale. In *Proceedings of the 23rd*

Internet Society (ISOC) Network and Distributed System Security Symposium (NDSS 2016). IEEE Computer Society, 2016. doi: 10.14722/ndss.2016.23477.

M. Perry. Experimental Defense for Website Traffic Fingerprinting. Tor project Blog. "https://blog.torproject.org/blog/experimental-defense-website-traffic-fingerprinting", 2011. (accessed: October 10, 2013).

T. Pulls. Adaptive padding early (APE). http://www.cs.kau.se/pulls/hot/thebasketcase-ape/, 2016a.

T. Pulls. A golang implementation of the kNN website fingerprinting attack. "https://github.com/pylls/go-knn", 2016b. (accessed: May, 2016).

N. Santhi and A. Vardy. On an improvement over Rényi's equivocation bound, 2006. Presented at the 44-th Annual Allerton Conference on Communication, Control, and Computing, September 2006. Available at http://arxiv.org/abs/cs/0608087.

SecureDrop. SecureDrop. securedrop.org. "https://securedrop.org/", 2016. (accessed: April 20, 2016).

V. Shmatikov and M.-H. Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *European Symposium on Research in Computer Security*, pages 18–33. Springer, 2006. doi: 10.1007/11863908_2.

R. Shokri, G. Theodorakopoulos, C. Troncoso, J.-P. Hubaux, and J.-Y. L. Boudec. Protecting location privacy: optimal strategy against localization attacks. In T. Yu, G. Danezis, and V. D. Gligor, editors, *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS 2012)*, pages 617–627. ACM, 2012. ISBN 978-1-4503-1651-4.

R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 3–18. IEEE, 2017.

G. Smith. On the foundations of quantitative information flow. In L. de Alfaro, editor, *Proceedings of the 12th International Conference on Foundations of Software Science and Computation Structures (FOSSACS 2009)*, volume 5504 of *LNCS*, pages 288–302, York, UK, 2009. Springer.

I. Steinwart. Support vector machines are universally consistent. *Journal of Complexity*, 18(3):768–791, 2002.

C. J. Stone. Consistent nonparametric regression. *The annals of statistics*, pages 595–620, 1977. doi: 10.1214/aos/1176343886.

P. Syverson, R. Dingledine, and N. Mathewson. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium*, pages 303–320. USENIX Association, 2004. doi: 10.21236/ada465464.

A. Tewari and P. L. Bartlett. On the consistency of multiclass classification methods. *Journal of Machine Learning Research*, 8:1007–1025, 2007. URL http://dl.acm.org/citation.cfm?id=1390325.

J. Ville. *Etude critique de la notion de collectif*. Gauthier-Villars Paris, 1939.

V. Vovk, I. Nouretdinov, and A. Gammerman. Testing exchangeability on-line. In *Machine Learning, Proceedings of the 20th International Conference ICML*, 2003.

V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic learning in a random world*. Springer Science & Business Media, 2005.

V. Vovk, H. Papadopoulos, and A. Gammerman. *Measures of Complexity: Festschrift for Alexey Chervonenkis*. Springer, 2015.

V. Vovk, V. Fedorova, I. Nouretdinov, and A. Gammerman. Criteria of efficiency for conformal prediction. In *Symposium on Conformal and Probabilistic Prediction with Applications*, pages 23–39. Springer, 2016.

T. Wang and I. Goldberg. Improved Website Fingerprinting on Tor. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 201–212. ACM, ACM, 2013. doi: 10.1145/2517840.2517851.

T. Wang and I. Goldberg. Walkie-talkie: An effective and efficient defense against website fingerprinting. Technical report, University of Waterloo, 2015.

T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg. Effective Attacks and Provable Defenses for Website Fingerprinting. In *USENIX Security Symposium*, pages 143–157. USENIX Association, 2014. URL https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/wang_tao.

J. Weston and C. Watkins. Multi-class support vector machines. Technical report, Citeseer, 1998.

D. H. Wolpert. The supervised learning no-free-lunch theorems. In *Soft computing and industry*, pages 25–42. Springer, 2002.

C. V. Wright, S. E. Coull, and F. Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *Network & Distributed System Security Symposium (NDSS)*. IEEE Computer Society, 2009. URL `http://www.isoc.org/isoc/conferences/ndss/09/pdf/14.pdf`.

S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282. IEEE, 2018.

Appendix

1 Formal analysis of the frequentist approach

This Appendix complements subsection 5.2.4. To better understand the behaviour of the frequentist approach for observations that were not in the training data, we derive a crude approximation of this estimate in terms of the size of training data n. The approximation makes the following assumptions:

1. each observation $o \in \mathcal{O}$ is equally likely to appear in training data (i.e., $P(o) = 1 - \frac{1}{|\mathcal{O}|}$);
2. if an observation appears in the training data, the frequentist approach outputs the secret minimising the Bayes risk;
3. the frequentist estimate knows the real priors π;
4. if an observation does not appear in the training data, then the frequentist approach outputs the secret with the maximum prior probability;

The first two assumptions are very strong, and thus this is just an approximation of the real trend of such estimate. However, in practice it approximates well the real trend Figure 1.

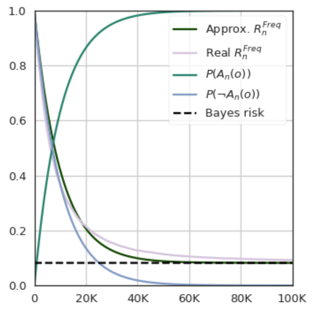

Figure 1: Approximation of the frequentist estimate as n grows for $R^* \approx 0.08$, $|O| = 10K$, and $|S| = 1K$; this is compared with the real frequentist estimate R_n^{Freq}.

Let $A_n(o)$ denote the event "observation o appears in a training set of n examples"; because of assumption 1), $P(A_n(o)) = 1 - \left(1 - \frac{1}{|\mathcal{O}|}\right)^n$. The conditional Bayes risk estimated with a frequentist approach given n examples is:

$$\begin{aligned}
r_n(o) =& r_n(o|A_n(o))P(A_n(o)) + r_n(o|\neg A_n(o))P(\neg A_n(o)) = \\
=& \left(1 - \max_{s\in\mathcal{S}} \frac{\hat{\mathcal{C}}_{s,o}\hat{\pi}(s)}{P(o)}\right)P(A_n(o)) + \\
&+ (1 - \max_{s\in\mathcal{S}} \hat{\pi}(s))P(\neg A_n(o)) \approx \\
\approx& \left(1 - \max_{s\in\mathcal{S}} \frac{\mathcal{C}_{s,o}\pi(s)}{P(o)}\right)P(A_n(o)) + \\
&+ (1 - \max_{s\in\mathcal{S}} \pi(s))P(\neg A_n(o))
\end{aligned}$$

Assumptions 2) and 3) were used in the last step. From this expression, we derive the frequentist estimate of R^* t step n:

$$\begin{aligned}
R_n^{Freq} &= \mathbb{E}r_n = \\
&= \sum_{o\in\mathcal{O}} P(o)\left(1 - \max_{s\in\mathcal{S}} \frac{\mathcal{C}_{s,o}\pi(s)}{P(o)}\right)P(A_n(o)) + \\
&\quad + \sum_{o\in\mathcal{O}} P(o)(1 - \max_{s\in\mathcal{S}} \pi(s))P(\neg A_n(o)) = \\
&= P(A_n(o))\left(\sum_{o\in\mathcal{O}} P(o) - \sum_{o\in\mathcal{O}} \max_{s\in\mathcal{S}} \mathcal{C}_{s,o}\pi(s))\right) + \\
&\quad + P(\neg A_n(o))(1 - \max_{s\in\mathcal{S}} \pi(s)) \sum_{o\in\mathcal{O}} P(o) = \\
&= P(A_n(o))\left(1 - \sum_{o\in\mathcal{O}} \max_{s\in\mathcal{S}} \mathcal{C}_{s,o}\pi(s))\right) + \\
&\quad + P(\neg A_n(o))(1 - \max_{s\in\mathcal{S}} \pi(s)) = \\
&= P(A_n(o))R^* + P(\neg A_n(o))R^\pi = \\
&= R^*\left(1 - \left(1 - \frac{1}{|\mathcal{O}|}\right)^n\right) + R^\pi\left(1 - \frac{1}{|\mathcal{O}|}\right)^n.
\end{aligned}$$

In the second step we used $P(A_n(o))$ as a constant, as allowed by assumption 1).

The expression of R_n indicates that $P(A_n(o))$ weights between random guessing according to priors-based random guessing and the Bayes risk; when $P(A_n(o)) \geq$

$P(\neg A_n(o))$, which happens for $n \geq -\frac{\log 2}{\log\left(1-\frac{1}{|\mathcal{O}|}\right)}$ the frequentist approach starts approximating using the actual Bayes risk (Fig. 1).

2 NN bound

The following result holds for the NN classifier. Let $L = |\mathcal{S}|$, and let R^{NN} be the probability that the NN classifier trained on a training set of size n misclassifies a test object. Then, as $n \to \infty$ (Cover and Hart, 1967):

$$R^* \leq R^{NN} \leq R^* \left(2 - \frac{L}{L-1} R^*\right).$$

This holds for arbitrary distributions on $\mathcal{O} \times \mathcal{S}$.

From this we obtain the NN lower bound on R^* shown in subsection 3.1.6.

Proof. Consider the right-hand side of the above inequality. We get:

$$\frac{L}{L-1}(R^*)^2 - 2R^* + R^{NN} \leq 0,$$

which has radices:

$$R^*_{1,2} = \frac{2 \pm \sqrt{4 - 4\frac{L}{L-1}R^{NN}}}{2\frac{L}{L-1}},$$

for which holds:

$$R^*_1 \leq R^* \leq R^*_2.$$

Then, by only considering the first one we obtain:

$$\frac{2 - \sqrt{4 - 4\frac{L}{L-1}R^{NN}}}{2\frac{L}{L-1}} \leq R^*,$$

which we can simplify to

$$\frac{L-1}{L}\left(1 - \sqrt{1 - 1\frac{L}{L-1}R^{NN}}\right) \leq R^*.$$

□

3 Webpage Fingerprinting

We include details on the attacks we evaluated in section 9.2.

3.1 Attacks

LL The feature set of this attack is composed of the count of packets with a certain direction and size, for each direction and size $\{\uparrow, \downarrow\} \times (0, ..., MTU]$, where $MTU = 1500$ is the maximum transmission unit. It uses the naïve Bayes classifier (NB) for classification, with kernel density estimation (KDE) for estimating the conditional probabilities (Liberatore and Levine, 2006).

VNG++ Its feature set includes total time span of a trace, total per-direction bandwidth, and *(direction, size)* of each sequence of contiguous outgoing packets (“bursts”). It uses NB as a classifier (Dyer et al., 2012).

k-NN The feature set comprises general features (e.g., bandwidth, and packet counts), unique packet lengths, features related to packet ordering and bursts, and the lengths of the first 20 packets accounting for direction (set to negative for incoming packets). An algorithm also determines a set of weights for the features, according to their importance; feature vectors are multiplied by these weights before classification.[1] For classification, it uses a custom modification of the k-Nearest Neighbours classifier, which works as follows. To predict the label for an object x, the classifier first determines the k closest objects to x. If all of them have the same label, it predicts that label; otherwise, it outputs an empty prediction. Manhattan distance is used as a distance metric for the classifier (Wang et al., 2014).

CUMUL The feature set of this attack includes basic information of packet sequences (e.g., total bandwidth, total in/out packets), together with the cumulative sum of packets' sizes. The attack employs a Support Vector Machine (SVM) classifier with an RBF kernel, and uses cross validation (CV) grid search to determine the optimal parameters for the kernel (Panchenko et al., 2016).

[1] In the original attack, weights are used by the distance metric. This is equivalent to multiplying weights before prediction.

k-FP This is the current state-of-the-art attack. Its feature set comes from a systematic analysis of the features proposed by previous research. Importance of features was evaluated with respect to the classifier used for the attack (k-Nearest Neighbours); the 20 most important features are in Table 9.1 (section 9.1). In this attack, the selected features are transformed using Random Forest (RF) as follows. First, feature values from the original feature set are extracted. Then, RF is applied to these values for generating leaves; leaves are then used as feature values for classification. As a classifier, this attack employs the custom modification of the k-Nearest Neighbours classifier that was used for the k-NN attack. Hamming distance is used as a distance metric (Hayes and Danezis, 2016).

4 ALPaCA

We provide details on the WF defence we introduced, ALPaCA. When defending a webpage, ALPaCA selects a *target* page T according to one of two variants: probabilistic (P-ALPaCA) or deterministic (D-ALPaCA). The target specifies the size of the HTML page once defended, and the size (and number) of all its objects.

4.1 Algorithms

We illustrate the algorithms we described in subsection 9.3.2.

ALPaCA The basic morphing algorithm for defending a webpage, and padding its content to a chosen target T is shown in Algorithm 2.

P-ALPaCA The probabilistic variant of ALPaCA selects the target T according to distribution estimated from data (subsection 4.2). After estimating said distributions, we run Algorithm 3.

D-ALPaCA The deterministic variant of ALPaCA generates a target as in Algorithm 4.

4.2 Distribution estimation for P-ALPaCA

We used Kernel Density Estimation (KDE) to estimate the distributions of number of objects (Figure 2), size of HTML pages (Figure 3) and size of objects (Figure 4). KDE is a non-parametric method for estimating a probability distribution given a

Algorithm 2 Pad a list of objects to a target

```
Require: $O$: list of original page objects
    $T$: list of target sizes
Ensure: $M$: list of morphed objects

  $M \leftarrow [\,]$
  $P \leftarrow [\,]$
  ▷ Morph the original objects.
  while $|M| < |O|$ do
      $\mathscr{o} \leftarrow \arg\min_{\mathscr{o} \in O} size(\mathscr{o})$
      ▷ Remove the target sizes smaller than $size(\mathscr{o})$.
      while $min(T) < size(\mathscr{o})$ do
          Remove $min(T)$ from $T$
          Append $min(T)$ to $P$
      end while
      if $T$ is empty then
          ▷ Cannot morph $O$ to $T$
          fail
      end if
      ▷ Note: the current $min(T)$ is larger than $size(\mathscr{o})$
      $t \leftarrow min(T)$
      $m \leftarrow \mathscr{o}$ padded to size $t$
      Append $m$ to $M$
  end while
  ▷ Add padding objects.
  Merge $P$ and $T$ into $P$
  for $p$ in $P$ do
      $m \leftarrow$ New padding object of size $p$
      Append $m$ to $M$
  end for
```

data sample, which provides smoother estimates than histograms. KDE requires to specify a kernel (Gaussian, in our case) and a bandwidth. The bandwidth impacts on the smoothness of the estimate: a larger bandwidth tends to provide better smoothness, but less fidelity to the original data. To determine the bandwidth for each of our distributions, we first performed Grid Search Cross Validation using `scikit-learn` library[2], to obtain a rough idea of the bandwidth ranges. Then, we manually trimmed the bandwidth to achieve what visually seemed to reflect well the variance of data, but also provided smooth distributions. For our pur-

[2] http://scikit-learn.org.

Algorithm 3 P-ALPaCA

```
Require: $O$: list of original page objects
  $D_n$: distribution over the number of objects
  $D_h$: distribution over the size of HTML pages
  $D_s$: distribution over the size of objects
  $html\_size$: size of the original HTML page
  $max\_bandwidth$: maximum page size

  ▷ We use $x \xleftarrow{\$} D$ to indicate that $x$ is sampled from distribution $D$
  $morphed \leftarrow False$
  while not $morphed$ do
      $T \leftarrow [\,]$
      $h \xleftarrow{\$} D_h$
      if $h < html\_size$ then
          continue
      end if
      $n \xleftarrow{\$} D_n$
      for $i$ in $1..n$ do
          $s \xleftarrow{\$} D_s$
          Append $s$ to $T$
      end for
      if $sum(T) < max\_bandwidth$ then
          Try morphing $O$ to target $T$ (Algorithm 2)
          If successful, $morphed \leftarrow True$
      end if
  end while
  Pad the HTML page to size $h$
```

poses, it was important to have smooth estimates to guarantee a good quality in sampling (e.g., to avoid spikes). We used a bandwidth of 2 for the distribution over objects, and of 2000 for both the HTML and object sizes distributions.

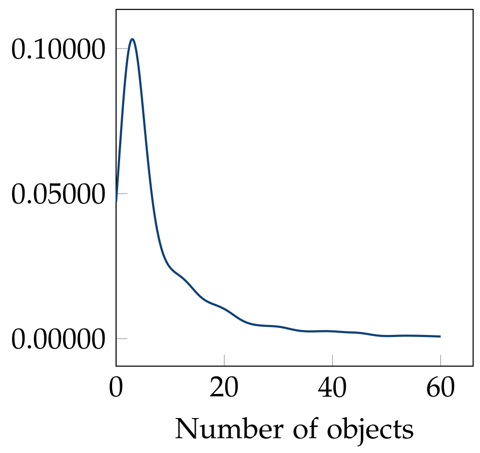

Figure 2: KDE distribution of the number of objects

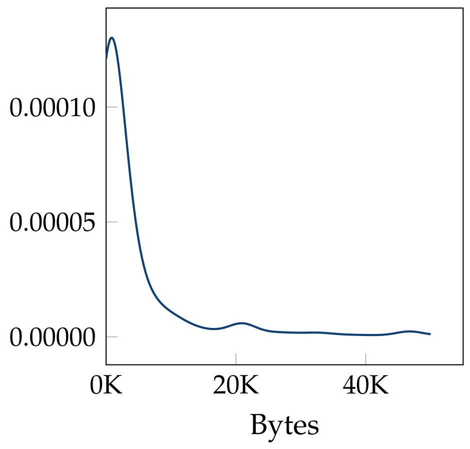

Figure 3: KDE distribution of the HTML sizes

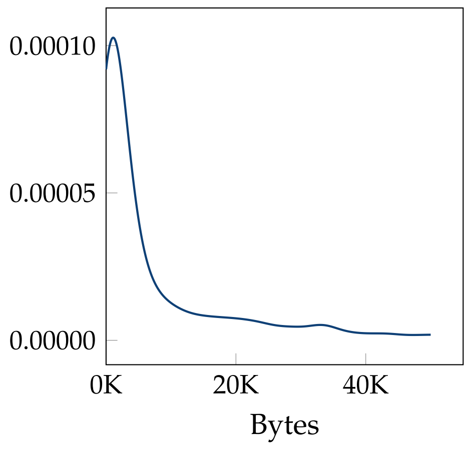

Figure 4: KDE distribution of the object sizes

Algorithm 4 D-ALPaCA

```
Require: $O$: list of original page objects
    $\sigma$: size parameter
    $\lambda$: number of objects parameter
    $html\_size$: size of the original HTML page
    $max\_s$: maximum size of a padding object (should be a multiple of $\sigma$)

    ▷ We use $x \xleftarrow{\$} S$ to indicate that $x$ is sampled uniformly at random from a set $S$
    $T \leftarrow [\,]$
    $h \leftarrow$ next multiple of $\sigma$ greater or equal to $html\_size$
    for $\not{o}$ in $O$ do
        $s \leftarrow$ next multiple of $\sigma$ greater or equal to $size(\not{o})$
        Append $s$ to $T$
    end for
    $n \leftarrow$ next multiple of $\lambda$ greater or equal to $size(O)$
    while $size(T) < n$ do
        $s \xleftarrow{\$} \{\sigma, 2\sigma, ..., max\_s\}$
        Append $s$ to $T$
    end while
    Morph $O$ to target $T$ (Algorithm 2)
    Pad the HTML page to size $h$
```