Security and Privacy Attacks with and against Machine Learning

Author:
Zhao, Benjamin

Publication Date:
2021

DOI:
https://doi.org/10.26190/unsworks/2052

Security and Privacy Attacks with and against Machine Learning

Benjamin Zi Hao Zhao

A thesis in fulfilment of the requirements for the degree of

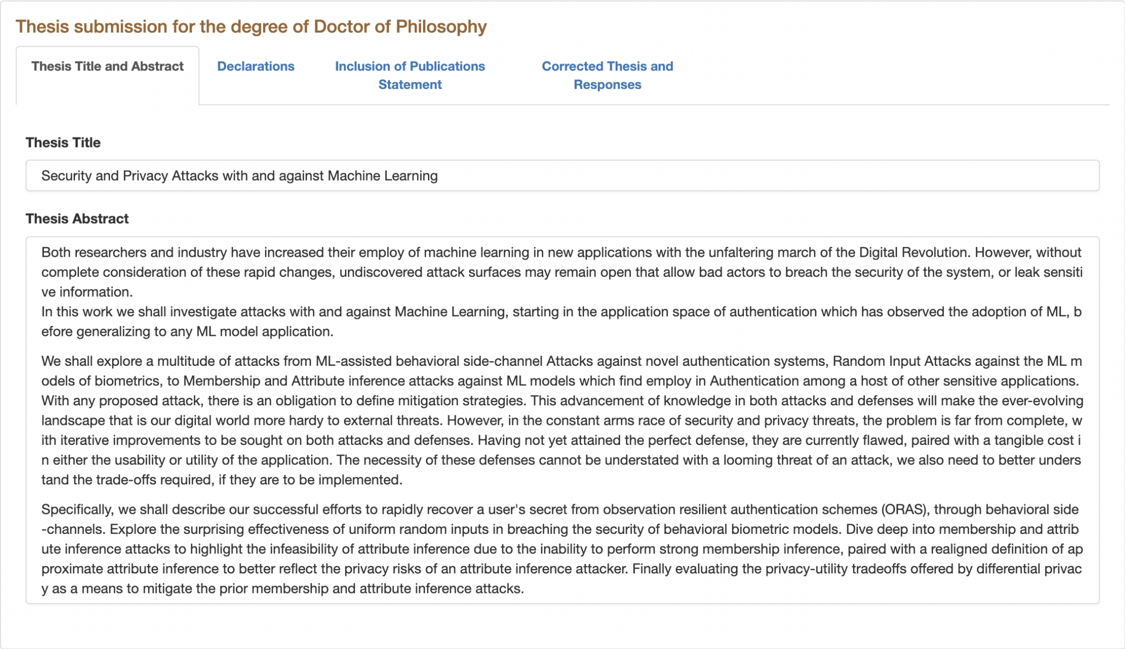
Doctor of Philosophy

School of Electrical Engineering and Telecommunications

Faculty of Engineering

The University of New South Wales

June 2021


Thesis Title and Abstract
Declarations
Inclusion of Publications Statement
Corrected Thesis and Responses

Thesis submission for the degree of Doctor of Philosophy

Thesis Title

Security and Privacy Attacks with and against Machine Learning

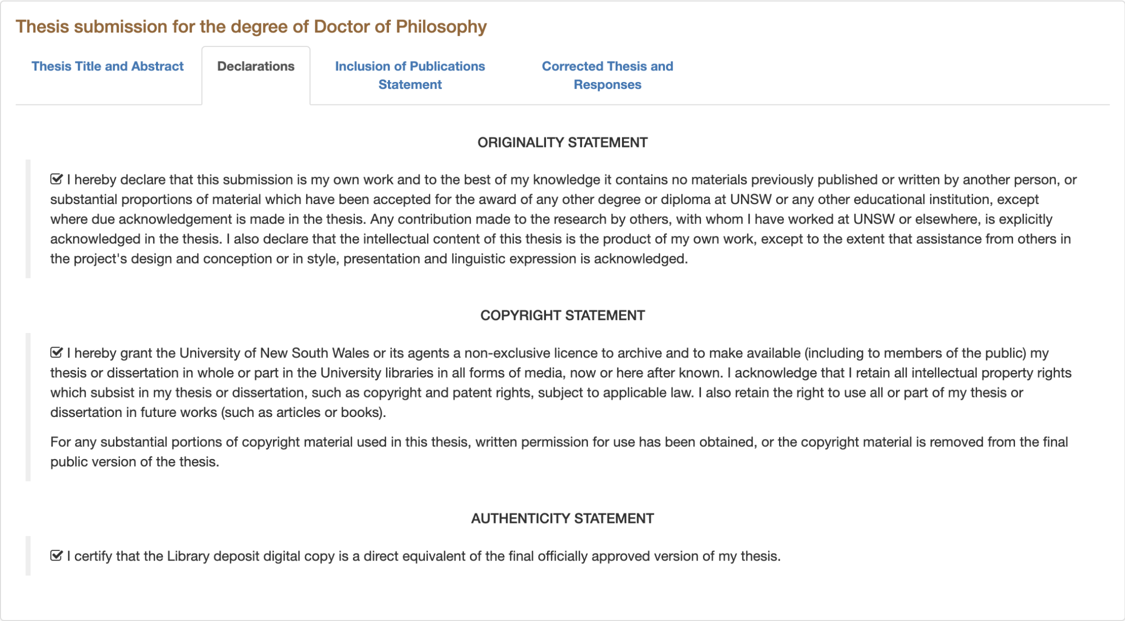
Thesis Abstract

Both researchers and industry have increased their employ of machine learning in new applications with the unfaltering march of the Digital Revolution. However, without complete consideration of these rapid changes, undiscovered attack surfaces may remain open that allow bad actors to breach the security of the system, or leak sensitive information.

In this work we shall investigate attacks with and against Machine Learning, starting in the application space of authentication which has observed the adoption of ML, before generalizing to any ML model application.

We shall explore a multitude of attacks from ML-assisted behavioral side-channel Attacks against novel authentication systems, Random Input Attacks against the ML models of biometrics, to Membership and Attribute inference attacks against ML models which find employ in Authentication among a host of other sensitive applications. With any proposed attack, there is an obligation to define mitigation strategies. This advancement of knowledge in both attacks and defenses will make the ever-evolving landscape that is our digital world more hardy to external threats. However, in the constant arms race of security and privacy threats, the problem is far from complete, with iterative improvements to be sought on both attacks and defenses. Having not yet attained the perfect defense, they are currently flawed, paired with a tangible cost in either the usability or utility of the application. The necessity of these defenses cannot be understated with a looming threat of an attack, we also need to better understand the trade-offs required, if they are to be implemented.

Specifically, we shall describe our successful efforts to rapidly recover a user's secret from observation resilient authentication schemes (ORAS), through behavioral side-channels. Explore the surprising effectiveness of uniform random inputs in breaching the security of behavioral biometric models. Dive deep into membership and attribute inference attacks to highlight the infeasibility of attribute inference due to the inability to perform strong membership inference, paired with a realigned definition of approximate attribute inference to better reflect the privacy risks of an attribute inference attacker. Finally evaluating the privacy-utility tradeoffs offered by differential privacy as a means to mitigate the prior membership and attribute inference attacks.



Thesis Title and Abstract
Declarations
Inclusion of Publications Statement
Corrected Thesis and Responses

Thesis submission for the degree of Doctor of Philosophy

ORIGINALITY STATEMENT

☑ I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

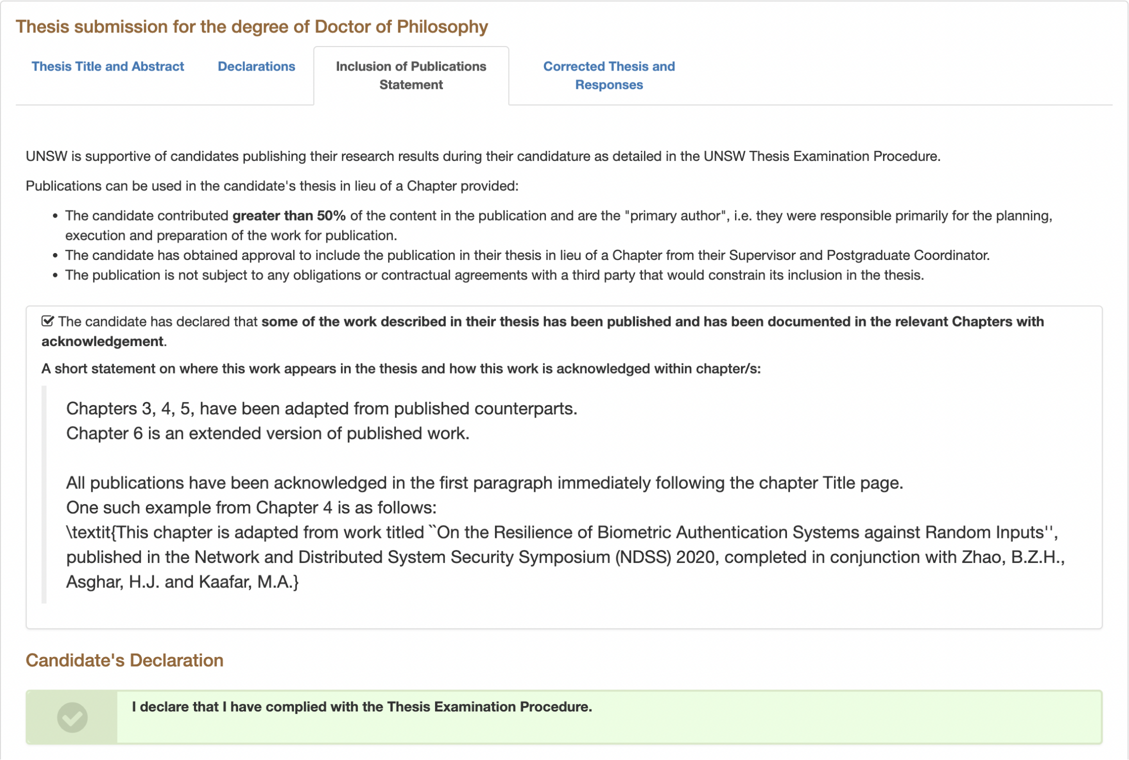
COPYRIGHT STATEMENT

☑ I hereby grant the University of New South Wales or its agents a non-exclusive licence to archive and to make available (including to members of the public) my thesis or dissertation in whole or part in the University libraries in all forms of media, now or here after known. I acknowledge that I retain all intellectual property rights which subsist in my thesis or dissertation, such as copyright and patent rights, subject to applicable law. I also retain the right to use all or part of my thesis or dissertation in future works (such as articles or books).

For any substantial portions of copyright material used in this thesis, written permission for use has been obtained, or the copyright material is removed from the final public version of the thesis.

AUTHENTICITY STATEMENT

☑ I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis.



Thesis Title and Abstract
Declarations
Inclusion of Publications Statement
Corrected Thesis and Responses

Thesis submission for the degree of Doctor of Philosophy

UNSW is supportive of candidates publishing their research results during their candidature as detailed in the UNSW Thesis Examination Procedure.

Publications can be used in the candidate's thesis in lieu of a Chapter provided:

- The candidate contributed **greater than 50%** of the content in the publication and are the "primary author", i.e. they were responsible primarily for the planning, execution and preparation of the work for publication.
- The candidate has obtained approval to include the publication in their thesis in lieu of a Chapter from their Supervisor and Postgraduate Coordinator.
- The publication is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in the thesis.

☑ The candidate has declared that **some of the work described in their thesis has been published and has been documented in the relevant Chapters with acknowledgement**.

A short statement on where this work appears in the thesis and how this work is acknowledged within chapter/s:

> Chapters 3, 4, 5, have been adapted from published counterparts.
> Chapter 6 is an extended version of published work.
>
> All publications have been acknowledged in the first paragraph immediately following the chapter Title page.
> One such example from Chapter 4 is as follows:
> \textit{This chapter is adapted from work titled ``On the Resilience of Biometric Authentication Systems against Random Inputs'', published in the Network and Distributed System Security Symposium (NDSS) 2020, completed in conjunction with Zhao, B.Z.H., Asghar, H.J. and Kaafar, M.A.}

Candidate's Declaration

✓ **I declare that I have complied with the Thesis Examination Procedure.**

Abstract

Both researchers and industry have increased their employ of machine learning in new applications with the unfaltering march of the Digital Revolution. However, without complete consideration of these rapid changes, undiscovered attack surfaces may remain open that allow bad actors to breach the security of the system, or leak sensitive information. In this work we shall investigate attacks with and against Machine Learning, starting in the application space of authentication which has observed the adoption of ML, before generalizing to any ML model application.

We shall explore a multitude of attacks from ML-assisted behavioral side-channel Attacks against novel authentication systems, Random Input Attacks against the ML models of biometrics, to Membership and Attribute inference attacks against ML models which find employ in Authentication among a host of other sensitive applications. With any proposed attack, there is an obligation to define mitigation strategies. This advancement of knowledge in both attacks and defenses will make the ever-evolving landscape that is our digital world more hardy to external threats. However, in the constant arms race of security and privacy threats, the problem is far from complete, with iterative improvements to be sought on both attacks and defenses. Having not yet attained the perfect defense, they are currently flawed, paired with a tangible cost in either the usability or utility of the application. The necessity of these defenses cannot be understated with a looming threat of an attack, we also need to better understand the trade-offs required, if they are to be implemented.

Specifically, we shall describe our successful efforts to rapidly recover a user's secret from observation resilient authentication schemes (ORAS), through behavioral side-channels. Explore the surprising effectiveness of uniform random inputs in breaching the security of behavioral biometric models. Dive deep into membership and attribute inference attacks to highlight the infeasibility of attribute inference due to the inability to perform strong membership inference, paired with a realigned definition of approximate attribute inference to better reflect the privacy risks of an attribute inference attacker. Finally evaluating the privacy-utility tradeoffs offered by differential privacy as a means to mitigate the prior membership and attribute inference attacks.

Acknowledgments

I would like to express my gratitude to my supervisors and mentors Hassan Asghar, Dali Kaafar, you have offered me many opportunities to grow as a person, guided me through unfamiliar territory, and have continually challenged me, allowing me to realize my competency as researcher.

I would also like to thank the many other collaborators, researchers and fellow students with whom I've been able to sit down and chat with, you have all assisted me on this journey in your own unique way. I do hope that I was also able to impart my own perspective on ideas and methods to your benefit.

There is no finite amount of thanks that will express my gratitude for all the people who have stood by me through this journey. To my family, you have been my safety net, financially and emotionally, supporting me through every step. And to my friends, a huge thank you.

The work in this thesis has taken me around the world, and has created memories that will be cherished for a lifetime. Thanks for all the great times everyone, and here's to the next challenge.

Publications and Presentations

List of Publications

- Zhao, B.Z.H., Ikram, M., Asghar, H.J., Kaafar, M.A., Chaabane, A. and Thilakarathna, K., 2019, July. A decade of mal-activity reporting: A retrospective analysis of internet malicious activity blacklists. In Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security (pp. 193-205).
- Zhao, B.Z.H., Asghar, H.J., Bhaskar, R. and Kaafar, M.A., 2019, November. On inferring training data attributes in machine learning models. Privacy Preserving Machine Learning (PPML) 2019, An ACM CCS Workshop.
- Zhao, B.Z.H., Asghar, H.J. and Kaafar, M.A., 2020, February. On the Resilience of Biometric Authentication Systems against Random Inputs. The Network and Distributed System Security Symposium (NDSS) 2020.
- Zhao, B.Z.H., Asghar, H.J., Kaafar, M.A., Trevisan, F. and Yuan, H., 2020, September. Exploiting Behavioral Side Channels in Observation Resilient Cognitive Authentication Schemes. ACM Transactions on Privacy and Security (TOPS), 24(1), pp.1-33.
- Zhao, B.Z.H., Agrawal, A., Coburn, C., Asghar, H.J., Bhaskar, R., Kaafar, M.A., Webb, D., and Dickinson, P., 2021, September. On the (In)Feasibility of Attribute Inference Attacks on Machine Learning Models. 6th IEEE European Symposium on Security and Privacy 2021.

List of Presentations

Oral presentations:

- ACM ASIA Conference on Computer and Communications Security (AsiaCCS), July 2019, Auckland, New Zealand.
- Privacy Preserving Machine Learning (PPML) an ACM CCS Workshop, November 2019, London, United Kingdom.

- The Network and Distributed System Security Symposium (NDSS),
  February 2020, San Diego, United States of America.
- The ACM Cloud Computing Security Workshop (CSSW) an ACM CCS Workshop,
  November 2020, Virtual.

Poster presentations:

- Privacy Preserving Machine Learning (PPML) an ACM CCS Workshop,
  November 2019, London, United Kingdom.

TPC Member/Reviewer

- PrivateNLP @ WSDM 2020
- PrivateNLP @ EMNLP 2020
- IEEE S&P 2021 Shadow PC
- Invited Reviewer for IEEE TIFS
- DPML @ ICLR 2021
- PrivateNLP @ NAACL 2021

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The world is in a constant state of change with innovations presenting themselves as the next disrupting force to the inertia of the status quo. A generous number of services have made a complete transition from a physical format to being entirely digital. With this, we are at an intersection whereby our digital identity is equally valuable and important as our physical identity. Elements of our physical identity, our names, looks, interest, history also taking residence in the digital space. With the digitization of this basic information, and the capture of information we previously did not even notice, there has been a meteoric rise of machine learning to harness this data for making more intelligent decisions. For example, recommending content/ads, predicting behaviors, or even tailoring medical treatments. In particular, with the use of machine learning, in security sensitive applications like authentication, and its' application on private data, additional security, and privacy risks are introduced that have not been previously understood. In this thesis, we shall explore both security and privacy attacks with the assistance of, and against machine learning models.

Thus we begin with protecting this sensitive information, to ensure that only the correct people can access said information. Authenticating people is not a new problem, with the three tenants of identification, *What you Have, What you Know, What you Are*, we can prove if you are a claimed individual. *What you Have*, may simply be a bank card, a license, or a key, a uniquely identifiable object that only you as an individual have access to. *What you Know*, takes the form of information that is only available to you, like that of a PIN, password, or your mother's maiden name. *What you Are*, is intrinsic to you, physical attributes that can be used to identify yourself, your face, fingerprint, or dental records, however, this also includes peculiar quirks or behaviors that make you unique, for example how you walk, talk or interact with objects.

These tenants have been the cornerstone of verifying an individual's identity, however, a fault point likes in the replicability of these identifiers for someone to impersonate you; The easiest of which is *What you Know*. *What you Know* is peculiar as, often to demonstrate *What you Know*, you would reveal to the verifier that you have this information. However, the very process of demonstrating you have this information has revealed the information. Which in the presence of a bad actor will take this information and use it to impersonate you. By impersonating you, the bad actors have the power to directly impact your life, such as stealing money, fraud, blackmail, and extortion, among others. Cybercrime as an entity has been estimated to cost 3 Trillion dollars globally in 2015, with forecasts to rise to 6 Trillion by 2021 [4].

A growing number of reported incidents indicate that this is no longer just a fictional possibility [5], prompting widespread proposals for alternative authentication schemes. One approach to alternative authentication schemes is to develop proof of knowledge systems for humans that prove the information is known, without directly revealing this information by relying on human cognition; These schemes are regarded as Observation Resilient Authentication Schemes (ORAS).

An example of such a scheme is the *Mod10* scheme [6]. The user has a 4-digit PIN as the secret. The challenge consists of a random 4-digit number (communicated through a covert channel). The user computes the modulo 10 sum of each of the four digits in the secret with the corresponding digits in the challenge, submitting the 4-digit remainder as their response. The use of the modulus operation is a common design element in many ORAS (e.g. [2, 7–12]), as it makes them resilient to observation by reducing information leakage as multiple possible secrets would yield the same final response.

In Chapter 3, we shall explore a subcategory of ORAS which often employs the modulus operation, the so called k-out-of-n ORAS. In these schemes, the secret is a mutually agreed upon set of items (between the user and the authentication service) of size k, selected from a larger pool of n items. A challenge contains a random subset of these n items, which are displayed on a device carried by the user. The cognitive function requires, the identification of any of the k secret items that may be sampled. It is important to note that the device itself does not store the user secret, and simply relays messages.

Different realizations of these schemes exist based on how the cognitive function is constructed [2,7,9], among others. An advantage of these schemes is that their security can be quantitatively analyzed by studying the mathematical properties of the cognitive function, and the information leaked through challenge-response pairs. However, this mathematical analysis only considers the records of challenge-response pairs, ignoring the interaction of the user with the "relay" device. when mentally computing the cognitive function. Observing human behavior while interacting with the device is likely to reveal more information about the secret, e.g., if the user dwells over a particular spot on the device's screen. These issues have been raised before [13,14]; however, there is no quantitative analysis of how this human behavior can be exploited to compromise the secret. In Chapter 3, we shall analyze how information obtained from user behavior while processing challenges in a wide class of ORAS (one that employs a modulus operation) can compromise the user's secret. We show how this adversary can launch an attack on these schemes to obtain the user secret after observing far fewer authentication rounds (number of challenge-response pairs) than attacks that only consider challenge-response transcripts.

Departing from the identification tenant of *What you Know*, an increasing number of solutions now allow users to authenticate with *What you Are.* Especially with the integration of biometrics into popular consumer devices. Biometric authentication systems are generally based on either physiological biometrics such as fingerprints [15], face [16,17], and voice [18,19]), or behavioral biometrics such as touch [20] and gait [21], the latter typically used for continuous and implicit authentication of users.

In Chapter 4 we shall consider a machine learning model trained on some user's data accessible as a black-box API for biometric authentication. Given an input (a biometric sample), the model outputs either an accept or reject, as its decision for whether the input belongs to the target user or not. Now imagine an attacker with access to the same API who has never observed the target user's inputs. The goal of the attacker is to impersonate the user by finding any accepting input. What would the success probability of such an attacker be?

These systems are mostly based on machine learning: a binary classifier is trained on the target user's data (positive class) and a subset of data from other users (negative class). This process is used to validate the performance of the machine learning classifier and hence the biometric system [2,21–29]. The resulting proportion of negative samples (other users' data) successfully gaining access (when they should have been rejected) produces the false positive rate (FPR, also referred to as False Acceptance Rate).

The FPR seems to be a good indicator of the success probability of finding an accepting sample. However, this assumes that the adversary is a human who submits samples using the same human computer interface as other users, e.g., a smartphone camera in case of face recognition. When the model is accessible via an API the adversary has more freedom in crafting its samples. This may happen when the biometric service is hosted on the cloud or within a secure enclave on the user's device. In particular, the attacker is free to sample uniform random inputs. It has previously been stated that the success probability of such an attack is exponentially small [30] or it can be derived from the FPR of the system [31,32].

In Chapter 4, we show to the contrary that uniform random inputs are accepted by biometric systems with a probability that is often higher and independent of the FPR. A simple toy example with a single feature can illustrate the reason for the efficacy of the attack. Consider a feature normalized within the interval $[0, 1]$. All of the target user's samples (the positive class) lie in the interval $[0, 0.5)$ and the other users' samples (the negative class) lie in the interval $(0.5, 1]$. A "classifier" decides the decision boundary of 0.5, resulting in identically zero FRR and FPR. However, a random sample has a 50% chance of being accepted by the biometric system. While an oversimplification, the success of the attack shows that the FPR and FRR, metrics used for reporting the accuracy of the classifier, cannot alone be used as proxies for assessing the security of the biometric authentication system.

These authentication systems seek to protect our digital identities, and by revealing potential weaknesses in their design, more robust iterations can be realized to better defend against bad actors. A digital parallel of our own physical world has been quite the feat, with the gradual recording and migration of information to machines. We can benefit from this digital clone with technologies like Machine Learning when applied to this information can optimize and produce tangible improvement to our daily lives. The ability to monetize Machine learning as a service through low-cost APIs trained on private datasets has accelerated the accessibility of machine learning in fields beyond technology, however, this information is at its core is still private information.

Consequently, this has caught the attention of privacy researchers who have previously shown that these models may leak information about the records in the training dataset via membership inference (MI) attacks. In an MI attack, the adversary with API access to the model can use the model's responses on input records of his/her choice to infer whether a target input was part of the training dataset or not. This can be a serious privacy breach when the underlying dataset is sensitive, e.g., medical data, mobility traces, and financial transactions [33,34]. Even the biometric models we will explore in Chapter 4 are vulnerable to such inference attacks, placing our biometric data at risk.

To date, membership inference attacks have been the primary focus of studies that have considered traits of the datasets and machine learning models that impact the attacks' likelihood and accuracy [1, 33–36]. In Chapter 5 we focus on a related, and perhaps a more likely attack in practice, where the adversary with partial background knowledge of a target's record seeks to complete its knowledge of the missing attributes by observing the model's responses. This attack is called *model inversion* [37,38], or in general *attribute inference* (AI) [35]. Yeom et al. [35] provide a formal definition of an AI adversary and argue that this adversary can infer the missing attribute values by using an MI adversary as a subroutine.

Beyond providing a formal definition, Yeom et al. experimentally validate the success of an AI attack on regression models, and conclude that more overfit models, have higher AI attack success [35, §6.3]. In the process to replicate their results on classification models (rather than regression models), where the adversary is given a partial record and its true label, we obtain unexpected results in Chapter 5. We discover that even if the target classification model is susceptible to MI attacks, AI attacks on the same model have a negligible advantage. Furthermore, the results persist even for highly overfitted models. We explore potential root causes in Chapter 5 and find that in order for AI attacks to be successful, the underlying MI attack, when used as a subroutine, should be able to infer membership in a stronger sense. Specifically, the MI attack should be able to distinguish between a member of the training dataset and any non-members that are *close* to that member. We call this, strong membership inference (SMI), parameterized by the distance from the training dataset.

We formulate the notion of SMI, and prove that a successful MI attack does not necessarily mean a successful SMI attack. Furthermore, we also formally show that a successful SMI attack is essential for an AI attack. This result implies that even a standalone AI attack, which does not use an MI attack as a subroutine, is bound to fail if SMI attacks are unsuccessful. We shall experimentally validate these results by evaluating several proposed MI attacks from the literature on several discrete and continuous datasets, and target machine learning models, and show that while these attacks are successful in inferring membership, they fall well short as an SMI attack, and consequently as an AI attack. On the positive side (from an attacker's point of view), we investigate a more relaxed notion of attribute inference, called *approximate attribute inference* (AAI), where the adversary is only tasked with finding attributes *close* to the target attributes, according to a given distance metric. We show that while AI attacks are not applicable, AAI attacks perform significantly better, and improve as the target model becomes more overfit. The AAI notion is also a natural extension of the (exact) AI notion for continuous attributes which has mostly been used in discrete settings [35].

While these Inference attack may appear to be devastating to the applicability of data in machine learning models, there exists techniques that allow for defending against the very inference attacks we have just described. Especially as companies monetizing machine learning models must comply with developing privacy regulations (e.g., EU and USA regulations such as COPPA [39] and GDPR [40], and recently e-Privacy [41] and CCPA [42]).

In order to preserve their models' privacy while still maximizing their ability to produce machine learning models that retain a high utility for their service, data-driven organizations are turning towards privacy-preserving ML (PPML) techniques, building on theoretical frameworks of Differential Privacy [43, 44] (*DP*) and/or Federated Learning [45] (FL). However, differentially private PPML methods often come with an intrinsic tradeoff between utility (e.g., as captured by the accuracy of the model) and the privacy guarantees offered by the technique applied to protect the private data.

A related initial investigation by [46] studies different *DP* compositions, and how these compositions can be applied to the training of a neural network or logistic regression model. [46] reports on the impact these privacy mechanisms have on the model's utility and the effectiveness of inference attacks on the resulting models.

Taking inspiration from [46], and to move towards the goal of understanding the tradeoff between privacy and utility of *DP*-enabled ML methods, we dive deeper into this problem in Chapter 6, setting out to assess how this inherent tradeoff depends on the (1) ML method used, (2) stage in the ML framework where the *DP* method is applied to protect the data or model, and (3) complexity of training data in use with respect to classes and attributes in the data.

In Chapter 6, we shall develop a comprehensive and systematic evaluation of a *DP*-enabled ML framework that enables a privacy ML researcher to study the Utility-Privacy tradeoff in depth for their data at hand. Our objective is to allow the selection of the best performing method yielding the highest predictive accuracy while still ensuring a solid level of privacy protection, by studying the different stages where *DP*-based noise can be applied: as an obfuscation to the *input data*, *during model training*, or at the *model finalization* by perturbing the learned model parameters.

Our exploration will include recent *DP* implementations of classical ML methods such as Naive Bayes, Logistic Regression, Random Forests, and Neural Networks, and will empirically measure their ability to fend off inference attacks we had previously explored in Chapter 5, while also measuring the model's core goal of providing accurate classifications. Crucially, we establish this standard evaluation framework to ensure each of these *DP* implementations are evaluated fairly.

We use both synthetic and real-world datasets to capture the aforementioned privacy and utility tradeoff. Our use of a synthetic dataset enables us to isolate the effects of *DP* noise, stages, and dataset complexity without the influence of data distributions. However, not to discount the importance of standard real-world datasets, we shall also perform our evaluation on a range of real data like CIFAR [47], Purchase [48], and the Netflix dataset [49].

With our experimentation in Chapter 6, we will make the following observations. Most notably, for a given amount of model utility, applying *DP* noise at stages later than the input phase permits the addition of more *DP* noise, thus providing higher privacy guarantees. This observation is consistent across all *DP*-ML algorithms. When considering utility and privacy as the function of the *DP* noise, we identify an "inflection point" for each function, an indicator of where the greatest change in utility and/or privacy will occur for a given *DP*-ML method. We find that this point on privacy function is more closely related to the Utility response, and the *DP*-ML method used, instead of *DP* privacy guarantees, as expected from the amount of *DP* noise applied to the process. Also, the data complexity of the dataset is unlikely to influence the inflection point of the utility or the privacy function. Finally, when privacy or utility comes with constraints, we provide recommendations for the best performing *DP*-ML method, and their expected utility and privacy guarantees.

With the defense work of Chapter 6, we are able to find solace in an ability to defend against attacks on our digital information, albeit at a cost to the performance of the model's performance. This leaves much room for improvement, particularly with the allure of slicing through longstanding problems with new insights learned from large datasets. But on the other side of the blade, it will also be shown that there is also room for improvement in the formulation of attacks, an offensive, and a defensive arms race.

So in summary, in Chapter 2, a literature review of key background concepts will be provided, with more focused and detailed reviews of related work reviewed within each of the respective chapters. Chapter 3 will describe our successful efforts to rapidly recover a user's secret from observation resilient authentication schemes (ORAS), through behavioral side-channels. Next, Chapter 4 will explore the surprising effectiveness of uniform random inputs in breaching the security of behavioral biometric models. We then transition to Chapter 5 where a deep dive of membership and attribute inference attacks are undertaken to highlight the infeasibility of attribute inference due to the inability to perform strong membership inference, paired with a realigned definition of approximate attribute inference to better reflect the privacy risks of an attribute inference attacker. We then explore the privacy - utility tradeoffs offered by differential privacy as a means to mitigate the previous inference attacks in Chapter 6. Before concluding in Chapter 7 with a discussion of the work contained within this thesis, and avenues of future work.

Chapter 2

Literature Review

This thesis presents the goal of developing, understanding, and defending security and privacy attacks created with or against Machine Learning. We shall start our investigations firmly in the realm of Authentication, before generalizing our attacks to a more general setting. Nevertheless, at every stage, as we explore both attacks and defenses, we shall continue to be mindful that there is an inherent trade-off between the perfect security and privacy of a system, and a system that is usable and fit for purpose. (i.e. achieving its original task).

To begin our journey, we shall first review two types of authentication schemes proposed as alternatives to conventional password systems, Cognitive schemes and Biometric schemes. Due to the contrasting nature of Cognitive and Biometric schemes, we will separately describe security attacks against Cognitive and Biometrics schemes. In cognitive scheme attacks we will describe existing algebraic attacks, and behavioral side-channels in exposing a user's secret password. The following sections will provide background about the validation techniques currently employed by biometric authentication schemes, followed by the description of key ML architectures currently leveraged in biometrics. This will be followed by an overview of literature about attacks that compromise the security and privacy of ML models. We then finally discuss differential privacy as the gold standard in protecting the privacy of data.

2.1 Alternative authentication schemes to conventional password systems

Let us now describe how Cognitive authentication and Biometric authentication schemes are defined, and how they function to authenticate a user. Due to a focus on the security and privacy issues about these alternative schemes, we will not detail their advantages and disadvantages over conventional passwords. Additionally, we defer the formal definitions of the respective schemes to Chapter 3 and 4.

2.1.1 Cognitive Authentication

Cognitive authentication schemes are similar to conventional passwords in the sense that there is a shared "secret" between the user (prover) and authentication service provider (verifier), but unlike passwords, this class of scheme leverages the cognitive abilities of the user to prove knowledge of the secret without directly communicating the secret to the verifier, formally known as a zero-knowledge proof. These secret objects take the form of images, or symbols that can be visually displayed, for a given scheme there will be a fixed number of pass-objects, of which a subsample acts as the shared secret between the user and the service provider.

During registration, either the user selects their secret-objects, or the verifier assigns a set of secret-objects to the user. For a login attempt, the verifier issues an authentication challenge, a random subsample of pass-objects, some of which may the part of the user's secret, to the user. The user then takes the authentication challenge and knowledge of their secret to mentally compute a predefined function for a response to return to the verifier. Additional challenges may be issued to the user to lower the likelihood of randomly guessing the response succeeding. Figure 2.1 displays this process between the user and the verifier.

The attacks on cognitive schemes in the following section target a specific subtype of

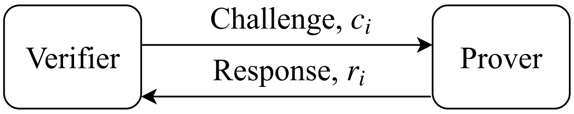


Figure 2.1: Interaction between verifier and prover for a cognitive authentication round

schemes, therefore we explain in depth how this type of scheme works.

Consider the toy example in Figure 2.2, the pass-objects within this example are the lettered tiles, from these 8 tiles, 3 have been selected as the user's secret. During a login challenge, a subset of these 8 tiles are sampled, in this toy example, the 4 tiles to the right are conveniently sampled, assigned random values, and displayed on a screen. The user recognizes that tiles C and G are secret tiles, and will mentally compute the function of addition, with the modulus of 4, $(1 + 3) \bmod 4 = 0$, therefore the returned response is 0.

The reason for the modulus operation is to increase the number of valid tile combinations to hide what the secret actually is. As far as an attacker knows, it is possible that only tile D may have been the user's secret.

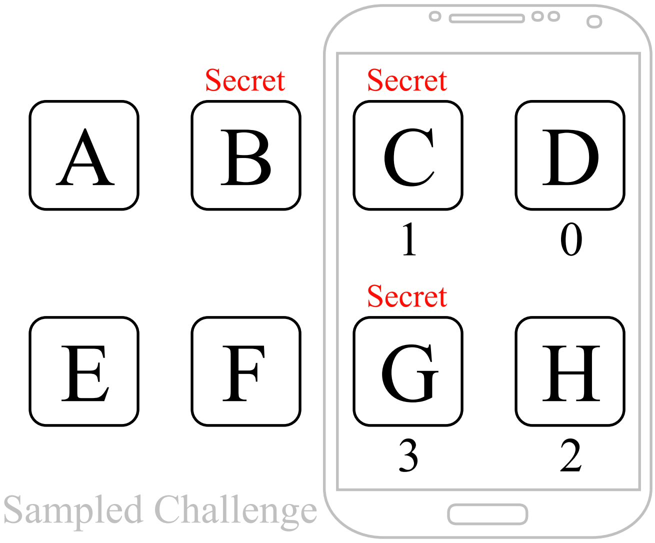


Figure 2.2: Example ORAS Scheme, $(n, k, l, d) = (8, 3, 4, 4)$

2.1.2 Biometrics Authentication

Biometrics is the term used to describe the measurement of human characteristics, therefore biometrics authentication is the utilization of human characteristic measurements to verify the claimed identity of a user. Biometrics can be decomposed into two subcategories: Physiological Biometrics and Behavioral Biometrics, whereby physiological characteristics embody "what" they are, in physical features like fingerprints, facial structure, iris patterns. Behavioral characteristics on the other hand capture "how" they conduct themselves, with measurements like handwriting, walking (gait), voice recognition.

Like the cognitive scheme, biometrics also requires a registration phase whereby a user is prompted to provide multiple biometric samples to train a ML model about the user, or to serve as a reference template of the user. During the login process, a test biometric sample is produced by the prover, this sample is provided to the ML model, or compared to the template to decide the sample's similarity to the registration samples. If the sample is sufficiently similar to the registration samples it is deemed successful and the prover will have been verified. Otherwise, the authentication attempt is rejected. The biometric sample will likely undergo feature extraction, where the biometric measurements have salient features extracted, for example, fingerprints have minutiae extracted before determining similarity.

Details about the verification method of proposed biometric schemes and examples of machine learning models are detailed in Section 2.1.4, whilst Figure 2.3 shows a generic process flow of the biometric samples. More details about specific instantiations of biometrics; Specifically, Face, Voice, Gait, and Touch can be found in Chapter 3.

2.1.3 Security Attacks on Cognitive Schemes

Cognitive schemes are designed from provably secure crypto-systems, however, the simplification of the cryptographic scheme to be mentally processed by a human being allows new attacks to be executed against the scheme. Cognitive schemes share a vital property

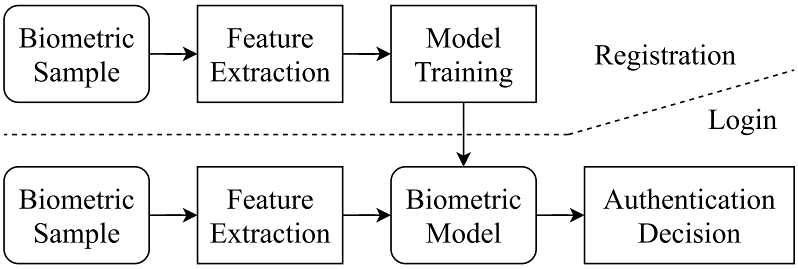


Figure 2.3: Biometric sample process flow during biometrics authentication

with passwords, they can be easily replaced when compromised and do not contain the same individual privacy risk as that of biometric templates. As such we regard attacks against cognitive schemes as strictly a security breach. Inherently Cognitive schemes will leak a small amount of information, but the scheme designer's objective is to maximize the number of observations before the secret needs to be renewed. The approaches below provide methods to harness this leaked information to reconstruct the user's secret. This will be relevant in Chapter 3, as we adopt and augment such approaches with behavioral side-channels to recover the secret with fewer observations.

2.1.3.1 Gaussian Elimination Attack

A Gaussian Elimination attack proposed by Asghar et al. [3] is an algebraic attack targeting a subclass of cognitive authentication protocols that involve the summation of positionally significant numerical weights, the positional information is dependent on the user's secret, and thus only the returning user would possess this information. The summed weights are then reduced to a smaller response space, either by a static mapping or with a modulus operation to increase the number of valid weight combinations that would produce the same user response. The attack observes and records authentication challenges and creates a system of linear equations which is dot produced with an unknown secret vector to produce a vector of recorded responses. This attack however is unable to handle intentional noise introduced to the response by the user, as is part of, and required by the Hopper-Blum protocol [7], one instance of a Cognitive authentication scheme. Gaussian elimination fundamentally works on linear systems, thus when a non-linear transformation, like a static mapping, is introduced within the cognitive function the number of observations greatly increases, due to a need to linearize the mapping step.

2.1.3.2 Frequency Attack

The Frequency Attack, or also known as a counting attack is another algebraic attack originally demonstrated by Yan et al. [50] and later generalized by Asghar et al. [9]. This attack exploits a bias in the resulting responses of the non-linear mapping of the aforementioned summed weights, the bias was such that the valid combination of secrets was more likely to produce a specific final response, thus over a sufficiently large number of observations a distinct separation between the secrets and non-secrets emerge. A mitigation strategy of the frequency attack is to modify the scheme parameters to increase the likelihood of longer combinations appearing in the subsamples challenge, thereby increasing the complexity of the attack by forcing counting in higher dimensions, and requiring more observations.

2.1.3.3 Timing Attack

Cagalj et al.'s work of timing attacks [51] exploits the behavioral side channel of the time required to complete a cognitive authentication challenge. In their attack, they identify that users require less time to mentally compute and respond to a challenge involving smaller PIN digits on a patented Mod10 scheme [6]. In addition to performing their attacks on the Hopper-Blum protocol [7]. Their attack is evaluated on experimentally obtained timing information, with which a table of weights for each possible secret item is updated, allowing for the deduction of the most likely items that form the basis for the target user's real secret over a sufficiently large number of observations. Since this attack, variations on the original Hopper-Blum protocol have been created to address security flaws exposed in critical algebraic attacks [9, 52], and usability limitations [2]. Timing information offers just a single dimension for finding relationships between user behaviors and the user's secret, as such there should exist additional unexplored human behaviors that may leak information about the cognitive secret, as we shall further explore in Chapter 3.

2.1.4 Measuring Biometrics

This thesis will contain a substantial amount of ML for capturing information from behavioral side-channels (Chapter 3), to function as the core component in biometrics schemes (Chapter 4). We shall provide a working overview of various ML techniques and procedures with an emphasis on its use in Biometric systems. Therefore in this section, we will describe metrics used in describing the performance of ML and the established validation techniques of biometric authentication schemes, followed by detailing various Machine Learning (ML) architectures. We shall defer discussion about direct attacks against machine learning models to Section 2.2.2, whereby these attacks are presented in a more general setting.

2.1.4.1 Performance metrics of a classification model

A classification task involves accepting an unknown input signal and making a judgment for which class the sample is more likely to originate from. In the case of *one* and *two* class classification tasks, the sample can either be part of the positive class, or part of the negative class. In the case of multi-class problems, there no longer a binary distinction, instead, averages over all classes may be taken, or a loss function used to measure how "close" the prediction is to the correct answer. The two most common measures of a system's performance are the True Positive Rate (TPR) and the False Positive Rate (FPR). Whereby the TPR measures the proportion of positive samples that have been correctly re-identified as positive, and the FPR is the proportion of negative samples which have been incorrectly identified as positive samples.

The False Negative Rate (FNR) is the proportion of positive samples that have been incorrectly labeled as negative and is closely related to the TPR, FNR = 1 - TPR. Similarly, for the True Negative Rate, the proportion of negative samples correctly identified as negative is related to the FPR, TNR = 1 - FPR.

When tuning the performance of a classifier, there is a trade-off between maximizing the TPR and minimizing the FPR, if the decision boundary is relaxed, more of the positive class samples are likely to be correctly identified, increasing TPR; however, the same relaxed boundary is more likely to accept negative samples, increasing the FPR. One means of deciding the final configuration of a biometric classifier is by taking the Equal Error Rate (EER), in which the FNR equals the FPR, thereby minimizing the errors of both the positive and negative classes. Multi-class systems can have their hyper-parameters tuned to minimize the aforementioned loss.

These metrics can be translated into real-world impacts on an authentication scheme's application. The TPR is a measure of the proportion that a returning user is correctly identified, this is one measure of the scheme's usability, as incorrectly rejecting a user will cause frustration in its use. FPR on the other hand is a measure of incorrectly accepting impostors, and therefore can be taken as a measure of security to prevent unauthorized access.

2.1.4.2 Validation Techniques

A proposed biometric authentication scheme needs to be validated in offering the claimed benefits, different validation techniques are used in response to increasingly capable attackers. We outline these validation techniques and relate them to the adversary the schemes were designed to protect against. Providing a light overview in preparation for Chapter 4.

Zero-Effort Attack The Zero-Effort Attack is a method of testing a biometric model by taking samples from other enrolled users and subjecting a target user's model to these samples. The samples from the other users form the negative test set, with samples from the target user withheld from training in a testing/training split, creating the positive test set. By comparing the model predictions with the ground truth of the test samples a TPR and FPR can be obtained. The type of adversary the zero-effort attack simulates is an attacker who has gained access to a secured device and simply performs an action with no other knowledge of the user in an attempt to gain access.

This is the most commonly used validation technique, as the biometric is often used in tandem with a physical token, such as the user's smartphone or an identification card to increase the difficulty of the attack. From the perspective of the scheme proposer, this method of validation does not require a secondary round of experiments with simulated adversaries to perform an informed attack on target users.

Shoulder-Surfer A Shoulder-Surfer attack provides an adversary with additional information about their target. The attacker is permitted to view a limited number of authentication attempts to gleam either the password or information about how the biometric was performed, before an attempt of replicating the target user's sample. This method of validation is typically used when a proposed biometric scheme is explicitly claiming observation-resilience [2].

Skilled Adversary A skilled adversary is an attacker who has been trained in replicating the biometric samples of a biometric modality, this type of adversary is commonly found in signature and handwriting based authentication schemes [53, 54]. A spoofed fingerprint or face mask physical recreation would also be considered an attack by a skilled adversary [55]. The performance metrics remain the same as the previous two methodologies, however, the quality of the negative test samples are increased, and thus increasingly similar to the positive samples.

2.2 Machine Learning

The core objective of Machine Learning is to learn trends or relationships about a data domain, from a representative dataset sampled from said domain. We have seen its employ in Biometric Authentication systems as an alternative to conventional password systems and in an increasing number of applications that seek to harness the mountains of data collected by governments and companies. In this section, we will provide an introduction to a few different machine learning architectures, finally reflecting. The subsequent section will expand on the possibility of security and privacy attacks on Machine Learning, leading us to the next sections where we expand on Active and Passive attacks against machine learning models.

2.2.1 Machine Learning Architectures

Machine Learning is an umbrella term describing the capability of a "machine" to learn trends from a dataset. The algorithm used by the machine to understand these trends determines the complexity, training time, and performance of the model. This learning can either be supervised or unsupervised. Supervised learners are provided the input data in addition to the correct classification label to separate the provided classes. Unsupervised learners, however, are only provided the input data with no label and are tasked to draw relationships between clusters from within the dataset. Within the domain of authentication, a majority of schemes involve supervised learning, however, unsupervised models are not unheard of [27, 56], instead finding greater application in identification from a group of users. For the remainder of this review, we refer to supervised learning.

Distance-Based Learners A distance-based learner does not create an explicit model from the training data in the same manner as the subsequent architectures, however is still loosely regarded as ML. This learner maintains a template of the positive training sample(s) and computes a similarity metric (distance) with the test sample, depending on if the distance between the positive template and the test sample exceeds a predetermined threshold, the sample is accepted or rejected. Examples of distance functions in authentication include Dynamic Time Warping (DTW) [2], Cosine Similarity [57] and Euclidean Distance [58].

Instead of using a threshold, the model designer may leverage k Nearest Neighbors (kNN), whereby the class of a test sample is determined by the majority label of the k nearest training samples by distance. Distance based learners are considered for their simplicity and transparent structures.

Naive Bayes The Naive Bayes classifier is a family of probabilistic classifiers that seeks to model the conditional probability between a set of inputs and predetermine output classes. The Naive Bayes family of classifiers are dubbed naive from the strong assumption of independence between the features of the input.

$$P(y|\mathbf{x}) = P(y)\frac{P(\mathbf{x}|y)}{P(\mathbf{x})}$$

Trees The basic Decision Tree (DT) structure is used to separate samples into their respective classes based on decisions about their feature values. With a single root node decision on a feature value is made to mode left or right of the node to a lower leaf node, at every subsequent node another division can be made to separate the leaf node into two more leaf nodes, this is repeated until all samples and their classes are separated. If a large number of leaves at the bottom of the tree exists the model may have overfitted to the training data, and thus perform poorly on new unseen test data. As such a technique known as pruning, or the removal of branches is performed as to not have overly specific final nodes.

An ensemble version of the DT exists, known as Random Forests, this variant constructs multiple decision trees (like that of a forest) by taking random subsamples of the training dataset, known as bagging. The bagging decreases the likelihood of the trained model overfitting to the training data and hence will perform more favorably on the testing data.

A final decision is reached by the ensemble of DTs by taking the result with the highest confidence, the label with the majority vote [58, 59].

Support Vector Machines Support Vector Machines (SVM) are a type of ML model which aims to construct a decision boundary between the class labels. It uses support vectors, also known as the samples within the training set that is the closest to samples of another class, to choose a position for constructing a boundary that maximizes the distance of separation between the support vectors and the created decision boundary. The shape of the decision boundary is dependent on the kernel chosen by the trainer, it is often Linear, Radial, or Polynomial in nature. This choice of kernel should be informed by the characteristics of the data to be classified, as a sub-optimal choice of kernel will be unable to create a good separation of the data leading to large classification errors. Depending on which side of the decision boundary a test sample lies, a classification is made and returned [60, 61]. SVM has the ability for one-class classification, where the decision is if a test sample falls within the single class or not [62, 63].

Neural Networks [64] A Neural Network (NN) simulates the same neural processes within the human brain for learning a given task. An input vector is transformed into a series of hidden internal layers through weighted summations at each layer before producing a final output. The weights of the summations are updated with training data fed forwards through the network to minimize the error of it's prediction on the output.

Deep Learning [65, 66] has the same neural structure as a neural network, however, the internal layers have a hierarchical structure, such that each progressive layer represents an increasingly abstract set of features automatically derived from the input data. As the abstracted internal layers of these networks represent unknown features that are automatically derived, there exists a possibility that these learned features parallel secondary information from within the dataset.

Federated Learning [67–69] takes the humble Neural network and distributes the learning task across multiple devices. By distributing the learning task, the resource demand on each individual device is reduced in comparison to if the model had been trained in a centralized location. Federated learning is also appealing due to the removal of the need to transmit and aggregate data at the central location for learning. This is an appealing advantage from the perspective of privacy as the data does not need to leave the device, instead, these devices only sending learned updates back to a global coordinator that manages aggregates the updates from all the collaborating devices. Once all the updates have been brought together, the global coordinator will transmit a final update back to the devices to maintain an updated and consistent model.

2.2.2 Security and Privacy Attacks on Machine Learning

With biometric systems as an example, it can be said that one of the earliest attacks on biometric systems involved skilled forgers reproducing signatures to fool a human being. By fooling this human to accept an erroneous signature as correct, we have achieved a security breach. We have observed the same objective of breaching security in machine learning, an attack that results in an ML model performing incorrectly allowing a negative sample to be accepted as a positive sample.

However, the alternative is also equally valid as an objective, that is to induce the classification of a positive input as negative. This alternate goal performs an effective denial of service for the positive users of the system. Degrading the overall performance, and rendering the service useless.

To date, there have been many attacks against Machine Learning proposed, with objectives evolving beyond simple misclassification.

To better understand the relationships between the attacks we will group the attack into either an active or passive attack depending on how it interacts with the model during the attack. For each attack, we will also describe the objective, and how each of these attacks are performed.

2.3 Active Attacks

We regard active attacks against machine learning models as an attack that requires the attacker to directly interfere with the training process of the model.

2.3.1 Poisoning Attacks

The objective of Poisoning attacks is to degrade the utility (predictive performance) of a given model. By inserting poisoned samples into the training data, the model will attempt to minimize the training loss to all samples included within its training dataset, including the poisoned samples (which when learning on will increase the loss of the overall objective).

Poisoning attacks are possible due the to model treating every sample of the training dataset as equally important. We will remark that if the attacker has access to the entire training dataset and can replace the entire dataset, the attack becomes trivial. However, to avoid detection, while still degrading performance, the attacker must reduce the number of poisoning samples needed to create the greatest change in the models' behavior [70].

2.3.2 Backdoor Attacks

Like the poisoning attack, a backdoor attack seeks to "poison" the model with a strong association between a pre-determined input (called a *Trigger*) and their desired output, with little to no impact on the behavior of the model when this input is not present. The input and output are dependent on the learning task, for example, in computer vision tasks, the input to a model is an image, if a small, but specific set of pixels on this image were replaced by the trigger, then the model would output the attacker's label, disregarding the content of the remaining image pixels [71]. Alternatively, consider a Natural language processing task, whereby the input is text, and the output can be a label of sentiment analysis, a translation, or question answering, by leveraging homograph replaced characters, or appended sentences as the trigger, the attacker can induce incorrect predictions, translations and answers respectively [72].

2.4 Passive Attacks

In passive attacks, we have attacks that do not directly interfere with the training process of a model, yet they are still able to compromise either the security or privacy of the model.

2.4.1 Adversarial Examples

Adversarial examples consist of a small perturbation that may be added to a test sample to cause the ML model to make a misclassification. These small perturbations can be undetectable by humans, however create a substantial error within the model's prediction [73, 74].

External to the authentication, adversarial examples have been applied to computer vision models which leverage deep learning to perform a prediction task such as recognizing street signs [75], or handwritten digits [76]. These adversarial examples have also been utilized to fool facial recognition scanners with 3d-printed glasses containing perturbations [77], directly compromising a biometric scheme's security.

An there are a few possible attack objectives when finding adversarial examples. One possible objective is to minimize the size of the permutation to maintain undetectability by humans, for example, a stop sign that still looks like a stop sign to a human observer, however, is interpreted as a right turn only sign by the model [75]. A second objective is the targeted ability of the sample, originally the task was to simply produce a misclassification, however, a more difficult problem is to find a sample that still appears to a human in its original form, however, the misclassification will produce a specific result. Under the previous attack setting, the stop sign classified as anything but a stop sign would be considered a successful attack, however, in a targeted attack, the attack is only considered successful if the stop sign is specifically interpreted as a specific predetermined target sign.

2.4.2 Transferability

Transferability is a property of the aforementioned adversarial examples, which observes the ability to transfer an adversarial example that is successful in deceiving one ML model to another ML model performing the same classification objective, creating another misclassification on the second model [76, 78]. Surprisingly, this transferability property has also been shown to persist across different architectures of ML models with the same classification objective [76]. The property implies that any developed attack on security has the possibility of also possessing this property and should be evaluated against multiple ML architectures. It has also been shown that a transferable adversarial example persists in fooling a ML model whence recaptured through the physical domain [79].

2.5 Privacy Attacks on Machine Learning

We define privacy breaches in machine learning as the ability for an attacker to learn information about the model and/or training data beyond its original trained intention. The scope of the following attacks occurs post-training, as we assume an attacker is unlikely to have modification access to the training dataset before training of the ML model. There exists a taxonomy of the Security and Privacy efforts in ML by Papernot et al, which elaborates on both attacks and defenses [80]. However here we shall present the most relevant topics to this thesis.

2.5.1 Model attacks

A model attack is capable of inferring information about the underlying training dataset, beyond the intent of the ML model.

One example of this attack is [81] by Ateniese et al., who were able to detect the presence of biases within the training data of a ML model unrelated to the objective of the model. They study two model objectives, the first, a speech-to-text recognition model, followed by an internet packet type classifier. Multiple models are constructed however some models were created with biases within their training datasets, the biases manifesting as if the speech originates from a speaker with an Indian accent, and for the packet classifier, the bias is if all packets originate from a specific destination, Google. The novelty of this attack is that they do not observe the input/output relationship of the ML model, instead, representing the learned parameters of the ML model, and using the parametric representation of the model as an input to an attack model to determine if the bias is present within the dataset.

2.5.2 Inversion attacks

An inversion attack is an attack that can recover sensitive information about an individual's training samples, provided partially known information about its target, and confidence outputs from a target ML model. The work of Fredrikson et al. exploits decision trees trained to predict an attribute about extramarital relationships, this sensitive attribute is recovered for individuals contained within the training dataset, with known non-sensitive values. Additionally, they can reconstruct the facial images used for the training of a facial recognition deep learner [37].

2.5.3 Model Extraction

Model extraction attack is one where an attacker, who is provided access to the ML to perform classification tasks, given a series of carefully crafted input samples and the corresponding output labels and confidences, the attacker can “steal” the machine learning model for their own use. Tramer et al. pioneers this attack [82], motivated by the development of pay-per classification services with models trained on private datasets, stealing the ML model would effectively deny revenue for the service provider. Whilst this is not a direct attack on privacy, the ability to steal the ML model allows for an unbounded number of queries towards the ML model, breaching any imposed “privacy budget” on query limitations.

2.6 Inference Attacks

Inference attacks shall be explored in depth in Chapter 5, as such we review different techniques to achieve these inference attacks. We will focus on providing an accessible high-level interpretation of these attacks while providing formal definitions and implementation details within Chapter 5.

Inference attacks are attacks that can infer additional information from a ML model that would otherwise not be available in the intended use of the ML model. Within this category of attacks, there are two distinct attack objectives, with an intertwined approach. These are the Membership Inference attack and the Attribute Inference attack.

The Membership Inference attack seeks to infer the membership of a given input vector, specifically, if the given vector input has been used for learning during the model’s training process. The privacy implications of a membership inference attack are subtle, as it is often dependent on the context of the data. For example, if the training dataset consists of data from a medical study, an attacker possessing information about a person, through a membership inference attack can infer if they were part of this medical study, if membership is detected, the attacker could infer if this person had a specific type of disease.

On the other hand Attribute Inference attacks seek to leverage partial information about a datapoint, and a model trained with the datapoint to reconstruct the missing information the attacker did not possess about the datapoint. In this setting, the privacy implication is clear. The attacker gains information in the form of datapoint attributes it previously did not possess.

With these considerations in mind, we journey through the different methods of performing Membership Inference and Attribute Inference, on the way highlighting existing similarities between the two. Be wary that each approach will require a differing amount of information or computation to succeed, which consequently results in differing attack effectiveness.

2.6.1 Membership Inference Attack

Recall that the overall objective of a Membership Inference attack is to determine if a given vector was used in the training process of a model.

In all scenarios described below, even in a white-box attacker setting, the training data strictly remains unknown to the attacker. If not the attack becomes trivial. This is in contrast to the white box setting that may be observed in other attacks against ML, specifically Poisoning and Backdoor attacks. The three examples we shall overview below are black-box attacks, though it is noted that grey-box (partial knowledge of the model) and white-box (full knowledge of the model) exist. In essence, white and grey-box attacks have access to more information at their disposal for inferring membership, consequently, we shall defer the details of one such white-box attack to Chapter 5.

2.6.1.1 Shadow Model Membership Inference (Shokri et al. [33])

The original inference attack is Shokri et al's work on Membership Inference. In this seminal work, Shokri et al.'s methodology [33] to achieve this attack is unique in that they introduce the concept of "Shadow Models" that are trained locally with one shadow model for each output label of the target model.

From each of these shadow models, the outputs are then provided to an additional attack model which learns from the outputs of the local Shadow models in predicting if a given sample was part of the original training dataset. By having additional shadow models, the attacker can obtain prediction confidence values for every classification class, a value that may not have been previously available from the targeted multi-class model. The intuition resulting in the success of the attack is with access to more confidence values, the attack ML model can observe more subtle changes in the model confidence values to learn about membership indicators.

The unfortunate downside of using Shadow Models is an upfront demand of training data from the same distribution as the target model. However, this requirement can be alleviated by the transfer of knowledge from an attack model trained on a differing distribution to the target distribution [34]. Additionally, [34] further reduces the data and computational demand of multiple shadow models (one for each class) by observing that a local multi-class classifier can provide the same granularity of confidence information for the attack model.

2.6.1.2 Loss Membership Inference [35]

Yeom et al. provide an alternative approach to Membership Inference while alleviating the need for Shadow models and the attack model altogether. Instead, this approach uses the loss of the predicted class (in classification) or predicted value (in regression) as a singular indicator of the membership status of a vector. The intuition behind this attack is that a vector used in the training process should produce a similar loss to the training loss observed during the training process. Understandably, the true label of the vector is required to compute this loss.

In practice, the attacker would use the training loss as a threshold for determining if a vector's loss makes it behave like a member or non-member.

2.6.1.3 Confidence Membership Inference [34]

Salem et al. [34] simplifies the approach even further compared to the Loss Membership Inference attack [35]. In a setting where the model may not return the prediction probabilities of every output class, or if the attacker does not have the ground truth class label to compute the loss with, this approach offers a viable method for performing membership inference.

The method simply leverages the maximum confidence value returned by the model and uses this value as a metric to gauge membership. The maximum confidence value will always be returned for a model that only returns a single prediction class (the prediction class returned will be the class with the highest confidence, and thus the highest prediction returned). The exception to this is for models that only return labels, however, this is an emerging area of research [83, 84]. The attack can leverage the maximum confidence for membership based on the simple intuition that an input a model has previously seen before (during training) will have already observed and learned the correct label, and will thus make the same prediction with very high confidence. Therefore the expectation is that members will have higher maximum confidence than non-members. In a practical attack, the threshold between members and non-members can be found before attack time either through a disjoint dataset of the same distribution or by transferring knowledge from a dataset of a differing distribution.

2.6.2 Attribute Inference Attacks

The core subject of Yeom et al.'s work is the relationship between membership inference and attribute inference and their joint reliance on overfitting. Yeom et al. demonstrate changes in the effectiveness of both attacks with variations in the "fit" of the model. Though we note that the attribute inference attack targets a regression model in [35]. As we shall explore later in Chapter 5, this is not the case for classification models.

At a high level, attribute inference seeks to infer information about a partial vector with access to a machine learning model.

The attribute inference attacks explored within this body of work are attribute inference attacks with membership inference as a subroutine. Simply, membership inference is evaluated on a set of all possible permutations of an input constructed from a partially known input vector. For example, consider a binary input vector of length 2 (b_0, b_1), if provided with $b_1 = 1$, then the two possible permutations of $(b_0, b_1)|b_1 = 1$ is $\{(0, 1), (1, 1)\}$. Membership inference is applied to both vectors, if one of the two vectors were part of the training dataset, then this one alone would be successfully detected as the member, whilst the other a non-member. From this, we can infer the member vector's b_0 attribute. In practice, vectors will be longer, have more possible values (e.g. continuous variables), more missing information (which in turn increases the search space), a vector's Label these factors including indirect factors of the "Fit" of the model, the Fairness of the model will also have an impact on the ability for an attacker to recover both membership and attribute information.

2.6.3 Positive applications of Inference Attacks

We have now reviewed various methods of inferring additional information from a machine learning model that was not intended in its original application. However, finding this information may also be beneficial for actors of justice that may seek to find if privacy and data agreements have been breached. Specifically, consider a company collecting personal information, or a data sharing agreement for a specific task A. However, the company takes the data it has received and uses it for a loosely related task B (a real possibility in the realm of Transfer Learning [85]). With access to the model of task B, an auditor could determine if data had been used beyond its original purpose, despite the obscuring of the exact record during the abstraction of rules from the training dataset. With an ability to audit the use of data within models, strong evidence can be provided to litigate breaches in data agreements.

There exist two such examples of auditing machine learning models. The auditing of Text Generation systems [86] and Automatic Speech Recognition [87]. Both these works utilize the shadow model and attack approach [33]. As previously stated [86] audits if a textual record was used in the training of the text generation model; However, [87] has a diverging objective, not only to infer the membership of a specific audio sample but to infer if any audio samples from a given user were used during the training process. The auditing of a user was successful [87], unfortunately, the exact reason was not explored at length in this work, though it is suggested that audio samples from a single user are sufficiently similar for membership inference to perform user membership inference. We shall later provide a concrete study as to why this approach works in Chapter 5, in essence, we prove the current generation of membership inference attacks are unable to successfully distinguish between two sufficiently similar samples, and stipulate that a strong membership inference attacker would not suffer the indistinguishability of close vectors.

2.7 Differential Privacy

Without diving too deep into the definitions, Differential privacy (DP) mathematically defines the protection offered in regards to the privacy of a single vector, whether that is representative of an individual, or a single temporal event [43]. ϵ-differential privacy is defined such that two neighboring sets of data D and D', differing by a single vector are indistinguishable up to a limit as described by a privacy budget ϵ. The output of a mechanism $\mathcal{M}$ applied on each dataset should also be indistinguishable from each other, up to our limit of ϵ. In other words, if differential privacy is applied to a dataset, all subsequent operations on this dataset still offer the same privacy guarantees as before. Alternatively, if differential privacy is applied during the learning process, or to a trained model, the original dataset will still be offered the same privacy guarantees.

There have been numerous relaxations to this foundational definition of differential privacy, of which more detail can be found in Chapter 6. However, in general, the relaxations provide flexibility to allow for a small amount of error in its offered protection. This flexibility decreases the aggressiveness of needing to protecting every data point, and thus permits less differential private noise to be applied. This in turn retains more of the original information of the dataset, and thus an increased utility from said data.

As a defense mechanism against the inference attacks of Section 2.6, with the added noise to the original datasets, an inference attacker cannot distinguish between the original and the now perturbed vector.

Chapter 3

Exploiting Behavioral Side Channels in Observation Resilient Cognitive Authentication Schemes

This chapter is adapted from work titled "Exploiting Behavioral Side Channels in Observation Resilient Cognitive Authentication Schemes", published in the journal ACM Transactions on Privacy and Security (TOPS), 24(1), pp.1-33, completed in conjunction with Zhao, B.Z.H., Asghar, H.J., Kaafar, M.A., Trevisan, F. and Yuan, H.

Observation Resilient Authentication Schemes (ORAS) are a class of shared secret challenge-response identification schemes where a user mentally computes the response via a cognitive function to authenticate themself such that eavesdroppers cannot readily extract the secret. Security evaluation of ORAS generally involves quantifying information leaked via observed challenge-response pairs. However, little work has evaluated information leaked via human behavior while interacting with these schemes. A common way to achieve observation resilience is by including a modulus operation in the cognitive function. This minimizes the information leaked about the secret due to the many-to-one map from the set of possible secrets to a given response. In this chapter, we show that user behavior can be used as a side-channel to obtain the secret in such ORAS. Specifically, the user's eye-movement patterns and associated timing information can deduce whether a modulus operation was performed (a fundamental design element), to leak information about the secret. We further show that the secret can still be retrieved if the deduction is erroneous, a more likely case in practice. We treat the vulnerability analytically, and propose a generic attack algorithm that iteratively obtains the secret despite the "faulty" modulus information. We demonstrate the attack on five ORAS, and show that the secret can be retrieved with considerably less challenge-response pairs than non-side-channel attacks (e.g., algebraic/statistical attacks). In particular, our attack is applicable on Mod10, a one-time-pad based scheme, for which no non-side-channel attack exists. We field test our attack with a small-scale eye-tracking user study.

3.1 Introduction

A longstanding issue with the prevailing methods of authenticating users via passwords and PINs is their vulnerability to observation. The user secrets (password or PIN) is entirely compromised after a single observation via, for instance, shoulder-surfing or a hidden camera. A growing number of reported incidents indicate that this is not just a theoretical vulnerability [5], prompting widespread proposals for alternative authentication schemes. These include biometric authentication (fingerprint, iris, etc.) and one-time passwords, either as standalone systems or in a multi-factor configuration alongside passwords. Another alternative is observation resilient challenge-response authentication schemes that rely on human cognition. In such schemes, the verifier (service provider) prompts the user to prove possession of a shared secret, through a series of challenges to whom the user has to respond to by (mentally) computing some cognitive function. The cognitive function is designed in a way that an eavesdropping adversary needs to observe multiple challenge-response pairs to retrieve the secret. We call these schemes observation resilient authentication schemes (ORAS).

An example of such schemes is the *Mod10* scheme [6]. The user has a 4-digit PIN as the secret. The challenge consists of a random 4-digit number (communicated through a covert channel). The user computes the modulo 10 sum of each of the four digits in the secret with the corresponding digits in the challenge, and submits the 4-digit response. The use of the modulus operation is a common design element in many ORAS (e.g. [2, 7–12]), as it makes them resilient to observation by reducing information leakage. For instance, the dot product of a random binary secret vector with a public binary vector, i.e., the challenge, leaks more information about the secret compared to when the dot product is reduced modulo 2 (only revealing its parity).

An interesting subcategory of ORAS which often employs the modulus operation is the so called k-out-of-n ORAS. In these schemes, the secret is a mutually agreed upon set of items (between the user and the service) of cardinality k, selected from a larger pool of n items. A challenge contains a random subset of the n items, which is displayed on a device carried by the user. The cognitive function requires, at least, the identification of any of the k secret items. The device itself does not store the user secret, and simply serves as an intermediary, relaying messages. Different realizations of these schemes exist based on how the cognitive function is constructed (which should be easy enough for the user to perform mentally). Examples of such schemes include the Hopper and Blum (HB) scheme [7], the (modified) FoxTail scheme (FT) [9], and BehavioCog (BC) [2], among others (see Section 3.2.3 for scheme descriptions). An advantage of these schemes is that their security can be quantitatively analyzed by studying the mathematical properties of the cognitive function, and the information leaked through challenge-response pairs.

The security analysis accompanying the proposals of almost all ORAS, including k-out-of-n variants, only considers “flat transcripts” of challenges-response pairs, ignoring entirely the interaction of the user with the “relay” device during computation of the cognitive function. Observing human behavior while interacting with the device is likely to reveal more information about the secret, e.g., if the user dwells over a particular spot on the device’s screen. These issues have been raised before [13, 14]; however there is no quantitative analysis of how such human behavior can be exploited to compromise the secret, barring some work on timing attacks which exploits the variation in time taken by humans when responding to challenges [51]. In this chapter, we analyze how information obtained from user behavior while processing challenges in a wide class of ORAS (one that employs a modulus operation) can compromise the user's secret. In particular, we consider an adversary which can not only observe challenge-response transcripts but also user's eye movements with varying accuracy. This information could be obtained through pinhole cameras like those found on ATMs [5], and does not require an adversary to have control of the device's camera. We show how this adversary can launch an attack on these schemes to obtain the user secret after observing far fewer authentication rounds (number of challenge-response pairs) than attacks which only consider challenge-response transcripts.

In more detail, the main contributions of this chapter are as follows:

- We analyze a wide class of ORAS in which the cognitive function involves a modulus operation. By using a generic k-out-of-n ORAS, we show in Section 3.3 that certain responses are more likely a result of a modulus or a non-modulus operation.[1] Furthermore, we show that knowing whether a modulus operation was performed or not in a given challenge can leak information that can lead to quicker retrieval of the secret.
- We propose an algorithm to obtain the user's secret using possibly faulty information about whether the *modulus event* has occurred or not in three proposed k-out-of-n ORAS from the literature (BehavioCog, FoxTail and HB). By simulating varying degrees of information accuracy about the modulus event, we show that the resulting attack retrieves the user secret in far fewer authentication rounds (challenge-response pairs) than (efficient) non-side-channel attacks, e.g., Gaussian elimination. For instance, even with an imbalanced simulated accuracy of 1.0 in detecting the modulus, and 0.6 in detecting non-modulus events, we can find the user's secret in 280 rounds for BehavioCog, 390 for FoxTail, and 909 for HB. This reduces to 474 rounds for

[1] e.g., consider the sum of two integers modulo 10. The sum $5 + 6$ requires a modulus operation, whereas $5 + 3$ does not.

BehavioCog, 666 for Foxtail, and 1555 for HB, when the latter is reduced to 0.35. In comparison, efficient algebraic attacks on these schemes (Gaussian elimination) require 900 rounds for BehavioCog [2] and 16,290 rounds for FoxTail [9], whereas the HB scheme does not have any efficient attack and hence no bound on the number of rounds. These results are shown in Section 3.4.

- In Section 3.5, we perform a small-scale eye-tracking user study with 11 users on BehavioCog, as a field test to evaluate how a user's eye-movement behavior during challenges can potentially expose information about the secret by indicating a modulus/no-modulus event. We identify and derive behavioral features from the eye-movement side-channel, e.g. total challenge time, and duration of last fixation. Using these features we train classifiers to predict the modulus and no-modulus events. We use leave-one-user-out verification to demonstrate event-specific behavioral information independent of users, and to avoid over-fitting user-specific behaviors.

- Continued in Section 3.5, we demonstrate real-world attack feasibility on k-out-of-n ORAS by considering adversaries with varying technological capabilities. Four adversarial levels are considered linked to the detail of information available; from the coarsest—only timing information, to the finest—timing information dwelling on a specific item. This information is obtainable with access to a camera directed at the user's face [88], a likely scenario with covert pinhole cameras already found in instances of ATM skimming [5]. In comparison to the aforementioned efficient algebraic attacks, we can deduce user secrets in 435, 589, and 1,346 rounds in BehavioCog, FoxTail and HB, respectively.[2]

- Finally in Section 3.6, we demonstrate that our attacks are applicable to other ORAS as well (not just k-out-of-n variants) as long as the cognitive function involves a modulus operation. Specifically, we evaluate the attack on PassGrids [11], a locations and modulo arithmetic based scheme. We also implicate the Mod-10 scheme [6],

[2] These numbers of required rounds (reported in Section 3.5) are from accuracy levels obtained via the user study. This is in contrast to the rounds required from simulated accuracy levels (reported in Section 3.4) as mentioned before. See Section 3.5.4 for the reason behind the discrepancies in the reported number of rounds.

which is necessarily a one-time pad using a covert channel to communicate the pad. Being a one-time pad, the scheme is secure against an unlimited number of authentication rounds observed. However, we show that the secret can be compromised with knowledge of the modulus event (obtainable through user behavioral information). With a 10% prediction error, it only takes an average of 36.1 rounds, to compromise a 4-digit PIN in Mod-10.

Compared to algebraic attacks (which only require passive observation of challenge-response pairs), behavioral side channels do require more effort from an attackers point of view. However, obtaining the resolution of user's eye movement information required in our attacks is not difficult given today's technology, and the attack in practice can be launched without much difficulty (e.g., by placing hidden cameras on frequently visited spots). Furthermore, we show that even limited side-channel information such as time to respond to challenges is enough to retrieve the user's secret. This information can be obtained even without hidden cameras. Our attacks suggest that the design of ORAS should explicitly consider user behavior while executing the schemes, as a threat and source of information leakage about the secret especially since these schemes are purported to be observation resilient.

The rest of the chapter is laid out as follows: Section 3.2 summarizes what an Observation Resilient Authentication Scheme is and how it functions. In Section 3.3, modulus-related biases are mathematically analyzed. The proposal and simulation of algorithms to exploit faulty oracle information is presented in Section 3.4. A realization of the attack is performed with eye-movement side-channel information obtained from an eye-tracking user study, to field-test an attacker's capabilities in Section 3.5. Finally our extension of the bias onto other authentication schemes is in Section 3.6.

3.2 Background

3.2.1 Observation Resilient Authentication Schemes

A (human) authentication scheme is a shared secret challenge-response authentication scheme consisting of a setup phase and an authentication phase. In the setup phase a secret S is shared between the prover (user) and a verifier (the authentication service). The authentication phase involves a series of challenges c from the verifier (displayed on the user's device) and responses r from the user, whereby the user mentally computes a public function f of c and S, returning the response r to the verifier. We shall call each challenge together with its corresponding response as a challenge-response pair or a challenge-response round, interchangeably. After a specified number of challenge-response rounds, the verifier accepts the user if the responses are correct; otherwise the user is rejected.

Threat Model We consider an eavesdropping adversary who can observe the interactions between the user and the server (during the authentication phase). Most prior work models this as giving the adversary one or more challenge-response pairs from the authentication phase. We extend this by also allowing the adversary to observe the interaction between the user and its device during the authentication phase (Figure 3.1). The *transcript* of a challenge-response round is defined as this entire interaction: from challenge receipt, user interaction with the device during computation of f, to response submission.

Observation Resilience A human authentication scheme is called observation resilient (ORAS) if no adversary (probabilistic polynomial time algorithm) can extract the secret with probability 1, after observing one or more challenge-response pairs. Note that this definition merely states what qualifies for an ORAS and does not reflect on the security of the ORAS. Indeed, an ORAS might only be secure for a few observations, before the secret can be extracted. For an ORAS to be secure, the probability of finding the secret should be small (or negligible) for a large number of challenge-response pairs. Since each challenge-response pair leaks some information about the secret, the goal of the designer is to use the ORAS for as many challenge-response rounds as possible before the adversary can extract the secret with non-negligible probability. Note that password-based authentication is not observation resilient under this definition, as the secret is recovered after one observation.

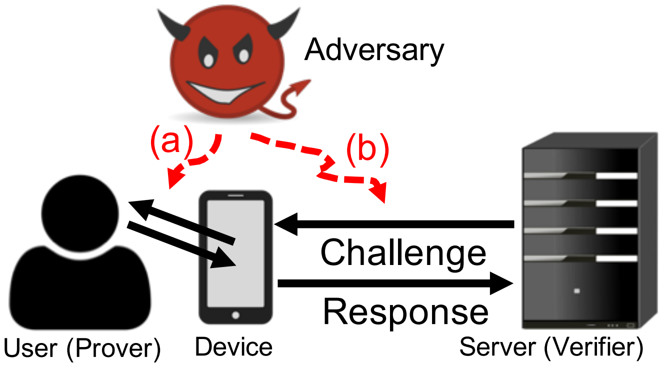


Figure 3.1: The threat model under consideration. Adversary can also observe the interaction between the user and the device.

3.2.2 k-out-of-n ORAS

In one class of ORAS the secret S is a random set of k items from a set of n (publicly known) items[3]. The elements of S are called the secret items, and the remaining $n - k$ items will be referred to as decoy items. We shall call these k-out-of-n ORAS. Different designs of these ORAS exist. In the following we focus on particular design elements which are promising in terms of both resistance to known attacks and usability (especially if employed in conjunction with other authentication factors such as behavioral biometrics). These design elements are:

- *Windowed Challenges:* A challenge is constructed by randomly selecting l out of n items. This can be visualized as a fixed window capable of enclosing l items. The set of n items is randomly shuffled each time, and the l items within the window are the challenge items (hence the name). If the *window size* l is small, the user can recognize

[3] Examples of items are images [89–91] or emoticons [2, 9, 13].

its secret items present in the challenge in a short amount of time. This is also desirable for deployment as a small set can be easily displayed on the small screens of smartphones [2]. However, if not designed carefully, these windowed challenges may compromise security. For instance, the Undercover [89] scheme requires *at least* one secret item to be present in all challenges. This results in an inherent bias, with the secret items appearing more frequently than the decoy items. This bias was exploited by Yan et al. [50] in a frequency analysis attack to extract the entire set of secret items after only a small number of observations. Subsequently, Asghar et al. [9] showed that if the windowed challenge of length l is sampled uniformly at random from the set of all possible $\binom{n}{l}$ challenges, then the above mentioned frequency-based attack can be mitigated.

- *Random Weights:* Each of the l items in the challenge is associated with a random integer, called its *weight*, from the set $\mathbb{Z}_d$, for a fixed integer $d \geq 2$. Note that for each challenge the weights are randomly sampled anew.

- *Modulus Operation:* The function f, to be mentally computed by the user, involves (at the minimum) summing the weights of the secret items present in the challenge and a modulo d operation on the sum. Notice that by construction, a challenge might not even contain any of the k secret items. We shall refer to it as the *empty event* or *empty case*, borrowing the term from [2]. How the function f is computed in an empty case depends on the scheme, as we shall discuss shortly.

Example 3.2.1. *We illustrate a k-out-of-n ORAS that satisfies the above design requirements. A windowed challenge can be represented by the n-element vector $\mathbf{c}$ whose ith element is the weight of the ith item, if present in the challenge, and 0 otherwise. With the same ordering, the secret can be represented as the binary vector $\mathbf{s}$ of Hamming weight k. One possible cognitive function f is the dot product modulo d, i.e, the response r is calculated as $\langle \mathbf{c}, \mathbf{s} \rangle \bmod d$. Consider the example in Figure 3.2, which has a pool of $n = 8$ items, with $k = 3$ secret items. A random challenge of $l = 4$ items has been sampled, with accompanying random values from $\mathbb{Z}_d = \mathbb{Z}_4$. The user computes* $r = \langle \mathbf{c}, \mathbf{s} \rangle \bmod d = \left\langle \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 3 & 2 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \right\rangle$

mod $4 = 0$. *The scheme is observation resilient as there are multiple candidates for the secret even after observing the challenge and this response. For example, the response 0 could have simply come from the weight of item D.*

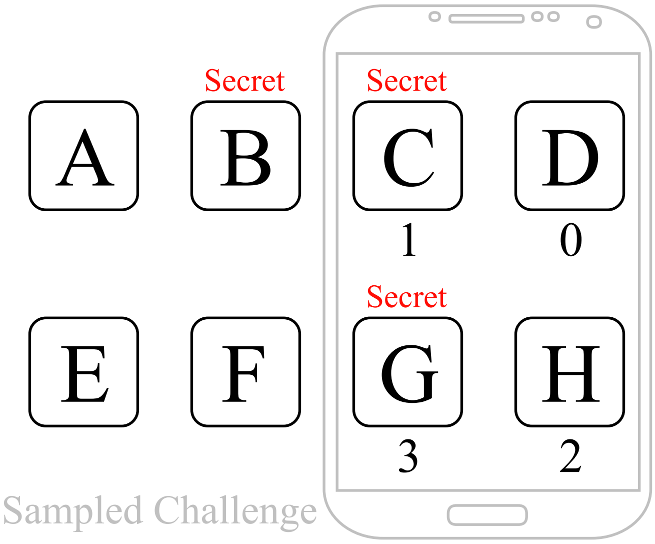


Figure 3.2: Example ORAS Scheme, $(n, k, l, d) = (8, 3, 4, 4)$

3.2.3 k-out-of-n ORAS Chosen for Analysis

We introduce three previously proposed ORAS that fall in the category of k-out-of-n ORAS described in Section 3.2.2. Note that with $O(n)$ challenge-response pairs in Example 3.2.1, the attacker can construct the secret using Gaussian elimination. The cognitive function in the three protocols is designed to increase the challenge-response pairs required to recover the secret through Gaussian elimination.

BehavioCog (BC) The BehavioCog [2] scheme is the same as in Example 3.2.1, except that it requires the user to submit a random response $r \in \mathbb{Z}_d$ in the case of an empty event (i.e., when none of the secret items are in the challenge). With this modification, Gaussian elimination requires $O(dn)$ challenge-response pairs [2]. One set of proposed parameters for the scheme is $(n, k, l, d) = (180, 14, 30, 5)$ [2], which we shall use in our analysis. We remark that BehavioCog was proposed with a behavioral biometric component to minimize authentication time. We disregard the biometric component and focus on the cognitive scheme.

FoxTail (FT) We chose the FoxTail scheme proposed by Asghar et al. [9] as a fix to secure the original FoxTail scheme [8] against a frequency attack [50]. The parameters we use for the scheme are $(n, k, l, d) = (180, 14, 30, 4)$ to allow for comparison between schemes. The cognitive function involves an additional step after the modulo $d = 4$ operation: the user is required to respond with 0, if the result is 0 or 1, and respond with 1, if the result is 2 or 3. In the case of an empty event, the user simply returns the response 0. The resulting non-linear map means that Gaussian elimination through linearization requires $\binom{n}{2} + n = O(n^2)$ challenge-response pairs [3].

Hopper & Blum (HB) The HB protocol [7] is one of the earliest ORAS proposed. The original proposal displays all n items to the user, with accompanying random binary weights. We modify the scheme to utilize windowed challenges, choosing parameters $(n, k, l, d) = (180, 14, 30, 2)$ (similar to BehavioCog). The protocol requires the user to intentionally flip the response bit (note that $d = 2$) with a fixed probability $\eta < 0.5$. The windowless HB was subject to timing attacks with the noise parameter $\eta = 0.2$ [51]; this value is maintained through the remainder of this chapter.[4] The HB protocol is based on the NP-Hard problem of learning parity in the presence of noise (LPN).

Rounds and Sessions Each authentication session consists of multiple challenge-response rounds. The number of rounds per session can be selected based on the success probability of randomly guessing the response (without knowledge of the secret). One common benchmark is 6-digit PIN, with a probability of randomly guessing the correct pin being $P_{\text{RG}} = 10^{-6}$ [2]. In BC (with the above parameters), the attacker can successfully guess

[4] We note that the timing attack from [51] is not applicable to the windowed HB protocol.

the response to a challenge with probability 0.256. Therefore, to achieve the security level of 10^{-6}, 10 rounds are required in a session. Similarly for FT, the answer could be guessed with probability 0.5, thus requiring 20 rounds per session for the same security level. In HB protocol, the user is accepted if the fraction of wrong answers are at most η [7]. For $\eta = 0.2$, this gives 51 rounds for a security level of 10^{-6}. Since this number of rounds is impractical, we lower the security level for HB to 10^{-4}, which gives us 34 rounds per session. In the remainder of this chapter, we will mostly use the number of rounds instead of sessions to discuss attacks, as the number of rounds within a session is ultimately at the discretion of the scheme designer.

3.2.4 Other ORAS

While we use k-out-of-n ORAS as the basis for our analysis, the results are applicable to other ORAS. More specifically, the results are applicable to any ORAS that uses a modulus operation. In Section 3.6, we shall give examples of these ORAS and our behavioral side channel attack on them.

3.3 The Modulus Event and Associated Biases

Given a challenge, we say that a modulus event occurs if the submitted response involves a modulus operation. For k-out-of-n ORAS, this happens if the sum of the secret items in a challenge is greater than d (otherwise the user does not need to reduce the sum modulo d). In this section, using a generic k-out-of-n ORAS, we show that

1. Depending on the parameters and the cognitive function, there is an imbalance in the likelihood of a modulus or a no-modulus event given different response values, e.g., a response of 0 is more likely to indicate a modulus event.
2. In a no-modulus event, the secret items have lower weights than the decoy items. Likewise in a modulus-event, the reverse is true.

The first of these observations will be used as one of the features to determine a modulus/non-modulus event in our classifiers in Section 3.5. The second observation is the basis of our algorithm to retrieve the secret in Section 3.4. While the response itself indicates if the modulus event has occurred or not, user behavior while computing the function f leaks further information about the event. This can be exploited by the adversary to increase confidence in predicting the modulus/no-modulus event to retrieve the secret.

Since the function f is mentally computed by the user, the adversary cannot know if the modulus event has occurred by simply looking at challenge-response pairs. However, user behavior while computing f leaks information about these events.

In what follows, we mathematically demonstrate that given a generic k-out-of-n ORAS, both above mentioned biases pertaining to the modulus event are linked to the (expected) number of secret items present in a challenge. The lower the number, the bigger the bias. Since this number is a function of the parameters (n, k, l), scheme designers need to choose appropriate values of these parameters to ensure that the expected number of secret items is large to minimize the biases.

3.3.1 Guessing a Modulus Event through Responses

Let us demonstrate this bias with the help of an example. Let G be the random variable representing the number of secret items present in a challenge. Thus, G takes on values in the set $\{0, 1, \dots, k\}$, where k is the total number of secrets items. Let g denote an instance of G in a specific round of authentication.

Example 3.3.1. *In Table 3.1, the number of secret item's present in the challenge is $g = 3$, and the responses are generated through the cognitive function in Example 3.2.1 with the modulus $d = 2$. Every combination of weights is equally probable (due to random sampling of weights). It is evident from the table that when the response is 0, it is more likely to be the result of having performed the modulus operation; whereas when the response is 1, it is more likely to be due to the absence of the modulus operation.*

Table 3.1: Modulo Bias in responses, 3 Secret Items, Binary weights.

Secret Weight 1	0	0	0	0	1	1	1	1
Secret Weight 2	0	0	1	1	0	0	1	1
Secret Weight 3	0	1	0	1	0	1	0	1
Modulus Event	No	No	No	Yes	No	Yes	Yes	Yes
Resulting Sum	0	1	1	2	1	2	2	3
User Response	0	1	1	0	1	0	0	1

We now generalize this to a generic ORAS. Let the random variable X denote the weight of an item in a challenge. Since each weight is sampled from a random uniform distribution over d,

$$\Pr(X = x) = \frac{1}{d}, \text{ for all } x \in \{0, 1, \ldots, d-1\}.$$

Let Y be the random variable denoting the sum of the weights of the g secret items. Then Y takes on values from the set

$$\{0, 1, 2, \ldots, (d-1)g\}.$$

We would like to determine $\Pr(Y = y \mid g)$, from which we can determine the probability of a modulus event by evaluating $\Pr(Y \geq d \mid g)$. To compute $\Pr(Y = y \mid g)$, we need to find the number of different ways g items with weights in $\mathbb{Z}_d$ can be summed to produce y. This is determined by the coefficient of z^y in the expansion of the polynomial $(z^0 + z^1 + \cdots + z^{d-1})^g$ [92, §1, pp. 23-24]. Alternatively, the probability can be evaluated without full expansion of the generating function via the following equation [92, §1, p. 24]:

$$\Pr(Y = y \mid g) = \frac{1}{d^g} \sum_{s=0}^{\lfloor y/d \rfloor} (-1)^s \binom{g}{s} \binom{y - sd + g - 1}{g - 1}. \tag{3.1}$$

Note that a modulus operation is *not* required when $Y = y < d$, whose probability is given by

$$\Pr(Y < d \mid g) = \sum_{y=0}^{d-1} \Pr(Y = y \mid g). \tag{3.2}$$

Similarly the probability that a modulus operation is required is

$$\Pr(Y \geq d \mid g) = \sum_{y=d}^{(d-1)g} \Pr(Y = y \mid g). \tag{3.3}$$

Equation 3.1 is dependent on g; in turn the probability of g items appearing in a challenge is dependent on $n, k,$ and l of the authentication scheme:

$$\Pr(G = g) = \frac{\binom{n-k}{l-g}\binom{k}{g}}{\binom{n}{l}}. \tag{3.4}$$

In Figure 3.3, we plot the probabilities of the modulus and no-modulus event given different values of g calculated through Eqs. 3.2 and 3.3. We see that if g is small, the no-modulus event is highly probable and a given response value might be biased towards a modulus or a no-modulus event. While this bias does not directly reveal information about the user's secret items, it can be used as an indicator of a particular challenge involving a modulus or a no-modulus event. And, as we shall show in the next section (Section 3.3.2), the knowledge of a modulus/no-modulus event, in turn, leaks information about the user's secret. Thus, from a security point of view, it is desirable to minimize this bias. This can be done by increasing the value of g, which makes the no-modulus event increasingly unlikely to happen (irrespective of the response). Asymptotically, we have the following result:

Theorem 3.3.1. *As $g \to \infty$, the probability of the modulus event approaches one, i.e., $\Pr(Y \geq d) \to 1$.*

Proof. Please see Appendix A.1. □

In practice, the likelihood of the no-modulus event vanishes much rapidly with an increasing g. For instance, for the case of $d = 2$, the probability is 0.1875 with $g = 5$, 0.0107 with $g = 10$, and 0.0005 with $g = 15$. We remark that the number of items present in a challenge g is a random variable dependent on the scheme parameters. Thus, to ensure the responses do not exhibit a bias towards the modulus or no-modulus event, the expected value of g needs to be high in an ORAS, which can be done by a combination of increasing k or l, and/or decreasing n (c.f. Equation 2). We will return to this in Section 7.1.

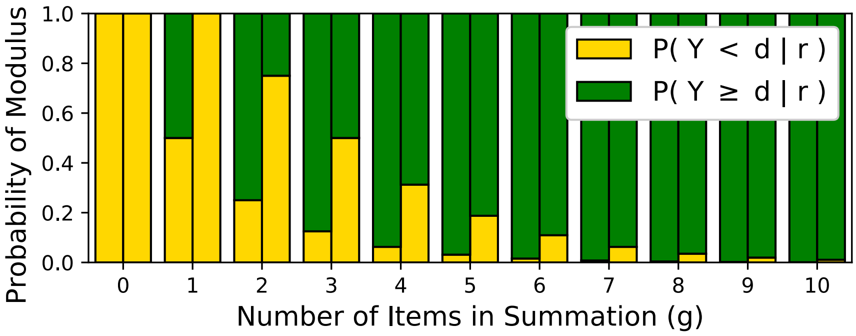


Figure 3.3: The probability of the modulus and no-modulus events given a user response against the number of secret items present in a challenge g when $d = 2$. The probability of modulus event increases with increasing g. The left and right hand columns respectively represent a user response of 0 and 1.

3.3.2 Weight Bias in a Modulus Event

We now show that given a modulus or a no-modulus event the expected weight of the secret items is biased away from the expected weight of the decoy items. Thus, the knowledge of a modulus/no-modulus event leaks information about the secret items. We will use this observation in our algorithm to retrieve the secret in Section 3.4. For now, we demonstrate this bias analytically using a generic k-out-of-n ORAS.

Recall that X denotes the weight of an item in a challenge. Clearly, the expected weight of any item within a challenge is $E[X] = (d-1)/2$. Denote by X_s, the random weight of a secret item. By construction, we have $E[X_s] = E[X] = (d-1)/2$. However, given the knowledge of a no-modulus event, the conditional expectation might not be the same. To see this, first note that

$$E[X] = E[X_s] = E[X_s \mid Y < d] \Pr(Y < d) \\ + E[X_s \mid Y \geq d] \Pr(Y \geq d), \tag{3.5}$$

where the conditional expectations are conditioned by no-modulus and modulus events, respectively. Since Y denotes the additive weight of g secret items, we have

$$E[X_s] = \frac{1}{g} E[Y].$$

We first show that the expected weight of secret items is less than or equal to the expected weight of decoy items in a no-modulus event, i.e., $E[X_s \mid Y < d] \leq E[X]$. The following lemma is used in the proof.

Lemma 3.3.2. *Let $g \geq 2$ be an integer, and let p be a strictly positive function, i.e., $p(i) > 0$, for all i in the domain of p. Then, for all $d \geq 1$*

$$\frac{1}{g}\sum_{i=0}^{d} ip(i) < \frac{d}{2}\sum_{i=0}^{d} p(i).$$

Proof. See Appendix A.2. □

Theorem 3.3.3. *Let $g \geq 1$. Then*

1. *$E[X_s \mid Y < d] = E[X]$, if $g = 1$.*
2. *$E[X_s \mid Y < d] < E[X]$, if $g \geq 2$.*

Proof. For part (1), when $g = 1$, only one secret item is present, thus it's weight will be uniformly sampled from 0 to $d-1$, and therefore in this case, $E[X_s|Y < d] = E[X]$.

For part (2), we have

$$\begin{aligned} E[X_s \mid Y < d] &= \frac{1}{g}E(Y \mid Y < d) \\ &= \frac{1}{g}\sum_{y} y \Pr(Y = y \mid Y < d) \\ &= \frac{1}{g}\sum_{y} y \frac{\Pr(Y = y, Y < d)}{\Pr(Y < d)} \\ &= \frac{1}{g}\sum_{y=0}^{d-1} y \frac{\Pr(Y = y)}{\Pr(Y < d)}. \end{aligned}$$

Invoking Lemma 3.3.2:

$$\begin{aligned} E[X_s \mid Y < d] &< \frac{1}{\Pr(Y < d)} \frac{d-1}{2}\sum_{y=0}^{d-1}\Pr(Y = y) \\ &= \frac{d-1}{2}\frac{\sum_{y=0}^{d-1} P(Y = y)}{\sum_{y=0}^{d-1} P(Y = y)} = E[X]. \end{aligned}$$ □

From this, it follows that expected weight of secret items is strictly greater than the expected weight of decoy items in a modulus event. That is:

Corollary 3.3.3.1. *$E[X_s \mid Y \geq d] > E[X]$.*

Proof. When $Y \geq d$, i.e., a modulus event, g is necessarily ≥ 2. From Eq. 3.5 and Theorem 3.3.3 part (2), we have

$$\begin{aligned} & E[X] < E[X] \Pr(Y < d) + E[X_s \mid Y \geq d] \Pr(Y \geq d) \\ \Rightarrow & E[X](1 - \Pr(Y < d)) < E[X_s \mid Y \geq d] \Pr(Y \geq d) \\ \Rightarrow & E[X] < E[X_s \mid Y \geq d] \end{aligned} \qquad \square$$

Thus, given a modulus event, weights of the secret items tend to be higher than decoy items. The reverse is true in a no-modulus event. We note that when $g = 0$, i.e., the empty event, the expected weight of the secret items is undefined (since they do not exist in the challenge).

Recall from Theorem 3.3.1 that as g increases, the probability of the no-modulus event approaches zero. Therefore, increasing g should also minimize the weight bias. This is demonstrated by the following corollary which shows that as g increases, the expected weight of a secret item approaches the global expectation $E[X]$.

Corollary 3.3.3.2. *As $g \to \infty$, $E[X_s \mid Y \geq d] \to E[X]$.*

Proof. From Theorem 3.3.1, we have $\Pr(Y \geq d) \to 1$ as $g \to \infty$. Consequently, $\Pr(Y < d) \to 0$. The result then follows from Eq. 3.5. □

Thus, scheme designers can minimize any weight bias by increasing the parameters (k, l), or decrease (n) to increase the expected value of g. Once again the above result is asymptotic, and in practice, the bias in expected weights vanishes quickly by a moderate increase in g since the probability of the no-modulus event decreases quickly. Unfortunately, the three ORASes under focus, do not have a sufficiently large expected value of g and are susceptible to revealing information about the secret items through this bias.

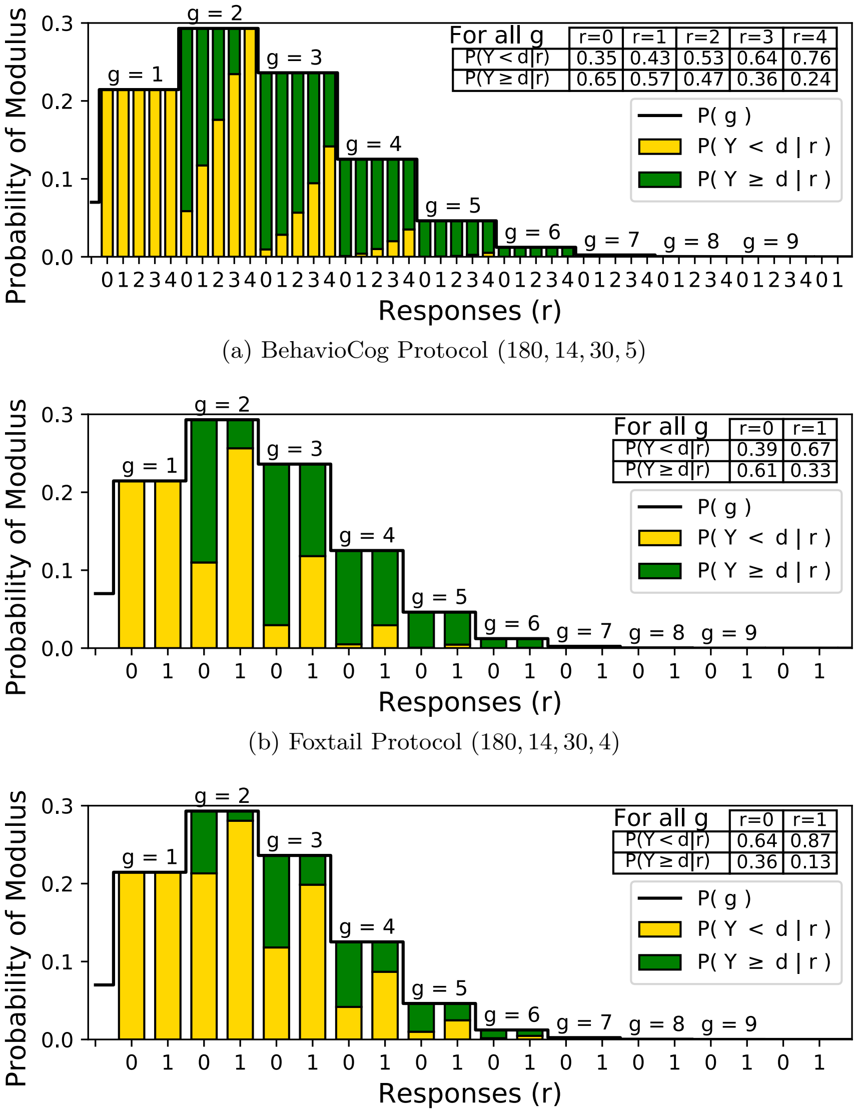
3.3.3 Biases in Specific ORAS

We now highlight these biases in the three instances of ORAS: BehavioCog, FoxTail, and HopperBlum. The cognitive functions in each of these schemes are slightly more involved than a simple mod operation on the sum (e.g., response flipping in HB). Therefore, the analytical results on the generic ORAS may not completely reflect the biases in these schemes.

3.3.3.1 Guessing a Modulus Event through Responses

First consider the BehavioCog scheme with parameters $(n, k, l, d) = (180, 14, 30, 5)$. Figure 3.4a shows the probability of a modulus and a no-modulus event given different response values broken down across different values of $g \geq 1$. The inset table in the figure shows the two probabilities irrespective of the value of g. Clearly, a higher response value indicates that it is more likely a no-modulus event. In contrast, lower response values are more likely to be a result of a modulus event. Not surprisingly, the probability of the no-modulus event decreases with increasing g. Recall that in BehavioCog, the user enters a random response in case of the empty event. However, the bias shown in the figure is for $g \geq 1$. The inset table on the other hand shows includes the empty event as well, and hence shows probabilities for all g.

The corresponding biases in the Foxtail and HB protocols are shown in Figures 3.4b and 3.4c, respectively. Recall that both Foxtail and HB have the response space $\{0, 1\}$. In Foxtail the response is “rounded” after a mod 4 operation, and in HB it is flipped with a fixed probability η (which we fix to 0.2). In Foxtail, we see that the response 0 is more likely to be from a non-modulus event, whereas the reverse is true of response 1. On the other hand, both responses in HB are more likely due to a no-modulus event, with response 1 being considerably more biased towards the no-modulus event. Thus, the bias of a particular response towards a modulus event depends on the scheme parameters as well as the cognitive function.



(a) BehavioCog Protocol $(180, 14, 30, 5)$

(b) Foxtail Protocol $(180, 14, 30, 4)$

(c) HopperBlum Protocol $(180, 14, 30, 2)$

Figure 3.4: Probabilities of modulus and no-modulus events given different response values in k-out-of-n ORAS. The probabilities are given for different values of g, showing the normalized probability for the event of the modulus operation. The overall probabilities irrespective of g are given in the table.

3.3.3.2 Weight bias in a Modulus Event

Table 3.2 shows the expected weight of the secret item(s) given a modulus and a no-modulus event for all three schemes. The expected weights are also shown against the number of secret items present in the challenge. The case $g = 1$ obviously does not involve any modulus operation, and so the expected weight, in this case, is equal to the overall expectation. This is shown in the table with the row labeled $E[X_s]$. In all three protocols, the expected weights of the secret items given a no-modulus event are lower than the overall expectation and decrease further as the number of secret items g increases. However, this also means that the no-modulus event becomes almost unlikely to occur. The expected weight of the secret items in a modulus event is higher for smaller values of g and approaches the overall expectation as we increase g. Noting in Figure 3.4 that higher values of g, say $g \geq 4$, the bias is less profound and is less likely to occur in a challenge (with the given parameters).

Table 3.2: Expected weights $E(X_s)$ of secret items.

$E(X_s \vert Y, g)$	All g	g=1	g=2	g=3	g=4	g=5	g=6	g=7	g=8	g=9
BC $Y < d$	1.20	2.00	1.33	1.00	0.80	0.67	0.57	0.50	0.44	0.40
d=5 $Y \geq d$	2.57	-	3.00	2.39	2.15	2.06	2.02	2.01	2.00	2.00
$E[X_s]$	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
FT $Y < d$	0.90	1.50	1.00	0.75	0.60	0.50	0.43	0.38	0.33	0.30
d=4 $Y \geq d$	1.98	-	2.33	1.84	1.64	1.56	1.52	1.51	1.50	1.50
$E[X_s]$	1.50	1.50	1.50	1.50	1.50	1.50	1.50	1.50	1.50	1.50
HB $Y < d$	0.30	0.50	0.33	0.25	0.20	0.17	0.14	0.13	0.11	0.10
d=2 $Y \geq d$	0.82	-	1.00	0.75	0.64	0.58	0.54	0.53	0.51	0.51
$E[X_s]$	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50

3.4 Attack Algorithm and the Faulty Oracle

In the previous section, we discussed how knowledge of the modulus or no-modulus event leaks information about the secret. In this section, we will construct an attack algorithm that retrieves the secret given access to an oracle that indicates a modulus or no-modulus event. A "perfect" oracle, however, is unrealistic in practice where we expect some error in our knowledge of the event. We therefore assume a *faulty* oracle which might erroneously indicate a modulus event. We will analyze the performance of the attack algorithm against varying accuracies of the faulty oracle.

More precisely we consider a faulty oracle, denoted $\mathcal{O}_{\text{mod}}$, which when given a challenge (and any auxiliary information) as input, returns -1 if it guesses that the user has *not* performed the modulus operation (i.e., the no-modulus event is the positive class), and +1 otherwise. The oracle can make two types of errors; *type 1 error* is when the oracle outputs -1 when it is a modulus event, and a *type 2 error* is when the oracle outputs $+1$ when in fact it is actually a no-modulus event. The true positive rate (TPR) is the probability of correctly guessing the modulus event, and therefore, $1 - \text{TPR}$ is the probability of *type 1 error*. Similarly, the true negative rate (TNR) is the probability of correctly guessing the no-modulus event. Thus, $1 - \text{TNR}$ is the probability of *type 2 error*. The oracle is parameterized by these two probabilities and we denote this by $\mathcal{O}^{\text{TPR,TNR}}_{\text{mod}}$.

Algorithm 1: Modulus Event Points Update

```
Input: Scheme parameters $(n, k, l, d)$, number of challenges $m$, penalty vectors
       $(u_0, \ldots, u_{d-1})$ where $0 = u_0 \geq \cdots \geq u_{d-1}$, and $(v_0, \ldots, v_{d-1})$, where
       $v_0 \leq \cdots \leq v_{d-1}$.
Output: A list of points $(p_1, p_2, \ldots, p_n)$, with top $k$ scores indicating secret items.
Initialize $(p_1, p_2, \ldots, p_n)$ to all zeroes.
for $j = 1$ to $m$ do
    Observe challenge $c$ containing items $i$ and weights $\text{wt}(i)$, auxiliary information
     'aux,' and response $r$.
    $b \leftarrow \mathcal{O}^{\text{TPR,TNR}}_{\text{mod}}(c, \text{aux})$.
    if $b = -1$ (no-modulus event) then
        for all items $i$ such that $\text{wt}(i) > r$ do
            penalize $p_i \leftarrow p_i + u_{\text{wt}(i)}$.
    else
        penalize $p_i \leftarrow p_i + v_{\text{wt}(i)}$.
return $(p_1, p_2, \ldots, p_n)$.
```

Algorithm 1 describes our algorithm to retrieve the secret with access to this faulty oracle, which we call the Modulus Event Points Update algorithm. The algorithm maintains a list of points $(p_1, \ldots, p_n)$ where p_i denotes the points for item i. Initially, all items have a score of 0. Upon receiving a challenge, the algorithm consults the faulty oracle. If the faulty oracle detects a no-modulus event, then it penalizes all items whose weights are greater than the response r (since such items would require a modulus event to produce the response r). Here items with higher weights are given higher penalties (as the expected weight of secret items is lower in the no-modulus event). When the oracle detects a modulus event, items with lower weights are given higher penalties as secret items are expected to have higher weights in a modulus event. This is reflected in the construction of the penalty vector $(v_0, \ldots, v_{d-1})$, where we have $v_0 \leq \cdots \leq v_{d-1}$. Note that in this case items are penalized irrespective of the response. This is because an item with any weight could have produced the response (since it is a composite of multiple weights reduced modulo d.

Since the oracle is faulty, secret items may also get penalized. However, the decoy items are penalized more with an increasing number of challenges, eventually leading to higher scores for the secret items. To show this, we consider a special case of the algorithm and show that the expected score of a secret item is higher than a decoy item with a large enough m, i.e., number of challenge-response pairs. Specifically, we consider the penalty vector $(v_0, \ldots, v_{d-1})$ to be all zeroes, i.e., no points update in case of the modulus event. Our simulations show that not updating the points at all when a modulus event is detected does indeed take the least number of samples to retrieve the secret.

Theorem 3.4.1. *Let the penalty vector $(v_0, \ldots, v_{d-1})$ be all zeroes. Furthermore, let the other penalty vector, i.e., $(u_0, \ldots, u_{d-1})$ be not identically zero. If TNR $> 1 -$ TPR, then for sufficiently large m, the expected score of a secret item is more than the score of a decoy item.*

Proof. See Appendix A.3. □

Obtaining an analytical estimate of the number of samples required to retrieve the secret through the algorithm is difficult. We therefore assess this through simulations. The penalty vectors chosen satisfy the condition of the theorem above. In particular, we use the penalty vector $(u_0, u_1, \ldots, u_{d-1}) = (-1, -1, \ldots, -1)$. With this penalty vector, the point update follows the pattern shown in Table 3.3, for each of the three schemes. We varied the TPR and TNR of the oracle between 0.6 and 1.0 with steps of 0.05. We

Table 3.3: Point update for BC, FT, HB. A cell is divided into a upper and lower half, representing the detection of a modulus and no-modulus respectively.

(a) BC

w r	0	1	2	3	4
0	0 0	0 -1	0 -1	0 -1	0 -1
1	0 0	0 0	0 -1	0 -1	0 -1
2	0 0	0 0	0 0	0 -1	0 -1
3	0 0	0 0	0 0	0 0	0 -1
4	0 0	0 0	0 0	0 0	0 0

(b) FT

w r	0	1	2	3
0	0 0	0 0	0 -1	0 -1
1	0 0	0 0	0 0	0 0

(c) HB

w r	0	1
0	0 0	0 -1
1	0 0	0 0

maintain the distinction between TPR and TNR to preserve the asymmetrical effects on our algorithm resulting from each type of error. For each pair of TPR and TNR, we ran 1,000 simulations of the attack algorithm on each of the three schemes. Instead of giving m, i.e., the number of challenges, as an algorithm input, we let it run until the top k items are the secret items. Tables 3.4 and 3.5 contains the average number of rounds required for all schemes.

We compare our results from Tables 3.4 and Table 3.5 to the samples required by best performing efficient algebraic attacks. For BehavioCog (in Table 3.4), the most efficient attack is Gaussian elimination, which finds the secret in 900 rounds [2]. For the FoxTail protocol (In Table 3.5), linearization followed by Gaussian elimination requires 16,290 observations [3]. In terms of a number of sessions, this is 90 sessions for BehavoCog (10 rounds per session) and 815 for FoxTail (20 rounds per session). In comparison, there are various ranges of accuracy levels for the faulty oracle which reduce the average number of sessions required to obtain the secret. Taking a realistic example of (TPR, TNR) = $(1.0, 0.6)$, BehavioCog would only need 279.7 rounds (28 sessions), whilst FoxTail would need 390.4 rounds (20 sessions), a substantially lower number of complete authentication sessions. For the HB protocol (In Table 3.5), no known efficient algebraic attack exists.

Table 3.4: Experimentally derived rounds required to reveal full user secret given varying TPR and TNR of side-channel classifier, for the Modulus applied on BehavioCog (900 Round Benchmark [2]).

1000 Iterations			Modulus Accuracy (TPR)								
			1.0	0.95	0.9	0.85	0.8	0.75	0.7	0.65	0.6
Non-Modulus Accuracy (TNR)	BehavioCog (BC)	1.0	165.978	262.996	345.960	451.254	565.272	678.194	823.412	982.746	1235.992
		0.95	174.384	280.342	385.178	488.884	612.682	766.068	922.540	1120.090	1342.758
		0.9	181.922	294.812	416.188	524.556	676.894	846.600	1025.982	1286.478	1580.582
		0.85	193.792	320.708	459.760	594.082	776.850	971.874	1178.938	1483.118	1891.290
		0.8	207.030	355.722	508.420	675.650	867.420	1077.646	1377.088	1761.734	2242.302
		0.75	216.218	381.012	549.340	730.038	960.940	1227.158	1629.738	2094.498	2808.506
		0.7	234.262	440.108	624.560	836.196	1115.732	1476.358	1949.384	2522.530	3340.990
		0.65	258.334	475.660	685.860	989.504	1331.370	1754.882	2372.312	3243.770	4297.246
		0.6	279.748	522.326	803.298	1157.238	1586.432	2203.824	2940.524	4112.454	5987.648
		0.55	303.129	616.505	948.832	1348.100	1911.934	2778.192	3909.067	5627.870	8521.225
		0.5	330.702	705.391	1126.371	1661.984	2392.133	3593.075	5396.283	8248.908	13557.03
		0.45	369.793	829.916	1344.850	2093.507	3149.585	5070.656	7984.902	13339.76	25016.93
		0.4	414.546	996.672	1683.427	2815.620	4455.89	7358.926	13297.78	25743.05	63722.74
		0.35	473.636	1234.009	2218.427	3832.711	6882.528	12585.91	26499.93	74077.08	>200000

This is not surprising as the protocol is based on the NP-Hard problem of learning parity with noise. For HB protocol, we require 909 rounds or approximately 27 sessions with 34 rounds per session (as discussed in Section 3.2.3). Thus, our attack shows an efficient attack based on side channel information.[5]

In comparison, Tables 3.4 and 3.5 shows that our attack algorithm with many different combinations of TPR-TNR far outperforms the aforementioned attacks. But, which combinations of TPR-TNR are fair and realistic? We see that if the TPR is high (close to 1), then our algorithm is less sensitive to decreasing TNR. This is somewhat evident from Algorithm 1, and our choice of the penalty vector. A low TNR means that a no-modulus event may be frequently misclassified as a modulus event. However, the algorithm never penalizes items in such a case (due to the use of a zero penalty vector). This is also due to the fact that in a no-modulus event, there is a large weight differential (as shown previously). For binary classification tasks, it is always possible to trade the TNR with diminishing FPR, as TPR and TNR are inversely related. The first few columns in the table correspond to this regime, and we see that in most cases we significantly outperform

[5] We note that Cagalj et al. [51] propose a side channel attack on the original HB protocol (without the window) which recovers the secret in 380 rounds. Unfortunately, this attack is not applicable to the windowed variant.

Table 3.5: Experimentally derived rounds required to reveal full user secret given varying TPR and TNR of side-channel classifier, for the Modulus applied on FoxTail (16,290 Round Benchmark [3]), and HopperBlum.

			Modulus Accuracy (TPR)								
1000 Iterations			1.0	0.95	0.9	0.85	0.8	0.75	0.7	0.65	0.6
Non-Modulus Accuracy (TNR)	FoxTail (FT)	1.0	234.099	361.049	462.091	590.253	716.030	851.755	996.776	1206.307	1431.352
		0.95	245.924	384.369	512.155	631.904	779.754	929.230	1132.713	1325.878	1574.523
		0.9	259.346	418.701	555.846	696.247	846.392	1039.581	1275.197	1514.627	1826.617
		0.85	275.493	450.995	591.611	751.928	945.971	1155.384	1412.938	1694.107	2061.644
		0.8	289.746	474.639	657.124	846.583	1073.885	1341.471	1625.042	2034.259	2407.814
		0.75	309.022	518.628	719.507	942.644	1184.645	1527.110	1873.568	2395.379	2942.497
		0.7	337.233	570.886	807.810	1076.267	1376.793	1780.750	2223.255	2780.624	3536.249
		0.65	360.027	654.276	903.714	1222.030	1561.489	2110.796	2593.069	3363.440	4372.477
		0.6	390.420	719.669	1027.850	1420.807	1857.157	2491.812	3303.234	4271.355	5663.664
		0.55	425.797	817.616	1211.947	1673.944	2309.427	3092.693	4160.624	5502.299	7751.852
		0.5	466.579	928.206	1432.573	2044.601	2804.419	3836.378	5499.969	7662.418	11093.98
		0.45	519.760	1112.707	1724.578	2495.029	3612.860	5179.980	7484.748	11356.39	17528.47
		0.4	581.398	1302.668	2123.358	3243.961	4803.101	7302.969	11542.69	18748.54	33004.62
		0.35	665.757	1592.206	2786.819	4429.673	7045.669	11521.79	19765.98	38068.37	83833.69
	HopperBlum (HB)	1.0	538.108	643.254	759.623	912.669	1056.444	1233.634	1448.241	1702.210	2011.441
		0.95	574.626	684.327	809.546	972.457	1144.787	1350.292	1616.565	1896.063	2251.269
		0.9	601.860	734.153	871.632	1041.118	1280.526	1502.996	1810.434	2120.654	2562.250
		0.85	636.235	780.374	957.234	1137.184	1364.196	1663.844	2056.685	2485.285	3047.063
		0.8	674.237	833.574	1031.503	1271.620	1537.344	1922.406	2356.007	2886.691	3512.182
		0.75	712.093	908.231	1129.152	1388.736	1757.375	2152.616	2687.703	3351.902	4257.354
		0.7	784.420	1000.224	1230.416	1580.273	1964.927	2466.244	3146.388	4113.707	5272.378
		0.65	852.850	1066.540	1383.679	1789.643	2253.984	2906.097	3840.653	4946.336	6689.905
		0.6	908.848	1181.096	1574.578	2081.068	2654.442	3515.377	4805.121	6448.616	8940.049
		0.55	983.286	1333.104	1812.558	2402.852	3259.621	4408.826	5955.009	8593.798	12597.03
		0.5	1087.780	1495.170	2119.745	2964.009	4068.224	5571.716	8035.144	12339.58	19706.51
		0.45	1225.173	1763.995	2510.148	3626.290	5112.787	7750.267	11859.26	19966.72	35981.45
		0.4	1356.933	2056.374	3105.767	4595.770	7067.243	11288.44	19362.25	37558.86	91942.35
		0.35	1554.770	2544.067	3914.772	6296.296	10456.32	18819.72	38944.80	102742.2	>200000

algebraic attacks.

On the other hand, in the case of BehavioCog, if both TPR and TNR are less than 0.8 then the performance of the side-channel classifier degrades in comparison to Gaussian elimination. However, in general, the classifier can be trained to favor the TNR over TPR, or vice versa, by varying the threshold. This means that we can fix the threshold to favor a high TPR, say 0.95 or 1.0, sacrificing the TNR as a result but still be able to outperform Gaussian elimination in terms of the number of rounds required to retrieve the secret, as is evident from the table (the first two columns under BC). Indeed, as we shall show in Section 3.5.4, we can train the classifier on data from our user study to achieve a TPR of 1.0 and a TNR of 0.38, and still able to obtain the secret after 435 rounds, less than half the number required via Gaussian elimination.

Increasing Confidence The results in Tables 3.4 and 3.5 show the average minimum number of rounds required before the first k items are the user's secret items. While, these numbers can be used as a reference on how many challenge-response pairs are required to find the secret via Algorithm 1 with high probability, the attacker can use a better strategy to increase its confidence on the first k items being the secret items. The idea is to rank items after each round according to their scores, and keeping track of the point differences between the neighbors of the kth ranked item. More precisely, let item(i) denote the item ranked i in the current round, where $1 \leq i \leq n$ (ties can be broken according to the initial order on the items). Note that the item ranked i might change over successive rounds. The attacker updates the (absolute) difference in points of the following pairs of items: $(\text{item}(k-1), \text{item}(k))$, $(\text{item}(k), \text{item}(k+1))$, and $(\text{item}(k+1), \text{item}(k+2))$. Let us denote these three point differences by $\text{diff}_{k-1,k}$, $\text{diff}_{k,k+1}$ and $\text{diff}_{k+1,k+2}$, respectively. For the first few rounds, the attacker cannot distinguish between these three. For a given (TPR, TNR), once the number of rounds passes the mark given in Tables 3.4 and 3.5, $\text{diff}_{k,k+1}$ starts deviating away from $\text{diff}_{k-1,k}$ and $\text{diff}_{k+1,k+2}$. The more rounds the attacker observes, the more $\text{diff}_{k,k+1}$ deviates away from the two. Thus, the attacker can increase its confidence that the top k are indeed the secret items by setting a threshold for the gap between $\text{diff}_{k-1,k}$ with respect to $\text{diff}_{k-1,k}$ and $\text{diff}_{k+1,k+2}$. Figure 3.5 shows this for BehavioCog with $(\text{TPR}, \text{TNR}) = (0.95, 0.95)$. We can see a divergence in the score differences after around 280 rounds, consistent with our simulated rounds required for this configuration (cf. Table 3.4).

3.5 Implementing the Attack Using Behavioral Side-Channel

In this section, we show that certain user behavior patterns while processing challenges can provide information about the modulus event. More specifically, we target the user's eye-movement together with the associated timing information. An eavesdropping adversary's ability to accurately guess the modulus event depends on the resolution of behavioral information available. We model this as adversaries with varying strength (Section 3.5.1);

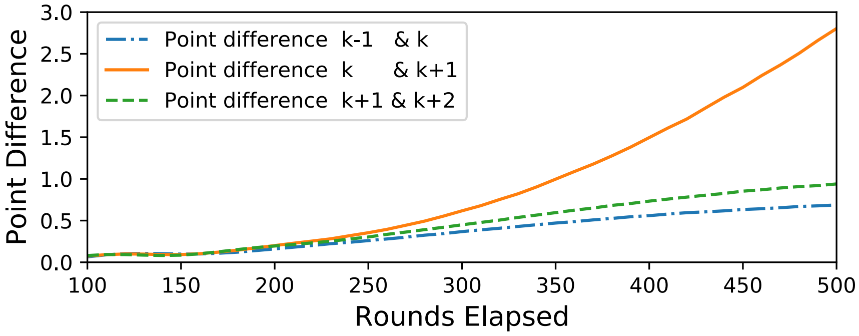


Figure 3.5: The point difference between the kth and $(k+1)$st ranked items versus the points difference between $(k-1)$st and kth ranked, and $(k+1)$st and $(k+2)$nd ranked items as a function of number of observed rounds. These results are for a faulty oracle with 0.95 EER on BehavioCog. Clearly, after around 280 (expected number of rounds to find the secret), there is growing divergence between the scores of secret and decoy items, indicating increased confidence in the top k items being the secret items.

from the weakest adversary with access to only meta information to the strongest adversary with high-resolution eye movement to screen mapping. We then identify potentially revealing behavior patterns through a user study by collecting data from an eye-tracker (Section 3.5.2). Following this, we identify features corresponding to these behavior patterns which are then used as input to machine learning classifiers (Section 3.5.3) to predict the modulus event (thus instantiating the faulty oracle of the previous section). The data from the user study is used to train and test the classifiers, and the resulting accuracy levels (TPR and TNR) are used as instances of the faulty oracle in the aforementioned attack algorithm.

3.5.1 Levels of Adversarial Strength

We define four different levels of adversaries differing by the resolution of behavioral information available to them. These levels are outlined below with real-world examples. Figure 3.6 illustrates them pictorially. We assume each adversary can access the challenge and responses in addition to the behavioral information.

- *Level 1 (L1):* An adversary with access to the challenge duration, i.e., time till user

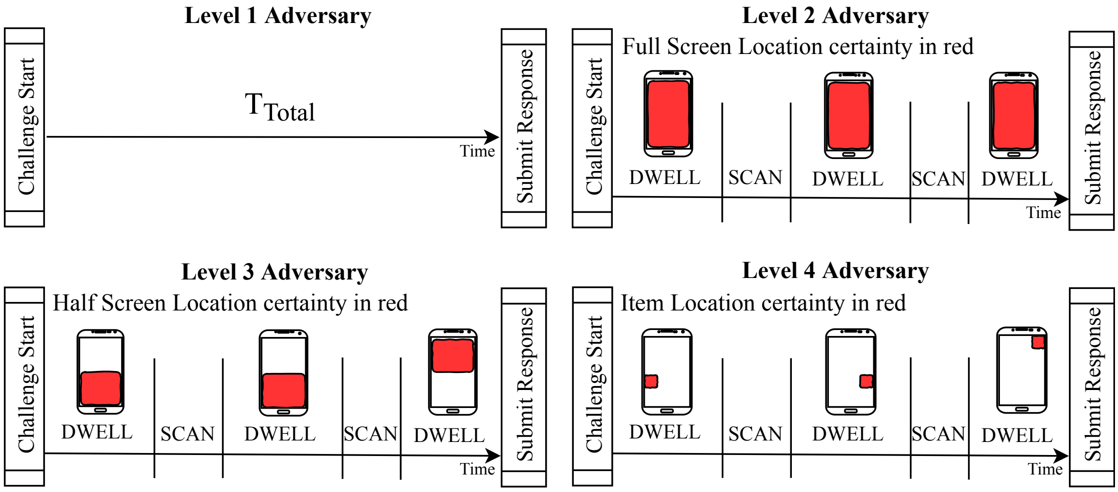


Figure 3.6: Four Levels of adversary capabilities in recovering eye-tracking information, Each level beyond L1 is provided with increasingly detailed location information, from no location information (L2), sectors (L3), to specific items (L4).

submits response. Examples include monitoring Internet traffic or the screen itself.

- *Level 2 (L2):* An adversary with further access to user dwell times, i.e., when the eye is stationary. This information can be obtained via a hidden camera facing the user, e.g., a pinhole camera mounted on an ATM, or a general surveillance camera. The resulting video feed of the user's eyes can be used to determine still positions through pupil detection and its lack of movement.

- *Level 3 (L3):* An adversary with further access to rudimentary positional information of dwells, e.g., lower half of the screen, top-left quadrant. This information can again be obtained via a hidden camera recording the user's face. Furthermore, we assume that the attacker has access to video-oculography to estimate gaze and to extract positional information from either the geometric model or appearance of the eyes [88, 93–95].

- *Level 4 (L4):* An adversary with further access to item-specific positional information of dwells. We assume the attacker employs a hidden camera to record a video of the user's face. The attacker has access to highly accurate video-oculography [88, 93–

95] to estimate item specific positional information (as compared to coarse-grained positions in L3).

Note: A webcam or the front camera of a smartphone or a laptop, are also possible examples of a hidden camera considered for adversary levels 2 to 4. However, it can be argued that the user's device is already compromised if an attacker has access to the in-device camera, and hence the protection provided by an ORAS might be superfluous. Therefore, we discard this as a possible attack vector, and instead consider off-device hidden cameras, examples of which are given above.

3.5.2 User Study

We recruited 11 postgraduate research students, in the 24-26 age range of mixed gender (6 males, 5 females) as participants in the eye-tracking experiment who were asked to process challenges from the BehavioCog scheme (specifically, the cognitive component in [2]) with parameters $(n, l, k, d) = (180, 30, 14, 5)$. The users were given training, and trial attempts to help them remember their secrets and to familiarize themselves with the scheme, followed by computing several random challenges. The position of their gaze, and field of view are recorded with a pair of SMI Eye Tracking Glasses (ETG2).[6] From all challenges attempted by the users, we sampled those that had correct responses such that there were roughly the same number of instances of $g \in \{0, 1, 2, 3\}$ secret items. This represented 81.4% of possible challenges within BehavioCog, resulting in a total of 64 challenge samples. A full breakdown of user samples with respect to the number of secrets present g, and whether they had performed a modulus operation is shown in Table 3.6.

The data from the study included (a) a recorded video of each challenge, (b) and an associated list of xy-coordinates. We performed a manual mapping process to link an xy-coordinate to a challenge item in the video. The task was partially automated by overlaying the timestamped xy-coordinate as a red dot on the corresponding frame of

[6] `https://www.smivision.com/eye-tracking/products/mobile-eye-tracking`

Table 3.6: User Contribution of Eye-tracking Samples

64 Samples	User	1	2	3	4	5	6	7	8	9	10	11	Total
Number of Secrets	0 Secrets	2	1	1	0	1	3	2	1	0	2	1	14
	1 Secrets	1	2	2	1	2	1	1	1	0	2	2	15
	2 Secrets	2	2	2	2	0	2	2	1	2	1	2	18
	3 Secrets	2	1	1	3	0	1	2	0	3	3	1	17
Modulus Status	Empty	2	1	1	0	1	3	2	1	0	2	1	14
	No Mod	3	4	4	3	2	3	5	2	3	3	4	36
	Mod	2	1	1	3	0	1	0	0	2	3	1	14

the video feed, as shown in Figure 3.7. The SMI eyetracker software provides its own classification of eye movement as either “Saccade” (rapid eye movement or scan), “Visual Intake” (low eye movement or dwell), and “Blink.” The mapped item over which the user’s focus is positioned corresponds to segments of visual intake separated by saccades or blinks. We noticed that the SMI software classification is highly sensitive to even the smallest eye movements, whereby if a user shifts their focus on different parts of the same item, a saccade may be registered. An attacker may not have the same luxury of information. So we treat sequential periods of visual intake separated by a saccade or blink of the same item as a continuous dwell on an item. The mapping of positional information to challenge items enables us to produce detailed information available to L3 and L4 adversaries. However, for less capable adversaries (L1 and L2) such positional information is not required.

3.5.2.1 Ethics Consideration

The participants were recruited via university mailing lists and posters, and were informed about why and how their data was to be used. Their consent for the collection of eye-tracking data was obtained, with monetary compensation provided to the participant at the completion of the experiment. Ethics approval for the conduct of the experiment and analysis of the data was obtained from our ethics review board prior to user recruitment. The recordings of our participant interactions may contain potentially identifiable information (fingerprints, skin tone), thus after the mapping of focal *xy*-coordinates and

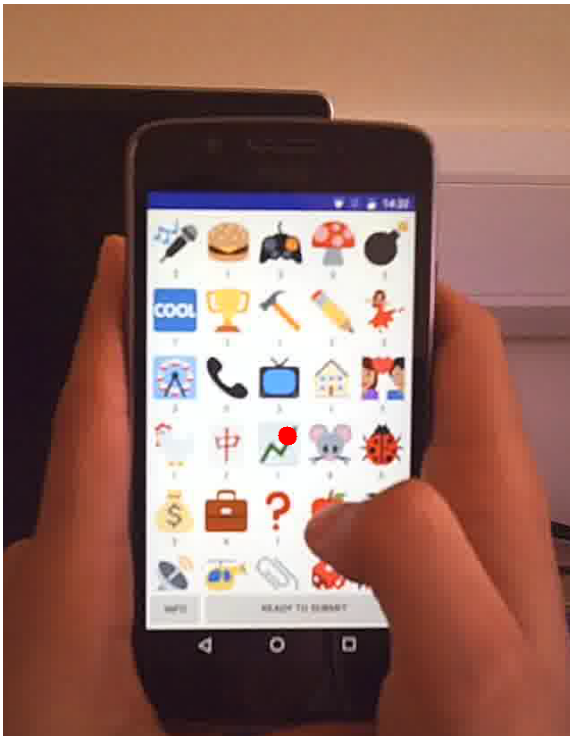


Figure 3.7: An image of the xy-coordinate (red dot) overlaid on a video feed.

challenge items, the recordings were encrypted at rest.

3.5.3 Features and Classifiers

From the eye-tracker study, we identified several features, e.g., minimum dwell time, number of vertical transitions (between the two screen halves). Features are categorized according to their availability to the four adversarial levels. Justifications and the hypotheses behind the choice of these features, i.e., how they intuitively reveal information about the modulus/no-modulus event, are included in Appendix A.4. The features are also listed in Table 3.7. Due to the small sample size, providing all features to a machine learning classifier is not recommended (due to the curse of dimensionality [96]). Thus features were ranked using the minimal-redundancy-maximal-relevance (mRMR) score [97]. The mRMR algorithm seeks to rank features to maximize the information gain provided by a feature for the task of separating the sample classes. The algorithm also accounts for redundant features; otherwise multiple similar features would be ranked highly, ignoring the fact that each subsequent feature would provide little new information. Table 3.7 details the feature rankings in ascending order of information gained.

Table 3.7: MRMR Rankings for features of modulus side channel classification.

		Adversary Level	L1	L2	L3	L4
Engineered Features	Level 1	Total Time of authentication	1	1	1	1
		Mean Challenge Weight	2	3	3	3
		User Challenge Response	3	5	5	5
	Level 2	Minimum Dwell Time		6	7	11
		10th Percentile Dwell Time		8	10	16
		Maximum Dwell Time		10	14	20
		90th Percentile Dwell Time		12	16	22
		Mean Dwell Time		11	15	21
		STD Dwell Time		15	19	25
		Number of Dwells		13	17	23
		Time to end from longest Dwell		14	18	24
		Dwell Consistency		2	2	2
		Duration of First Fixation		7	9	15
		Duration of Last Fixation		9	12	18
		Longest Dwell Consistency		4	4	4
	Level 3	Vertical Transitions (Even)			6	10
		Vertical Transitions			8	13
		Horizontal Transitions			11	17
		Time to end from screen bottom			13	19
	Level 4	Number of largest revisits				8
		Number of unvisited items				14
		Longest repeating sequence				7
		Weight of 1st Longest Dwell				6
		Weight of 2nd Longest Dwell				9
		Weight of 3rd Longest Dwell				12

A comprehensive selection of classification algorithms was tested from the Python machine learning library `scikit-learn` [98]. Specifically, we use Support Vector Machines (linear and radial kernel), Naive Bayes, AdaBoost and Random Forest classifiers. Each algorithm was tested on an increasing number of features for each adversary level, as determined by the mRMR algorithm. Each of our classifiers are used in a two-class configuration for the modulus/no-modulus event.

We adopt leave-one-user-out verification as the most rigorous form of model validation, allowing the demonstration of generic behaviors irrespective of user. The method is a proactive assurance against overfitting; with the low number of available training samples, the inclusion of any user specific samples would risk the trained model learning user-specific behavior instead of generic behaviors across the entire group of users. This also represents a realistic attack scenario whereby the attacker has no prior knowledge of the target user.

As previously observed in our simulations (Table 3.5), the performance of the attack algorithms is disproportionately sensitive to the accuracy of detecting one class over the other (modulus event accuracies are more important than the no-modulus accuracy). Each of the classifiers return a prediction probability score for each class label. By default, a threshold is set to 50%, a sample is classified as belonging to the first class if the score returned by the classifier is 50% or above. We can favor either class by altering this threshold to tighten or loosen the conditions for being classified into the first class, thus controlling the trade-off between TPR and TNR.

The best performing classifier (algorithm, features, threshold), is then found by using the TPR and TNR values in the faulty oracle in the points update algorithm and simulating challenge-response rounds in the scheme. The simulation is repeated 1,000 times to obtain an average number of rounds. The classifier with the lowest number of rounds to resolve the secret, is chosen. This process is then repeated for all adversarial levels. We note that a single global threshold is used across every testing sample, irrespective of the validation fold. To obtain a single value of the pair (TPR, TNR), we aggregate the test samples from each fold into a set. We acknowledge the low number of test samples prevents us from directly attacking a user, instead having to adopt challenge-response simulations. Additionally with a larger test group of users, more data can be leveraged to train better performing machine learning models.

3.5.4 Modulus Event Side Channel

After training and testing classifiers on the ranked features, the results of the simulations with the classifiers as faulty oracles (given by corresponding TPR and TNR) are presented in Table 3.8. With only one feature, i.e., “Total Time”, the AdaBoost classifier was able to obtain a (no-mod, mod) accuracy of (0.38, 1.0). Unfortunately, the additional features provided to the classifier in adversarial levels L2 and L3 did not improve our algorithm performance further; Until L4, where the naive Bayes classifier is able to achieve an accuracy of (0.4, 1.0) with 7 features. Note that we did not change the threshold over the default 0.5 by a large degree to obtain a TPR of 1.0. The results for the higher level adversaries do not show a significant improvement over lower level adversaries. But this may be due to our limited field study. Since these accuracy levels are dependent on the data from the user study, a larger user study might reflect better on the influence of other features in classification accuracy. On the other hand, a Level 1 adversary with only the total time of the challenge can sufficiently separate the modulus and the non-modulus challenges, demonstrating the practicality of our attack and the need to consider user behavior when designing ORAS.

Table 3.8: Best adversary level classifier exploitation of the modulus operation information.

Adver. Level	No-Mod Acc.	Mod Acc.	Rounds required	Classifier used	# Features, Threshold
L1	0.38	1.00	435.04	Adaboost	1, 0.51
L2	0.38	1.00	435.04	Adaboost	1, 0.51
L3	0.38	1.00	435.04	Adaboost	1, 0.51
L4	0.40	1.00	411.89	Naive Bayes	7, 0.59

With these oracle accuracies, our simulations show that it will take approximately 435 observations on average for a L1-L3 adversary, and 412 for L4 adversary to find the user’s secret. This is half of the rounds needed by the Gaussian elimination attack (900 rounds) in the BehavioCog scheme [2]. By extending these simulations to the FoxTail and HopperBlum schemes, we observe 589 and 1,346 rounds, respectively, for an L4 adversary, and 618 and 1,415 rounds, respectively, for L1-L3 adversaries. Recall that the linearisation/-Gaussian elimination attack on Foxtail requires 16,290 rounds, whereas the HB protocol has no efficient algebraic or statistical attack.

Finally, comparing the number of rounds in Table 3.8 against the numbers reported in

Table 3.5, we see that the number of rounds required by the best classifier via the user study is larger than the simulated attacks. However, we reiterate that this is due to the best accuracy level through our limited user study, which is not indicative of the best accuracy level achievable in practice. With a larger user study we would expect to obtain better accuracy levels, matching those in Table 3.5, e.g., (TPR, TNR) $= (1.0, 0.6)$, and thus retrieving the secret in a smaller number of rounds.

Attack Performance without Timing Information It may appear from the lack of improvement in L2 and L3 adversaries' performance that the eye movement related features do not show any gain over simply timing based information. This is particularly problematic from an attacker's point-of-view as scheme designers can easily mask timing information by mandating a minimum time before the user can submit a response in each authentication round. However, the eye movement features are also fairly accurate indicators of the modulus/no-modulus event. To demonstrate this, an experiment with the Total Time feature excluded from the feature set. With only a Naive Bayes classifier, we are able to obtain (TPR, TNR) $= (1.0, 0.38)$ with the second ranked feature (Dwell Consistency) at a threshold of 0.76. This offers performance equivalent to the adversaries L1-L3 in Table 3.8. This feature is part of the feature set of adversaries L2 to L4, and hence demonstrates that observing eye movement patterns can successfully retrieve the secret.

Per-User Accuracy Rate Until now we have reported system-wide accuracies to determine an attacker's performance. Since the dataset is small, we are interested in how the modulus detection accuracy varies between users, to see if the system-wide values are good representatives. We therefore report modulus detection accuracies for each user within our study for the four selected configurations (corresponding to adversary levels noted in Table 3.8). These are shown in Table 3.9.

First, we see that for all users against all adversary levels, we achieve a TPR of 1.0. In case of TNR, against adversary levels L1-L3, 7 out of the 11 users are within ±0.2 of the

Table 3.9: Modulus detection accuracy separated on a per-user basis. It is observed that in L1-3, the TNR is approximately equal between users. Under L4 however, there appears to be more variance in the performance of the classifier. Where no accuracy is reported for TPR, no positive user samples exist. The total number of user positive and negative samples are noted in the last row of the table.

User	1		2		3		4		5		6		7		8		9		10		11	
Adv.	TNR	TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR	TPR
L1	0.4	1	0.0	1	0.6	1	0.333	1	0.333	-	0.333	1	0.286	-	0.667	-	0.667	1	0.4	1	0.4	1
L2	0.4	1	0.0	1	0.6	1	0.333	1	0.333	-	0.333	1	0.286	-	0.667	-	0.667	1	0.4	1	0.4	1
L3	0.4	1	0.0	1	0.6	1	0.333	1	0.333	-	0.333	1	0.286	-	0.667	-	0.667	1	0.4	1	0.4	1
L4	0.2	1	0.4	1	0.2	1	0.0	1	0.0	-	0.833	1	0.714	-	0.333	-	0.333	1	0.2	1	0.6	1
Total	5	2	5	1	5	1	3	3	3	0	6	1	7	0	3	0	3	2	5	3	5	1

system wide TNR of 0.38 (cf. Table 3.8). Three other users have TNRs between 0.6 and 0.667, slightly off the mark from the system TNR. One user, however, is an outlier with a TNR of 0.0. On the other hand, again, 7 out of 11 users against adversary level 4 are within ±0.2 of the system TNR of 0.4. However, the outliers in this case are further adrift, with 2 of the users exhibiting a TNR of 0.0, and 2 others showing a TNR of more than 0.714. Note, that higher than average TNR is not a problem from the attack's perspective, as this would require fewer observations before the secret can be retrieved (cf. Tables 3.4 and 3.5). Thus, we can conclude that the system-wide performance of the attack is mostly representative of its performance per-user: the attack can be carried out against most users in the system, with TNR of most users being close to the system-wide TNR. This indicates that the classifiers are unlikely to have overfit. The exception being the outliers who exhibit a TNR of 0.0. The prevalence of such users in the general population would require a larger study, which we leave as future work.

3.6 Application to Other ORAS

In this section, we show that the attack is applicable to other ORAS which do not fit the description of k-out-of-n ORAS, as long as they contain a modulus operation. We use two such ORAS: PassGrids [11] and Mod10 [6], and present slightly modified point update algorithms tailored to these schemes. Both PassGrids and Mod10 use a modulus of $d = 10$, and due to their fundamentally different construction from k-out-of-n ORAS, not all side-channel features previously used are relevant (e.g., no items to gaze at in

Mod10). Coarse timing information, however, is still relevant, due to the *problem size effect* as studied by LeFevre, Sadesky, and Bisanz. [99]. The problem size effect observes relatively slower latency (timing) on arithmetic problems with sums greater than 10. For PassGrids and Mod10 with a modular operator of $d = 10$, slower latency then is a close indicator of the modulus/no-modulus event. Thus, we may think of the faulty oracles in the attack algorithms on these schemes being initiated by classifiers that use such timing related information to classify modulus/no-modulus events. Throughout this section we will use symmetrical oracle accuracies despite our earlier observation of an asymmetrical response to classifier errors, this is to provide simpler performance references of hypothetical attackers.

These schemes can be configured with secrets of variable length. For example, a PIN can be 4 or 6 digits in length. Each secret digit and the challenge cognitive function are independent of the other secret digits. As such, we assume our attacker is capable of obtaining oracle information for each sequential challenge (pass-item/digit) and has knowledge of when a challenge (pin digit entry) starts and stops. This notion was not applicable for the previous schemes of BehavioCog, FoxTail, and HopperBlum, as the secret items collectively produce a single final response. We remark that while we do have oracle information about the individual digits, we do not stop updating points on any digit until all digits are ranked highest, i.e., the complete secret has been found.

3.6.1 PassGrids

The PassGrids system [11] consists of a series of schemes that are modifications of the commonplace PIN authentication systems. The schemes are designed to be resistant to observation. We consider the version of their scheme called “PGx+4.” This scheme is implemented on a 6×6 grid with 36 possible locations. A challenge consists of an assignment of a random digit $\{0, \dots, 9\}$ to each of the 36 locations. The digits are generated so that each appears an approximately equal number of times, i.e., 3-4 times. The user’s secret is a set of four tuples of the form: (i, x_i, y_i), where i is a random location, $x_i \in \{1, \dots, 9\}$ and

$y_i \in \{0, \ldots, 9\}$. For each secret tuple s, given the challenge c, the response is computed as $r_i = f(s, c) = c_i x_i + y_i \bmod 10$, where c_i is the digit corresponding to location i in the challenge. The secret space is thus of size $36 \cdot 9 \cdot 10 = 3240$ for a 1-length secret, and consequently a 4-length secret would have $\frac{(3240)!}{(3240-4)!} \approx 2^{46.6}$ possible secrets. This scheme offers a degree of observation resilience (<10 observations). Once again, we see that the modulus operation is used to provide observation resilience.

```
Algorithm 2: PassGrid Points Update
Input:  Number of challenges $m$; A set of secrets $S$ where $s \in S$ is a tuple $(i, x, y)$,
        where $i$ is one of 36 locations, $x \in \{1, \ldots, 9\}$ and $y \in \{0, \ldots, 9\}$; size of $S$ as
        $n = 36 \times 9 \times 10$.
Output: A list of points $(p_1, p_2, \ldots, p_n)$, with top score indicating the target secret.
Initialize $(p_1, p_2, \ldots, p_n)$ to all zeroes.
for $j = 1$ to $m$ do
    Observe challenge $c$, auxiliary information ‘aux,’ and response $r$.
    $b \leftarrow \mathcal{O}^{\text{TPR,TNR}}_{\text{mod}}(c, \text{aux})$.
    if $f(s, c) \neq r$, for $s \in S$ then
        penalize $s$ by 10
    else
        if $b = -1$ (no-modulus event) & $(f(s, c) \geq 10)$ or $b = +1$ (modulus event)
          & $(f(s, c) < 10)$ then
            penalize $s$ by 3
return $(p_1, p_2, \ldots, p_n)$.
```

We implemented the PGx+4 scheme and ran our attack algorithm, Algorithm 2, on it. The attack algorithm is a point update algorithm that penalizes locations and operands (x_i's and y_i's) that do not agree with the challenge response. We also to a lesser extent penalize secrets that do agree with the response but do not agree with the modulus oracle. The algorithm is shown for a 1-length secret for clarity, but can simply be extended to a 4-length secret by parallel execution.

We selected a point update vector of 10 for secrets that conflict with the response and 3 for

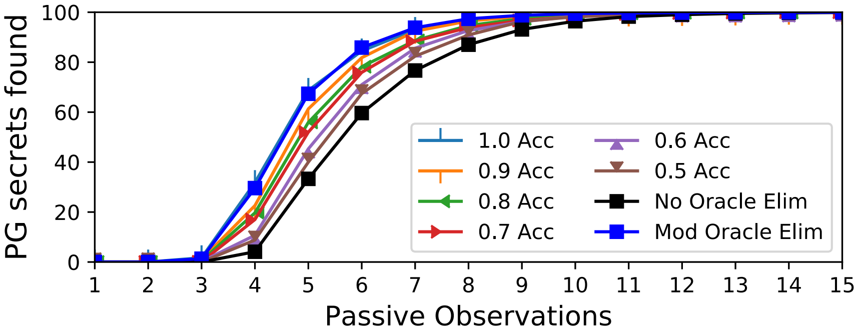


Figure 3.8: The CDF of 1000 PassGrid user secrets found over a increasing number of observations attained by an attacker, with varying degrees of modulus information accuracy. We note that the square markers results eliminate possible secrets with perfect oracles, instead of updating points.

secrets that conflict with the modulus oracle. While we have chosen 10 and 3, as long as the first value is larger than the second, the performance will fall within the performance bounds from the direct elimination of secrets (perfect knowledge).

Figure 3.8 displays a CDF on the percentage of 1000 PassGrids that is found after a given number of observations. This demonstrates the modulus information can be used to enhance an attack on this scheme.

3.6.2 Mod10

The Mod10 method [6,51] is a patented method proposed as an alternative to commonplace PIN authentication. The scheme combines each of the digits in the user's PIN with a one time pad (OTP) communicated through a protected channel. More specifically, for the ith PIN digit $s_i \in \{0, \ldots, 9\}$, the verifier covertly communicates an OTP $o_i \in \{0, \ldots, 9\}$. The user responds with $r_i = s_i + o_i \bmod 10$. It follows that each of the digits is equally likely to be the secret even after observing the response r_i.

We consider the effect of the faulty modulus oracle in obtaining the user's PIN. The knowledge of a modulus event divides the response space, as seen in Table 3.10. This knowledge reduces the number of possible secret pin digits which could be combined with

Table 3.10: Response and Modulus operation (mod performed shaded) of a given secret digit and one time pad.

Response		User Secret Digit									
of Sum		0	1	2	3	4	5	6	7	8	9
One Time Pad digit	0	0	1	2	3	4	5	6	7	8	9
	1	1	2	3	4	5	6	7	8	9	0
	2	2	3	4	5	6	7	8	9	0	1
	3	3	4	5	6	7	8	9	0	1	2
	4	4	5	6	7	8	9	0	1	2	3
	5	5	6	7	8	9	0	1	2	3	4
	6	6	7	8	9	0	1	2	3	4	5
	7	7	8	9	0	1	2	3	4	5	6
	8	8	9	0	1	2	3	4	5	6	7
	9	9	0	1	2	3	4	5	6	7	8

an unknown random OTP to produce the given response. The user's 4-digit secret PIN can be found by updating points to reward secret digits that agree with the response, and the mod-oracle, for each of the 4 secret digits. The algorithm for one PIN digit is shown in Algorithm 3, which can be extended in a modular way to all 4 digits. In the case of a no-modulus event, all secret digits less than or equal to the response r are rewarded (since digits greater than r would require a modulus operation regardless of the OTP). More precisely, since it is a no-modulus event, we necessarily have $s_i + o_i = r_i$ (even without reducing the result modulo 10). If $s_i > r_i$, then this implies $o_i < 0$, a contradiction. Hence, $s_i \leq r_i$. Thus, we reward the points p_0 to p_r in the algorithm. On the other hand, in case of the modulus event, all secret digits greater than the response r_i are rewarded. This follows from the fact that in a modulus event, we necessarily have $o_i + s_i = r_i + 10$. Since, $o_i \leq 9$, this gives us $r_i + 10 \leq 9 + s_i$, and hence $s_i \geq r_i + 1$. Therefore, all points p_{r+1} to p_9 are rewarded in the algorithm. By simulating users on the Mod10 scheme, and using symmetrical oracle accuracies (same TPR and TNR for no-modulus/modulus events) we can find the PIN in (Mod Accuracy, Average Rounds): (1.0, 24.4), (0.9, 36.1), (0.8, 60.20), (0.7, 118.37), (0.6, 409.76). This result is also visually displayed in Figure 3.9.

We note that Cagalj et al. [51] demonstrate timing attacks on the same scheme, exploiting the differences in the user's cognitive load in the addition of the one time pad to their

Algorithm 3: MOD10 POINTS UPDATE

Input: m responses.

Output: A list of points $(p_0, p_1, \ldots, p_9)$, with top score indicating the target secret digit.

```
Initialize $(p_0, p_1, \ldots, p_9)$ to all zeroes.
for $j = 1$ to $m$ do
    Observe auxiliary information 'aux,' and response $r$.
    $b \leftarrow \mathcal{O}_{\text{mod}}^{\text{TPR,TNR}}(c, \text{aux})$.
    if $b = -1$ (no-modulus event) then
        reward $(p_0, \ldots, p_r)$
    else
        reward $(p_{r+1}, \ldots, p_9)$
return $(p_0, p_1, \ldots, p_9)$.
```

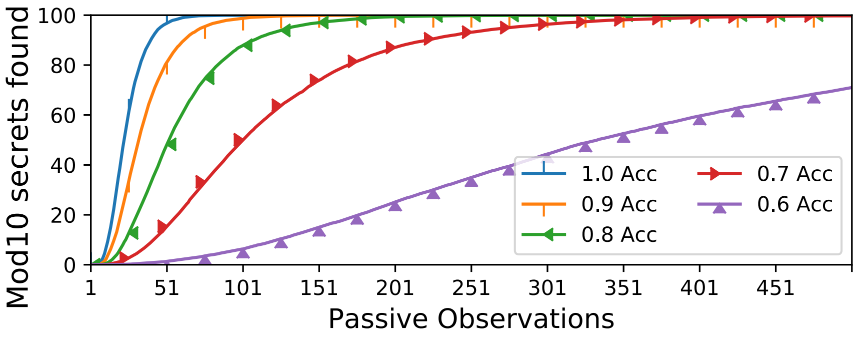


Figure 3.9: The CDF of Mod10 user secrets found over a increasing number of observations attained by an attacker, with varying modulus oracle accuracy.

secret digit. They are able to reduce the entropy of the unknown pin digit by 0.5 bits over an observation, and effectively reducing the candidate size of the pin digit from 10 to 6 (with 90% confidence) over 90 observations. However, their attack does not retrieve the entire secret.

3.7 Related Work

We focus on related work on side-channel attacks on ORAS as well as password and PIN authentication schemes. The most related work to ours is the timing attack from Cagalj et al. [51] who exploit coarse-grained timing information as a side channel. Coarse-grained means that the timing information is limited to the overall time taken to respond to a challenge. They exploit the fact that a user's time to respond to a challenge is directly proportional to the cognitive load (which varies due to randomized weights in the challenge). They demonstrate the susceptibility of the (full) HB protocol and the Mod10 scheme to their timing attack. In contrast, we exploit further information (features other than the total time taken) obtainable via observing the user's eye movement patterns coupled with the observation that a modulus event indicates a high cognitive-load challenge. As a result, our attack is applicable to a broader class of k-out-of-n ORAS (as well as other ORAS that use a modulus operation). Note that the simple timing attack from [51] does not apply to the windowed HB protocol considered in this chapter.

To the best of our knowledge, this is the only work that explores side-channel attacks on ORAS. However, there are numerous studies on side-channel attacks on PIN and password-based schemes, which we summarize next.

Kune and Kim present a side-channel attack that extracts the user's PIN by observing the time taken as the finger travels between PIN digits on the keypad [100]. This timing information enables the attacker to derive the distance traveled, and thus infer potential key pairs the user was moving between. With the key pairs, the attacker reduces the possible space of secrets, to eventually find the user's secret. The attack from [101] uses the position of the phone during PIN entry to determine the location of the secret digit using gyroscope information. The smudge attack [102] is able to infer a user's pass-pattern from the oily residue remaining on the screen from the user's finger when in contact with the screen. We note that the challenge-response pairs in the authentication systems considered in this thesis are already assumed to be known to the attacker, and as a result, these side channel attacks are not applicable to our case.

There has also been some work on using hidden and/or on-device cameras to steal a user's PIN entry. The work in [103] shows how an attacker can use computer vision to determine the exact digit or keyboard letter pressed through a distant camera even if the angle is not optimal (directly facing the screen). Likewise, [104] shows a similar attacker capability who has access to a front facing camera feed of the user, to identify which digit was pressed on an on-screen number pad. They observe that the user (in one-handed operation) may tilt the phone, and consequently the camera to press a digit in a particular location. Thus if the location can be derived from the position of a stationary reference (e.g. the user's face) on the camera feed, so can the secret digit. Our work in this chapter relates to these two works in terms of using a camera recording to detect eye movement patterns; however, as discussed before, the task of retrieving the secret in our case is more involved (as opposed to mere detection of password letters entered).

3.8 Conclusion

We have investigated and successfully exploited the modulus event present in existing observation resilient schemes. We have shown that proposed schemes are vulnerable to eye-movement based side-channel information which indicates the occurrence of the mod events. Through this chapter, we have presented the algorithms to exploit the weight bias in the modulus event, with an attempt at leveraging timing and positional focus found in a user's unconscious behavior in solving the authentication challenges. With the algorithm independent to the side-channel, we speculate there may exist other behavioral features that can be measured and utilized to better improve the overall attack. The development of algorithms to exploit cognitive schemes that involve the modulus like PassGrids or Mod10 demonstrate the value of this leaked information. In this chapter, through analysis, we are able to derive why these side-channels leak information about the secret, present remedies to reduce the amount of information released in observation-resilient authentication schemes, and serves to inform future scheme designers.

Chapter 4

On the Resilience of Biometric Authentication Systems against Random Inputs

This chapter is adapted from work titled "On the Resilience of Biometric Authentication Systems against Random Inputs", published in the Network and Distributed System Security Symposium (NDSS) 2020, completed in conjunction with Zhao, B.Z.H., Asghar, H.J. and Kaafar, M.A.

Moving beyond the ORAS of the previous chapter, we assess the security of machine learning based biometric authentication systems against an attacker who submits uniform random inputs, either as feature vectors or raw inputs, in order to find an *accepting sample* of a target user. Unlike Chapter 3, we do not seek to recover the biometric template of the target user, instead only trick the model to gain access. The average false positive rate (FPR) of the system, i.e., the rate at which an impostor is incorrectly accepted as the legitimate user, may be interpreted as a measure of the success probability of such an attack. However, we show that the success rate is often higher than the FPR. In particular, for one reconstructed biometric system with an average FPR of 0.03, the success rate was as high as 0.78. This has implications for the security of the system, as an attacker with only the knowledge of the length of the feature space can impersonate the user with less than 2 attempts on average. We provide detailed analysis of why the attack is successful and validate our results using four different biometric modalities and four different machine learning classifiers. Finally, we propose mitigation techniques that render such attacks ineffective, with little to no effect on the accuracy of the system.

4.1 Introduction

Consider a machine learning model trained on some user's data accessible as a black-box API for biometric authentication. Given an input (a biometric sample), the model outputs a binary decision, i.e., accept or reject, as its prediction for whether the input belongs to the target user or not. Now imagine an attacker with access to the same API who has never observed the target user's inputs. The goal of the attacker is to impersonate the user by finding an *accepting sample* (input). What is the success probability of such an attacker?

Biometric authentication systems are generally based on either physiological biometrics such as fingerprints [15], face [16, 17], and voice [18, 19]), or behavioral biometrics such as touch [20] and gait [21], the latter category generally used for continuous and implicit authentication of users. These systems are mostly based on machine learning: a binary classifier is trained on the target user's data (positive class) and a subset of data from other users (negative class). This process is used to validate the performance of the machine learning classifier and hence the biometric system [2, 21–29]. The resulting proportion of negative samples (other users' data) successfully gaining access (when they should have been rejected) produces the false positive rate (FPR, also referred as False Acceptance Rate). The target user's model is also verified for their own samples, establishing the false reject rate (FRR). The parameters of the model can be adjusted to obtain the equal error rate (EER) at which point the FPR equals FRR.

Returning to our question, the FPR seems to be a good indicator of the success probability of finding an accepting sample. However, this implicitly assumes that the adversary is a human who submits samples using the same human computer interface as other users, e.g., a smartphone camera in case of face recognition. When the model is accessible via an API the adversary has more freedom in choosing its probing samples. This may happen when the biometric service is hosted on the cloud (online setting) or within a secure enclave on the user's device (local setting). In particular, the attacker is free to sample uniform random inputs. It has previously been stated that the success probability of such an attack is exponentially small [30] or it can be derived from the FPR of the system [31, 32].[1]

In this chapter, we show that uniform random inputs are accepted by biometric systems with a probability that is often higher and independent of the FPR. Moreover, this applies to the setting where the API to the biometric system can be queried using feature vectors after processing raw input as well as at the raw input level. A simple toy example with a single feature can illustrate the reason for the efficacy of the attack. Suppose the feature is normalized within the interval $[0, 1]$. All of the target user's samples (the positive class) lie in the interval $[0, 0.5)$ and the other users' samples (the negative class) lie in the interval $(0.5, 1]$. A "classifier" decides the decision boundary of 0.5, resulting in identically zero FRR and FPR. However, a random sample has a 50% chance of being accepted by the biometric system.[2] The success of the attack shows that the FPR and FRR, metrics used for reporting the accuracy of the classifier, cannot alone be used as proxies for assessing the security of the biometric authentication system.

Our main contributions are as follows:

- We theoretically and experimentally show that in machine learning-based biometric

[1] We note that these observations are made for *distance-based* authentication algorithms and not machine-learning model based algorithms. See Sections 4.5.3 and 4.7 for more details.

[2] This example is an oversimplification. In practice, the training data is almost never nicely separated between the two classes. Also, in higher dimensions, one expects exponentially small volume covered by samples from the positive and negative classes as is explained in Section 4.3.

authentication systems, the *acceptance region*, defined as the region in feature space where the feature vectors are accepted by the classifier, is significantly larger than the true positive region, i.e., the region where the target users samples lie. Moreover, this is true even in higher dimensions, where the true positive region tends to be exponentially small [105].

- As a consequence of the above, we show that an attacker who has access to a biometric system via a black-box feature vector API, can find an accepting sample by simply generating random inputs, at a rate which in many cases is higher than implicated by the FPR. For instance, the success probability of the attack is as high as 0.78 for one of the systems whose EER is only 0.03. The attack requires minimum knowledge of the system: the attacker only needs to know the length of the input feature vector, and permissible range of each feature value (if not normalized).

- We show that the success rate of a random input attack can also be higher than FPR if the attacker can only access the API at the raw input level (before feature extraction). For instance, on one system with an EER of 0.05, the success rate was 0.12. We show that the exponentially small region spanned by these raw random inputs rarely overlaps with the true positive region of any user in the system, owing to the success probability of the attack. Once again the attack only requires minimum knowledge of the system, i.e., the range of values taken by each raw input.

- To analyze real-world applicability of the attack, we reconstruct four biometric authentication schemes. Two of them are physiological, i.e., face recognition [17] and voice authentication [18]. The other two use behavioral traits, i.e., gait authentication [106], and touch (swipes) authentication [20, 107]. For each of these modalities, we use four different classifiers to construct the corresponding biometric system. The classifiers are linear support vector machines (SVM), radial SVM, random forests, and deep neural networks. For each of these systems, we ensure that our implementation has comparable performance to the reference.

- Our experimental evaluations show that the average acceptance region is higher than the EER in 9 out of 16 authentication configurations (classifier-modality pairs), and only one in the remaining 7 has the (measured) average acceptance region of zero. Moreover, for some users, this discrepancy is even higher. For example, in one user model (voice authentication using random forests) the success rate of the random (feature) input is 0.55, when the model's EER is only 0.03, consistent with the system average EER of 0.03.

- We propose mitigation techniques for both the random feature vector and raw input attacks. For the former, we propose the inclusion of beta-distributed noise in the training data, which "tightens" the acceptance region around the true positive region. For the latter, we add feature vectors extracted from a sample of raw inputs in the training data. Both strategies have minimal impact on the FPR and TPR of the system. The mitigation strategy renders the acceptance region to virtually 0 for 6 of the 16 authentication configurations, and for 15 out of 16, makes it lower than FPR. For reproducibility, we have made our codebase public.[3]

We note that a key difference in the use of machine learning in biometric authentication as compared to its use in other areas (e.g., predicting likelihood of diseases through a healthcare dataset) is that the system should only output its decision: accept or reject [108], and not the detailed confidence values, i.e., confidence of the accept or reject decision. This makes our setting different from *membership inference* attacks where it is assumed that the model returns a prediction vector, where each element is the confidence (probability) that the associated class is the likely label of the input sample [33, 34]. In other words, less information is leaked in biometric authentication. Confidence vectors can potentially allow an adversary to find an *accepting sample* by using a hill climbing approach [31], for instance.

[3] Our code is available at: `https://imathatguy.github.io/Acceptance-Region`

4.2 Background and Threat Model

4.2.1 Biometric Authentication Systems

The use of machine learning for authentication is a binary classification problem.[4] The positive class is the target user class, and the negative class is the class of one or more other users. The target user's data for training is obtained during the registration or enrollment phase. For the negative class, the usual process is to use the data of a subset of other users enrolled in the system [2, 21–29, 110]. Following best machine learning practice, the data (from both classes) is split into a training and test set. The model is learned over the training set, and the performance of the classifier, its misclassification rate, is evaluated on the test set.

A raw biometric sample is usually processed to extract relevant features such as fingerprint minutiae or frequency energy components of speech. This defines the feature space for classification. As noted earlier, the security of the biometric system is evaluated via the misclassification rates of the underlying classifier. Two types of error can arise. A type 1 error is when a positive sample (target user sample) has been erroneously classified as negative, which forms the false reject rate (FRR). Type 2 error occurs when a negative sample (from other users) has been misclassified as a positive sample, resulting in the false positive rate (FPR). By tuning the parameters of the classifier, an equal error rate (EER) can be determined which is the rate at which FRR equals FPR. One can also evaluate the performance of the classifier through the receiver operator characteristic (ROC) curve, which shows the full relationship between FRR and FPR as the classifier parameters are varied.

Once a biometric system is set up, i.e., classifier trained, the system takes as input a biometric sample and outputs accept or reject. In a continuous authentication setting,

[4] We note that sometimes a *discrimination model* [109] may also be considered where the goal is to identify the test sample as belonging to one of n users registered in the system. Our focus is on the authentication model. Also, see Section 7.2.

where the user is continually being authenticated in the background, the biometric system requires a continuous stream of user raw inputs. It has been shown that in continuous authentication systems the performance improves if the decisions is made on the *average* feature vector from a set of feature vectors [107, 111, 112].

4.2.2 Biometric API: The Setting

We consider the setting where the biometric system can be accessed via an API. More specifically, the API can be queried by submitting a biometric sample. The response is a binary accept/reject decision.[5] The biometric system could be *local*, in which case the system is implemented in a secure enclave (a trusted computing module), or *cloud-based* (online), in which the decision part of the system resides in a remote server. We consider two types of APIs. The first type requires raw biometric samples, e.g., the input image in the case of face recognition. The second type accepts a feature vector, implying that the feature extraction phase is carried out before the API query. This might be desirable for the following reasons.

- Often the raw input is rather large. For instance, in the case of face recognition, without compression, an image will need every pixel's RGB information to be sent to the server for feature extraction and authentication. In the case of an image of pixel size 60×60, this would require approximately 10.8 KB of data. If the feature extraction was offloaded to the user device, it would produce a 512 length feature embedding, which can take as little as 512 bytes. This also applies to continuous authentication which inherently requires a continual stream of user raw inputs. But often decisions are only made on an *average* of a set of feature vectors [107, 111, 112]. In such systems, only sending the resultant extracted average feature vector to the cloud also reduces communication cost.

[5] For continuous authentication systems, we assume that the decision is returned after a fixed number of one or more biometric samples.

- Recent studies have shown that raw sensory inputs can often be used to track users [113]. Thus, they convey more information than what is simply required for authentication. In this sense, extracting features at the client side serves as an *information minimization* mechanism, only sending the relevant information (extracted feature vectors) to the server to minimize privacy loss.

- Since the machine learning algorithm only compares samples in the feature space, only the feature representation of the template is stored in the system. In this case, it makes sense to do feature extraction prior to querying the system.

From now onwards, when referring to a biometric API we shall assume the *feature vector* based API as the default. We shall explicitly state when the discourse changes to raw inputs. Figure 4.1 illustrates the two APIs.

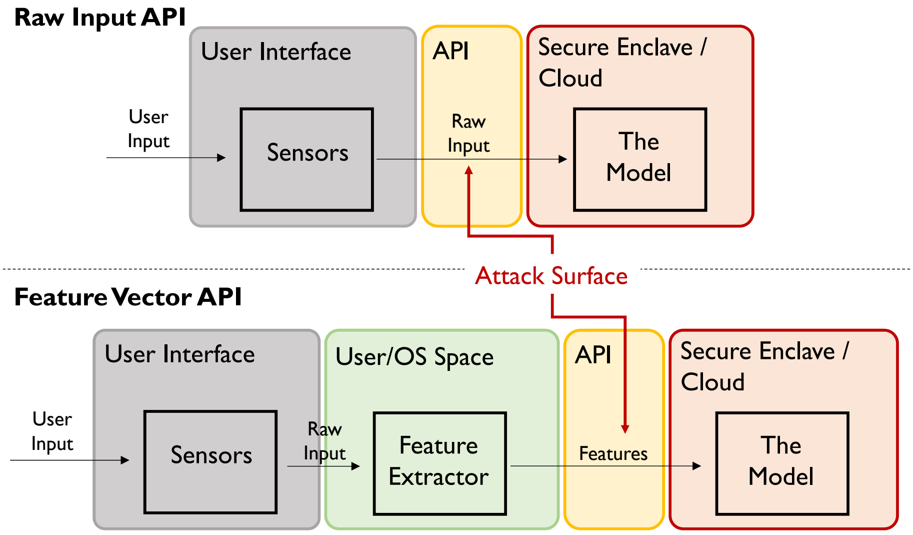


Figure 4.1: The threat model and the two types of biometric API.

4.2.3 Threat Model and Assumptions

We consider an adversary who has access to the API to a biometric system trained with the data of a target user whom the adversary wishes to impersonate. More specifically, the adversary wishes to find an *accepting sample*, i.e., a feature vector for which the system returns "accept." In the case of the raw input API, the adversary is assumed to be in search for a raw input that results in an accepting sample after feature extraction. We assume that the adversary has the means to bypass the end user interface, e.g., touchscreen or phone camera, and can thus *programmatically* submit input samples. There are a number of ways in which this is possible.

In the online setting, a mis-configured API may provide the adversary access to the authentication pipeline. In the local setting, if the feature extractor is contained within a secure environment, raw sensory information must be passed to this protected feature extraction process. To achieve this an attacker may construct their own samples through OS utilities. An example is the Monkey Runner [114] on Android, a tool allowing developers to run a sequence of predefined inputs for product development and testing. Additionally, prior work [115] has shown the possibility of compromising the hardware contained within a mobile device, e.g., a compromised digitizer can inject additional touch events.

Outside of literature, it is difficult to know the exact implementation of real-world systems. However, taking face recognition as an example, we believe our system architecture is similar to real world facial authentication schemes, drawing parallels to pending patent US15/864,232 [116]. Additionally, there are API services dedicated to hosting different components of the facial recognition pipeline. Clarifai, for example, hosts machine learning models dedicated to the extraction of face embeddings within an uploaded image [117]. A developer could then use any number of Machine Learning as a Service (MLaaS) providers to perform the final authentication step, without needing to pay premiums associated with an end-to-end facial recognition product.

We make the following assumptions about the biometric API.

- The input feature length, i.e., the number of features used by the model, is public knowledge.
- Each feature in the feature space is min-max normalized. Thus, each feature takes

value in the real interval $[0, 1]$. This is merely for convenience of analysis. Absent this, the attacker can still assume plausible universal bounds for all features in the feature space.

- The attacker knows the identifier related to the user, e.g., the username, he/she wishes to impersonate.

Beyond this, we do not assume the attacker to have any knowledge of the underlying biometric system including the biometric modality, the classifier being used, the target user's past samples, or any other dataset which would allow the attacker to infer population distribution of the feature space of the given modality.

4.3 Acceptance Region and Proposed Attack

4.3.1 Motivation and Attack Overview

Given a feature space, machine learning classifiers learn the region where feature vectors are classified as positive features and the region where vectors are classified as negative features. We call the former, the acceptance region and the latter the rejection region. Even though the acceptance region is learnt through the data from the target user, it does not necessarily tightly surround the region covered by the target user's samples. Leaving aside the fact that this is desirable so as to not make the model "overfitted" to the training data, this implies that even vectors that do not follow the distribution of the target user's samples, may be accepted. In fact, these vectors may bear no resemblance to any positive or negative samples from the dataset. Consider a toy example, where the feature space consists of only two vectors. The two-dimensional plane in Figure 4.2 shows the distribution of the positive and negative samples in the training and testing datasets. A linear classifier may learn the acceptance and rejection regions split via the decision boundary shown in the figure. This decision boundary divides the two dimensional feature space in half. Even though there is a small overlap between the positive and negative classes, when evaluated against the negative and positive samples from the dataset there would be an acceptably low false positive rate. However, if we construct a vector by uniformly sampling the two features from the interval $[0, 1]$, with probability $1/2$ it will be an accepting sample. If this model could be queried through an API, an attacker is expected to find an accepting sample in two attempts. Arguably, such a system is insecure.

Figure 4.2 illustrates that the acceptance region can be larger than the region covered by the target user's samples. However, in the same example, the area covered by the target user's samples is also quite high, e.g., the convex hull of the samples. As we discuss next, in higher dimensions, the area covered by the positive and negative examples is expected to be concentrated in an exponentially small region [105]. However, the acceptance region does not necessarily follow the same trend.

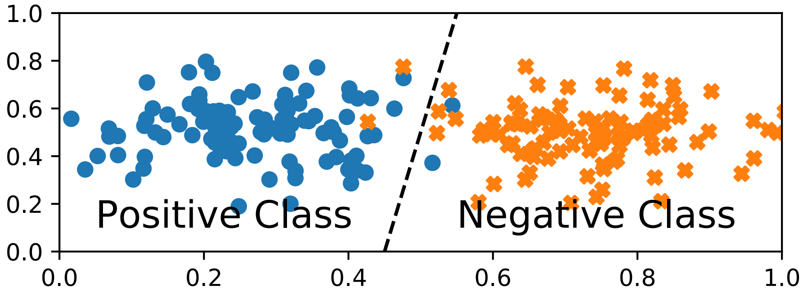


Figure 4.2: Example feature space separation by a linear boundary between two classes. This demonstrates low FPR and FRR of test sample classification, yet allows approximately 50% of the feature space to be accepted as positive.

4.3.2 Acceptance Region

Notations. Let $\mathbb{I} := [0, 1]$ denote the unit interval $[0, 1]$, and let $\mathbb{I}^n := [0, 1]^n$ denote the n-dimensional unit cube with one vertex at the origin. The unit cube represents the feature space with each (min-max normalized) feature taking values in $\mathbb{I}$. Let f denote a model, i.e., an output of a machine learning algorithm (classifier) trained on a dataset $D = \{(\mathbf{x}_i, y_i)\}_{i \in [m]}$, where each $\mathbf{x}_i$ is a feature vector and $y_i \in \{+1, -1\}$ its label. The label $+1$ indicates the positive class (target user) and -1 the negative class (one or more other users of the authentication system). We may denote a positive example in $\mathbf{x} \in D$ by $\mathbf{x}^+$, and any negative example by $\mathbf{x}^-$. The model f takes feature vectors $\mathbf{x} \in \mathbb{I}^n$ as input and outputs a *predicted* label $\hat{y} \in \{+1, -1\}$.

Definitions. *Acceptance region* of a model f is defined as

$$A_f \coloneqq \{\mathbf{x} \in \mathbb{I}^n : f(\mathbf{x}) = +1\}, \tag{4.1}$$

The n-dimensional volume of A_f is denoted $\mathrm{Vol}_n(A_f)$. The definition of acceptance region is analogous to the notion of decision regions in decision theory [118, §1.5]. We will often misuse the word acceptance region to mean both the region or the volume covered by the region where there is no fear of ambiguity. Let FRR and FPR be evaluated on the training dataset D.[6] Let $\mathbf{x} \leftarrow\$ \mathbb{I}^n$ denote sampling a feature vector $\mathbf{x}$ uniformly at random from $\mathbb{I}^n$. In a *random input attack*, the adversary samples $\mathbf{x} \leftarrow\$ \mathbb{I}^n$ and gives it as input to f. The attack is successful if $f(\mathbf{x}) = +1$. The success probability of a random guess is defined as

$$\Pr[f(\mathbf{x}) = +1 : \mathbf{x} \leftarrow\$ \mathbb{I}^n]. \tag{4.2}$$

Since the n-volume of the unit cube is 1, we immediately see that the above probability is exactly $\mathrm{Vol}_n(A_f)$. Thus, we shall use the volume of the acceptance region as a direct measure of the success probability of random guess. Finally, we define the *rejection region* as $\mathbb{I}^n - A_f$. It follows that the volume of the rejection region is $1 - \mathrm{Vol}_n(A_f)$.

Existence Results. Our first observation is that even if the FPR of a model is zero, its acceptance region can still be non-zero. Note that this is not evident from the fact that there are positive examples in the training dataset D: the dataset is finite and there are an infinite number of vectors in $\mathbb{I}^n$, and hence the probability of picking these finite positive examples is zero.

Proposition 4.3.1. *There exists a dataset D and a classifier with output f such that $FRR = FPR = 0$, and $Vol_n(A_f) > 0$.*

Proof. Assume a dataset D that is linearly separable. This means that there exists a

[6] In practice, the FRR and FPR are evaluated against a subset of D called a holdout or testing dataset.

hyperplane denoted $H(\mathbf{x})$ such that for any positive example $\mathbf{x}^+ \in D$, we have $H(\mathbf{x}^+) > 0$ and for any negative example in D we have $H(\mathbf{x}^-) < 0$. Consider the perceptron as an example of a classifier which constructs a linear model: $f_{\mathbf{w},b}(\mathbf{x}) = +1$ if $\langle \mathbf{w}, \mathbf{x} \rangle + b > 0$, and -1 otherwise. Since the data is linearly separable, the perceptron convergence theorem states that the perceptron learning algorithm will find a solution, i.e., a separating hyperplane [119]. Intersecting this hyperplane $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ with the unit cube creates two sectors. The sector where $f_{\mathbf{w},b}(\mathbf{x}) = +1$ is exactly the acceptance region $A_{f_{\mathbf{w},b}}$. The n-volume of $A_{f_{\mathbf{w},b}}$ cannot be zero, since otherwise it is one of the sides of the unit cube with dimension less than n, implying that all points $\langle \mathbf{w}, \mathbf{x} \rangle + b > 0$ lie outside the unit cube. A contradiction, since FRR $= 0$ (there is at least one positive example). □

A non-zero acceptance region is not necessarily a threat. Of practical concern is a *non-negligible* volume. Indeed, the volume may be negligible requiring a large number of queries to f before an *accepting sample* is produced. The following result shows that there are cases in which the acceptance region can be non-negligible.

Proposition 4.3.2. *There exists a dataset D and a classifier with output f such that FRR $=$ FPR $= 0$, and Vol$_n(A_f) \geq 1/2$.*

Proof. Consider again the perceptron as an example of a classifier. Let D be a dataset such that for all positive examples $\mathbf{x}^+$, we have $x_1^+ > 0.5$, and for all negative examples $x_1^- < 0.5$. The rest of the features may take any value in $\mathbb{I}$. The resulting data is linearly separable by the ($n - 1$ dimensional) hyperplane $x_1 - 0.5 = 0$. Initialize the perceptron learning algorithm with $w_1 = 1$, $w_i = 0$ for all $2 \leq i \leq n$, and $b = -0.5$. The algorithm then trivially stops with this hyperplane. Clearly, with this hyperplane, we have FRR $= 0$, FPR $= 0$, and the acceptance region is $1/2$. □

The above examples illustrate the high probability of success of the random input attack due to a non-negligible acceptance region. However, the example used is contrived. In practice, datasets with a “nice” distribution as above are seldom encountered, and the model is more likely to exhibit non-zero generalization error (a tradeoff between FRR and

FPR). Also, in practice, more sophisticated classifiers such as the support vector machine or deep neural networks are used instead of the perceptron. However, we shall demonstrate that the issue persists in real datasets and classifiers used in practice. We remark that we are interested in the case when $\text{Vol}_n(A_f) > \text{FPR}$, since arguably it is misleading to use the FPR as a measure of security of such an authentication system. This could happen even when the FPR is non-zero. When and why would this case occur? We explain this in the following.

Real Data and High Dimensions. We first discretize the feature space. For a given positive integer B, let $\mathbb{I}_B$ denote the binned (or discrete) version of the interval $\mathbb{I}$ partitioned into B equally sized bins. Clearly, each bin is of width $1/B$. Let $\mathbb{I}_B^n$ denote the discretized feature space. Given a set of feature values from $\mathbb{I}$, we say that a bin in $\mathbb{I}_B$ is *filled* if there are $> \epsilon_n$ feature values falling in that bin, where ϵ_n is a cutoff to filter outliers. The number of filled bins is denoted by α. Clearly $\alpha \leq B$. See Figure 4.3.

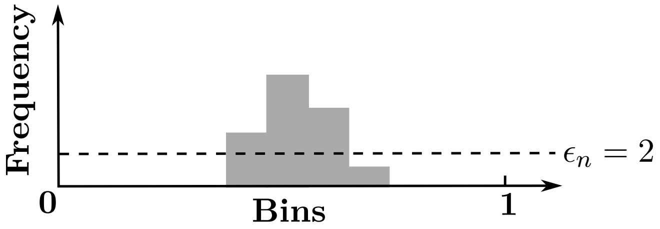


Figure 4.3: The binned version $\mathbb{I}_B$ of the unit interval $\mathbb{I}$. Each bin is of width $1/B$ (B not specified). The number of filled bins is $\alpha = 3$, with a cut-off of $\epsilon_n = 2$.

For the ith feature, let α_i^+ denote the number of bins filled by all positive examples in D. We define:

$$R^+ := \frac{1}{B^n} \prod_{i=1}^{n} \alpha_i^+$$

as the *volume of the true positive region.* We define α_i^- and R^- analogously as the volume of the true negative region. Let $c \in [0, 1]$ be a constant. If each of the α_i^+'s is at most cB, then we see that $R^+ = c^{-n}$. For instance, if $c \leq 1/2$, then $R^+ \leq 2^{-n}$. In other words, the volume of the region spanned by the user's own samples is exponentially small as compared to the volume of the unit cube. In practice, the user's data is expected to be normally distributed across each feature, implying that the α_i^+'s are much smaller than $B/2$, which makes the above volume a loose upper bound. The same is true of the α_i^-'s. Figure 4.4 shows the filled bins from one of the features in the Face Dataset (see Section 4.4.1). For the same dataset, the average true positive region is 5.781×10^{-98} (with a standard deviation of $\pm 2.074 \times 10^{-96}$) and the average true negative region is 1.302×10^{-55} (with a standard deviation of $\pm 2.172 \times 10^{-54}$) computed over 10,000 iterations considering a 80% sample of the target user's data, and a balanced sample of other users.[7]

We thus expect a random vector from $\mathbb{I}^n$ to be outside the region spanned by the target user with overwhelming probability. Thus, if a classifier defines an acceptance region tightly surrounding the target user's data, the volume of the acceptance region will be negligible, and hence the random input attack will not be a threat. However, as we shall show in the next sections, this is not the case in practice.

Factors Affecting Acceptance Region. We list a few factors which affect the volume of the acceptance region.

- One reason for a high acceptance region is that the classifier is not penalized for classifying *empty space* in the feature space as either positive or negative. For instance, consider Figure 4.4. There is significant empty space for the feature depicted in the figure: none of the positive or negative samples have the projected feature value in this space. A classifier is generally trained with an objective to minimize the misclassification rate or a loss function (where, for instance, there is an asymmetrical penalty between true positives and false positives) [118]. These functions take input from the dataset D. Thus, empty regions in the feature space which do not have examples in D can be classified as either of the two classes without being penalized during training, resulting in a non-negligible acceptance region.

[7] We compute the true positive and negative region by only considering the minimum and maximum feature values covered by each user for each feature with binning equal to the floating point precision of the system. Thus, this is a conservative estimate of the true positive region.

- The acceptance region is also expected to be big if there is high variance in the feature values taken by the positive examples. In this case, the α_i^+'s will be much closer to B, resulting in a non-negligible volume R^+.

- On the other hand, the acceptance region is likely to be small if the variances of the feature values in the negative examples are high. The classifier, in order to minimize the FPR, will then increase the region where samples are rejected, which would in turn make the acceptance region closer in volume to the true positive region.

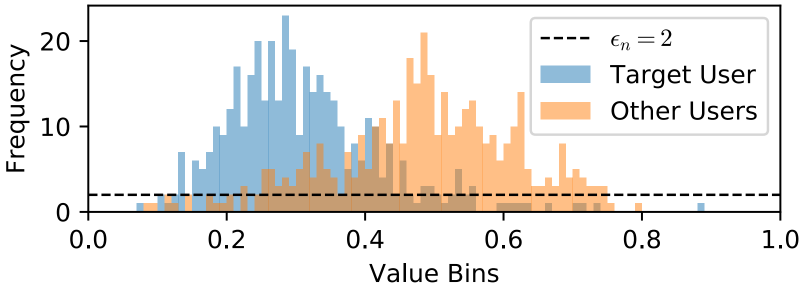


Figure 4.4: The histogram of feature values of one of the features in the Face Dataset (cf. § 4.4.1). Here we have $B = 100$. The number of filled bins for the target user is $\alpha_i^+ = 35$ (with 400 samples), and for the negative class (10 users; same number of total samples) it is $\alpha_i^- = 50$. A total of 24 bins are not filled by any of the two classes, implying that (approximately) 0.24 of the region for this feature is empty.

We empirically verify these observations in Section 4.5. The last observation also hints at a possible method to tighten the acceptance region around the region spanned by the target user: generate random noise around the target user's vectors and treat it as belonging to the negative class. We demonstrate the effectiveness of this method in Section 4.6. Jumping ahead, if the noise is generated from an appropriate distribution, this will have minimal impact on the FRR and FPR of the model.

4.4 Evaluation on Biometric Systems

To evaluate the issue of acceptance region on real-world biometric systems, we chose four different modalities: gait, touch, face, and voice. The last two modalities are used as examples of user authentication at the point of entry into a secured system, whilst gait and touch are often used in continuous authentication systems [120]. We first describe the four biometric datasets, followed by our evaluation methodology, the machine learning algorithms used, and finally our results and observations.

4.4.1 The Biometric Datasets

4.4.1.1 Activity Type (Gait) Dataset

The activity type dataset [121], which we will refer to as the "gait" dataset, was collected for human activity recognition. Specifically, its aim is to provide a dataset for determining if a user is sitting, laying down, walking, running, walking upstairs or downstairs, etc. However, as the dataset retains the unique identifiers for users per biometric record, we re-purpose the dataset for authentication. This dataset contains 30 users, with an average of 343 ± 35 (mean $\pm$ SD) biometric samples per user, there is an equal number of activity type samples for each user. For the purpose of authentication, we do not isolate a specific type of activity. Instead, we include them as values of an additional feature. The activity type feature increases the total number of features to 562. We will refer to these features as *engineered* features as they are manually defined (e.g., by an expert) as opposed to *latent* features extracted from a pre-trained neural network for the face and voice datasets.

4.4.1.2 Touch Dataset

The UMDAA-02 Touch Dataset [20] is a challenge dataset to provide data for researchers to perform baseline evaluations of new touch-based authentication systems. Data was collected from 35 users, with an average of 3667 ± 3012 swipes per user. This dataset was collected by lending mobile devices to the participants over a prolonged period of time. The uncontrolled nature of the collection produces a dataset that accurately reflects swipe interactions with constant and regular use of the device. This dataset contains every touch interaction performed by the user including taps. In a pre-processing step, we only consider sequences with more than 5 data points as swipes. Additionally, we set four binary features to indicate the direction of the swipe, determined from the dominant vertical and horizontal displacement. We retained all other features in [20] bar inter-stroke time, as we wished to treat each swipe independently, without chronological order. We substitute this feature with half-time of the stroke. This produces a total of 27 engineered touch features.

4.4.1.3 Face Dataset

FaceNet [17] proposes a system based on neural networks that can effectively learn embeddings (feature vectors) that represent uniquely identifiable facial information from images. Unlike engineered features, these embeddings may not be directly explainable as they are automatically extracted by the underlying neural network. This neural network can be trained from any dataset containing the labeled faces of individuals. There are many sources from which we can obtain face datasets, CASIA-WebFace [16], VGGFace2 [122] and Labeled Faces in the Wild (LFW) [123] are examples of such datasets. However, with a pre-trained model, we can conserve the time and resources required to re-train the network. The source code for FaceNet [124] contains two pre-trained models available for public use (at the time of writing): one trained on CASIA-WebFace, and another trained on VGGFace2. We opt to use a model pre-trained on VGGFace2[8] , while retaining CASIA-WebFace as our dataset for classifier training. We choose to use different datasets for the training of the embeddings and the classifiers to simulate the exposure of the model to never before seen data. Our face dataset is a subset of CASIA-WebFace containing only the top 100 identities with the largest number of face images (producing 447 ± 103 images per individual). This model produces 512 latent features from input images of pixel size 160x160 which have been centered and aligned. Recall that face alignment involves finding a bounding box on the face of an image, before cropping and resizing to the requested dimensions.

[8] (20180402-114759) is the identifier of pre-trained model used.

4.4.1.4 Speaker Verification (Utterances)

VoxCeleb [18], and VoxCeleb2 [19] are corpuses of spoken recordings by celebrities in online media. These recordings are text-independent, i.e., the phrase uttered by the user is not predetermined. Text-independent speaker verification schemes depart from text-dependent verification schemes in which the individual is bound to repeat a pre-determined speech content. Thus, the task of text-independent verification (or identification) is to distinguish how the user speaks as an individual, instead of how the user utters a specific phrase. The former objective is an arguably harder task. Despite the increased difficulty, researchers have trained neural networks to convert speaker utterances into a set of latent features representing how individuals speak. These works have also released their models to the public, increasing the accessibility of speaker verification to developers. We opt to use the pre-trained model of VoxCeleb [18], with utterances from VoxCeleb2 [19]. From VoxCeleb2, we only use the test portion of the dataset, which contains 118 Users with an average of 406 ± 87 utterances. VoxCeleb was trained as a Siamese neural network [125] for one-shot comparison between two audio samples. A Siamese network consists of two identical branches that produce two equal size outputs from two independent inputs for distance comparison. To fit the pre-trained model into our evaluation of ML-based models, we extract embeddings from one of the twin networks and disregard the second branch. The 1024-length embedding is then used as the feature vector within our evaluation.

4.4.2 Evaluation Methodology

In our creation of biometric models for each user, we seek to obtain the baseline performance of the model with respect to the ability of negative user samples gaining access (i.e. FPR), and the measured Acceptance Region (AR). We use the following methodology to evaluate these metrics for each dataset and each classification algorithm.

1. We min-max normalize each extracted feature over the entire dataset between 0-1.

2. We partition the dataset into a $(70, 30\%)$ split for training and testing sets, respec-

tively.

3. For both training and testing samples, we further sample an equal number of negative samples from every other user such that the total number of negative samples are approximately equal to the number of samples from the target user, representing the positive class, i.e., the positive and negative classes are balanced.

4. Using the balanced training set from step 3, we train a two-class classifier defining the target user set as the positive class, and all remaining users as negative.

5. We test the trained model using the balanced testing set from step 3. This establishes the FRR and FPR of the system.

6. We uniformly sample one million vectors from $\mathbb{I}_n$, where n is the dimension of the extracted features. Testing the set of vectors against the model measures the acceptance region (AR).

7. We record the confidence values of the test prediction for the user's positive test samples, other users' negative test samples, and the uniformly sampled vectors. These confidence values produce ROC curves for FRR, FPR, and AR.

8. Repeat steps 3-7 by iterating through every user in the dataset as the target user.

Remark 4.4.1. *In general, the decision regions (accept and reject in the case of authentication) learned by the classifiers can be quite complex [126]. Hence, it is difficult to determine them analytically, despite the availability of learned model parameters. We instead use a Monte Carlo method by sampling random feature vectors from $\mathbb{I}_n$ where each feature value is sampled uniformly at random from $\mathbb{I}$. With enough samples (one million used in our experiments, and averaged over 50 repetitions), the fraction of random samples accepted by the classifier serves as an estimate of the acceptance region as defined by Eq. 4.2 due to the law of large numbers.*

Remark 4.4.2. *Our evaluation of the biometric systems is using the mock attacker model (samples from the negative class modeled as belonging to an attacker) as it is commonly used [127]. We acknowledge that there are other attack models such as excluding the data of* *the attacker from the training set [127]. Having the attacker data included in the training dataset, as in the mock attacker model, yields better EER. On the other hand, it is also likely to lower the AR of the system, due to increased variance in the negative training dataset. Thus, the use of this model does not inflate our results.*

Remark 4.4.3. *We have used balanced datasets in our experiments, i.e., the number of positive and negative samples being the same. It is true that a balanced dataset is not ideal for minimizing AR; more negative samples may reduce the acceptance region. However, an unbalanced dataset, e.g., more negative samples than positive samples, may be biased towards the negative class, resulting in misleadingly high accuracy [127, 128]. A balanced dataset yields the best EER without being biased towards the positive or negative class.*

4.4.3 Machine Learning Classifiers

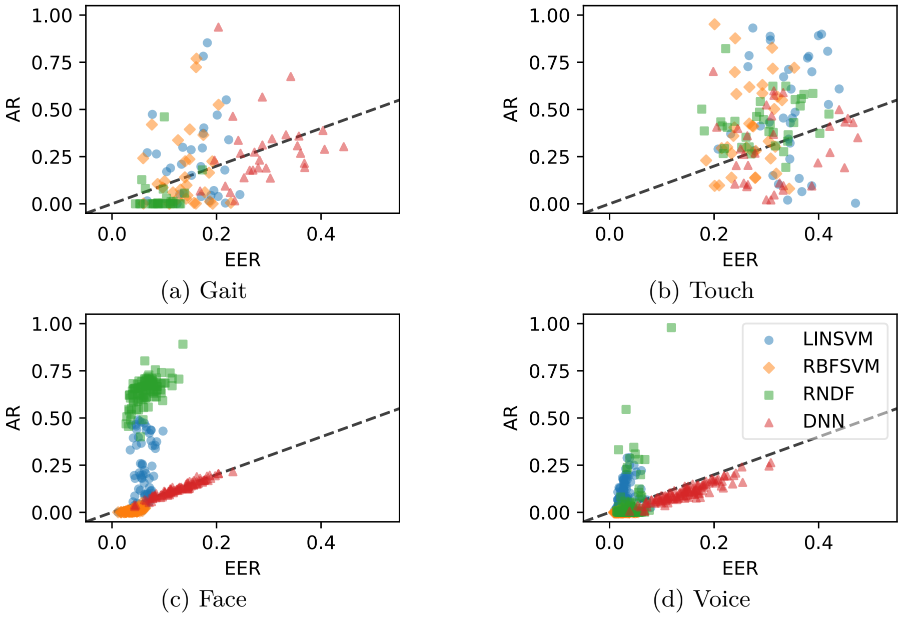
Our initial hypothesis (Section 4.3) stipulates that AR is related to the training data distribution, and not necessarily to any weakness of the classifiers learning from the data. To demonstrate this distinction, we elected four different machine learning algorithms: Support Vector Machines (SVM) with a linear kernel (LinSVM), SVM with a radial basis function kernel (RBFSVM), Random Forests (RNDF), and Deep Neural Networks (DNN). Briefly, SVM uses the training data to construct a decision boundary that maximizes the distance between the closest points of different classes (known as support vectors). The shape of this boundary is dictated by the kernel used; we test both a linear and a radial kernel. Random Forests is an aggregation of multiple decision tree learners formally known as an ensemble method. Multiple learners in the aggregation are created through bagging, whereby the training dataset is split into multiple subsets, each subset training a distinct decision tree. The decisions from the multiple models are then aggregated to produce the random forest's final decision. DNNs are a class of machine learning models that contain hidden layers between an input and an output layer; each layer containing neurons that activate as a function of previous layers. Specifically, we implement a convolutional neural network with hidden layers leading to a final layer of our two classes, accept and reject. All four of these machine learning models are trained as supervised learners. As such, we provide the ground truth labels to the model during training.

The linear SVM was trained with $C = 10^4$, and default values included within Scikit-learn's Python library for the remaining parameters [98]. For radial SVM we also used $C = 10^4$ while keeping the remaining parameters as default. The Random Forests classifier was configured with 100 estimators. DNNs were trained with TensorFlow Estimators [129] with a varying number of internal layers depending on the dataset. The exact configurations are noted in Appendix B.2.

Remark 4.4.4. *We reiterate that our trained models are reconstructions of past works. However, we endeavor that our models recreate error rates similar to the originally reported values on the same dataset. On Mahbub et al.'s touch dataset [20], the authors achieved 0.22 EER with a RNDF classifier, by averaging 16 swipes for a single authentication session. We are able to achieve a comparable EER of 0.21 on RNDF without averaging. For face authentication, we evaluate a subset of CASIA-Webface, consequently, there is no direct comparison. The original Facenet accuracy in verifying pairs of LFW [123] faces is 98.87% [17], but our adoption of model-based authentication is closer to [130], unfortunately, the authors have fixed a threshold for 0 FPR without reporting their TPR. Nagrani, Chung, and Zisserman's voice authenticator [18] reports an EER of 0.078 on a neural network. Our classifiers achieve EERs of 0.03, 0.02, 0.04, and 0.12, which are within range of this benchmark. Our gait authenticator is the exception, it has not been evaluated for authentication with its mixture of activity types. However, a review of gait authentication schemes can be found at [131].*

4.4.4 Acceptance Region: Feature Vector API

In this section, we evaluate the acceptance region (AR) by comparing it against FPR for all 16 authentication configurations (four datasets and four classifiers). In particular, we display ROC curves showing the trade-off between FPR and FRR against the acceptance region (AR) curve as the model thresholds are varied. These results are averaged over all users. While this gives an average picture of the disparity between AR and FPR, it



(a) Gait

(b) Touch

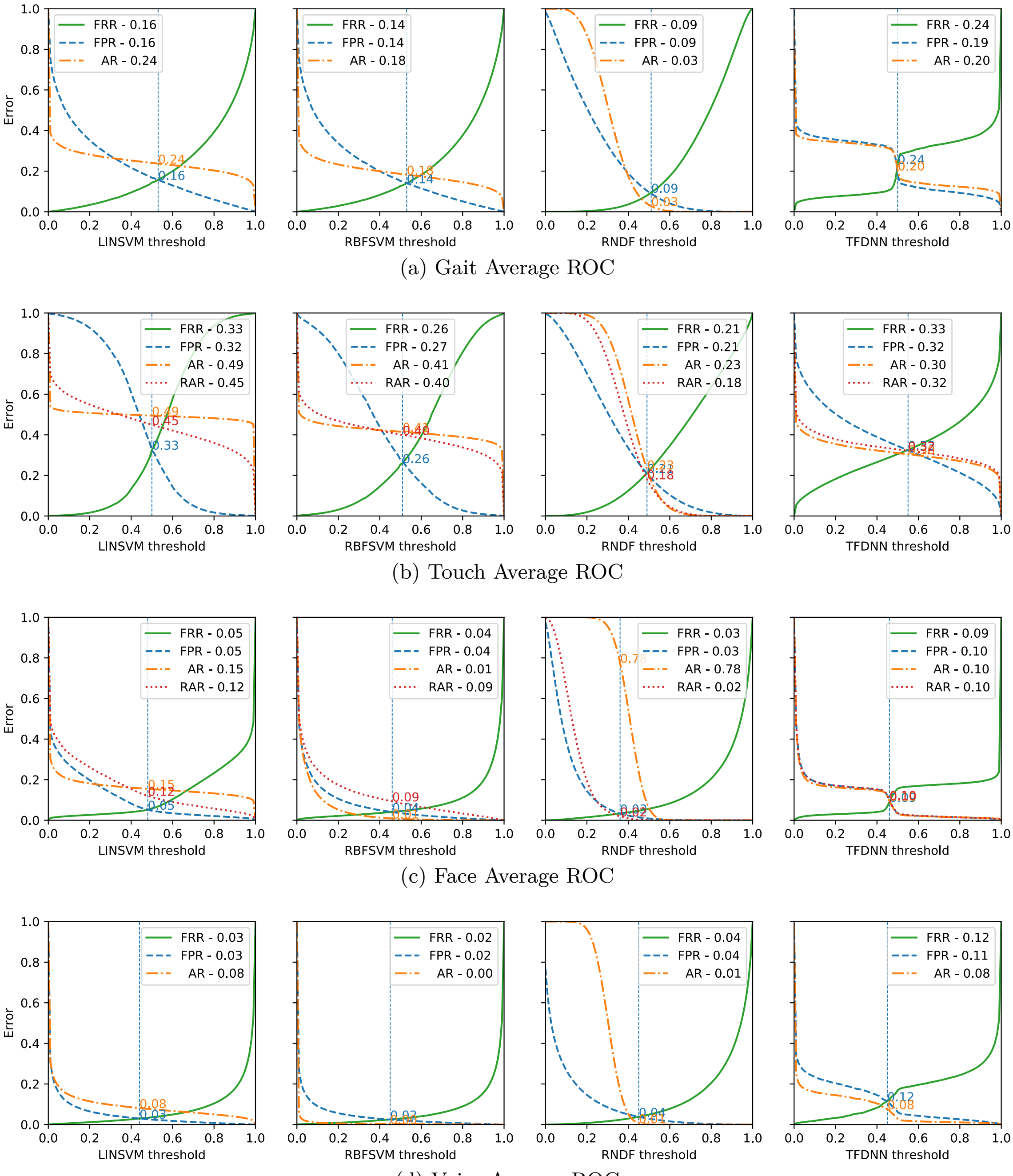
(c) Face

(d) Voice

Figure 4.5: Individual user scatter of AR and FPR. In a majority of configurations, there is no clear relationship between AR and FPR, with the exception of the RBFSVM and DNN classifiers for face and voice authentication.

does not highlight that for some users AR may be substantially higher than FPR, and vice versa. In such a case, the average AR might be misleading. Thus, we also show scattered plots showing per-user AR and FPR, where the FPR is evaluated at EER. The per-user results have been averaged over 50 repetitions to remove any bias resulting from the sampled/generated vectors. The individual user AR versus FPR scatter plots are shown in Figure 4.5, and the (average) AR curves against the ROC curves are shown in Figure 4.6.

Remark 4.4.5. *EER is computed in a best effort manner, with only 100 discretized threshold values, to mitigate the storage demands of the 1M uniformly random vectors measuring AR. Unfortunately, there are some instances whereby the FRR and FPR do not match exactly, as the threshold step induces a large change in both FRR and FPR. Only 1/16 classifiers exhibit an FPR-FRR discrepancy greater than 1%.*



(a) Gait Average ROC

(b) Touch Average ROC

(c) Face Average ROC

(d) Voice Average ROC

Figure 4.6: ROC curve versus the AR and RAR curves for all configurations. The EER is shown as a dotted vertical blue line. The FRR, FPR, AR and RAR values shown in the legend are evaluated at EER (FPR = FRR). The RAR is only evaluated on the Touch and Face datasets.

4.4.4.1 Gait Authentication

Figure 4.5a shows AR against FPR of every user in the activity type (gait) dataset. Recall that in this figure FPR is evaluated at EER. The dotted straight line is the line where AR equals FPR (or ERR). We note that there is a significant proportion of users for which AR is greater than FPR, even when the latter is reasonably low. For instance, in some cases, AR is close to 1.0 when the FPR is around 0.2. Thus, a random input attack on systems trained for these target users will be successful at a rate significantly higher than what is suggested by FPR. We also note that the two SVM classifiers have higher instances of users for whom AR surpasses FPR. Figure 4.6a shows the AR curve averaged across all users against the FPR and FRR curves for all four classifiers. We can see that AR is higher than the ERR (represented by the dotted vertical line) for the two SVM classifiers. For the remaining two classifiers, AR is lower than EER. However, by looking at the AR curve for RNDF, we see that the AR curve is well above the FPR curve when FRR ≤ 0.3. This can be especially problematic if the threshold is set so as to minimize false rejection at the expense of false positives. We also note that the AR curve for DNN closely follows the FPR curve, which may suggest that the AR is not as problematic for this classifier. However, by looking at Figure 4.5a, we see that this is misleading since for some users the AR is significantly higher than FPR, making them vulnerable to random input attacks. Also, note that the AR generally decreases as the threshold is changed at the expense of FRR. However, except for RNDF, the AR for the other three classifiers is significantly higher than zero even for FRR values close to 1.

4.4.4.2 Touch (Swipe) Authentication

The touch authenticator has the highest EER of all four biometric modalities. Very few users attained an EER lower than 0.2 as seen in Figure 4.5b. This is mainly because we consider the setting where the classification decision is being made after each input sample. Previous work has shown EER to improve if the decision is made on an average vector of a few samples some work [2, 107, 111]. Nevertheless, since our focus is on AR, we stick to the per-sample decision setting. Figure 4.5b shows that more than half of the users have ARs larger than FPR, and in some cases where the FPR is fairly low (say 0.2), the AR is higher than 0.5. Unlike gait authentication where the RNDF classifier had ARs less than FPR for the majority of the users, all four algorithms for touch authentication display high vulnerability to the AR based random input attack. When viewing average results in Figure 4.6b, we observe the average AR curve to be very 'flat' for both SVM classifiers and DNN. This indicates that AR for these classifiers remains mostly unchanged even if the threshold is moved closer to the extremes. RNDF once again is the exception, with the AR curve approaching 0 as the threshold is increased.

4.4.4.3 Face Authentication

Figure 4.5c shows that AR is either lower or comparable to FPR for RBFSVM and DNN. Thus, the FPR serves as a good measure of AR in these systems. However, AR for most users is significantly higher than FPR for LinSVM and RNDF. This is true even though the EER of these systems is comparable to the other two as seen in Figure 4.6c. For LINSVM, we have an average AR of 0.15 compared to an EER of 0.05. For RNDF, the situation is worse with the AR reaching 0.78 against an EER of 0.03. We also note that while the AR is equal to FPR for DNN, its value of 0.10 is still worrisome to be resistant to the random input attack. The relatively high FPR for DNN as compared to RBFSVM is likely due to a limited set of training data available in training the neural network.

4.4.4.4 Voice Authentication

Figure 4.5d shows that once again LinSVM and RNDF have a significant proportion of users with AR higher than FPR, whereas for both RBFSVM and DNN the AR of users is comparable to FPR. Looking at the average ARs in Figure 4.6d, we see that interestingly RNDF exhibits an average AR of 0.01 well below the ERR of 0.04. The average suppresses the fact that there is one user in the system with an AR close to 1.0 even with an EER of approximately 0.1, and two other users with an AR of 0.5 and 0.3 for which the EER is significantly below 0.1. Thus these specific users are more susceptible to the random input attack. Only LinSVM has an average AR (0.08) higher than EER (0.03). The average AR of DNN is lower than EER (0.11), but it is still significantly high (0.08). For RBFSVM we have an average AR close to 0.

Observations

In almost every configuration, we can observe that the average AR is either higher than the FPR or at best comparable to it. Furthermore, for some users, the AR is higher than FPR even though the average over all users may not reflect this trend. This demonstrates that an attacker with no prior knowledge of the system can launch an attack against it via the feature vector API. Moreover, for both the linear and radial SVM kernels, and some instances of the DNN classifier, we observe a relatively flat AR curve as the threshold is varied, unlike the gradual convergence to 1 experienced by the FPR and FRR curves. These classifiers thus have a substantial acceptance region that accept samples as positives irrespective of the threshold. Random Forests is the only classifier where the AR curve shows significant drop as the threshold is varied. Random forests sub-divide the training dataset in a process called bagging, where each sub-division is used to train one tree within the forest. With different subsets of data, different training data points will be closer to different empty regions in feature space, thus producing varied predictions. Because the prediction confidence of RNDF is computed from the proportion of trees agreeing with a prediction, the lack of consensus within the ensemble of trees for the empty space may be the reason for the non-flat AR curve.

4.4.5 Acceptance Rate: Raw Input API

The results from the feature vector API are not necessarily reflective of the success rate of a random input attack via the raw input API. One reason for this is that the feature vectors extracted from raw inputs may or may not span the entire feature space, and as a consequence the entire acceptance region. For this reason, we use the term *raw acceptance* *rate* (RAR) to evaluate the probability of successfully finding an accepting sample via raw random inputs. To evaluate RAR, we select the touch and face biometric datasets. The raw input of the touch authenticator is a time-series, whereas for the face authentication system it is an image.

4.4.5.1 Raw Touch Inputs

We used a continuous auto-regressive process (CAR) [132] to generate random time-series. We opted for CAR due to the extremely high likelihood of time-series values having a dependence on previous values. This time-series was then min-max scaled to approximate sensor bounds. For example the `x`-position has a maximum and minimum value of 1980 and 0 respectively, as dictated by the number of pixels on a smartphone screen. Both the duration and length of the time-series were randomly sampled from reasonable bounds: 0.5 to 2.0 seconds and 30 to 200 data-points, respectively. The time-series was subsequently parsed by the same feature extraction process as a legitimate time-series, and the outputs scaled on a feature min-max scale previously fit on real user data. In total, we generate 100,000 time-series, which are used to measure RAR over 50 repetitions of the experiment.

The results of our experiments are shown in Figure 4.6b, with the curve labeled RAR showing the raw acceptance rate as the threshold of each of the classifiers is changed. As we can see, the RAR is large and comparable to AR. This seems to indicate that the region spanned by random inputs covers the acceptance region. However, on closer examination, this happens to be false. The average volume covered by the true positive region for the touch dataset (cf. Section 4.3) is less than $1.289 \times 10^{-4} \pm 5.462 \times 10^{-4}$, yet the volume occupied by the feature vectors extracted from raw inputs is less than 2.609×10^{-6}. This is significantly smaller than the AR for all four classifiers. We will return to this observation shortly.

4.4.5.2 Raw Face Inputs

We generated 100,000 images of size 160x160 pixels, with uniformly sampled RGB values. Feature embeddings were then extracted from the generated images with the pre-trained Facenet model (cf. Section 4.4.1.3). This set of 100,000 raw input vectors, was parsed by a min-max scaler fitted to real user data. We did not align the noisy images, as there is no facial information within the image to align. Note that alignment is normally used in face authentication to detect facial boundaries within an image. Again, we aggregate results over 50 repetitions to remove any potential biases.

The results from these raw inputs are shown in Figure 4.6c. We note that the RAR curve behaves much more similar to the FPR curve than what was previously observed for raw touch inputs. Also, in the particular example of RBFSVM, we obtain a RAR of 0.09 which is significantly higher than the AR (0.01) at an equal error rate. We again computed the true positive region and found that the average is $6.562 \times 10^{-94} \pm 6.521 \times 10^{-93}$. However, the volume covered by the raw inputs (after feature extraction) is only 4.670×10^{-390}, which is negligible compared to the ARs (0.15, 0.01, 0.78, and 0.10 for all four classifiers). Additional analysis shows that only one other user's feature space overlapped with the space of raw inputs, with an overlapped area of 8.317×10^{-407}, many orders of magnitude smaller than both the positive users and the raw feature space itself.

Observations

The threat of a random input attack via raw random inputs is also high, and in some cases greater than the FPR. However, the region spanned by the feature vectors from these raw inputs is exponentially small and hence does not span the acceptance region. Furthermore, the region also does not coincide with any true positive region. This implies that raw inputs may result in a high raw acceptance rate due to the fact that the training data does not have representative vectors in the region spanned by raw inputs. We shall return to this observation when we discuss mitigation strategies in Section 4.6.

4.5 Synthetic Dataset

The analysis in the previous section was limited in the sense that we could not isolate the reasons behind the discrepancy between AR and FPR. Indeed, we saw that for some configurations (dataset-classifier pairs), the AR curve nicely followed the FPR curve, e.g., the face dataset and DNN (Figure 4.5c), whereas for others this was not the case. In order to better understand the factors affecting AR, in this section, we attempt to empirically verify the hypothesized factors affecting the acceptance region outlined in Section 4.3. Namely, high feature variance in a target user's samples is likely to increase AR, and low feature variance in the user samples in the negative class is expected to result in high AR. In both these cases, we expect to achieve a reasonably low EER, but AR may still be significantly greater than FPR. Moreover, if these factors are indeed true, we expect to see similar behavior across all classifiers. To test this we create a synthetic model of a biometric dataset.

4.5.1 Simulating a Biometric Dataset

Let $\mathcal{N}(\mu, \sigma^2)$, denote the normal distribution with mean μ and standard deviation σ. We assume each feature to be normally distributed across all users with slight variations in mean and standard deviation across all features and users. More specifically, our methodology for generating the synthetic dataset is as follows.

1. We model the mean of all n features taking values in the unit interval $\mathbb{I}$ as a normally distributed random variable $\mathcal{N}(\mu_{\text{mn}}, \sigma^2_{\text{mn}}) = \mathcal{N}(0.5, 0.1^2)$. Similarly we model the standard deviation of all n features as another normally distributed random variable $\mathcal{N}(\mu_{\text{var}}, \sigma^2_{\text{var}}) = \mathcal{N}(0.1, 0.07^2)$.

2. For each feature $i \in [n]$, we first sample $\mu_i \leftarrow \mathcal{N}(\mu_{\text{mn}}, \sigma^2_{\text{mn}})$ and $\sigma_i \leftarrow \mathcal{N}(\mu_{\text{var}}, \sigma^2_{\text{var}})$. The resulting normal distribution $\mathcal{N}(\mu_i, \sigma_i^2)$ serves as the population distribution of the mean of the feature i.

3. For each user u, we sample the mean $\mu_{u,i} \leftarrow \mathcal{N}(\mu_i, \sigma_i^2)$. The variance $\sigma_{u,i}^2$ is chosen as the control variable. User u's samples for the ith feature are generated as i.i.d. random variables $\mathcal{N}(\mu_{u,i}, \sigma_{u,i}^2)$, which serves as user u's distribution for the ith feature.

We evaluate the same four types of ML architectures, LinSVM, RBFSVM, RNDF, and DNN. Due to the large number of potential configurations, we evaluate the model performance at a fixed threshold of 0.5. For the experiments, we choose 50 (synthetic) users, with 50 features in the feature space. Each experimental run is repeated 50 times to reduce any potential biases arising from the random process.

4.5.2 Effects of Feature Variance on Acceptance Region

A machine learning model aims to include as many positive samples within the positive region and as many negative samples in the negative region as possible. Given our use of a balanced dataset, neither is favored. As such it is speculated that if elements of the dataset are more varied, the machine learner will learn a boundary encompassing a larger region, which will result in a larger AR.

4.5.2.1 Variable Isolated User Variance and Fixed Population Variance

We first treat one out of the 50 users as an outlier, which we call the *isolated* user. The variance $\sigma_{u,i}^2$ is fixed at $(0.2)^2$ for all other users u and for all features $i \in [n]$. We vary the variance $\sigma_{u_{\text{tgt}},i}$ of the isolated user u_{tgt} from 0.05 to 0.35 in increments of 0.05. Figure 4.7 plots the user's standard deviation ($\sigma_{u_{\text{tgt}},i}$) relative to the fixed population standard deviation ($\sigma_{u,i}$) of 0.2. It is clear the overall AR, FRR, and FPR of the users are not affected by changing feature variance of a single user, despite the isolated user's samples included as part of training and testing data of other users. Conversely, when viewing the AR, FRR, and FPR of the isolated user, we observe a slight increase in FRR and FPR as the relative variance increases. This is due to the positive samples being spread out due to increased variance in the isolated user's samples. However, this is accompanied by a substantially large increase in the acceptance region of this user, approaching 1, i.e., the entire feature space. Furthermore, this trend is visible for all four classifiers.

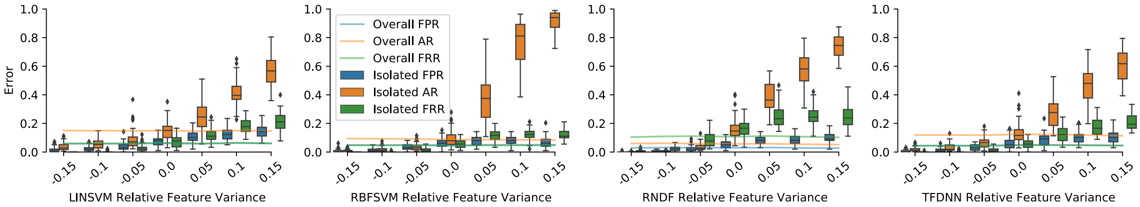


Figure 4.7: A comparison between FPR, AR, four different ML architectures. Trained on synthetic data of 50 features of 50 user, of increasing variance within features for a singular user, repeated 50 times. Note how the system level AR and FPR remains unchanging, despite the isolated user's AR increasing substantially.

4.5.2.2 Fixed Isolated User Variance and Variable Population Variance

In this experiment, we fix the variance $\sigma^2_{u_{\text{tgt}},i}$ of the isolated user (u_{tgt}) at $(0.2)^2$. The $\sigma^2_{u,i}$ of the remaining population is sampled from a normal distribution $\sigma_{u,i} \leftarrow \mathcal{N}(\mu_i, \sigma_i^2)$. Where μ_i and σ_i is sampled from the following distributions $\mathcal{N}(\mu_{\text{mn}}, \sigma^2_{\text{mn}}) = \mathcal{N}(\mu_{\text{mn}}, 0.05^2)$ and $\mathcal{N}(\mu_{\text{var}}, \sigma^2_{\text{var}}) = \mathcal{N}(0.03, 0.02^2)$, respectively. μ_{mn} is varied between 0.05 and 0.35 in increments on 0.05 This sampling permits a small amount of variation between features.

The results are shown in Figure 4.8. Inspecting the average AR, FRR, and FPR of the system, it is evident there is a continual increase of all 3 metrics as the relative variance increases. This increase is expected as the majority of users' feature values have high variance, presenting an increasingly difficult problem for the machine learner to reduce misclassification errors. However, in all four classifiers, the average AR curve is either comparable or lower than the FPR curve as the relative variance increases. For the isolated user, we see that when the relative variance of all other users is lower than this user (to the left), the AR is significantly higher even though the FPR and FRR are minimal in all four classifiers. This shows that less variance in the population samples will result in a high AR, as the classifier need not tighten AR around the true positive region, due to lack of high variance negative samples. On the other hand, AR of the isolated user decreases as the relative variance of the population increases.

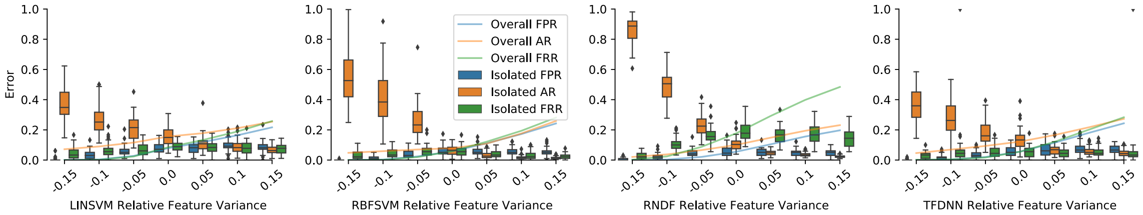


Figure 4.8: A comparison between FPR, AR, four different ML architectures. Trained on synthetic data of 50 features of 50 user, of increasing variance within features of all other users except a singular user, repeated 50 times. The x-axis denotes the relative SD of the population compared with the isolated user.

4.5.3 On Distance Based Classifiers

As noted earlier, it has been stated that random inputs are ineffective against distance-based classification algorithms [30]. This is in contrast to the machine learning based algorithms evaluated in this chapter. We take a brief interlude to experimentally evaluate this claim on the cosine similarity distance-based classifier. We sample 50 features with means distributed as $\mathcal{N}(\mu_{\text{mn}}, \sigma^2_{\text{mn}}) = \mathcal{N}(0.2, 0.05^2)$ and variance distributed as $\mathcal{N}(\mu_{\text{var}}, \sigma^2_{\text{var}}) = \mathcal{N}(0.03, 0.02^2)$. Cosine similarity is computed between two vectors of the same length. As our positive training data contains more than one training sample, we use the average of these samples as the representative template of the user [112]. We use a fixed number of 50 users, with the experiment repeated 50 times. Recall that our evaluation at each threshold is best-effort; we use 1,000 threshold bins for the evaluation of the cosine similarity classifier, since the FRR and FPR rapidly change over a small range of thresholds.

Figure 4.9 displays three classical machine learning algorithms of linear SVM, radial SVM, and random forests, alongside a distance-based cosine similarity classifier. It is clear from the figure, that the AR is near zero for cosine similarity, unlike the other classifiers using the same synthetic dataset. This, however, comes at the cost of higher EER. This suggests that distance-based classifiers are effective in minimizing the AR of model, but at the expense of accuracy in the system. We leave further investigation of distance-based classifiers as future work.

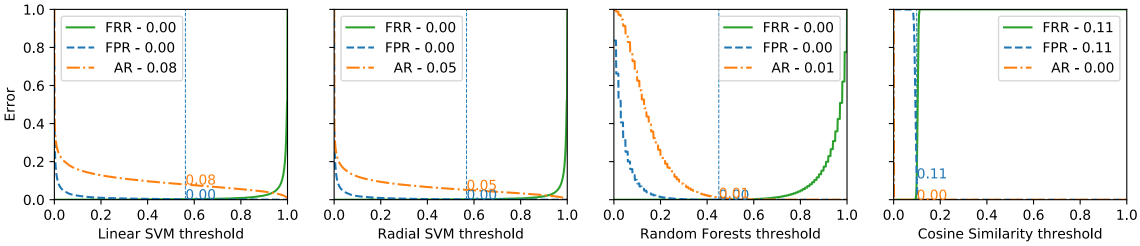


Figure 4.9: ROC Curves versus the AR curve for different ML architectures, including a cosine similarity distance-based classifier. Trained on synthetic data of 50 features of 50 user, with fixed mean and variance for features of all users, repeated 50 times.

4.5.4 Effects of Increasing Synthetic Users

The real-world datasets used in Section 4.4 have a variable number of users. Our binary classification task aggregates negative user samples into a negative class, resulting in distributions and variances of the negative class which depend on the number of users in the datasets. Thus, in this test, we investigate the impact on TPR, FPR, and AR by varying the number of users in the dataset. We use the synthetic dataset configured in the same manner as in Section 4.5.3. We increase the number of users within the synthetic dataset, from 25 to 150, in increments of 25. Note that the split between positive and negative samples is still balanced (see Remark 4.4.3).

In Figure 4.10, we observe that with the addition of more users, there is a slight increase in the FPR. This is expected as the likelihood of user features being similar between any two users will increase with more users in the population. As the training of the classifier uses samples from other users as a negative class, the increased number of negative users slightly lowers the AR of the classifier, with an increased variation of the negative training set (from additional users) covering more of the feature space. However, both these changes are relatively minor despite the multi-fold increase in the number of users. Thus, the AR of the classifiers remains relatively stable with an increasing number of users.

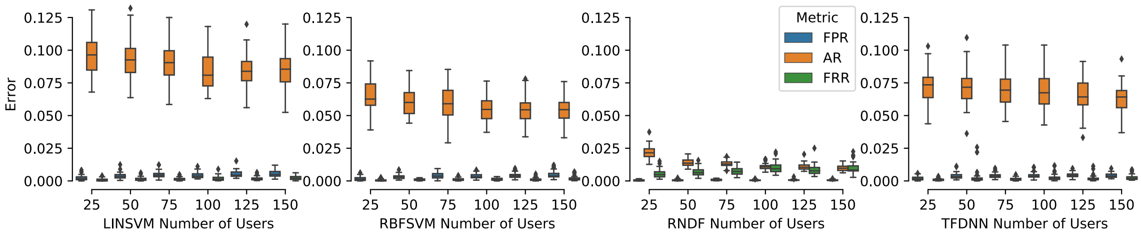


Figure 4.10: A comparison between FPR and AR of four different ML architectures. Trained on synthetic data of 50 features per user, with a variable number of users, repeated 50 times.

4.6 Mitigation

In the previous section, we validated that higher variance in the samples in the negative class as compared to the variance of samples from the target user class reduces AR. The data from the negative class is obtained from real user samples, and therefore scheme designers cannot control the variance. However, this gives us a simple idea to minimize AR: generate noise vectors around the target user's vectors and treat them as part of the negative class for training the model. This will result in the tightening of the acceptance region around the true positive region. We remark that the noise generated is independent of the negative training samples.

4.6.1 The Beta Distribution

More specifically, we generate additional negative training samples by sampling noisy vectors where each feature value is sampled from a beta distribution. We generate samples equal to the number of samples in the positive class. Thus creating a dataset with a third of the samples as positive, another third as negative samples from other users, and finally the remaining third of feature vectors treated as negative samples from the beta distribution dependent on the positive user. The procedure is as follows. For the ith feature, let μ_i denote the mean value for the given target user. We use the beta distribution with parameters $\alpha_i = |0.5 - \mu_i| + 0.5$ and $\beta_i = 0.5$. We denote the resulting beta distribution by $\mathcal{B}e(\alpha_i, \beta_i)$. Then a noisy sample $\mathbf{x}$ is constructed by sampling its ith element x_i from the distribution $\mathcal{B}e(\alpha_i, \beta_i)$ if $\mu_i \leq 0.5$, and from $1 - \mathcal{B}e(\alpha_i, \beta_i)$ otherwise. The two cases ensure that we add symmetric noise as the mean moves over to either side of 0.5.

Table 4.1: Equal Error Rate and AR with and without the mitigation strategy. Green (resp., red) shades highlight improvement (resp., deterioration) in FPR and AR. Color intensity is proportional to degree of performance change.

	Linear SVM				Radial Svm				Random Forest				Deep Neural Network			
Biometric	Normal		Mitigation		Normal		Mitigation		Normal		Mitigation		Normal		Mitigation	
Modality	FPR	AR	FPR	AR	FPR	AR	FPR	AR	FPR	AR	FPR	AR	FPR	AR	FPR	AR
Gait	0.160	0.24	0.160	0.04	0.140	0.18	0.140	0.04	0.09	0.03	0.09	0.00	0.215	0.20	0.170	0.00
Touch	0.325	0.49	0.340	0.01	0.265	0.41	0.265	0.03	0.21	0.23	0.21	0.00	0.325	0.30	0.375	0.00
Face	0.050	0.15	0.065	0.11	0.040	0.01	0.040	0.01	0.03	0.78	0.03	0.00	0.095	0.10	0.065	0.04
Voice	0.030	0.08	0.030	0.06	0.020	0.00	0.020	0.00	0.04	0.01	0.04	0.00	0.115	0.08	0.090	0.02

Results on AR. In Table 4.1, we show the resulting FPR and AR after the addition of beta noise at the equal error rate. The detailed ROC curves are shown in Figure B.2 in Appendix B.1. In every configuration (classifier-dataset pairs), we see a significant decrease in AR. The AR is now lower than FPR in every configuration. In 14 out of 16 cases, the AR is $\leq$ 0.04. The two exceptions are LinSVM (with face and voice datasets). We further see that in 13 out of 16 instances the FPR either remains unchanged or improves! The 3 instances where the FPR degrades are LinSVM with face and face datasets both by +0.015, and DNN with Touch where the difference is +0.05. Thus, adding beta distributed noise does indeed decrease the AR with minimal impact on FPR. This agrees with our postulate that high AR was likely due to loose decision boundaries drawn by the classifier, and the addition of beta noise tightens this around the true positive region. Figure B.1 in Appendix B.1 displays individual user FPRs and ARs.

Table 4.2: Equal Error Rate and RAR with and without the mitigation strategy. The AR values remain the same as in Table 4.1. β-RAR indicates RAR treated with only β noise. RAR indicates the inclusion of both β noise and raw random input samples.

	Linear SVM					Radial Svm					Random Forest					Deep Neural Network				
Biometric	Normal		Mitigation			Normal		Mitigation			Normal		Mitigation			Normal		Mitigation		
Modality	FPR	RAR	FPR	β-RAR	RAR	FPR	RAR	FPR	β-RAR	RAR	FPR	RAR	FPR	β-RAR	RAR	FPR	RAR	FPR	β-RAR	RAR
Touch Raw	0.325	0.45	0.345	0.44	0.00	0.265	0.40	0.265	0.36	0.01	0.21	0.18	0.215	0.05	0.00	0.325	0.32	0.38	0.26	0.00
Face Raw	0.050	0.12	0.075	0.14	0.00	0.040	0.09	0.040	0.09	0.00	0.03	0.02	0.030	0.01	0.00	0.095	0.10	0.07	0.06	0.03

Results on RAR. Interestingly, beta distributed noise only marginally reduces the raw acceptance rate as can be seen in Table 4.2 (columns labeled β-RAR). The reason for this lies in the volume of the region spanned by random raw inputs. We previously saw in Section 4.4.5 that it was (a) exponentially small and (b) many orders of magnitude smaller than the true positive region. Thus, it is unlikely that beta distributed noise will lie in this region to aid the model to label them as negative samples. Consequently, we sought another means to mitigate this attack surface.

4.6.2 Feature Vectors from Raw Inputs as Negative Samples

Our mitigation strategy to reduce RAR is to include a subset of raw input vectors in the training process, whose cardinality is equal to the number of positive user samples in the training dataset. The training dataset now contains 1/4th each of raw input vectors, beta-noise, positive samples, and samples from other users.

Results on AR and RAR. Table 4.2 shows that the mitigation strategy reduces the RAR to less than or equal to 0.03 in all instances (columns labeled RAR). The resulting FPR is marginally higher than the FPR from only beta-distributed noise in some cases (Table 4.1). Thus, the inclusion of beta-distributed noise in conjunction with a subset of raw inputs in the training data reduces both AR and RAR with minimal impact on FPR and FRR.

4.7 Related Work

There are several mentions of attacks similar to the random input attack discussed in this chapter. Pagnin et al. [30] define a *blind brute-force attack* on biometric systems where the attacker submits random inputs to find an accepting sample. The inputs are n-element vectors whose elements are integers in the set $\{0, 1, \ldots, q-1\}$. The authors conclude that the probability of success of this attack is exponential in n, assuming that the authentication is done via a distance function (discarding any vector outside the ball of radius determined by the system threshold). They concluded that a blind brute force attack is not effective in recovering an accepting sample. While this may apply to distance-based matching, the same conclusion cannot be made about machine learning based algorithms whose decision functions are more involved. Indeed, we have shown that the acceptance region for machine learning classifiers is not exponentially small. It has also been argued that the success rate of random input attacks can be determined by the false positive rate (FPR), at least in the case of fingerprint and face authentication [31, 32]. We have shown that for sophisticated machine learning classifiers this conclusion is not true, and random input attacks in many instances succeed at a rate higher than FPR. A more involved method is hill-climbing [31, 133] which seeks an accepting sample via exploiting the confidence scores returned by the matching algorithm. The authentication systems considered in this chapter do not return confidence scores.

Serwadda and Phoha [61] use a robotic finger and population statistics of touch behavior on smartphones to launch a physical attack on touch-based biometric authentication systems. Their attack reduces the accuracy of the system by increasing the EER. In contrast, our work does not assume any knowledge of population biometric statistics, e.g., the population distribution of feature space. It is an interesting area of work to investigate whether a robotic finger can be programmed to generate raw inputs used in our attack.

Garcia et al. [108] use explainable-AI techniques [130] to construct queries (feature vectors) to find an accepting sample in machine learning based biometric authentication systems. On a system with 0 FPR, they show that their attack is successful in breaching the system with up to 93% success rate. However, their attack is more involved: it requires the construction of a seed dataset containing representative accepting and rejecting samples of a user set chosen by the adversary. This dataset trains a neural network as a substitute to the classifier of the authentication system. The adversary then uses explainable AI techniques to obtain an accepting sample of a target user (not in the seed dataset) in as few queries as possible, by updating the substitute network. The authors also report a random feature vector attack, however, the attack is only successful on one out of 16 victims. The random feature vector is constructed by sampling each feature value via a normal distribution (distribution parameters not stated), unlike the uniform distribution in our case. We also note that they propose including images with randomly perturbed pixels as a counter-measure to defend against the aforementioned random input attack. This is different from our proposed beta-distributed noise mitigation technique, as it is agnostic to the underlying biometric modality.

The frog-boiling attack [134, 135] studies the impact of gradual variations in training data samples to manipulate the classifier decision boundary. In this work we do not consider the adversary with access to the training process, nor do we evaluate models with an iterative update process. If this threat model is considered for the problem addressed in this chapter, then an adversary may seek to maximize the acceptance region of a model by gradually poisoning the training dataset. As we have demonstrated in Section 4.5, the relative variance between the user's data and population dataset directly impacts AR. Thus the manipulation of a user's training samples to be more varied would be effective in increasing the AR. Likewise, in our mitigation technique, we have shown that beta-distributed noise is effective in the minimization of AR. However, an adversary might poison the training data by labeling beta noise as positive samples resulting in a maximization of the acceptance region to near 100% of the feature space.

Our work is different from another line of work that targets machine learning models in general. For instance, the work in [75] shows an *evasion attack* where the adversary, through only blackbox access to a neural network, forces the classifier to misclassify an input by slightly perturbing the input even though the perturbed sample is perceptually similar to the original sample, e.g., noisy images. The attack can be applicable to the authentication setting as well. However, it relies on the confidence values (probability vectors) returned by the classifier, which is not the case in authentication. Similarly, the work in [82] shows how to steal a machine learning model, i.e., retrieve its undisclosed parameters, which only returns class labels (accept/reject decision in the case of authentication). They describe several techniques including the Lowd and Meek attack [136] to retrieve a model sufficiently similar to the target model. The machine learning models considered in their attack are for applications different from authentication where one expects to find an accepting sample with negligible probability.

There are also proposals to defend against the above mentioned evasion attacks. The goal is to make the classifiers *robust* against adversarial inputs in the sense that classification is constant within a ball of a certain radius around each input [137, 138]. Madry et al. [137] propose a theoretical framework that formalizes defense against adversarial attacks by including adversarially perturbed samples in the loss function of DNNs. They show that it is possible to train DNNs robust against a wide range of adversarial input attacks. Cao and Gong [139] propose another defense where given a test input, random points within a hypercube surrounding the input are sampled, and the majority label returned by the already trained DNN is assigned to the test input. Randomized smoothing [138] creates a separate classifier from any classifier such that its prediction within a Gaussian noise region (ball) around any input is constant, and consequently less likely to produce an erroneous prediction. We note that in evasion attacks there is a notion of *nearness*, i.e., the adversary is given an input and seeks to add a small amount of noise such that the resultant erroneously labeled input is close to the original input. In contrast, in our case, the random input need not be close to the target user's samples or even follow the same distribution. Furthermore, we have shown that even a conservative estimate of the true positive region is negligible in comparison to the entirety of the feature space (Section 4.3.2). Thus, it is unclear whether such defenses apply to uniform random inputs, as opposed to random perturbations of inputs.

Membership inference attacks [33, 34] attempt to determine if a record obtained by an adversary was part of the original training data of the model. Whilst this attack does not compromise the security of the model, it breaches the privacy of the individual records. These attacks create a *shadow model* [33] to mimic the behavior of the target model. Salem et al. [34] construct a shadow model using only positive class samples and negative noise generated via uniformly random feature vectors. However, it is hypothesized that these random samples belong to non-members, i.e., the negative class [34, §V.B]. We have shown that a large portion of these random inputs may also belong to the positive class.

Finally, we point to other works in literature analyzing the security of biometric authentication systems. Sugrim et al. [128] survey and evaluate a range of performance metrics used in biometric authentication schemes. They seek to motivate scheme designers to leverage robust metrics to provide a complete description of the system, including a proposal of the new metric: Frequency Count Score (FCS). The FCS metric shows a distribution of scores of legitimate and unauthorized users, identifying the overlap between the two distributions which helps to select the appropriate threshold for the classification decision. The FCS, however, is dependent on the negative class or samples of other users, which does not include random inputs. The work in [140] investigates the accuracy of authentication systems reported on a small number of participants when evaluated over an increasing number of users. The authors suggest that the performance limits of a system with a small number of participants should be evaluated iteratively by increasing the participant count until the performance degrades below a tolerable limit.

4.8 Conclusion

It is important to assess the security of biometric authentication systems against random input attacks akin to the security of passwords against random guess attacks. We have demonstrated that without intentionally including random inputs as part of the training process of the underlying machine learning algorithm, the authentication system is likely to be susceptible to random input attacks at a rate higher than indicated by EER. Absent any other detection mechanism, e.g., liveliness detection, this renders the system vulnerable. The mitigation measures proposed in this chapter can be adopted to defend against such attacks.

Chapter 5

On the (In)Feasibility of Attribute Inference Attacks on Machine Learning Models

This chapter is adapted from work titled "On the (In)Feasibility of Attribute Inference Attacks on Machine Learning Models", accepted in the 6th IEEE European Symposium on Security and Privacy 2021, completed in conjunction with Zhao, B.Z.H., Agrawal, A., Coburn, C., Asghar, H.J., Bhaskar, R., Kaafar, M.A., Webb, D., and Dickinson, P.

With an increase in low-cost machine learning APIs, advanced machine learning models may be trained on private datasets and monetized by providing them as a service. In addition, an ease of use has allowed the deployment of models in an increasing number of domains, including Biometric Authentication models, which we observed in Chapter 4. However, privacy researchers have demonstrated that these models may leak information about records in the training dataset via membership inference attacks. In this chapter, we take a closer look at another inference attack reported in literature, called attribute inference, whereby an attacker tries to infer missing attributes of a partially known record used in the training dataset by accessing the machine learning model as an API. We show that even if a classification model succumbs to membership inference attacks, it is unlikely to be susceptible to attribute inference attacks. We demonstrate that this is because membership inference attacks fail to distinguish a member from a nearby non-member. We call the ability of an attacker to distinguish the two (similar) vectors as strong membership inference. We show that membership inference attacks cannot infer membership in this strong setting, and hence inferring attributes is infeasible. However, under a relaxed notion of attribute inference, called approximate attribute inference, we show that it is possible to infer attributes close to the true attributes. We verify our results on three publicly available datasets, five membership, and three attribute inference attacks reported in literature.

5.1 Introduction

The introduction of low-cost machine learning APIs from Google, Microsoft, Amazon, IBM, etc., has enabled many companies to monetize advanced machine learning models trained on private datasets by exposing them as a service. This also caught the attention privacy researchers who have shown that these models may leak information about the records in the training dataset via membership inference (MI) attacks. In an MI attack, the adversary (for instance, a user of the service) with API access to the model, can use the model's responses (class labels and probability/confidence of each label) on input records of his/her choice to infer whether a target input was part of the training dataset or not. This can be a serious privacy breach when the underlying dataset is sensitive, e.g., medical data, mobility traces and financial transactions [33, 34].

To date, membership inference attacks have been the primary focus of studies that have contemplated on traits of the datasets and machine learning models that impact the attacks' likelihood and accuracy [1, 33–36]. Our focus is on a related, and perhaps a more likely attack in practice, where the adversary with partial background knowledge of a target's record seeks to complete its knowledge of the missing attributes by observing the model's responses. This attack is called *model inversion* [37, 38], or in general *attribute* *inference* (AI) [35]. Yeom et al. [35] provide a formal definition of an AI adversary, and argue that this adversary can infer the missing attribute values by using an MI adversary as a subroutine. More precisely, for a missing attribute with t possible values, the AI adversary constructs t different input (feature) vectors, gives them as input to the MI adversary, and outputs the attribute value which corresponds to the vector that the MI adversary deems to be in the training dataset.

Beyond providing a formal definition, Yeom et al. experimentally validate the success of an AI attack on regression models, and conclude that the more overfit the model, the higher the success of the AI attack [35, §6.3]. Seeking to replicate their results on classification models (rather than regression models), where the adversary is given a partial record and its true label, our results in this chapter turn out to be different. We show that even if the target classification model is susceptible to MI attacks, AI attacks on the same model have negligible advantage. Furthermore, the results persist even for highly overfitted models. We explore the reasons behind this failure, and find that in order for AI attacks to be successful, the underlying MI attack, used as a subroutine, should be able to infer membership in a stronger sense. More precisely, the MI attack should be able to distinguish between a member of the training dataset and any non-members that are *close* to that member, according to a suitable distance metric (we consider several such distance metrics based on the nature of the dataset). We call this, strong membership inference (SMI), parameterized by the distance from the training dataset.

We formulate the notion of SMI, and prove that a successful MI attack does not necessarily mean a successful SMI attack. Furthermore, we also formally show that a successful SMI attack is essential for an AI attack. This result implies that even a standalone AI attack, which does not use an MI attack as a subroutine, is bound to fail if SMI attacks are unsuccessful. We experimentally validate these results by evaluating several proposed MI attacks from the literature on several discrete and continuous datasets, and target machine learning models, and show that while these attacks are successful in inferring membership, they fall well short as an SMI attack, and consequently as an AI attack. On the positive side (from an attacker's point of view), we investigate a more relaxed notion of attribute inference, called *approximate attribute inference* (AAI), where the adversary is only tasked with finding attributes *close* to the target attributes, according to a given distance metric. We show that while AI attacks are not applicable, AAI attacks perform significantly better, and improve as the target model becomes more overfit. The AAI notion is also a natural extension of the (exact) AI notion for continuous attributes which has mostly been used in discrete settings [35].

In more detail, our main contributions are as follows.

- We provide a formal treatment of membership, attribute, and approximate attribute inference attacks, and propose a new definition of strong membership inference (SMI), building on the work from [35] on the definitions of MI and AI in Section 5.2. We formally prove that an SMI adversary is *strictly stronger* than an MI adversary (Theorem 5.2.1), and that SMI is necessary for AI (Theorem 5.2.2).

- We experimentally validate our theoretical findings through an extensive set of experiments involving five MI attacks, three black-box and two white-box, from [1,33–35], eight datasets (constructed from 3 main binary and continuous datasets), and several target machine learning models (neural networks, support vector machines, logistic regression, and random forests) (cf. Section 5.3). Our results in Section 5.4 validate our formal separation and show that while these attacks are successful to infer membership, they are ineffective in inferring membership at distances close to the training dataset (SMI).

- In Section 5.5, we further construct 3 AI attacks using the MI attacks of [33, 35] and [34] as a subroutine, and show via experiments that these attacks are not effective in inferring attributes, even if we increase the overfitting levels of the target model. On the other hand, we show that our constructed AI attacks can approximately infer attributes (AAI), with the advantage increasing as the level of overfit of the target model increases.

- Our other key findings include explanation behind the seemingly contradictory conclusions about AI attacks on regression models [35] and classification models (our focus) in Section 5.5.1. We also show that the success of an MI attack is dependent on the class label of the vector; if the corresponding class occupies an overwhelmingly large portion of the feature space, then training records belonging to this class are harder to distinguish from non-members (cf. Section 5.4.1.3). This gives one plausible reason why MI attacks have always performed poorly on target models for binary classification problems [33, 34].

5.2 Formal Treatment of Membership and Attribute Inference Attacks

In this section, we formally introduce the privacy notions of strong membership inference (SMI), and recap the notions of membership, attribute and approximate attribute inference. In order to define them, we need rigorous definitions of a distance metric on the feature space, missing (features) attributes of a feature vector and its relation to distance, and how the probability distribution on the feature space behaves around feature vectors. We first define these concepts in the next section followed by privacy definitions in Section 5.2.2.

5.2.1 Notation and Definitions

Feature Space. Let $\mathbb{D}$ denote a subset of the real space $\mathbb{R}$. We assume the feature space to be $\mathbb{D}^m$, where each point $\mathbf{x} \in \mathbb{D}^m$ is called a feature vector consisting of m elements/features. We assume the output space to be $Y = \mathbb{R}^*$. Let $\mathcal{D}$ be a distribution over $\mathbb{D}^m$. The *training* dataset X is defined as a multiset of n elements drawn i.i.d. from $\mathbb{D}^m$ with distribution $\mathcal{D}$. Each $\mathbf{x} \in X$ is accompanied by its *true* label $\mathbf{y} \in Y$. We denote this mapping by c, which we call the *target concept* following standard terminology [141, 142]. Thus, for each $\mathbf{x} \in X$, $c(\mathbf{x})$ denotes is true label. The term label is used generically; it may be discrete, denoting different classes, or it may be continuous, denoting the confidence or probability score for the different classes. The support of distribution $\mathcal{D}$ is defined as

$\text{supp}(\mathcal{D}) = \{\mathbf{x} \in \mathbb{D}^m \mid p_{\mathbf{x}} > 0\}$, where $p_{\mathbf{x}}$ is $\Pr_{\mathcal{D}}(\mathbf{x})$ if $\mathbb{D}^m$ is discrete and $f_{\mathcal{D}}(\mathbf{x})$ if $\mathbb{D}^m$ is continuous, f being the probability density function. The notation $a \leftarrow\$ A$ indicates sampling an element a from some set A uniformly at random. The notation $\mathbf{x} \leftarrow \mathcal{D}$ denotes sampling a feature vector according to the distribution $\mathcal{D}$. Similarly, the notation $X \leftarrow \mathcal{D}^n$ denotes sampling a multiset of n feature vectors (training set) drawn i.i.d. from $\mathcal{D}$.

Machine Learning Models. A machine learning model h_X trained on X, takes as input $\mathbf{x} \in \mathbb{D}^m$ and outputs a label $\mathbf{y} \in Y$. Let $L : Y \times Y \to \mathbb{R}$ denote a loss function. The training loss of h, denoted, $L_{\text{tr}}(h)$, determines how much h differs from c on all $\mathbf{x} \in X$. Similarly we define the test loss of h by $L_{\text{test}}(h)$, which is evaluated by computing $h(\mathbf{x})$ and $c(\mathbf{x})$ over the distribution $\mathcal{D}$. For instance, if Y is discrete, then L can be the 0-1 loss function, which evaluates to $L(h(\mathbf{x}), c(\mathbf{x})) = 0$, if $h(\mathbf{x}) = c(\mathbf{x})$, and 1 otherwise [35]. The generalization error of h is defined as

$$\text{err}(h) = L_{\text{tr}}(h) - L_{\text{test}}(h). \tag{5.1}$$

The exact form of the loss function L depends on the learning problem. More specifically, it depends on the nature of Y. If the learning problem is that of classification among k different classes, which is our focus, we have $|Y| = k$. The true label of a sample $\mathbf{x}$ is then a k-element vector $\mathbf{y} \in Y$ with 1 in the position corresponding to the true class, and 0 in all other places. A classifier h_X however, may output a vector $\mathbf{y}' \in Y$ such that each element $y_i \in [0, 1]$ and $\|\mathbf{y}'\|_1 = 1$.

Metrics. The notions of SMI and AAI, informally introduced in the introduction, are based on the ability to distinguish nearby vectors in the feature space. The notion of "nearness" is based on a distance metric on the feature space $\mathbb{D}^m$. The examples of metrics used in this chapter are Hamming distance d_H for binary datasets, i.e., over the domain $\mathbb{D}^m = \{0, 1\}^m$, and Manhattan distance d_M for normalized continuous datasets, i.e., over $\mathbb{D}^m = [-1, 1]^m$. In general, our results generalize to any *conserving* metric (See Appendix C.3). The following defines the distance of a non-member vector from the training dataset.

Definition 5.2.1 (Distance and Neighbors). *Let d be a (conserving) metric on $\mathbb{D}^m$. Let r be a positive real number and let $\mathbf{x} \in \mathbb{D}^m$. The set of r-neighbors of $\mathbf{x}$ is the r-ball centered at $\mathbf{x}$ defined as*

$$B_d(\mathbf{x}, r) = \{\mathbf{x}' \in \mathbb{D}^m \mid d(\mathbf{x}, \mathbf{x}') \leq r\}.$$

A member of $B_d(\mathbf{x}, r)$ is called an r-neighbor of $\mathbf{x}$. The distance of a vector $\mathbf{x} \in \mathbb{D}^m$ from a set $X \subseteq \mathbb{D}^m$ is defined as $\min_{\mathbf{x}' \in X} d(\mathbf{x}, \mathbf{x}')$. We call $\mathbf{x}'$ the nearest neighbor of $\mathbf{x}$ in X. □

For attribute inference, we define the notion of a vector with missing attributes as *portion*:

Definition 5.2.2 (Portions). *We introduce a special symbol $*$ called star, and define $\mathbb{D}^* = \mathbb{D} \cup \{*\}$. Let S be a subset of indexes from $[m]$, which we call the set of unknown features. We define the map $\phi_S : \mathbb{D}^m \to \mathbb{D}^{*m}$, which given as input a feature vector $\mathbf{x}$ outputs a vector $\mathbf{x}^*$, such that $x_i^* = *$ for each $i \in S$ and $x_i^* = x_i$ for all $i \notin S$. We call $\mathbf{x}^* = \phi_S(\mathbf{x})$ a* portion *of $\mathbf{x}$ under S, or simply a portion of $\mathbf{x}$ if reference to the set S is not relevant. The set of features that are* masked, *i.e., replaced by $*$, in $\phi_S(\mathbf{x})$ will be called the* unknown part *of $\mathbf{x}^*$.* □

Definition 5.2.3 (Siblings). *Define the set:*

$$\Phi_S(\mathbf{x}) = \{\mathbf{x}' \in \mathbb{D}^m \mid \phi_S(\mathbf{x}) = \phi_S(\mathbf{x}')\},$$

then $\Phi_S(\mathbf{x})$ is called the set of siblings of $\mathbf{x}$ under S, and any member of the set a sibling of $\mathbf{x}$ under S. Note that $\mathbf{x}$ is also a sibling of itself.

For attribute inference, the algorithm will be given a portion $\mathbf{x}^* = \phi_S(\mathbf{x})$, such that the feature corresponding to the set S will be missing (unknown). The set $\Phi_S(\mathbf{x})$ contains all vectors which could possibly have the portion $\mathbf{x}^*$, including the original vector $\mathbf{x}$. These are the possible *candidates* of the portion, and the algorithm would need to distinguish them from $\mathbf{x}$. In Appendix C.3, we show that given a vector $\mathbf{x}$, all of its possible portions with i unknown features are within a ball whose radius can be determined through i. This result is useful to show the link between attribute inference and strong membership inference, as we shall see later.

In some of our inference definitions, we would need to sample vectors in the vicinity of some feature vector $\mathbf{x}$. Depending on the distribution $\mathcal{D}$, it may well be the case that the vectors around $\mathbf{x}$ have a negligible probability of being sampled as feature vectors. Thus, the adversary may simply be able to infer non-membership by checking which vector is not likely to be sampled under $\mathcal{D}$ [35]. To overcome this technical issue, we assume that the distribution $\mathcal{D}$ is such that there is at least one vector within a small radius around $\mathbf{x}$ which is assigned a similar probability as $\mathbf{x}$. This is made precise by the following definitions.

Definition 5.2.4 (Induced Distribution)**.** *Let Z be a set of feature vectors. Define $Z_{\mathcal{D}} = supp(\mathcal{D}) \cap Z$. We say that a vector $\mathbf{z}$ is sampled from Z according to the distribution induced by $\mathcal{D}$ if the resulting random variable has probability mass function $\frac{p_{\mathbf{z}}}{\sum_{\mathbf{z}' \in Z_{\mathcal{D}}} p_{\mathbf{z}'}}$ or the probability density function $\frac{p_{\mathbf{z}}}{\int_{Z_{\mathcal{D}}} f_{\mathcal{D}}(\mathbf{z}')d\mathbf{z}'}$ in the continuous case.* □

Note that the probabilities are only defined if $Z_{\mathcal{D}}$ is non-empty. We shall always assume this to be the case.

Definition 5.2.5 (Indistinguishable Neighbor Assumption)**.** *Let $r > 0$, and let d be a metric. Let $\mathbf{x} \leftarrow \mathcal{D}$. Let $\mathbf{x}'$ be sampled from $B_d(\mathbf{x}, r)$ according to the distribution induced by $\mathcal{D}$. Let $\mathcal{A}$ be any algorithm (distinguisher) taking as input a feature vector $\mathbf{x}$ and a distribution $\mathcal{D}$, which outputs 1 if $\mathbf{x} \leftarrow \mathcal{D}$ and 0, otherwise. Let $b \leftarrow\$ \{0, 1\}$. Let $\mathcal{A}$ be given $\mathbf{x}$, if $b = 1$ and $\mathbf{x}'$, if $b = 0$. Then*

$$\Pr[\mathcal{A}(\mathbf{x}, \mathcal{D}) = 1] - \Pr[\mathcal{A}(\mathbf{x}', \mathcal{D}) = 1] \leq \epsilon(r). \tag{5.2}$$

We call $\epsilon(r)$, the r-neighbor distinguishability advantage, and assume it to be negligible for small r. □

The above assumption states around any vector $\mathbf{x}$, there are some vectors sampled according to the distribution induced by $\mathcal{D}$ that are indistinguishable from $\mathbf{x}$ under $\mathcal{D}$. Note that this does not apply to all neighbors of $\mathbf{x}$ (which may be out of distribution). Put in other words, it states that around any vector $\mathbf{x}$, there are neighborhood vectors which have similar probability of being sampled under $\mathcal{D}$. It is easy to see why this assumption should hold on datasets with continuous attributes, as minor changes in the attributes would hardly be off-distribution. We argue that this is also a plausible assumption for discrete datasets. For instance, consider the Purchase (shopping transactions) dataset [48], which records the items bought by customers; 1 if the corresponding item is purchased by the customer and 0, otherwise. Given any vector $\mathbf{x}$, a nearby vector where a few item purchases have been removed can barely be considered an anomaly. Further note that the ability to distinguish increases, the further we move from the original vector, since now there are other vectors likely to be sampled through the induced distribution which are starkly different from $\mathbf{x}$, i.e., at greater distance from $\mathbf{x}$. Hence, the advantage $\epsilon(r)$ is defined as a function of r. To experimentally validate our claim, we trained a generative adversarial network (GAN) on the Purchase dataset to see if it can distinguish between original and nearby vectors. The results shown in Appendix C.2.2 are in agreement with our assumption.

Decision Regions. Our final definition in this section is that of decision regions, i.e., regions in the feature space assigned to a given class. We shall show later that performance of membership inference is linked to the volume of decision regions. Let $k \geq 2$ be the number of classes.

Definition 5.2.6. *Given a classifier h_X, for each class $j \in [k]$, we define its* decision region *(DR) as*

$$\mathcal{R}_j = \{\mathbf{x} \in \mathbb{D}^m : h_X(\mathbf{x}) = j\} \tag{5.3}$$

This is analogous to the definition of acceptance region in [143]. Similar to [143], we sample a large number of feature vectors from $\mathbb{D}^m$ uniformly at random, and use the fraction of vectors labelled j by h_X to estimate the *fractional volume* of the decision region $\mathcal{R}_j$. Overloading notation, we shall use decision region to mean both the region and its fractional volume. A class is said to *dominate* another class if the DR of the former is larger than the DR of the latter. The class with the largest DR shall be called the *most dominant* class.

5.2.2 Formal Results: Relationship between Variants of Membership and Attribute Inference

Membership Inference. Our first definition is that of membership inference which is derived from the definition in [35].

Experiment 1 (Membership Inference (MI) [35])**.** Let $\mathcal{A}$ be the adversary, let $X \leftarrow \mathcal{D}^n$ be the input dataset.

1. Construct model h_X.
2. Sample $b \leftarrow\$ \{0, 1\}$.
3. If $b = 0$, sample $\mathbf{x} \leftarrow \mathcal{D}$.
4. Else if $b = 1$, sample $\mathbf{x} \leftarrow\$ X$.
5. $\mathcal{A}$ receives $\mathbf{x}$, $c(\mathbf{x})$ and oracle access to h_X.
6. $\mathcal{A}$ announces $b' \in \{0, 1\}$. If $b' = b$, output 1, else output 0.

Using the True Label. Note that in addition to the vector $\mathbf{x}$, its true label $c(\mathbf{x})$ is also given to the adversary. This then allows the adversary to compute the loss function $L(h_X(\mathbf{x}), c(\mathbf{x}))$ from the output of the model h_X. This is considered for instance in [35], the shadow model technique in [33] and the shadow model variants of membership inference attacks in [34]. However, note that the true label is not necessarily required as is demonstrated in one of the attacks in [34] which only uses the knowledge of the input sample and the prediction returned by h_X. In this case, the adversary simply ignores the true label $c(\mathbf{x})$. The same is true in all the other experiments (definitions) to follow.

Let $\text{Exp}_{\text{MI}}(\mathcal{A}, h, n, \mathcal{D})$ denote the output of the above experiment.

Definition 5.2.7 (Membership Inference Advantage)**.** *The membership inference advantage of $\mathcal{A}$ on the classifier h, i.e., $Adv_{MI}(\mathcal{A}, h, n, \mathcal{D})$, is defined as*

$$\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]$$
$$= \Pr[b' = 0 \mid b = 0] - \Pr[b' = 0 \mid b = 1]$$

It is the thesis of this chapter that an MI adversary with a significant advantage in distinguishing between members and non-members is due to the fact that non-members are at a significant distance away from member vectors. If on the other hand a non-member vector is close to a member vector, then the adversary may not be able to distinguish between the two. We therefore present another definition of membership inference, called strong membership inference (SMI) defined next. The definition challenges the adversary to distinguish between two neighboring feature vectors. The closeness of the two vectors is controlled by the parameter r in the definition. We show later why such a strong inference attacker is a better starting point for constructing an attribute inference attacker in the spirit of [35].

Experiment 2 (r-Strong Membership Inference (SMI))**.** Let $\mathcal{A}$ be the adversary, let $X \leftarrow \mathcal{D}^n$ be the input dataset, let d be a (conserving) metric, and let $r > 0$ be a real number.

1. Construct model h_X.
2. Sample $b \leftarrow\$ \{0, 1\}$.
3. Sample $\mathbf{x}_0 \leftarrow\$ X$.
4. If $b = 0$, sample $\mathbf{x}$ from $B_d(\mathbf{x}_0, r)$ according to the distribution induced by $\mathcal{D}$ (cf. Definition 5.2.4).
5. Else if $b = 1$, $\mathbf{x} = \mathbf{x}_0$.
6. $\mathcal{A}$ receives $\mathbf{x}$, $c(\mathbf{x})$ and oracle access to h_X.
7. $\mathcal{A}$ announces $b' \in \{0, 1\}$. If $b' = b$, output 1, else output 0.

Definition 5.2.8 (Strong Membership Inference Advantage)**.** *The SMI advantage of $\mathcal{A}$ on the classifier h, i.e., $\mathit{Adv}_{SMI}(\mathcal{A}, h, r, n, \mathcal{D})$, is defined as*

$$\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]$$

$$= \Pr[b' = 0 \mid b = 0] - \Pr[b' = 0 \mid b = 1]$$

Relationship between MI and SMI. SMI is the same as MI if r is large enough to encompass all feature vectors in the support of $\mathcal{D}$. Otherwise, the next theorem shows that the two definitions are not equivalent.

Theorem 5.2.1. *There exists a domain $\mathbb{D}^m$, a distribution $\mathcal{D}$ on the domain, an $r > 0$, a dataset $X \leftarrow \mathcal{D}^n$, a classifier h, and an algorithm $\mathcal{A}$ such that an MI adversary gains non-negligible advantage using $\mathcal{A}$ whereas an SMI adversary has 0 advantage using the same algorithm.*

Proof. See Appendix C.4. □

The proof of the above result essentially constructs a dataset such that the output of the classifier is constant around any vector $\mathbf{x}$ in the dataset. In a real-world dataset, this implies that we assume the output of the classifier to be nearly constant around any feature vector $\mathbf{x}$, thus making it hard for an SMI attack to distinguish non-members in the vicinity of members. We shall later show that this assumption holds for real-world datasets and classifiers.

Attribute Inference. We first start with the definition of attribute inference derived from [35].

Experiment 3 (Attribute Inference (AI) [35])**.** Let $\mathcal{A}$ be the adversary, let $X \leftarrow \mathcal{D}^n$ be the input dataset, and let S be a subset of $[m]$ with cardinality m' such that $1 \leq m' < m$.

1. Construct model h_X.
2. Sample $b \leftarrow\$ \{0, 1\}$.
3. If $b = 0$, sample $\mathbf{x} \leftarrow \mathcal{D}$.
4. Else if $b = 1$, sample $\mathbf{x} \leftarrow\$ X$.
5. Let $\mathbf{x}^* = \phi_S(\mathbf{x})$ be a portion of $\mathbf{x}$.
6. $\mathcal{A}$ receives $\mathbf{x}^*$, $c(\mathbf{x})$ and oracle access to h_X.

7. $\mathcal{A}$ announces $\mathbf{x}' \in \mathbb{D}^m$. If $\mathbf{x}' = \mathbf{x}$ output 1, else output 0.

Definition 5.2.9 (Attribute Inference Advantage)**.** *The AI advantage of $\mathcal{A}$ on the classifier h, i.e., $Adv_{AI}(\mathcal{A}, h_X, m', n, \mathcal{D})$, is defined as*

$$\begin{aligned} &\Pr[Exp_{AI}(\mathcal{A}, h_X, m', n, \mathcal{D}) = 1 \mid b = 1] \\ -\ &\Pr[Exp_{AI}(\mathcal{A}, h_X, m', n, \mathcal{D}) = 1 \mid b = 0]. \end{aligned}$$

The above definition mirrors the one from [35]. However, the attribute inference covered in [35] is more general; it considers arbitrary background knowledge about $\mathbf{x}$, and not necessarily a portion. The version that we consider is called the model inversion attack [35, 38]. We remark that the above definition is by no means the standard definition of AI. We refer the reader to Section 5.6 for a discussion on other definitions of AI proposed in literature.

Inferring through the Distribution vs the Model. Note that these definitions purposely define advantage as the difference between inferring through the distribution alone versus inferring via access to the model. For instance, one way to infer the missing features is to exploit statistical correlations between the observed features and the label. But notice that this can be done directly through knowledge of the distribution, irrespective of access to the model. The AI advantage will therefore be negligible for such a strategy. Hence, the definitions only define an AI attack as advantageous if it can infer more through the model as opposed to through statistical trends of the feature vectors. The same applies to approximate attribute inference to be defined shortly. See Section 5.6 for further discussion on this point. Correlations can indeed be a privacy issue if the distribution is not known to the attacker. But this definition is outside the scope of this thesis, where we consider the distribution to be known by the attack algorithm.

Relationship between AI and SMI. It is easy to see how an AI adversary can use an SMI adversary to infer attributes. Given a portion $\mathbf{x}^* = \phi_S(\mathbf{x})$, the AI adversary uses the size of S, i.e., m', to choose an r according to Corollary C.3.1.1, in Appendix C.3, and then runs the SMI adversary with input r and each possible *sibling* of the vector $\mathbf{x}$

(Even though the set S is not explicitly given to the AI adversary, it is implicit from the portion). Whenever, the SMI adversary outputs 1, i.e., predicts the corresponding vector to be a member, our AI adversary outputs that vector as its guess for $\mathbf{x}$. Thus SMI $\Rightarrow$ AI.

In the other direction, the following theorem shows that AI implies SMI, or in other words $\neg$SMI $\Rightarrow$ $\neg$AI. Therefore, if an SMI adversary has negligible advantage, then we cannot hope to find an AI adversary with significant advantage.

Theorem 5.2.2. *Let $\mathcal{A}$ be an AI adversary with advantage δ. Then there exists an SMI adversary $\mathcal{B}$ with advantage $\delta + \epsilon(r)$, assuming $\epsilon(r)$, the r-neighbor distinguishability advantage, is negligible for small r.*

Proof. Consider an SMI adversary $\mathcal{B}$ which is given $\mathbf{x}$. SMI chooses a random index, or alternatively, a random index set S of cardinality 1. The adversary $\mathcal{B}$ constructs $\mathbf{x}^* = \phi_S(\mathbf{x})$ and gives it to $\mathcal{A}$. Upon receiving $\mathbf{x}'$ from $\mathcal{A}$, the adversary $\mathcal{B}$ checks if $\mathbf{x}' = \mathbf{x}$. If yes, it returns 1. Else it returns 0. The advantage of adversary $\mathcal{B}$ is

$$\begin{aligned}&\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]\\&= \Pr[\mathrm{Exp}_{\mathrm{AI}}(\mathcal{A}, h_X, 1, n, \mathcal{D}) = 1 \mid b = 1]\\&- \Pr[\mathrm{Exp}^*_{\mathrm{AI}}(\mathcal{A}, h_X, 1, n, \mathcal{D}) = 1 \mid b = 0],\end{aligned} \tag{5.4}$$

where $\Pr[\mathrm{Exp}^*_{\mathrm{AI}}(\mathcal{A}, h_X, 1, n, \mathcal{D}) = 1 \mid b = 0]$ denotes the version of Experiment 4, where $\mathbf{x} \leftarrow \mathcal{D}$ in Step 3 is replaced with $\mathbf{x}_0 \leftarrow\$ X, \mathbf{x} \leftarrow B_d(\mathbf{x}_0, r)$, according to the distribution induced by $\mathcal{D}$. From Eq. 5.2 for any algorithm $\mathcal{C}$, we see that:

$$\begin{aligned}&\Pr[\mathrm{Exp}_{\mathrm{AI}}(\mathcal{A}, h_X, 1, n, \mathcal{D}) = 1 \mid b = 0]\\&- \Pr[\mathrm{Exp}^*_{\mathrm{AI}}(\mathcal{A}, h_X, 1, n, \mathcal{D}) = 1 \mid b = 0]\\&\leq \Pr[\mathcal{C}(\mathbf{x}, \mathcal{D}) = 1] - \Pr[\mathcal{C}(\mathbf{x}', \mathcal{D}) = 1] \leq \epsilon(r),\end{aligned}$$

where $\epsilon(r)$ is the r-neighbor distinguishability advantage. Thus, Eq. 5.4 becomes

$$\begin{aligned}&\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]\\&\leq \Pr[\mathrm{Exp}_{\mathrm{AI}}(\mathcal{A}, h_X, 1, n, \mathcal{D}) = 1 \mid b = 1]\\&- \Pr[\mathrm{Exp}_{\mathrm{AI}}(\mathcal{A}, h_X, 1, n, \mathcal{D}) = 1 \mid b = 0] + \epsilon(r)\\&= \delta + \epsilon(r).\end{aligned}$$

Under the indistinguishable neighbor assumption 5.2.5, we assume $\epsilon(r)$ to be negligible for small r. □

Theorem 2, together with the previous result, shows that SMI $\Leftrightarrow$ AI, provided the r-neighbor distinguishability assumption holds. If $\epsilon(r)$ is large, then the advantage does not translate, as now the neighbor vector (sampled from the induced distribution) does not follow the distribution $\mathcal{D}$ expected by the AI algorithm $\mathcal{A}$ in Experiment 4. This observation is mirrored by our experiments where we show that constructing an attacker that can *exactly* predict the missing values of a portion of a member vector with high probability is highly unlikely. Since this equivalence is under the r-neighbor distinguishability assumption, SMI is not identical to the notion of AI. This is true in particular for datasets where the assumption fails to hold. For instance, a location dataset with sparse locations. However, the assumption should hold for most real-world datasets, such as the ones considered in this chapter. We remark that in its raw form the definition may be overly strict for continuous attributes. To overcome this, in our experiments we apply binning, and flag any continuous attribute value as correctly identified if it falls in the correct bin (See Section 5.5.1 for the CIFAR dataset). Even with this judicious interpretation of the definition, our experimental results show that the adversary does not have much advantage in predicting the missing attributes. This leads to the definition of approximate AI, that requires the attacker to predict the missing values only "approximately close" to a member vector.

Experiment 4 (Approximate Attribute Inference (AAI))**.** Let $\mathcal{A}$ be the adversary, let $X \leftarrow \mathcal{D}^n$ be the input dataset, let S be a subset of $[m]$ with cardinality m' such that $1 \leq m' < m$, and let $\alpha \geq 0$ be a distance parameter.

1. Construct model h_X.
2. Sample $b \leftarrow\!\$\ \{0, 1\}$.
3. If $b = 0$, sample $\mathbf{x} \leftarrow \mathcal{D}$.
4. Else if $b = 1$, sample $\mathbf{x} \leftarrow\!\$\ X$.

5. Let $\mathbf{x}^* = \phi_S(\mathbf{x})$ be a portion of $\mathbf{x}$.

6. $\mathcal{A}$ receives $\mathbf{x}^*$ and oracle access to h_X.

7. $\mathcal{A}$ announces $\mathbf{x}' \in \mathbb{D}^m$. If $d(\mathbf{x}', \mathbf{x}) \leq \alpha$ output 1, else 0.

Definition 5.2.10 (Approx. Attribute Inference Advantage)**.** *The AAI advantage of $\mathcal{A}$ on the classifier h, i.e., $Adv_{AI}(\mathcal{A}, h_X, m', n, \alpha, \mathcal{D})$, is defined as*

$$\begin{aligned} &\Pr[Exp_{AI}(\mathcal{A}, h_X, m', n, \alpha, \mathcal{D}) = 1 \mid b = 1] \\ -&\Pr[Exp_{AI}(\mathcal{A}, h_X, m', n, \alpha, \mathcal{D}) = 1 \mid b = 0]. \end{aligned}$$

Note that with $\alpha = 0$, Experiment 3 becomes a special case of Experiment 4. It is easy to see that AI ⇒ AAI, but the converse is not necessarily true.

Depending on the distance metric, the AAI advantage definition can have different interpretations. For instance, if the distance metric is Euclidean distance, then this captures the notion of mean squared error. Similarly, the Manhattan distance metric gives the absolute error interpretation. The parameter α should be set carefully to avoid degenerate cases, e.g., if α is set too small, then an adversary whose guess is always slightly off α would be deemed less advantageous than an adversary with only one guess within α and the remaining deviating significantly from α. For our experiments, we set α as the distance of a random guess from the target vector.

Computing Advantages in Practice. As most prior work on membership inference uses the Area Under the Curve (AUC) of a Receiver Operating Characteristics (ROC) curve as a measure of aggregated classification performance of the MI attacker (viewed as a binary classifier), we use the same metric in our experiments in Section 5.3. In Appendix C.5, we show how our advantage definitions 5.2.7 and 5.2.8 are related to the AUC statistic. For the evaluation of AI and AAI attacks we employ the advantage metrics defined in Definitions 5.2.9 and 5.2.10.

5.3 Experimental Methodology

In this section, we describe the datasets, instances of MI and AI attacks used, and how we carry out membership and attribute inference attacks in our experiments in Sections 5.4 and 5.5. We first evaluate the performance of several MI attacks in terms of MI advantage (Definition 5.2.7) with increasing distance of the challenge vectors from the training set (Section 5.4). We then evaluate the performance of AI attacks in terms of AI advantage (Definition 5.2.9) which use MI attacks as a subroutine (Section 5.5.1). Finally, we study the performance of the same AI attacks in the sense of approximate attribute inference (Definition 5.2.10). These experiments demonstrate the shortcomings of MI and AI definitions and the need for our newly proposed definitions, i.e., SMI and AAI.

5.3.1 Data and Machine Learning Models

We evaluate MI and AI attacks on three different datasets: (a) *Location:* a social network locations check-in dataset obtained from Foursquare [144], (b) *Purchase:* a shopping transactions dataset [48], and (c) *CIFAR* an image dataset [47]. These datasets have previously been used to demonstrate MI [33,34,46] and AI attacks [46]. The first two datasets are binary, with 467 binary features in Location and 599 in Purchase, whereas the CIFAR dataset was processed, using principal component analysis (PCA), to yield 50 continuous features normalized between -1 and 1 [46]. We applied k-means clustering to obtain class labels in both the Location and Purchase datasets. The number of classes in the Location dataset is 30 and for the Purchase dataset, we create 5 variants differing in the number of classes (2, 10, 20, 50, 100), as is done in [34]. Finally, the CIFAR dataset contains 100 class labels for the images, with an additional set of 20 labels which are a superset of the 100 classes, e.g. the label “flowers” is the superset of orchids, poppies, roses, sunflowers, and tulips. We call the two datasets CIFAR-100 and CIFAR-20.

We predominantly explore the neural network as our target model. However, later in Section 5.4.2, we show that our observations generalize to Logistic Regression, Support

Vector Machine, and Random Forest classifiers. The exact configurations of these models for each experiment are detailed in Appendix C.1.

5.3.2 MI and AI Adversaries

We use five MI attacks from literature as examples of an MI adversary (Definition 5.2.7), and three AI attacks as examples of an AI adversary (Definition 5.2.9).

5.3.2.1 MI Attacks

Our MI attacks include three black-box attacks: the shadow model based attack from Shokri et al. [33], the attack from Yeom et al. based on prediction loss [35], and the attack from Salem et al. based on maximum prediction confidence [34], and two variants (local and global) of a white-box attack from Nasr et al. [1]. Recall that in an MI attack, the attacker is given a member or a non-member vector with optionally its true label, and is asked to infer membership.

Shadow MI [33]. This attack trains a machine learning model, called an attack model, to discern membership of a given vector from the prediction output vector (confidence of every class label). This attack model leverages outputs from shadow models which are trained with a disjoint dataset to mirror the behaviour of the target model.

Loss MI [35]. This attack eliminates the high computational cost of training shadow and attack models by evaluating the prediction loss of a vector on the target model directly. This attack, in practice, may use the target model training loss as a loss threshold to determine membership.

Conf MI [34]. Conf MI, short for Confidence, is even simpler than Loss MI; instead of computing the prediction loss, the attack simple uses the confidence value of the most likely label. With less information available to the attack, it performs worse than both

Loss MI and Shadow MI (as we shall see in Section 5.4). However, it is arguably a more practical attack, requiring less information.

Local White Box (WB) and Global White Box (WB) MI [1]. The three previous attacks are all black-box attacks with little to no information about the target model, and only API access to the model. An alternative form of MI attack is a white-box membership inference attack, which in a federated setting, may offer additional information for an adversary to launch an MI attack. Despite the federated setting, we suspect any observations we perform on the black-box setting should be reflected in a white-box setting. Nasr et al. attack [1] is a standalone attack targeting federated machine learning models in a white-box setting. The white-box setting lends additional hidden layer information and intermediate model states from the training process to better inform the attack model. This information includes the final layer gradients, outputs and the true label, obtained from intermediate and final states of the target model.

The federated setting consists of multiple parties, each training models independently and contributing parameters to a central server. The server aggregates these parameters before sending the results back to each party to replace their individual model. Two different attacks are tested: the *Global WB MI* attack, where the attacker has server level information and attacks each of the parties individually (in the case of a Malicious MLaaS provider); and the *Local WB MI* attack whereby the attacker is an external or contributing party attacking the server or MLaaS provider.

5.3.2.2 Attribute Inference (AI) Attacks

We use three AI attacks as examples of an AI adversary. All three attacks use an MI attack as a subroutine as mentioned in Section 5.2. We, therefore, use the same names for them as the underlying MI attacks. Briefly, our general procedure to evaluate an AI attack is as follows. Given a portion $\mathbf{x}^* = \phi_S(\mathbf{x})$ for a set S of unknown features (cf. Definition 5.2.2), we first construct all siblings of $\mathbf{x}$ (cf. Definition 5.2.3), by trying all possible permutations of the missing attribute(s), i.e., features. We then give each sibling as input to the MI attack. From the set of siblings, the vector with the highest membership confidence from the underlying MI attack is deemed the original vector $\mathbf{x}$, and thus its attributes identified as the missing attributes.

Shadow AI. The basis of this attack is to use the attack model from Shadow MI [33] for AI. While the MI version of the attack only uses the final decision (member or non-member), in the AI attack, we use the prediction confidence from the attack model to gauge which vector is most likely the original vector, and thus infer attributes.

Loss AI [35]. This attack follows the original proposal from Yeom et al. to use the training loss as the deciding factor for attribute inference. Given all siblings, the vector that achieves the prediction loss (from the target model) closest to the training loss, is flagged as the original vector.

Conf AI [145]. Recall that Conf MI [34] uses the single largest prediction confidence of the vector to deduce its membership. We repeat the same process, and flag the highest confidence vector (prediction confidence from the target model) from all siblings as the original vector.

Note. Although both Local WB and Global WB MI attacks can also be used to perform AI, we opted against, as they are computationally more demanding than other attacks. Fortunately, as we shall show, Local WB and Global WB MI attacks show similar trends as the other 3 MI attacks we use as subroutines for AI.

5.3.3 Attack Methodology

Prior to inference, we must first train a target model on a given dataset. To do so we split the dataset into training and testing sets. We describe the exact training/testing data split, the architecture of the neural network, and other hyper-parameters in Appendix C.1. These models have been tuned to replicate models observed in prior works. The training set is used to train the target model, and the prediction accuracy of the target model is evaluated on the testing set. We tune our target models to produce prediction accuracies comparable to [33] (exact attack accuracy values are reported in Table C.1 in Appendix C.1). From the training and testing sets we then sample 1000 vectors each to serve as our member and non-member sets. With the target model prepared, we take the following steps to launch MI and AI attacks.

MI. For MI, we obtain AUCs by evaluating the member and non-member subsets with either the MI attack model (for Shadow, Local WB and Global WB MI), or the target model (for Loss and Conf MI) for a membership confidence score.

AI. For AI, we take our set of member and non-members, and then use the top most informative features according to the Minimal Redundancy Maximal Relevance (mRMR) criterion [97]. Intuitively, the informative features are likely to have more influence on the classifier's output. This also follows previous work [35, 146] where it is shown that informative features, i.e., those with more influence, have a positive impact on attribute inference, albeit the results apply for Boolean and binary variables. Thus, the use of most informative features increases the likelihood of an AI attack. The set of most informative features forms the set S of unknown features. For each vector, we then create its portion based on S, and generate all siblings of the vector, only one of which is the original vector with the target attribute values. With this set of siblings, for each member and non-member vector, we perform an MI attack. This produces a measure of membership confidence (either as attack model probability, prediction loss, or prediction confidence, c.f. Section 5.3.2.2). From this measure, the sibling with the highest membership confidence is regarded as the correct vector, and consequently containing the correct missing attributes. For AI, we regard the attack as a success when the recovered sibling is exactly equal to the original vector (Experiment 3). For AAI, we regard the attack a success when the recovered sibling is within a given α distance away from the correct attributes (Experiment 4).

5.4 Membership Inference

We first show results from MI attacks highlighting the need for our definition of strong membership inference (SMI) (Experiment 2). Two key findings are:

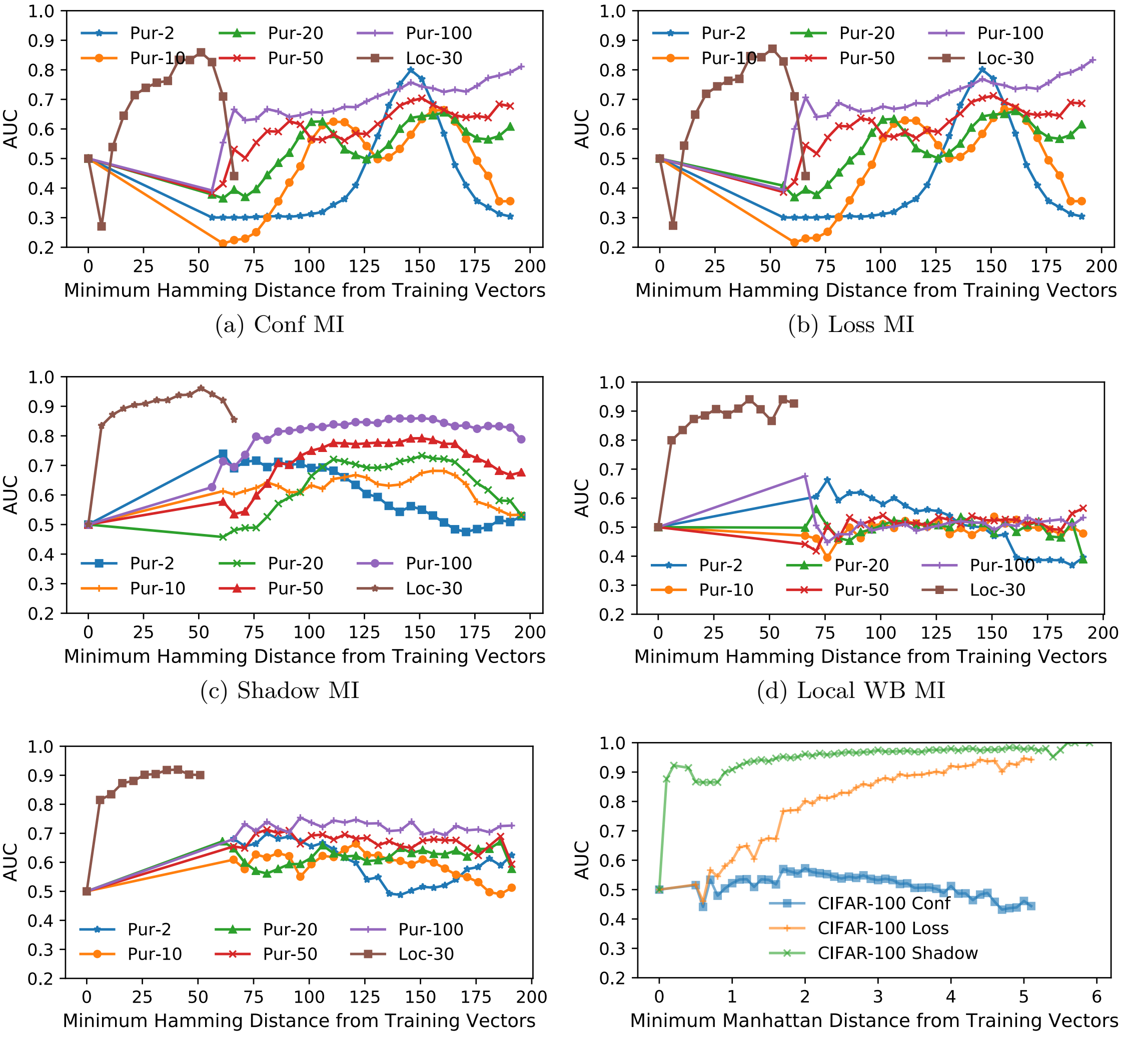
- MI attacks perform better if the non-members are at a greater distance from the training dataset. This observation is crucial for attribute inference, as we shall see in the next section.
- MI attack performance is not uniform across all classes in the dataset. In fact, it is inversely related to the dominance of the class, i.e., the decision region of the class (Definition 5.3).

5.4.1 MI Attacks on Neural Networks

We first inspect the performance of the five MI attacks (See Section 5.3.2.1) on members and non-member vectors from the original dataset as a function of their distance from the training dataset (Definition 5.2.1). We observe that the vectors in the original dataset are quite far away from each other, consequently lacking MI performance information at small distances. Thus we follow this analysis with MI performance on synthetically generated vectors, to illustrate a complete picture of MI performance as a function of distance from the training dataset (Section 5.4.1.2). We also explore the relationship between MI attack performance and the decision region of a class (Section 5.4.1.3).

5.4.1.1 MI Performance on the Original Dataset as a Function of Distance

After training the target model, we compute the distance of each non-member vector from the training set. Recall from Section 5.2, we use Hamming distance d_H for Location and Purchase datasets (which are binary), and Manhattan distance d_M for the continuous (normalized) CIFAR datasets. The vectors are then grouped according to their distance from the training dataset (the distance is 0 for members). We then calculate AUC for each distance by taking the membership score of each vector in this distance group as the negative class, and all member vectors as the positive class. This test is repeated 50 times (10 for the WB MI attacks due to computational resource limitations), and the AUC is computed on the aggregation of all confidence values (Figure 5.1).



(a) Conf MI

(b) Loss MI

(c) Shadow MI

(d) Local WB MI

(e) Global WB MI

(f) CIFAR-100

Figure 5.1: Increasing AUC of various MI attacks with increasing Hamming distance of original non-members from the training dataset on target models. Subplot (f) compares the difference in attack AUC between MI attacks on CIFAR-100 (CIFAR-20 can be found in Appendix C.2.1).

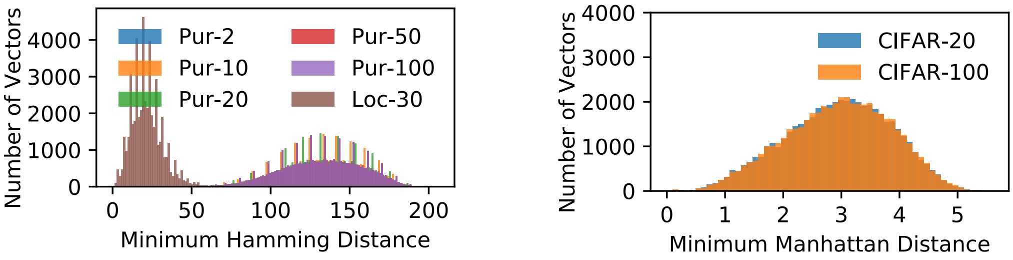
Results. From Figs. 5.1a to 5.1e, we observe that for the Location dataset the AUC improves as the distance of non-members from the training dataset increases in all five MI attacks, with the AUC being closer to random guess (0.5) for non-members closest to the training dataset. From the same figures, we can see that this trend is less obvious for the Purchase datasets. This is mainly because non-members in the Purchase datasets are at a greater distance from the training dataset. The same observation can be made for CIFAR-100 in Figure 5.1f (results for CIFAR-20 are in Appendix C.2.1). This gives a first indication that SMI (Experiment 2) is less successful than MI (Experiment 1).

An issue with the results in Figure 5.1 is that there is a lack of vectors close to and farthest away from the training datasets. This is evident from the distribution of distances displayed in Figure 5.2. Observe that there is little data available when we attempt to inspect AUC for distances close to the original dataset. As the non-members in the original Purchase datasets do not provide a full picture of how the MI performance behaves across all distances, and hence MI performance, in the next section, we generate synthetic vectors allowing us to control the distance (Hamming or Manhattan) from the training dataset providing a more complete picture.

A few other observations are worth highlighting:

- Consistent with what has been previously reported on MI attacks, the attack accuracy improves on target models with a greater number of classes [33, 34]. Higher number of classes is also linked to a higher degree of overfitness (Table C.1).

- The AUC performance of the Loss and Conf MI attacks is almost identical. Recall that Conf MI uses the maximum confidence value of the prediction, while Loss MI uses the prediction loss. Note that the prediction loss for a classification model is simply the loss between the confidence of the true label and 1. Given that a (good) target model is likely to predict the correct label of the vector, it follows that, most of the times, the maximum prediction confidence (as used in Conf MI) will be equal to the confidence used to compute the loss in Loss MI.



(a) Hamming distance (b) Manhattan distance

Figure 5.2: Histogram of distances of non-members from members in our training datasets. This data distribution is consistent across all attacks.

- Some of the AUCs exhibit peaks; an increase as the distance from the training dataset increases followed by a decrease. This is due to the decision regions (DR) learnt by the classifiers. We shall elaborate on this in Sections 5.4.1.2 and 5.4.1.3.

- Another peculiar observation is that some of the AUCs drop below 0.5, meaning that the strategy employed by the corresponding MI attack predicts flips and applies more to non-members than to members. The potential reason behind this is the same as the observation above which we shall explain in Section 5.4.1.3.

Observation 1. *In the MI attacks reported in literature, the distance of non-members from the training dataset is large. In general, an MI attack is more likely to accurately predict a non-member, the greater its distance from the training dataset.*

5.4.1.2 MI Performance on Synthetic Non-Members as a Function of Distance

Ideally, synthetic vectors should follow the original data distribution. Unfortunately, this would not yield vectors close to the training dataset as can be seen from Figure 5.2. To circumvent this, we take existing vectors and create synthetic vectors by flipping or perturbing some of the features. This creates synthetic vectors that are deliberately off-manifold, but still close to a training vector, where the majority of unaltered features still follow the original data distribution, while allowing us to control distance from the training dataset.

To generate synthetic vectors for the binary datasets (Location and Purchase), we (a) randomly select a member of the training set, (b) randomly select features to invert, (c) and vary the number of features and generate 5 non-members for each distance group, ranging from Hamming distance 1 to, 467 for Location, and 599 for Purchase. For CIFAR datasets, we define Manhattan distance groups at increments of 0.05 from the training dataset, starting from 0.05 to 5. We then produce non-members by randomly selecting features and adding additive perturbations to the feature values of the original vector. The process is repeated 5 times for each Manhattan distance group. The entire process is repeated for all selected 1000 member vectors for each dataset. The distance to the training dataset is recomputed for all non-members, to cater for the event that the nearest neighbor of a non-member in the training dataset has changed. The vectors thus generated are non-members, with the same label as the original member, unless, by chance, any of them collides with a member, in which case we discard it. We also ensure that the nearest neighbor in the dataset of the newly generated vector is of the same label as the base member vector, if not, this generated vector is discarded.

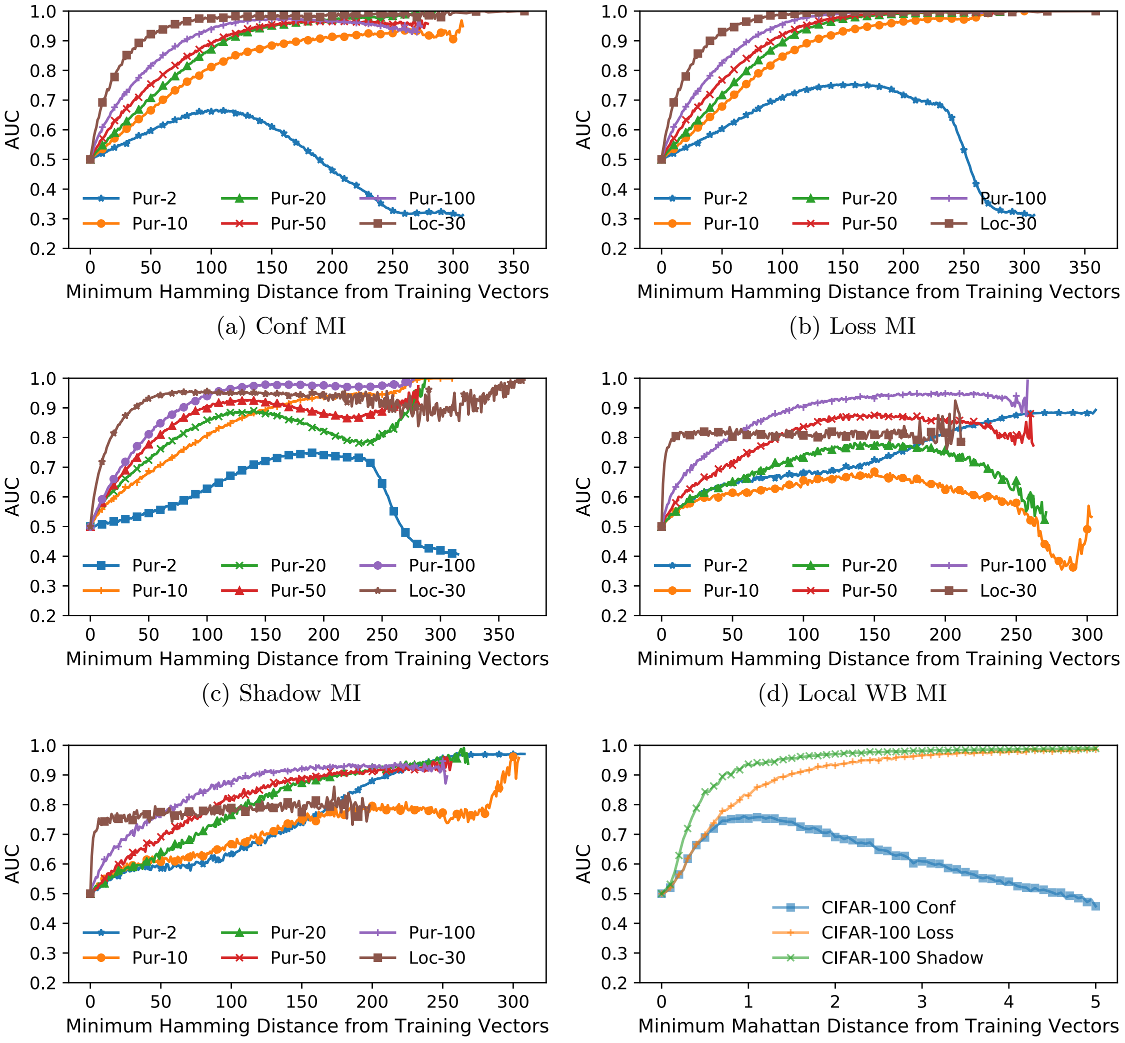
Results. The AUCs of the five MI attacks are displayed in Figure 5.3. For all five attacks, we observe that the AUC is close to 0.5 for vectors close to the training dataset, and starts improving as the distance from training dataset increases. It is also evident that the higher the number of classes, the steeper the improvement in AUC as the Hamming distance increases for the Location and Purchase datasets. This is more obvious through the magnified Figure 5.6, where we show AUC of the Conf MI attack on the Location, Purchase and CIFAR datasets at smaller distances from the datasets. The AUC is below 0.6 for Hamming distances of less than 5 and Manhattan distance of less than 0.2. This implies that the MI attack is not successful enough in the stronger sense, i.e., in the sense of SMI (Definition 5.2.8). This has implications for attribute inference, as we shall see in Section 5.5.

On datasets with higher number of classes, the AUCs of Loss MI (Figure 5.3b), Local WB (Figure 5.3d) and Global WB (Figure 5.3e) MI, show little change after a certain distance, even if the distance of non-members from the training dataset increases. On the other hand, on the Purchase datasets, for smaller number of classes (2, 10 and 20), Conf (Figure 5.3a), Loss (Figure 5.3b) and Shadow (Figure 5.3c) MI attacks observe an increase in AUC followed by a decrease. For the 10 and 20 class variants, we see a second incline in the AUC performance of Shadow MI around a Hamming distance of 250. The reason for this is that at certain distances a non-member vector $\mathbf{x}'$ with a class label j, might be in the decision region of another class, even when the nearest neighbor of $\mathbf{x}'$ in the dataset has the class label j. We elaborate this in the next section. Interestingly, in Figure 5.3f, the AUC curves of Conf and Loss MI diverge as the Manhattan distance from the training dataset grows greater than 0.7-0.8. This is because at larger Manhattan distance, the target model starts giving incorrect label predictions. The Loss MI attack detects this (as it computes loss with the predicted confidence). On the other hand, Conf MI only uses the highest confidence. It is therefore unable to detect this, showing worse performance. Finally, we note that a few of the AUC lines are ragged, especially at distances furthest away from the datasets. This is exhibited by attack model based MI attacks (Shadow, Local and Global WB). This is because the underlying attack models have less exposure to vectors at large distances as a result of the data distribution (c.f. Figure 5.2a, corresponding to distances where the AUC lines becomes ragged). The AUC curves of Loss and Conf MI are smooth throughout.

Observation 2. *The existing success of MI is a consequence of most non-member vectors being very different to members in terms of distance. For non-member vectors very close to members, the MI attacks perform similar to a random guess (0.5 AUC), and hence fail in the sense of SMI. Thus, the incumbent definition of MI does not capture the behavior of an MI adversary for non-members at distances close to the training data, i.e., SMI, which is essential for launching attribute inference attacks (Theorem 5.2.2).*

5.4.1.3 MI performance on Synthetic Non-Members as a Function of Class Label and Distance

The results thus far have been averaged over members and non-members from all classes. However, as we shall show, the performance of the MI attacks is not consistent over all



(a) Conf MI

(b) Loss MI

(c) Shadow MI

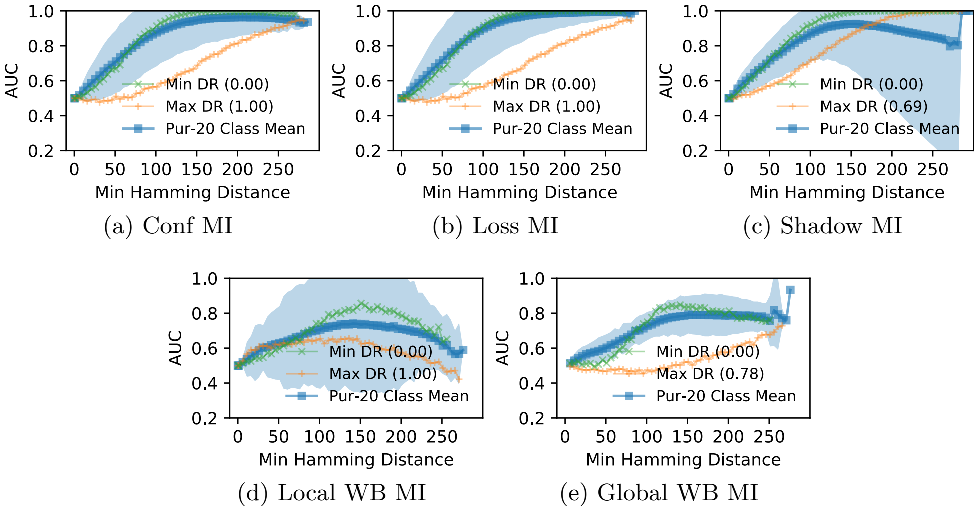
(d) Local WB MI

(e) Global WB MI

(f) CIFAR-100

Figure 5.3: Increasing AUC of various MI attacks with increasing Hamming distance of synthetic non-members from the training dataset on target models. (f) compares the difference in attack AUC between MI attacks on CIFAR-100 (CIFAR-20 can be found in Appendix C.2.1).

classes. In fact, the more dominant a class, i.e., the larger the decision region (DR) of the class (Definition 5.2.6), the less likely it is to be susceptible to membership inference. We empirically measure the decision region of a given class by sampling one million vectors from the feature space by sampling each feature uniformly at random within feature bounds (see feature bounds in Section 5.3.1). A similar approach had been adopted in [143]



(a) Conf MI

(b) Loss MI

(c) Shadow MI

(d) Local WB MI

(e) Global WB MI

Figure 5.4: Increasing AUC of various MI adversaries with increasing Hamming distance of synthetic non-members from the training dataset on target models, with a separation of class labels depending on the size of the Decision Region (DR), for the Purchase-20 dataset.

for binary classification (C.f. Chapter 4).

For per-class analysis, we train the target model and generate the synthetic vectors as before, except that now not only do we group synthetic vectors by the distance from the training dataset, but also according to the class label of the nearest training dataset vector. Due to space restrictions, we only show results for the Purchase-20 dataset. Results from the other datasets are in agreement with the conclusions drawn here, and are presented in Appendix C.2.1. In the figures, we highlight the AUC performance of the most dominant (largest DR) and least dominant (smallest DR) classes.

Results. Each plot in Figure 5.4 has 4 salient features. A blue line representing the mean AUC of all classes, an accompanying blue shaded area representing 2 standard deviations of AUC between classes, a green and blue line representing the class with the smallest DR, and the largest DR, respectively. From Figure 5.4, we observe that across all MI attacks, the AUC of the most dominant class is well below the average. In particular, at distances close to the dataset.

This can be explained as follows. Near the dataset, a non-member vector with class label j (which is also the label of its nearest neighbor in the dataset) is likely to lie in the decision region $\mathcal{R}_j$ of class j. As we move away from the dataset, by varying the distance, the corresponding non-member vectors shift further away from the spot in the decision region occupied by their nearest neighbors in the dataset. At certain distance, depending on the target or attack model, the decision region changes to a decision region occupied by a different class, even though the nearest neighbor still has the class label j. These non-members are then likely to be misclassified as member vectors of another class, since they lie deep in the decision region of another class. This phenomenon is particularly true if one class overwhelmingly dominates other classes, thus occupying the bulk of the decision region. In this case, the attack will not be able to distinguish between members and non-members from the dominating class.

This is most evident from the results on the 2-Purchase dataset (Figure C.2a-e in Appendix C.2.1), in which one of the two classes overwhelmingly dominates the other class (a DR of almost 1). The AUC performance of the dominant class is poor, whereas it is high for the other class, bringing the average AUC close to 0.5. This partly explains why the reported performance of MI attacks on 2-Purchase has always been comparatively poorer in the literature [33, 34]. The per-class analysis on the remaining binary datasets is in Appendix C.2.1.

Observation 3. *If a class overwhelmingly dominates other classes, i.e., occupies a significant portion of the decision region in the feature space, then it is least susceptible to MI and SMI. An MI or SMI attack is unable to efficiently distinguish between members and non-members from this class.*

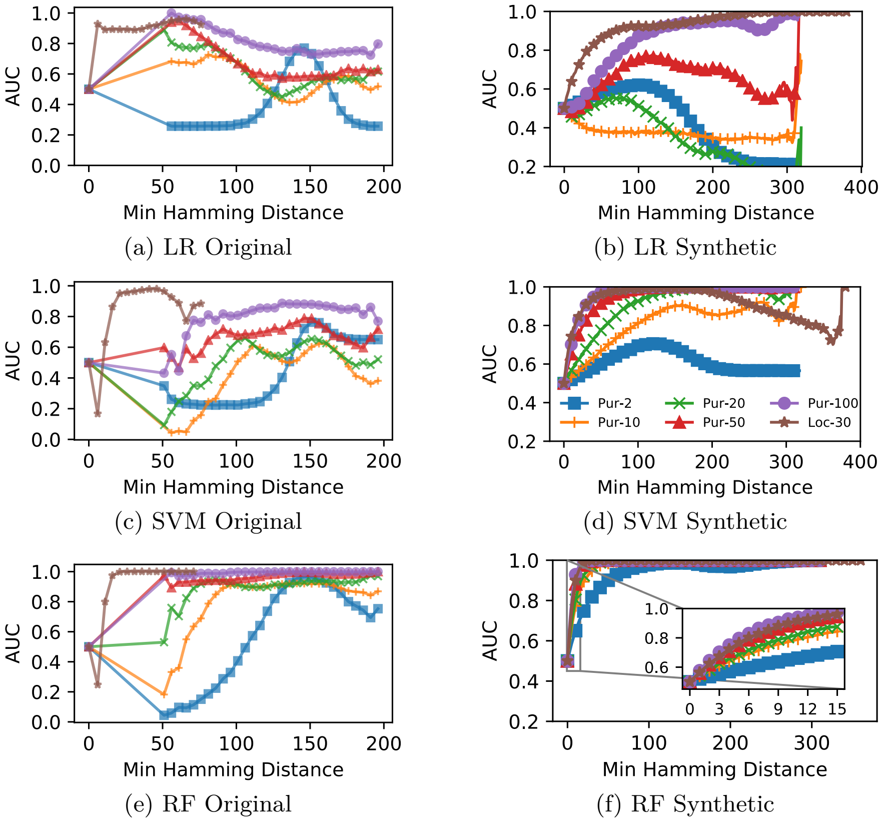
Tuning Attack Models for SMI. It may be argued that these MI attacks are not specifically trained to distinguish between members and nearby (synthetic) non-members, which may explain their poor performance in terms of SMI. We performed additional experiments where we tuned the training process of these attack models to further include nearby synthetic non-members. We observe even with tuning, the attack model is unable to achieve SMI. Details appear in Appendix C.2.4.

5.4.2 Generalization to Other Machine Learning Models

In this section, we demonstrate that the previous observations are not just limited to neural networks, and generalize to other machine learning models as well. More specifically, we use Logistic Regression (LR), Support Vector Machines (SVM) and Random Forests (RF) classifiers as the target classification models. Since our observations are consistent across all MI attacks, we only evaluate the Conf MI attack as it requires the least amount of information about the target model, making it the most portable attack between different machine learning target models.

Results. Figs. 5.5a, 5.5c, 5.5e display the AUCs on the original non-members from the datasets. We see that, in general, they exhibit the same as the neural network: the AUC improves as the distance of non-members from the dataset increases, with the AUC performance closer to 0.5 near the dataset. This trend in the AUCs is more prominent on the synthetic non-members shown in Figs 5.5b, 5.5d, 5.5f. An interesting observation is that the AUC of the RF model is very high even for non-member vectors close to the dataset, across all datasets. The main reason for this is that the RF model in general is more overfitted than the other models (see Table C.1 of Appendix C.1). This may seem to suggest that it is possible to launch a successful SMI attack on an RF-based target model. However, if we zoom into distances close to the training dataset, i.e., inset Figure 5.5f, we see that the AUC is close to 0.5 for Hamming distance ≤ 2. Thus, it is still difficult to launch an SMI attack for small distances.

Observation 4. *The observation that an MI attack is unable to distinguish between members and nearby non-members (strong membership inference) is consistent across different machine learning target models.*



(a) LR Original

(b) LR Synthetic

(c) SVM Original

(d) SVM Synthetic

(e) RF Original

(f) RF Synthetic

Figure 5.5: Increasing AUC of MI with increasing Hamming distance of original and synthetic non-members from the training dataset on target models with various ML algorithms. Inset (f): Zoomed in view of small hamming distances.

5.5 Attribute Inference

In this section, we first present the results of our experiments using the three attribute inference (AI) attacks described in Section 5.3.2.2. We show that all three AI attacks have negligible advantage in inferring the missing attributes of a target vector. On the other hand, for the same three attacks, we show that approximate attribute inference attack (AAI) advantage (Definition 5.2.10) is significant, thereby suggesting that these attacks can approximately guess the missing attributes with a probability better than a random guess. We only focus on neural networks as the target model, since we have already shown that the results generalize to other machine learning models. We also study the effect of overfitting on the success advantage of both AI and AIA attacks in the last subsection.

5.5.1 Attribute Inference Attacks

To perform AI experiments (Experiment 3), we train the model exactly as described in Section 5.3.3. We then (a) randomly select a member of the training set, or a non-member (from the testing set), (b) we mask a select number of most informative feature values as determined by mRMR [97] on the entire dataset to create the set S of unknown features (15 binary features for Location and Purchase; 5 continuous features for CIFAR datasets), (c) and generate all possible siblings of the vector under S (2 value bins per feature for Location and Purchase, and up to 10 value bins per feature for CIFAR). We then evaluate the AI attacks by giving each of the generated siblings to the underlying MI attack, and flagging those siblings that the corresponding MI attack identifies as a member vector. Again, the decision to use the most informative features from mRMR is to improve the likelihood of success for AI, as differences in the most informative features are likely to have the largest influence on the output of the classification model. We determine the AI attack to be successful, if the original member vector is in this set of *flagged siblings*. If there are more than one flagged sibling (excluding the original vector), we treat it as a tie and regard the attack as only partially successful. We add a fraction (determined by the number of ties) to its success count. For instance, 1/100 if there is a tie between 100 candidates. We then compute the AI advantage as the difference in the success counts between members, and non-members divided by the total counts of the tested members and non-members, respectively. We note that we also performed Experiment 3 on a single missing feature (as is done in other works [35,46]). The results are shown in Appendix. C.2.3. For this section, we focus on the expanded number of missing features, which is a more general case. The results for single feature AI, as we shall see, are only slightly better than multiple missing features.

Results. Across all attacks, we observe negligible AI advantages irrespective of the dataset and the attack (see Table 5.1). Moreover, the advantages are also very low for more

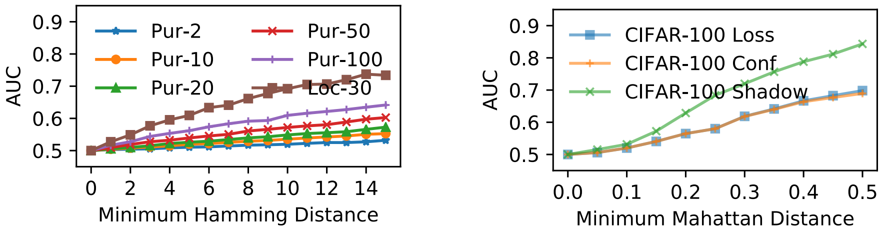
Table 5.1: Attribute Inference (Experiment 3) Advantage, where the adversary seeks to infer the exact attributes. The results below are normalized when dealing with ties.

AI	Loc-30	Pur-2	Pur-10	Pur-20	Pur-50	Pur-100	CIF-20	CIF-100
Conf	7.78E-4	1.38E-5	-3.69E-4	2.16E-4	2.00E-3	1.65E-3	-3.32E-7	4.14E-7
Loss	7.76E-4	-9.79E-5	5.57E-3	6.69E-3	4.59E-3	5.09E-3	3.33E-4	7.80E-4
Shadow	8.00E-4	-2.00E-4	2.17E-3	2.63E-3	4.10E-3	4.20E-3	2.26E-4	7.99E-4

overfitted target models (Location-30, Purchase-50, Purchase-100). This suggests that an AI attack is difficult to launch, even though the same target model and datasets are susceptible to MI attacks. Our conclusion runs counter to the results from Yeom et al. on the success of attribute inference [35], who demonstrate that on regression problems, a Loss AI attack can successfully infer attributes (using Loss MI attack as a subroutine), and the more overfit the target model, the more successful the attack. But this is easily reconciled by noting that our results apply to the classification problem, where the true label given to the attacker is discrete (class label). This is in contrast to the regression problem, where the true label (response) is a continuous value. The latter provides more information to the attack algorithm, which can be employed to launch a loss-based attack, i.e., Loss AI. The link to overfitting merits further exploration, and we defer this to Section 5.5.3.

A closer look at the Location dataset sheds more light on the reasons behind the failure of the AI attack. Previously, in Section 5.4.1.2, we observed that the performance of the Loss MI attack on the Location dataset reaches AUC greater than ≥ 0.7, significantly higher than other datasets. In Figure 5.6a we focus on the Loss MI attack on non-members at Hamming distances 1 to 15 from the dataset. We can see that the AUC reaches 0.7 at Hamming distance 10 but remains close to 0.5 between distance 1 to 3. Thus, while the Loss MI attack should easily be able to discard siblings of the original vector at Hamming distances greater than 10, it fails at closer distances and thereby resulting in an overall negligible advantage for the corresponding AI attack. The same reasoning applies to the CIFAR-100 dataset (Figure 5.6b), although under Manhattan distance.

Observation 5. *It is difficult to infer (exact) attributes of a target vector in the training dataset from a machine learning model trained for a classification task, even if it is* *susceptible to membership inference.*



(a) Loss MI - Location and Purchase, 15 hamming distance.

(b) CIFAR-100, zoomed to 0.5 Manhattan distance.

Figure 5.6: Closer inspection of Hamming and Manhattan distance for select datasets and MI attacks previously seen in Figure 5.3. Note at small distances from the training vectors, the AUC is close to 0.5, suggesting a poor AI attack.

5.5.2 Approximate Attribute Inference Attacks

Since an MI attack starts performing better as the distance of non-member vectors from the dataset increases, this suggests that the relaxed notion of approximate attribute inference (AAI) defined in Experiment 4 may be realizable in practice. Recall that an AAI adversary is given a portion $\mathbf{x}^*$ of a vector $\mathbf{x}$, and is asked to return a vector $\mathbf{x}'$ such that $d(\mathbf{x}, \mathbf{x}') \leq \alpha$, where the parameter α determines closeness to the exact attributes. In this section, we evaluate AAI attacks. These are essentially AI attacks, but the success is determined by the parameter α. To set an appropriate value of α, we need to take into account any algorithm that randomly guesses the missing features without even using the output of the classifier. Over all challenge vectors, the average distance of the guessed vectors from the target vectors will approach the expected distance of a vector $\mathbf{x}'$ from $\mathbf{x}$ whose missing features are randomly generated. We therefore set α equivalent to this expected distance. This means that any algorithm that successfully guesses more the missing features within an α distance of the target vector is non-trivial. Note that guessing missing features trivially due to correlations in the data distribution is already covered by the way our AAI definition is constructed, i.e., learning via the model versus via the distribution. Thus, for the Location and Purchase datasets, where we have 15 unknown features, we set $\alpha = 7.5$, and for the CIFAR dataset, with 5 unknown continuous features (normalized between -1 and 1), we set $\alpha = 3.33$, which is the average distance of a random guess from the original values (See Appendix C.5).

Table 5.2: Approximate AI Advantage (Definition 5.2.10), where the adversary seeks to infer approximate attributes (α = 7.5 for Location and Purchase, $\alpha = 3.33$ for CIFAR). Results with ties are normalized.

AAI	Loc-30	Pur-2	Pur-10	Pur-20	Pur-50	Pur-100	CIF-20	CIF-100
Conf	0.1609	0.0366	0.0516	0.0502	0.0958	0.1307	-0.0004	0.0016
Loss	0.1030	0.0125	0.0516	0.0541	0.0789	0.1012	0.0300	0.0325
Shadow	0.0554	0.0054	0.0067	0.0149	0.0766	0.0964	0.0339	0.0445

Results. Table 5.2 shows the AAI advantage (Definition 5.2.10) of the three AI attacks on all datasets. Overall, the AAI advantage is considerably higher than the AI advantage (from Table 5.1), reaching up to 0.1609 for the Loss AI attack on the Location dataset. However, the advantage obtained is still lower than the theoretical maximum of 1. Furthermore, the advantage is higher for more overfitted datasets, i.e., Location, Purchase-50, Purchase-100, and CIFAR-100. This indicates that increasingly the level of overfitting may improve the attack accuracy, which we shall explore in the next section. Interestingly, Shadow AI either performs worse or comparable to Conf AI and Loss AI, even though the latter attacks have less information available to them. The advantages seen in Table 5.2 exceed AI with one missing feature (See Appendix C.2.3), despite the increased inference difficulty, with more missing features.

Like Yeom et al. [35], our current evaluation, regards the measure of success as an adversary's ability to infer attributes with a single guess, reported as an average over multiple vectors; However, we acknowledge there are additional measures of success. For example *top-k*, whereby an attacker has the opportunity to submit their top k guesses.

Observation 6. *It is possible to infer attributes approximately close to their true values with a success rate significantly greater than random guess when the target model is susceptible to membership inference.*

5.5.3 AI, AAI and Relation to Overfitting

In both AI and AAI attacks, we observed greater advantage on more overfitted target models. To explore this further, we focus on the Purchase-100 dataset and the Shadow AI attack. We define the overfitting level of a model as the generalization error (GE) as defined in Eq. 5.1. To alter GE, and hence the degree of overfitting, we vary the amount of training data, while maintaining proportional splits between training and testing sets. As we increase the training data size from 20,000 (20K) to 200,000 (200K), the generalization error decreases from 0.368 down to 0.193 as shown in Table 5.3.

Table 5.3: Approximate AI (Experiment 4) Advantage, where the Shadow adversary seeks to infer approximate attributes ($\alpha = 7.5$) from various states of generalized Purchase-100 Models, trained with different amounts of data to simulate the effect of overfitting. The results below are normalized when dealing with ties.

Dataset Size	20K	40K	60K	80K	100K	150K	200K
Overfitting	0.368	0.301	0.271	0.251	0.237	0.211	0.193
Shadow AI	0.0024	0.0046	0.0021	0.0052	0.0040	0.0049	0.0033
Shadow AAI	0.118	0.098	0.096	0.078	0.066	0.046	0.026

Results. From the “Shadow AI” row of Table 5.3, we can see that increasing the overfitting level has little to no impact on the AI advantage (the Shadow AI result in Table 5.1 corresponds to a dataset size of 40K). Returning to the comparison with the findings of Yeom et al. on the effectiveness of AI on regression tasks in Section 5.5.1, our results indicate that for a classification problem, AI remains ineffective even if we increase the degree of overfit. On the other hand, there is a positive correlation between overfitting level and the AAI advantage, evident from the row labeled “Shadow AAI” in Table 5.3. As the overfitting level increases from 0.193 up to 0.368, the AAI advantage improves from 0.026 to 0.118.

Observation 7. *The more overfitted a target classification model, the more susceptible it is to approximate attribute inference. On the other hand, attribute inference remains hard even with increased overfitting levels.*

5.6 Related Work

The three black-box MI attacks evaluated in this Chapter were proposed by Shokri et al. [33], Salem et al. [34] and Yeom et al. [35]. All three works have used a split of a real dataset into training and testing sets, and demonstrated the effectiveness of MI using the testing sets. We have shown that most vectors in the testing set, i.e., non-members, are expected to be far from the training set, which explains why the relationship of MI performance to distance from members was not identified in these works. We have also shown that our results apply in the white-box setting, by evaluating the MI attacks from Nasr et al. [1], who proposed passive and active white box attacks targeting both standalone and federated models. Of course, the research on MI is not limited to these works. For instance, in [147] black and white box MI attacks are evaluated on generative adversarial networks; in [148] a new MI attack is proposed based on the loss-based MI attack from Yeom et al., and in [36] the authors show that even if MI attacks are ineffective as a whole on a dataset, they have disparate effectiveness on different sub-groups in the dataset. We have already demonstrated that our observations generalize to other MI attacks and models, since the underlying principle remains the same, i.e., ML models are less susceptible to strong membership inference in the classification setting.

The central theme of this Chapter is on the feasibility of attribute inference, also known as model inversion [37, 38, 149, 150]. A criticism of these works on model inversion is that they essentially exploit the correlation between the attributes and the true label, to infer the missing attributes [33]. Finding such correlations is the very purpose of the learning task, and therefore, the missing attributes would be learned regardless of whether the challenge vector is a member or a non-member [33]. The model inversion or attribute inference definition from Yeom et al. [35] avoids this issue by defining the AI advantage as the difference between inferring attributes with the model and without the model (i.e., through the distribution). Indeed, our definitions of AI and AAI use the same approach, based on their work. Yeom et al. [35] are also the first to formally relate MI attacks to AI attacks. They also formalize the role of overfitting to the effectiveness of MI and AI attacks, a link which was previously experimentally identified and demonstrated in [33, 34]. As mentioned previously, they demonstrate that AI attacks are feasible on regression problems, with the accuracy of the attacks improving with the level of overfit. Although the AI attack performance is not as significant as the MI attack, it is still quite substantial reaching an advantage of up to 0.5 on one of the datasets [35]. We have shown that for classification problems, only approximate attribute inference seems to be feasible. Apart from [35], Jayamaran and Evans [46] have also experimentally evaluated attribute inference attacks on classification models. Even though the goal of their analysis is to evaluate privacy leakage from classification models treated with differential privacy, their results with lower privacy (higher values of the privacy parameter ϵ [44]) can be considered as closer to the non-private setting. These results also show low AI advantages as compared to MI attacks, although the authors do not delve into the reasons.

Another related area is the investigation of factors effecting membership inference. Sablayrolles et al. [151] seek the optimal strategy for membership inference and find that such a strategy depends only on the loss function, implying that, asymptotically, knowledge of the model parameters (white box setting) does not provide any benefits over black box access. However, their treatment does not explore distance-based impact on membership inference as is done in our work. Long et al. [152] explore the performance of membership inference focused on training data records which are more vulnerable, in contrast to looking at membership inference performance as an aggregate over the entire training dataset. They find that records which have fewer neighbors are more vulnerable, as their presence or absence has more influence on the model's output. They also state that it is difficult for an MI attack to distinguish between a member and its non-member neighbors. Unlike [152], we formally prove the distinction between MI and SMI, and how this separation negatively impacts AI (and AAI) on classification models.

On the definitional side, Wu et al. [146] present an initial formal definition of attribute inference as the difference in inferring from the output of the model versus through the distribution (without access to the model). The definition from Yeom et al. [35], which is the basis of our related definition, follows the same line of thinking. In addition to membership and attribute inference, Melis et al. [153] also consider *property inference*, which is a property of a subset of training points within a class but not true of the entire class. They show that it is possible to infer properties that are independent of what characterizes the class through unintended learning by the machine learning algorithm. Unlike membership or attribute inference which is tied to individual data points, their property inference relates to multiple training points (subsets).

This is similar to other attacks on machine learning models, such as *model extraction* [82], which apply to the entire model itself and not necessarily to individuals in the training dataset. In a model extraction attack, unknown parameters of the model are retrieved to construct similarly behaving models (hence stealing the model in a proprietary sense). On the defense side, it has been demonstrated that MI and AI attacks can be mitigated by the use of *differential privacy* [44, 46, 154], although, this comes at a potential loss in utility [46, 150, 155]. Our findings on the infeasibility of AI attacks indicate that we may only need protection against (the weaker) approximate attribute inference, for which tailored differentially private learning algorithms can be constructed offering better utility. This is particularly useful for applications where membership inference is less of a concern, or may even be desirable. A case in point being machine learning auditors, based on membership inference attacks, to prevent unauthorized use of personal data [86, 87]. Additionally only evaluating defenses against AI may mask potential privacy leakage though AIA, an arguably simpler attack and thus a more difficult task to defend.

Finally Adversarial examples are vectors with applied perturbations close to the original target that result in large variations in the model's behavior, commonly observed as a mis-prediction [156]. In the setting of MI or AI, given an adversarial example of a vector within the training dataset, the large difference between the behavior of the known and adversarial example would allow for their distinction. However, as Long et al. [152] state, the majority of the neighborhood around the vector would have a minimal difference on the model output; with the adversarial example behaving as an exception, rather than the norm. Though combative methods have been developed to train models robust to adversarial examples [157], we speculate that robust adversarial models will only have a minor positive impact on the mitigation of the MI/AI attack, as robust models should preserve the regular behavior of the model, to only mitigate the behavior of the adversarial examples. Though this warrants further investigation.

5.7 Conclusion

Our results show that it is infeasible for an attacker to correctly infer missing attributes of a target individual whose data is used to train a machine learning model for a classification problem owing to the inability of membership inference attacks to distinguish between members and nearby non-members. For applications, where the privacy concern is attribute inference, and not membership inference, defense mechanisms tailored to protect against approximate attribute inference can be constructed. As a future direction, it will be interesting to explore whether the approximate attribute inference attacks mentioned in this chapter can be improved to infer missing attributes as close as possible to the original attributes.

Chapter 6

Not one but many Tradeoffs: Privacy Vs. Utility in Differentially Private Machine Learning

This chapter is an extended version from work titled "Not one but many Tradeoffs: Privacy Vs. Utility in Differentially Private Machine Learning", accepted in the 2020 ACM SIGSAC Conference on Cloud Computing Security (CCSW'20), completed in conjunction with Zhao, B.Z.H., Kaafar, M.A., and Kourtellis, N.

As seen in the Chapter 4, machine learning models are vulnerable to security attacks, from Chapter 5, models continue to be vulnerable to privacy attacks. However, this has not dissuaded data holders from seeking to protect their user's privacy, whilst still maximizing their ability to produce machine models with high quality predictions. In this chapter, we empirically evaluate various implementations of differential privacy (DP) and measure their ability to fend off real-world privacy attacks, in addition to measuring their core goal of providing accurate classifications. We establish an evaluation framework to ensure each of these implementations are fairly evaluated. Our selection of DP implementations adds DP noise at different positions within the framework, either at the point of data collection/release, during updates while training of the model, or after training by perturbing learned model parameters. We evaluate each implementation across a range of privacy budgets, and datasets, each implementation providing the same mathematical privacy guarantees. By measuring the models' resistance to real world attacks of membership and attribute inference, and their classification accuracy. we determine which implementations provide the most desirable tradeoff between privacy and utility. We found that the number of classes of a given dataset is unlikely to influence where the privacy and utility tradeoff occurs. Additionally, in the scenario that high privacy constraints are required, perturbing input training data does not trade off as much utility, as compared to noise added later in the ML process.

6.1 Introduction

Advanced machine learning (ML) techniques enable accurate data analytics for various application domains. This promoted the commercial deployment of ML as a service (offered by data giants, such as Google and Amazon) which allows data-driven businesses to train models on sensitive data while offering third party (paid) access to these models. Although commercially attractive, these services can be vulnerable to model theft and privacy infringements potentially not compliant with developing privacy regulations (e.g., EU and USA regulations such as COPPA [39] and GDPR [40], and most recently e-Privacy [41] and CCPA [42]). To preserve their models' privacy while still maximizing their ability to produce ML and deep learning (DL) models that have high utility for their services, data-driven organizations are turning towards leveraging privacy-preserving ML (PPML) techniques, building on theoretical frameworks of Differential Privacy [43,44] (*DP*) and/or Federated Learning [45] (FL). However, differentially private PPML methods often come with an intrinsic tradeoff between utility (e.g., as captured by the accuracy of the model) and the privacy guarantees offered by the technique applied to protect user data.

A recent initial investigation in [46] studies different *DP* compositions, and how these compositions can be applied to the training of a neural network or logistic regression model. [46] reports on the impact these privacy mechanisms have on the model's utility, and the effectiveness of inference attacks on the resulting models. Inspired by [46], and towards the goal of understanding the tradeoff between privacy and utility of *DP*-enabled ML methods, we dive deeper into this problem and, in this study, we set to assess how this inherent tradeoff depends on the (1) ML method used, (2) stage in the ML framework where the *DP* method is applied to protect the data or model, and (3) complexity of training data in use with respect to classes and attributes in the data.

We develop a comprehensive and systematic evaluation of a *DP*-enabled ML framework that enables a privacy ML researcher to study the Utility-Privacy tradeoff in depth for their data at hand. Our objective is to allow the selection of the best performing method yielding the highest predictive accuracy while still ensuring a solid level of privacy protection, by studying the different stages where *DP*-based noise can be applied: as an obfuscation to the input data, during model training, or at the model finalization by perturbing the learned model parameters. Equally important, the study's objective is to inform privacy ML researchers what privacy threshold to apply in their framework, and what are the privacy guarantees expected from the selected setup, vs. the utility of the chosen ML method.

We study various recent *DP* implementations of classical ML and DL methods such as Naive Bayes, Logistic Regression, Random Forests, and Neural Networks, and empirically measure their ability to fend off black-box privacy attacks that may be practically launched in the real-world, while also measuring the model's core goal of providing accurate classifications. Crucially, we establish this standard evaluation framework to ensure each of these *DP* implementations are evaluated fairly.

In particular, we study and test how ML performance and privacy are impacted when *DP* noise is added at different stages of the ML pipeline: Stage (1) by adding noise to the input data before the ML/DL training phase. Stage (2) where *DP* noise is added during model updates, i.e. while training the selected model. Stage (3) after the model training is performed, by perturbing learned model parameters. We evaluate each *DP*-enabled ML implementation across a range of privacy budgets, each instance providing the same mathematical privacy guarantees. We measure different metrics to capture the aforementioned tradeoff: privacy offered to the model and data (resistance to membership and attribute inference attacks) and model utility (classification accuracy).

We use both synthetic and real-world datasets to capture the aforementioned privacy and utility tradeoff. Our use of a synthetic dataset enables us to isolate the effects of *DP* noise, stages, and dataset complexity without the influence of data distributions. However, not to discount the importance of standard real-world datasets, we also perform our evaluation on a range of real data like CIFAR [47], Purchase [48], and the Netflix dataset [49] in which we provide the same pre-processing treatment as Purchase [48].

With our experimentation, we make the following observations. Most notably, for a given amount of model utility, applying *DP* noise at stages later than the input phase permits the addition of more *DP* noise, thus providing higher privacy guarantees. This observation is consistent across all *DP*-ML algorithms.

When considering utility and privacy as a function of the *DP* noise, we identify an "inflection point" for each function, an indicator of where the greatest change in utility and/or privacy will occur for a given *DP*-ML method. We find that this point on privacy function is more closely related to the Utility response, and the *DP*-ML method used, instead of *DP* privacy guarantees, as expected from the amount of *DP* noise applied to the process. Also, the data complexity of the dataset is unlikely to influence the inflection point of the utility or the privacy function. Finally, when privacy or utility comes with constraints, we provide recommendations for the best performing *DP*-ML method, and their expected utility and privacy guarantees.

We contribute our open sourced framework[1] for reproducibility purposes, as well as for other researchers to build on it and study privacy and utility thresholds of newly proposed *DP*-ML methods.

[1] Source code available at `https://github.com/PrivateUtility/PrivateUtility`

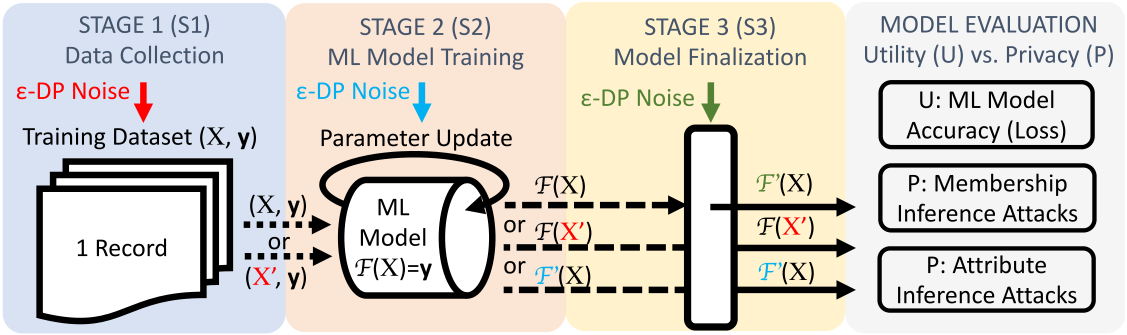


Figure 6.1: Our instantiation of the proposed methodology, with the three possible Stages that *DP* noise can be introduced in the ML pipeline to guarantee data privacy, and performance metrics used to assess privacy-utility tradeoff.

6.2 Methodology

6.2.1 Overview

In this Section, we provide details of the building blocks needed to study the privacy-utility tradeoff as a comprehensive and modular methodology. Our methodology encompasses the following:

- *DP* noise definitions (Section 6.2.2)
- Stages of the ML pipeline at which *DP* noise is added (Section 6.2.3)
- ML algorithms that are *DP*-enabled (Section 6.2.4)
- Privacy metrics, assessed with privacy attacks on data (Section 6.2.5)
- ML utility metrics (Section 6.2.6)

In this chapter, we provide an instantiation of this methodology (Figure 6.1) to evaluate the privacy-utility tradeoff in *DP*-enabled ML algorithms. Next, we cover details for each of these building blocks, and in Section 6.3, we provide details of their implementation.

Note that our methodology can be extended to account for other considerations in the privacy-utility tradeoff analysis. This could include Resource metrics (e.g. required computational resources for training ML models) or various datasets characteristics in use.

6.2.2 Differential Privacy

Differential privacy (DP) mathematically defines the protection offered in regards to the privacy of a single data vector, whether that is representative of an individual, or a single temporal event [43]. The ϵ-differential privacy is defined such that two neighboring sets of data D and D', differing by a single vector are indistinguishable up to a limit as described by a privacy budget ϵ. The output of a mechanism $\mathcal{M}$ applied on each dataset should also be indistinguishable from each other, up to our limit of ϵ. In other words:

$$Pr[\mathcal{M}(D) \in S] \leq Pr[\mathcal{M}(D') \in S] * e^{\epsilon} \tag{6.1}$$

Many differentially private ML algorithms support relaxations of the DP definition. There are two main relevant relaxations of ϵ-DP: (ϵ, δ)-DP [44], and (α, ϵ)-DP (Renyi-DP) [158]. Both relaxations provide eased requirements for DP while preserving properties such as composition and core privacy guarantees.

We will not be using these relaxations in this chapter, however, we note they are reducible to (ϵ)-DP, the focus definition in this thesis. In fact, (ϵ, δ)-DP [44] is equivalent to (ϵ)-DP, when $\delta = 0$, and (α, ϵ)-DP [158] is reduced to ϵ-DP when $\alpha = \inf$. Also, the authors in [159], given a set of assumptions, derive the upper bound of ϵ-differential privacy as p/ϵ, where p is the dataset dimensionality.

6.2.3 ML Pipeline Stages for *DP* Noise Injection

As noted by [46], there are three general positions in which DP noise can be applied to a ML task, to preserve the privacy of the data used, or the model built. These three positions of entry in the ML pipeline are visualized in Figure 6.1. To make the next observations more concrete, let the function $\mathcal{F}$ map the training dataset $\mathbf{X}$ to class labels

$\mathbf{y}$, that is, $\mathcal{F}(\mathbf{X}) = \mathbf{y}$. Then, the goal of the ML model is to approximately learn this relationship between dataset and labels as best as possible. Next, we discuss each of these three Stages:

Stage 1 (S1): *Before the learning process.* During the collection or release of data ($\mathbf{X}$), and before aggregation at the server, if local *DP* noise is applied on every data record, the data (**X'**) are protected before being used in a ML pipeline. Alternatively, when releasing a dataset to the public domain, the owner can train a data generator to create a synthetic dataset containing the same data semantics as the real data, but with the synthetic data governed by the rules of *DP*. Consequently, the $\mathcal{F}$(**X'**) model learned is *DP*-enabled.

Stage 2 (S2): *During the learning process.* In this stage, each step of the model update is restricted as to not excessively alter the model with the added *DP* noise ($\mathcal{F}'(\mathbf{X})$), and thus compromise the privacy of a given batch of records. The classic example for this Stage is the Tensorflow Privacy, which deploys a *DP* stochastic gradient descent algorithm [154].

Stage 3 (S3): *After the learning process.* After the data modeling has finished, the learned parameters of the model can be perturbed, by adding *DP* noise on them ($\mathcal{F}'(\mathbf{X})$) to remove dependencies between learned parameters and training data.

6.2.4 *DP*-based ML Algorithms

A literature review on existing ML methods that provide *DP* protection to the data or model revealed that various realizations can be loosely divided into the three Stages outlined above. We identified four key ML classification algorithms of interest: Naive Bayes (*NB*), Logistic Regression (*LR*), Random Forests (*RF*), and Neural Networks (*NN*). Next, we describe the approach used by each one in learning on data in a supervised setting, while applying *DP* noise in each Stage. We remark that [46] focused primarily on different *DP* compositions for *NN* and *LR*, both leveraging empirical risk minimization in the learning process, and loosely mapping to our S2 and S3 Stages, respectively. However, they did not offer direct comparisons of these ML algorithms across all Stages, as we do.

Table 6.1: *DP*-enabled ML methods used in each pipeline Stage.

	Stage where *DP* noise is applied		
ML Method	S1	S2	S3
Naive Bayes	X		X
Logistic Regression	X		X
Random Forests	X	X	
Neural Network	X	X	

In fact, we compare these and other *DP*-based ML methods, summarized in Table 6.1, as applicable in each Stage.

6.2.4.1 S1: Manipulation of Laplacian Noise

At this stage, we apply *DP* Laplacian noise [113] directly on the dataset, and thus, this process is independent of the ML algorithm used in subsequent steps of the framework. As a result of this independence, we can employ all four ML algorithms, in their non-private versions, on the modified, *DP* data. In particular, *DP* is provided by the addition of noise to every vector in the dataset. Laplacian noise is independently sampled for every feature value, of every data vector from the distribution $Lap(0, \beta_i)$, where $\beta_i = \frac{S_i}{\epsilon/p}$, and S_i is the value range of the ith feature [113] (Algorithm 4).

Algorithm 4: Direct addition of *DP* noise to dataset before ML.

```
Input: Training Dataset $X$, where $x := \{x_0, ..., x_i, ..., x_p\}$
Result: Differentially private Training Dataset $X'$
for $x$ in $X$ do
    $x' = x + b$; where $b := \{b_0, ..., b_i, ..., b_p\}; b_i \in Lap(0, \beta_i)$
Proceed with learning task $\mathcal{F}$ on $X'$.
```

Adding *DP* noise in S1 is ML independent and permits more flexibility, as any ML or DL method can be employed after S1 for training. The application of noise is dependent on data types and their complexity with respect to features and values allowed.

Remark 6.2.1. DP *noise is applied with the assumption that features are independent from each other, meaning a maximal amount of noise must be applied to each feature to* *ensure* DP*. With knowledge of feature dependence, hypothetically less noise can be applied to the dependent features as there is less uniquely identifying information between the dependent features.*

6.2.4.2 S2: *DP*-based Random Forests

Random Forests (*RF*) [160] are an ensemble of decision trees produced through bagging (bootstrap aggregation), whereby the training dataset is resampled from the training data to introduce dataset variability in the training of each tree. By applying feature randomization, a restriction in the number of features available to a tree when splitting a leaf node, the *RF* is able to produce an ensemble of varied trees each with its own decision, from which the final classification decision is made (for example majority vote). Differentially private random forests with Smooth Sensitivity [161] create conventional random forests, but instead of retaining the exact frequencies of data under each subsequent branch from the leaf, only the majority label is retained after the application of an exponential mechanism dependent on the privacy budget available (Algorithm 5):

Algorithm 5: *DP*-based *RF* with Smooth Sensitivity [161].

```
Input: Boosted and Bagged Training Dataset $X$
Result: Differentially private decision tree $T'$
Compute $T$, a regular tree from $X$;
for each node within the tree do
    retain Majority Label from datapoint frequencies through exponential
      mechanism [161];
Return $T'$ to the Forest;
```

Remark 6.2.2. *Computing the confidence of a prediction from the* RF *has been complicated due to the omission of distributions in the leaf nodes of trees. It is however possible to compute a confidence value from the majority voting aggregation of predictions of trees within the* RF*.*

6.2.4.3 S2: *DP*-based Neural Networks

Neural Networks (*NN*) are designed to mimic the functionality observed within brains [162]. They contain multiple layers of neurons (some hidden) that are activated depending on the activation of neurons in the previous layer. The influence of a previous layer's neurons on the current neuron varies depending on a weight or parameter value learned during the training phase. The very final layer is often a decision layer that corresponds to each of the classes present in the classification problem. The degree of activation of this last layer is analogous to the *confidence* of the class prediction.

The approach employed by Tensorflow-Privacy's [154] implementation of *DP*-enabled *NN* involves the use of a *DP* stochastic gradient descent (SGD) algorithm. The SGD algorithm seeks to find network parameters θ to learn function $\mathcal{F}$. The *DP*-based SGD first clips or limits the size of gradient update, to not be heavily impacted by one batch of data. Additional noise is added to the updated gradient depending on the values of ϵ, and batch sensitivity (Algorithm 6):

Algorithm 6: *DP*-based Stochastic Gradient Decent [154].

```
Input: Training Dataset $X$
Result: Differentially private parameters $\theta'$
$\theta' \leftarrow$ RAND, initialize the parameters randomly;
for batch $t \in T$ do
    compute gradient $\Delta\theta$, clip gradients $\Delta\theta$, add DP-noise $b$
    $\theta' = \theta' + \Delta\theta + b$
Complete $\mathcal{F}'$ learning task with $\theta'$.
```

6.2.4.4 S3: *DP*-based Naive Bayes

The Naive Bayes (*NB*) [163] classification algorithm learns probabilistic distributions of the output classes informed by the input feature values. The algorithm is considered "naive", as it assumes independence between features. The distributions are learned directly from the training dataset. The simple formulation of the model enables the Naive

Bayes classifiers to both be trained, and to make predictions relatively quickly.

IBM *NB* [164] implements an (ϵ)-*DP NB*, originally by [165]. The approach adds noise to the learned distributions that relate the input feature to the output decision. Algorithm 7 shows the Laplacian noise addition to the mean and standard deviation (μ, σ) computed from training dataset X. A more complete algorithm for handling both categorical and continuous data can be found in [165].

Algorithm 7: *DP*-based Naive Bayes provided by IBM [164].

```
Input: Training Dataset $X$
Result: Differentially private model distributions $\theta'$
Compute ($\mu$, $\sigma$) from $X$;
for $i$ in $features$ do
    Compute scaling factor $S_{(\mu,i)}$ and $S_{(\sigma,i)}$ from feature mean $\mu_i$, feature STD $\sigma_i$,
      and $\epsilon$;
    $\mu'_i = \mu_i + b_i$; where $b_i \in Lap(0, S_{(\mu,i)})$;
    $\sigma'_i = \sigma_i + b_i$; where $b_i \in Lap(0, S_{(\sigma,i)})$;
Compute output priors $P(y|x)$ from ($\mu'$, $\sigma'$);
```

6.2.4.5 S3: *DP*-based Logistic Regression

The Logistic Regression (*LR*) [166] algorithm employs the logistic function (sigmoid function) to model the probability of a binary outcome (binary classification) with the input feature vector. In the multi-class setting, every output class has an independent logistic function predicting the binary objective of the specific class (or not). The final classification is an aggregation of these independent functions. *DP*-based *LR* [167] works by adding Laplacian noise proportional to the Sensitivity of the feature and ϵ to the objective function of the model (loss function). After the noise addition, $\mathcal{F}'$ can be evaluated (Algorithm 8):

Remark 6.2.3. *We regard the minimization of the objective function on the noisy objective function of* LR *as Stage 3, as the* DP *noise is applied with access to the entire set of data,*

Algorithm 8: *DP*-based Logistic Regression by IBM [164].

```
Input: Training Dataset $X$
Result: Differentially private objective function $\mathcal{J}'$
Compute $(\mu, \sigma)$ from $X$; $\mathcal{J}' = \mathcal{J} + \mathbf{b}$;
where $\mathbf{b} \in Lap(0, \beta)$; specific details on $\beta$ available in [164]
```

and the noise is not dependent on any partial split of the data.

6.2.5 Privacy Attacks & Privacy Metrics

Traditionally, privacy has been measured with theoretical metrics such as information leakage [168, 169] and mutual information [170]. However, recent privacy attacks such as membership inference (MI) [33–35] and attribute inference (AI) [35, 145] have been introduced [46] as alternatives to measure the privacy risk of ML models.

In this work, we quantify the privacy offered by the implementation of *DP*, through the effectiveness of these two well-known privacy attacks (MI and AI). The threat model adopted by these attacks falls under the category of black-box attacks, with an adversary only having access to the input and output of the ML model. In fact, for the current generation of MI attacks [33–35], only one query is required for the vector in question (disregarding queries needed to train an attack model), whereas AI attacks need multiple queries, one for any possible value in the unknown attribute.

6.2.5.1 Membership Inference Attack

MI attack [33–35] defines an attacker that tries to determine if a specific data record has been included within the training data of a given ML model, or not. The attack objective is related to the definition of *DP*, as according to *DP*, two datasets with or without an ϵ proportion of records should be indistinguishable from each other. Of course, this is problematic if a privacy ML practitioner is seeking to maintain the confidentiality of their training data or to adhere to privacy regulations governing the data used in training. In literature, there are three realizations of the MI attack [33–35]. In this work, we focus on attacks described by Salem et al. [34] (*ConfMI*) and Yeom et al. [35] (*LossMI*).

ConfMI [**34**] attack works on the premise that a ML model is more confident about a prediction on an input vector it has previously encountered (in the training set), than an input vector it has not previously encountered (in the testing set). Thus, a vector with higher prediction confidence on any class label is more likely to be a member vector. A threshold can be found from a similarly distributed dataset to make a final distinction if an input is a member or non-member. Indeed, this attacker does not know the vector's classification truth, and the prediction confidence is a single value of the most probable class, irrespective of if it is the correct prediction.

LossMI [**35**] attack is similar to *ConfMI*. However, they use prediction loss, requiring the true label of the input vector. Additionally, instead of finding a threshold from a similar data distribution, the model training loss is assumed known and used as the threshold. The additional information needed makes the *LossMI* attack more difficult to perform than *ConfMI*, but more effective.

Remark 6.2.4. *If a classifier does not directly return a value of training loss from the training process, we compute the training loss as the average loss across the training dataset.*

6.2.5.2 Attribute Inference Attack

AI attack is an extension of the MI attack, however, instead of only determining if a record is included within the training set, the adversary seeks to recover the exact value of a missing attribute that could be masked due to its sensitivity (e.g., the diagnosis for the type of cancer of an individual). In particular, if a record vector has a dimensionality of n (i.e., n features), the adversary is assumed to have $n-1$ true features of the original record. Their objective is to infer the n^{th} feature's sensitive value. In general, AI attacks are more difficult to mount than MI attacks due to the requirements of the attacker.

The first method of AI (***LossAI***) follows work by [35] and [46] in evaluating every binned permutation of a vector and its unknown attribute, and selecting the value that produces a loss closest to the model's training loss. The second attack (***ConfAI***) follows work by [145] and [34], by selecting the vector permutation that produces the highest model confidence as the most likely real attribute.

To date, many implementations of the AI attack (e.g., as in [46]) bin numerical features for a binary evaluation. In this work, we go beyond the state-of-art and increase the number of allowable (binned) values in the inference of a vector's attribute, from two bins up to a maximum of 10 bins, depending on the unique values of an attribute. For instance, if an attribute is binary, two bins are required. A numerical feature with 6 distinct values will require 6 bins, and a continuous feature will be binned into 10 value bins.

6.2.5.3 Measuring Privacy Leaks: Adversary Advantage

The *adversary advantage* can be described as the improvement of a privacy attack observed on a set of input vectors that were included in the training set, as opposed to not being included in the training set. The rate at which the privacy attack succeeds on the positive class (member vectors) is the True Positive Rate (TPR), while the rate at which privacy attack is incorrectly predicted on the negative class (non-members) is the False Positive Rate (FPR). As such, the advantage can be formulated as $ADV = TPR - FPR$. A rigorous definition of the advantage is provided in [35]. It is clear novel attacks, and their advantage can be added to our framework. Here, we measure the impact of the four aforementioned attacks.

6.2.6 ML Utility Metrics

The objective of ML is to learn trends from a training dataset, and then predict the label of a previously unseen input instance. To evaluate the effectiveness of a trained model, predictions are made on a holdout set (not used in training), said predictions are then compared to known true labels. The proportion of the holdout set that is correctly re-predicted as the true labels represents the accuracy (ACC) of the trained model: $ACC = n_{correct}/n_{holdout}$.

Accuracy is a simple measure of ML prediction performance. Other commonly used metrics are AUC, Precision, Recall, or F-Score. Also, new metrics such as model fairness [171] and minimization of computational processes [172, 173] can be important in a privacy-utility tradeoff. All such utility metrics can be added to our framework.

We focus on **Accuracy Loss ($\boldsymbol{ACL}$)**, defined as the ratio of performance lost when DP is applied to the ML process (m), in comparison to an equivalent ML model trained with no DP applied (i.e., $\epsilon = \inf$):

$$\texttt{Accuracy Loss (}ACL\texttt{)} = 1 - \frac{ACC_{(m,\epsilon)}}{ACC_{(m,\epsilon=\inf)}} \tag{6.2}$$

6.3 Experimental Investigation

In this section, we detail how the methodology introduced earlier is instantiated[2] to experimentally investigate the tradeoff between ML model performance with respect to prediction, vs. privacy guarantees provided to data used to train said model. In particular, with our experimentation, we are interested in answering questions of:

1. What is the inflection point in the tradeoff between ML model accuracy and privacy leak? Is this inflection point consistent across various types of privacy attacks?
2. Does the stage of the DP-enabled ML framework in which the DP noise is applied impact this inflection point?
3. Is there a ML method that outperforms others at both prediction and privacy guarantees, consistently across datasets?

[2] Our code and data is provided at `https://github.com/PrivateUtility/PrivateUtility`

We seek to empirically identify important parameters that affect the manifestation of this privacy-utility tradeoff. To this end, in Section 6.3.1, we detail the experimental procedures that vary the *DP* noise amount (ϵ), where it is applied in the framework (Stages), different *DP*-ML algorithms implemented, and metrics used. Then, we describe the training datasets used, both synthetic and real (Section 6.3.2), providing details on the number of classes and type of attributes (continuous, binary). In the next Section 6.4, we present our experimental results and extract key takeaway messages.

6.3.1 Experimental Framework

First, we detail implementations of *DP*-ML methods used, as well as metrics to assess ML performance and privacy when *DP* noise is applied. We note that Section 6.2 already provided details for the privacy attacks and ML methods used. Then, we outline the common steps shared between all evaluations of the *DP*-ML methods. We bootstrapped our framework implementation from [174], but make the following crucial extensions:

- Accommodate the new ML algorithms to run in this framework,
- Adapt code to improve framework resource consumption,
- Add implementation of MI attack proposed by Salem et al. [34],
- Add implementation of AI attack proposed by Zhao et al. [145] (c.f. Chapter 5),
- Adapt AI attack of Yeom et al. [35] to support multiple bin values instead of only binary,
- Add synthetic data generation for tradeoff & benchmark studies.

6.3.1.1 Machine & Deep Learning Methods

We used implementations of ML algorithms explained in Section 6.2.4 readily available online. *Tensorflow-Privacy* [154] has code in [175]. DP *Random Forests* [161] has code in [176]. We emphasize the modifications done to *DP RF*, as it did not adhere to standard function calls of sk-learn [177] APIs. *IBM Naive Bayes* and *IBM Logistic Regression* [164] have code in [178].

6.3.1.2 Performance Metrics & Privacy Budget

In our experiments with the various ML methods and datasets, we measure different performance metrics. For prediction performance of a trained model, we measure Accuracy Loss (*ACL*) (See Section 6.2.6), We perform four MI and AI attacks (See Section 6.2.5), to quantify privacy leaks. Finally, in order to vary the amount of *DP* noise applied in each framework Stage and in each ML method, we use different values for the privacy budget $\epsilon = \{0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000\}$.

6.3.1.3 Experimental Steps

To perform the evaluation for: 1) a given dataset, on 2) a *DP*-based ML method, with 3) a privacy budget ϵ, we first sample from the dataset two sets of 10,000 samples each, forming our training and testing sets. Then, we train the ML model with the training set. In the case of S1 *DP* noise, we apply noise to the training set prior to the model training. Model training configurations can be found Appendix D.1.

Each model's prediction accuracy is obtained on the unseen testing set. With a trained model, the *ConfMI*, *LossMI*, *ConfAI*, and *LossAI* attacks are performed. In MI attacks, the training set constitutes the membership set, whilst the testing set is the non-member test set. In AI attacks, we consider up to 10 unique values for the unknown protected attribute (whilst accounting for continuous features). The attack is repeated on 20 different attributes, randomly selected to be the protected attribute. Then, the entire training and attack process is repeated 5 (10) times for synthetic (real) data, with training and testing sets sampled anew, to reduce the impact of biases arising from the data or *DP* noise.

6.3.2 Experimental Datasets

6.3.2.1 Synthetic Data

We generated data by uniformly sampling 100k vectors from a normalized feature space of 50 features. From these 100k vectors, we apply k-means clustering onto the dataset to artificially create labels of 2, 5, 10, 20, 50, 100, and 200 classes. This results in 7 different datasets of a varying number of classes, however, they all contain the same vectors originally sampled.

6.3.2.2 Real-World Data

We used three real datasets to study the tradeoff in our *DP*-enabled framework (summary in Table 6.2):

Table 6.2: Summary of datasets used in our experimental investigation, with respect to the number of instances available, classes provided (or constructed), and attributes available.

Dataset	Instances	Classes	Attributes
Synthetic	100,000	2, 5, 10, 20, 50, 100, 200	50
CIFAR [47]	50,000	20, 100	50
Purchase [48]	200,000	2, 10, 20, 50, 100	599
Netflix [49]	100,000	2, 10, 20, 50, 100	1000

CIFAR-100 [47]: The *CIFAR* dataset consists of 50k tiny images of various objects, that can be labeled according to 100 types. They can also be re-classified under 20 type super-classes. This dataset has been pre-processed with principal component analysis as in [46], to extract 50 key features to represent each of the images.

Purchase [48]: The *Purchase* dataset contains 200k user records of item purchases made from a set of 599 products. The values are binary, indicating if users had or not bought one of the 599 items. We perform a similar pre-processing step as in [33], by encoding a single user's transaction history as a binary vector, followed by the k-means clustering of users into purchaser groups. We consider label complexities of $k = \{2, 10, 20, 50, 100\}$.

Netflix Prize [49]**:** The *Netflix* dataset was first released in 2006 and contains ratings (from 1 to 5) by viewers on the Netflix platform for movies they watched. This dataset was also used in [35]. However, insufficient pre-processing details were provided for us to replicate their exact dataset. Therefore, we performed the following steps: (1) Sample the user ratings of the top 1000 rated (based on the number of ratings, not rating score) movies within the dataset. (2) Every user has their ratings assembled into a feature vector, with unrated movies filled in with a zero value. (3) If a user has not rated any of the 1000 most popular movies, the user is excluded from the dataset. (4) Then, we apply k-means clustering (as in *Purchase*) to obtain viewer groupings of $k = \{2, 10, 20, 50, 100\}$.

6.4 Experimental Results

In this section, we present our results for different experiments using our evaluation framework, to answer the questions posed in Section 6.3. We first analyze results with synthetic data, while controlling class complexity, and extract generalized patterns related to the privacy-utility tradeoff. We shall compare these patterns with results on real data to assess how the tradeoff manifests on real-world datasets.

6.4.1 Privacy-Utility Tradeoff on Synthetic Data

We perform experiments on controlled, synthetic datasets to discover generalizable properties that can be drawn regarding the privacy-utility tradeoff. The synthetic dataset allows us to remove the effect of data-specific biases (in a controlled manner), that may otherwise be present in the real data.

6.4.1.1 ML accuracy vs. *DP* noise

In Figure 6.2, we analyze *ACL* and its inflection point for the different ML algorithms, while varying class complexity, amount of *DP* noise, and the stage at which it is applied. When applying large amounts of *DP* noise (i.e., small ϵ) at the input stage (Stage 1), we observe that the *ACL* is equivalent to a random guess irrespective of the ML algorithm in use. It is not until ϵ=10 ($\sim$100 for *NN*) that the ML algorithm is capable of outperforming a random guess.

In Stage 2, *RF* struggles to learn the task even at high values of ϵ (the *ACL* never reaches 0). However, both *RF* and *NN* exhibit a notable inflection point at ϵ=1 and 10, respectively.

Finally, at Stage 3, the inflection point for *NB* occurs at ϵ=0.01, while the first inflection point for *LR* occurs at ϵ=0.1.

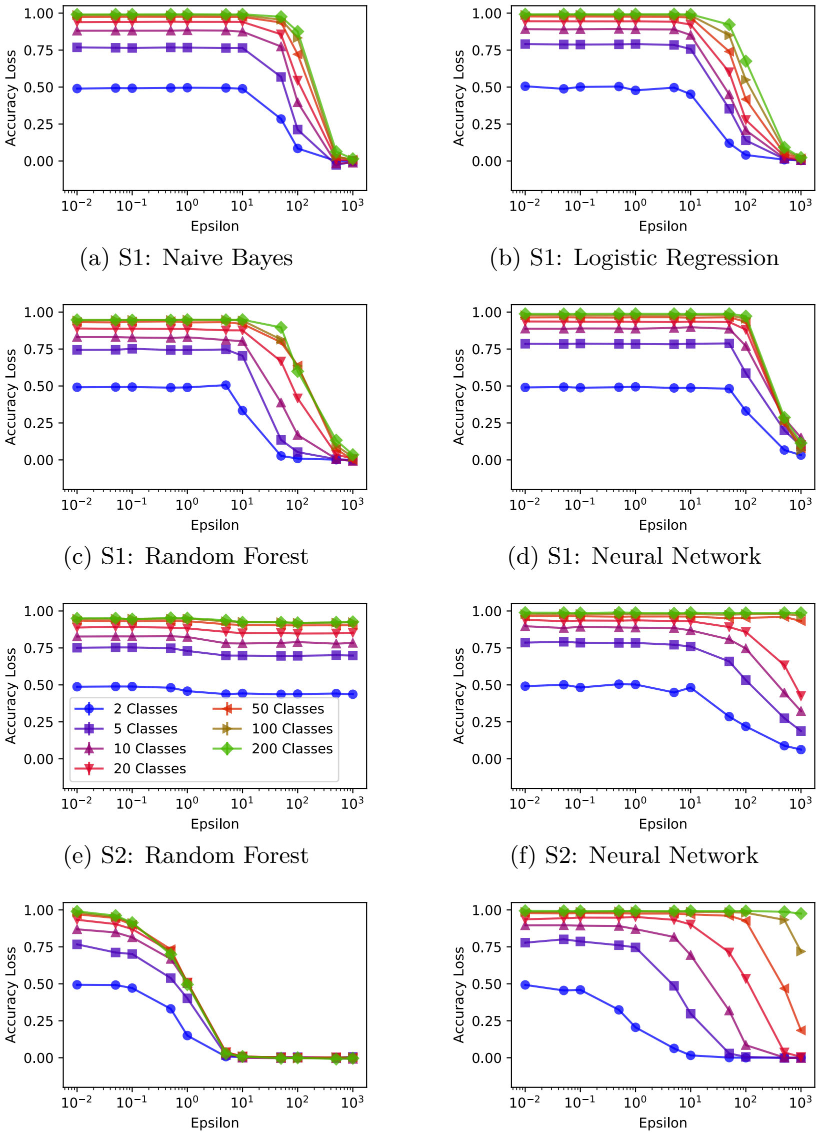
When we compare ML performance across Stages, we observe that from S1 to S3, there is an increasing amount of *DP* noise that can be applied to the ML method, before the accuracy of the system is reduced to a random guess.

Notably, given that the synthetic dataset is generated with the same underlying data vectors but with different class complexities, we observe that the inflection point occurs at about the same value of ϵ, irrespective of the number of classes.

While this inflection point does not vary across class complexities, the complexity of each dataset has a direct impact on the maximum *ACL* (due to the random guess).

6.4.1.2 Membership Inference Attacks vs. *DP* noise

In Figure 6.3, we analyze the results on *ConfMI* attack (similar results for *LossMI* are in Figure D.1 in Appendix D.2). We first analyze the inflection point of the privacy advantage of the attacker, for each of the framework stages, followed by an analysis on



(a) S1: Naive Bayes

(b) S1: Logistic Regression

(c) S1: Random Forest

(d) S1: Neural Network

(e) S2: Random Forest

(f) S2: Neural Network

(g) S3: Naive Bayes

(h) S3: Logistic Regression

Figure 6.2: Accuracy Loss for each ML method used, when different amount of *DP* noise is applied at framework Stages 1, 2 or 3, and for synthetic dataset complexities used. The underlying complexity of data vectors in each dataset remains the same.

the class complexity.

Across all ML methods in Stage 1, there is a clear inflection point at ϵ=100, where an attacker until this point has an advantage. It is interesting to note that in comparison to the *ACL*, the *ConfMI* advantage reaches zero before the accuracy is completely diminished.

In Stage 2, the absolute advantage is rather small, resulting in a seemingly high variance. Finally, in Stage 3, the inflection points for *NB* and *LR* occur at ϵ=1∼10 and ϵ=10∼100, respectively. We note that the gradient of decreasing *ConfMI* advantage (i.e., while ϵ is decreasing), is similar between S1 and S3, while the inflection points in S3 occur at smaller ϵ values than S1.

Across Stages, we note that for ML methods in S2, the *ConfMI* struggles with very low advantages, in comparison to S1 and S3. Similar to what was observed in *ACL*, the class complexity appears to have little effect on the inflection point of the *ConfMI* attack. However, where the attack is effective, a higher class complexity is more vulnerable to *ConfMI* attack.

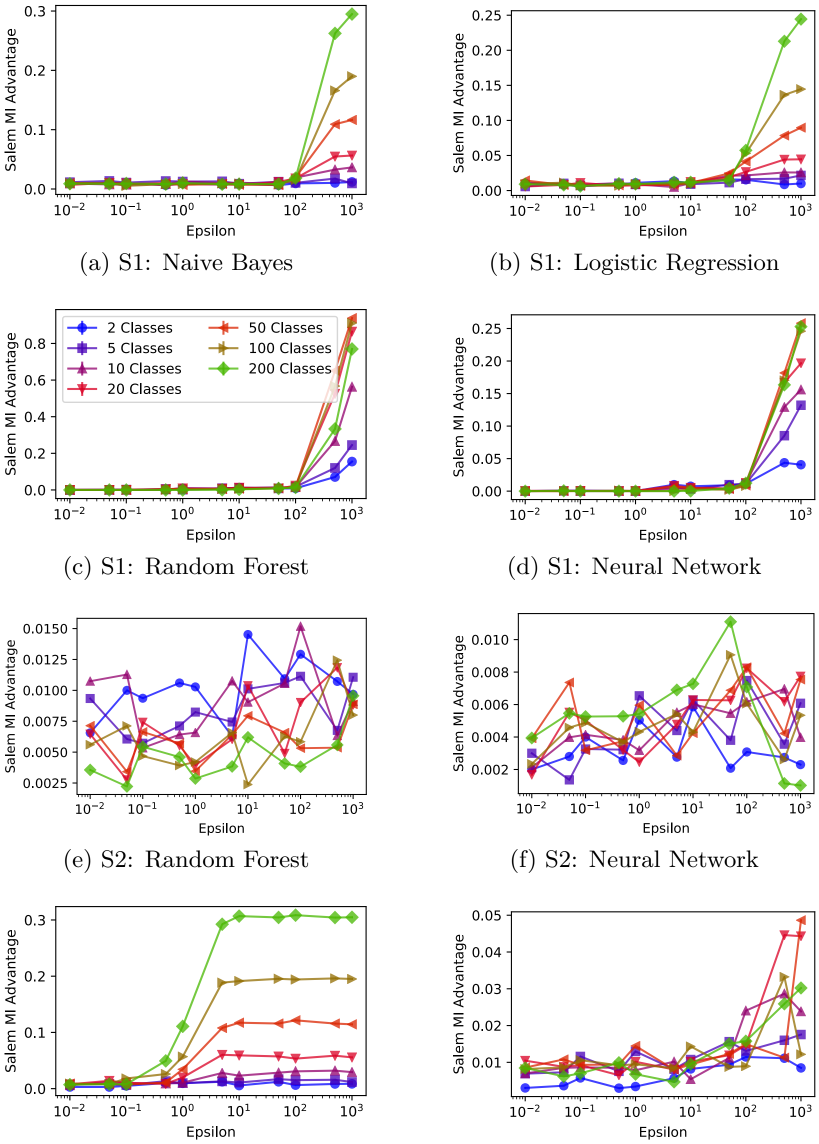
6.4.1.3 Attribute Inference Attacks vs. *DP* noise

In Figure 6.4, we analyze the results on *LossAI* attack (similar results for *ConfAI* are in Figure D.2 in Appendix D.3). We study the inflection point of the attack for each stage and across stages, and across class complexity.

Across S1, there is an inflection point in the attack effectiveness at ϵ=10∼100. We note that the absolute advantage of the attack differs depending on the ML algorithm used. For S2, *RF* has inflection point at ϵ=1, while *NN* at ϵ≈100. Interestingly, we note that in S2, while *RF* was observed to have difficulty learning the task (i.e., high *ACL* at different ϵ values), it is still vulnerable to expose attributes of the training data under *LossAI* attack. Lastly, for S3, *NB* has inflection point at ϵ=0.1, while *LR* at ϵ≈100.

The *LossAI* attack does not perform well against S2:*NN* and S3:*LR*. Interestingly, they both rely on a Stochastic Gradient Descent approach to learn the classification task. However, adding noise at stages S2 and S3 seem to have a bigger impact on the inflection points of *NN* and *LR* than when adding noise at Stage S1.

Again, we observe that number of classes does not impact the position of the inflection point for a given DP ML technique.



(a) S1: Naive Bayes

(b) S1: Logistic Regression

(c) S1: Random Forest

(d) S1: Neural Network

(e) S2: Random Forest

(f) S2: Neural Network

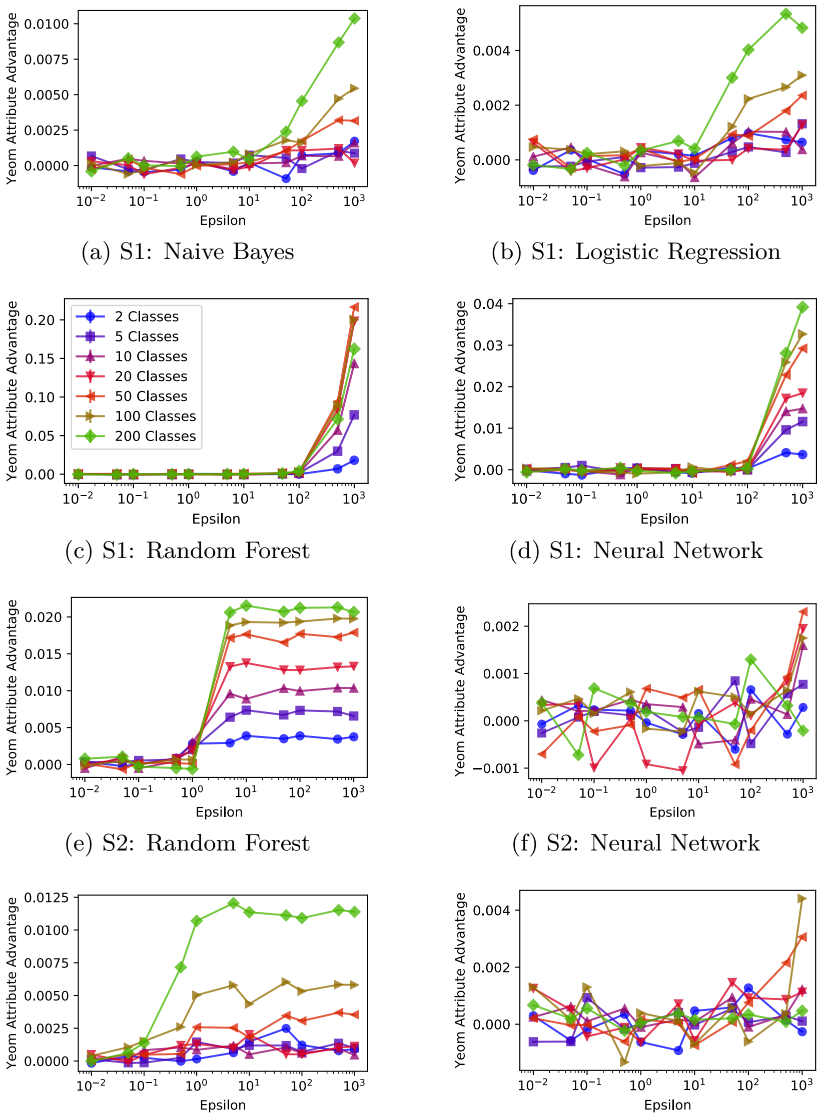
(g) S3: Naive Bayes

(h) S3: Logistic Regression

Figure 6.3: Advantage of *ConfMI* attack for each ML method, when different amount of *DP* noise is applied at Stages 1, 2 or 3, and for different synthetic dataset complexities. The underlying complexity of data vectors in each dataset remains the same.

6.4.1.4 Summary of Results on Synthetic Data

We saw evidence of a measurable inflection point and the tradeoff between the utility and privacy, as measured by MI and AI attacks. The observable ϵ value in which this inflection



(a) S1: Naive Bayes

(b) S1: Logistic Regression

(c) S1: Random Forest

(d) S1: Neural Network

(e) S2: Random Forest

(f) S2: Neural Network

(g) S3: Naive Bayes

(h) S3: Logistic Regression

Figure 6.4: Advantage of *LossAI* attack for each ML method, for different amount of *DP* noise applied at Stage 1, 2, or 3, for different synthetic dataset complexities. The underlying complexity of data vectors in each dataset remains the same.

occurs, is largely dependent on the Stage in which the *DP* noise is applied, and to a much lesser extent on the algorithm used. Between *ACL* and privacy advantage results, class complexity has little impact on where the tradeoff is observed.

6.4.2 Privacy-Utility Tradeoff on Real Data

Here, we present an analysis of results on real data, highlighting pattern similarities and differences compared to the synthetic data.

6.4.2.1 ML accuracy vs. *DP* noise

Next, we analyze the *ACL* on real data in a similar fashion as with the synthetic data, but grouping results of all datasets by class complexity (number of classes) to facilitate comparison. We discuss model performance at each stage and across stages, and the impact of class complexity on *ACL*.

Stage 1 (S1): When the *DP* noise is applied in S1, i.e., directly on the dataset, in Figures 6.5(a-d), we observe similar trends with *ACL* in synthetic data. However, *ACL* remains high until ϵ increases to ∼100. Up to that point, the modeling process is unable to learn the dataset rules, and the *ACL* is indicative of random guesses from the model, and this is true regardless of the model used.

Also, when the smallest amount of *DP* noise (i.e., ϵ=1000) is applied, we find that *LR* performs the best and achieves the lowest *ACL* ($0 < ACL < 0.1$) among the four ML methods. *NB* is also found to perform well at this noise level, even though its *ACL* is higher ($0.2 < ACL < 0.4$). *RF* achieves a wider range of *ACL* ($0.1 < ACL < 0.7$), which indicates a high dependence on the complexity of the dataset. Specifically, datasets with many classes (50 or 100) do not allow *RF* to achieve high accuracy when a minimal amount of *DP* noise is applied (high ϵ). Finally, *NN* performs the worst, since its *ACL* is high ($0.4 < ACL < 0.7$), regardless of dataset complexity.

Stage 2 (S2): Interestingly, as seen in Figure 6.5(e,f), when the *DP* noise is applied at S2, i.e., during model training, we notice that *ACL* is at its highest until $\epsilon \approx 1$ for *RF*, and $\epsilon \approx 10$ for *NN*. In particular, we observe that the accuracy of both models is generally low, and also highly depended on the dataset used ($0.1 < ACL < 0.9$). However, it seems that

NN is more effective than *RF* in modeling the data at hand, when very low *DP* noise is applied (i.e., ϵ=1000).

Stage 3 (S3): When the *DP* noise is applied at S3, i.e., after the model was trained but before it is used, we see (Figure 6.5(g,h)) the *ACL* at its highest until $\epsilon \approx 0.1$ for *NB*, and $\epsilon \approx 1 \sim 10$ for *LR*. When this inflection point is passed, and *NB* is applied, the lowest *ACL* ≈ 0 is achieved, and this performance is consistent across all datasets and class complexities. Alternatively, *LR* achieves low *ACL* only for low class complexity datasets. This association of the inflection point with class complexity is an exception to the norm and only evident with S3: *LR*. This effect is more pronounced when viewing the synthetic dataset (c.f., Figure 6.2(h)).

Remark 6.4.1. *We observe that* ACL *drops below* 0.0 *in S1:* LR *and S3:* NB*, indicating a model accuracy higher than if no privacy was applied. It is likely that the small amounts of* DP *noise applied have assisted in generalizing the model to predict better on unseen data. However, as the* DP *noise continues to increase, a diminished model performance returns. These may be interesting cases where a practitioner can seek to obtain smaller* ϵ *at no cost to model performance.*

Dataset class complexity: Generally, we know that datasets with high class complexity are harder to model with ML methods, and thus, their accuracy achieved would be expected to be low, in presence of no *DP* noise. Indeed, in the above experimentation, we notice that in several occasions, datasets with 50 or 100 classes are difficult to model with high accuracy and high *DP* noise. When small amount of *DP* noise is applied on low-complexity datasets with 2, 10 or even 20 classes, and especially in S1 and S3, the tested ML methods perform fairly well, with low *ACL*.

Comparing ML Performance Across *DP*-ML framework

Stages: To offer stronger protection guarantees for the given data, more *DP* noise must be added on the data (i.e., move towards the left hand-side of the aforementioned plots). When adding more noise, it appears that the *ACL* is affected in a similar fashion, for any ML method used, and regardless of the Stage at which we apply the noise, or dataset class complexity. There is an amount of *DP* noise that when it is added, it obscures much of the data variability, and consequently increases the *ACL* of each trained model. Interestingly, as identified earlier at the analysis of results from each Stage, and even on the results with synthetic data, this inflection point moves to higher levels of *DP* noise (i.e., lower values of ϵ), as the noise is added in later Stages in the framework. In particular, we notice that the *ACL* is drastically reduced when:

Stage 1: Inflection point of $\epsilon > 100$

Stage 2: Inflection point of $\epsilon > 1 \sim 10$

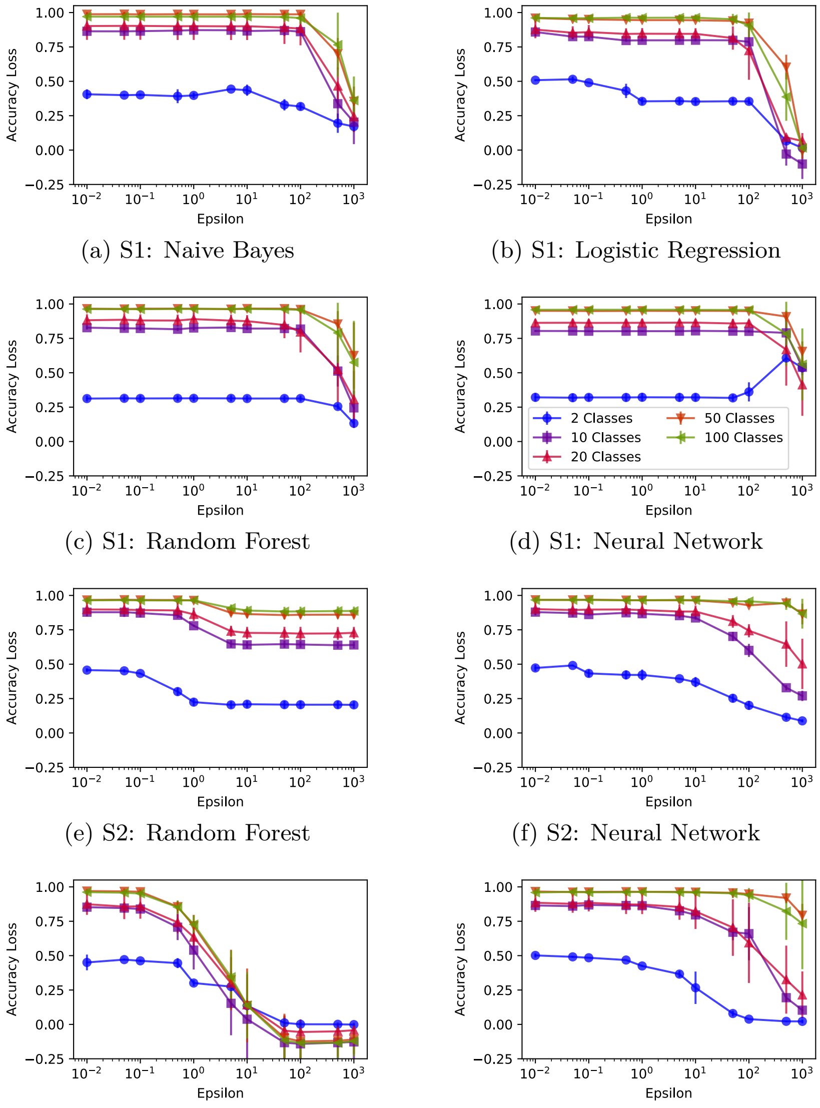
Stage 3: Inflection point of $\epsilon > 0.1 \sim 1$

Furthermore, it appears that the various models perform differently depending on the Stage the *DP* noise is applied. *NB* is more effective when used at S3 than S1, as for the same amount of *DP* noise, the model accuracy is better (i.e., *ACL* is lower). For similar reasons, *LR* is more effective when used at S3. However, if the *DP*-enabled ML framework requires consistent ML performance (i.e., low *ACL*) across datasets of different class complexities (i.e., 2-100 classes), then *DP* noise may need to be applied at S1. *RF* and *NN* perform better across all datasets when low noise is applied at S1.

6.4.2.2 Membership Inference Attacks vs. *DP* noise

Next, we analyze the advantage of an attacker when mounting *ConfMI* attack, in a similar fashion as with the synthetic data, but grouping results of all real datasets by class complexity. We discuss the effectiveness of *ConfMI* attack on individual models per stage and across stages, and the impact of class complexity on the attack. Finally, we also compare the two MI attacks, *ConfMI* and *LossMI*.

Stage 1 (S1): When *DP* is applied at S1, we notice that *ConfMI* performance is generally low and close to zero, up to $\epsilon{\approx}100$ for *NB* and *LR*, in Figures 6.6(a-d). For *RF*, the inflection point is higher ($\sim$500), indicating that *RF* may be more resilient to such MI attacks, for lower *DP* noise levels. In contrast, *NN* shows a non-zero advantage from $\epsilon{\approx}10$



(a) S1: Naive Bayes

(b) S1: Logistic Regression

(c) S1: Random Forest

(d) S1: Neural Network

(e) S2: Random Forest

(f) S2: Neural Network

(g) S3: Naive Bayes

(h) S3: Logistic Regression

Figure 6.5: Accuracy Loss for each of the ML methods used, when different amount of *DP* noise is applied at Stage 1, 2 or 3 of the framework, and for different real datasets used. We summarize the datasets by the number of classes used.

and on. Moving from left to right in the ϵ-axis, and until these thresholds are reached, the *ConfMI* attacker does not gain any privacy advantage from discerning if data records were being included in the training dataset of the given model or not.

Stage 2 (S2): When *DP* noise is applied at S2, the *ConfMI* advantage is low until ϵ≈0.5

for *RF*, and for any ϵ for *NN*, in Figures 6.6(e,f). In general, the effectiveness of this attack on models built with *DP* noise added at this Stage is low. In particular, we observe that for the *NN*, the attacker's advantage is overall low (*ConfMI* <0.008), regardless of the amount of *DP* noise applied, and for *RF*, *ConfMI* <0.025. Surprisingly, *RF* allows an attacker to learn relatively more on a dataset with lower class complexity, in contrast to *NN* and previous observations made in [33] and [34].

Stage 3 (S3): When *DP* is applied at S3, *ConfMI* advantage increases when ϵ>1 for *NB*, and ϵ>100 for *LR*, in Figure 6.6(g,h). This means that when *NB* is trained, datasets with high class complexity are more vulnerable than with *LR*.

Interestingly, all aforementioned results demonstrate similar patterns with the results on synthetic data (i.e., Figure 6.3 and 6.6), apart from *RF* built with noise applied in S2, as commented earlier.

Comparing *ConfMI* Across *DP*-ML framework Stages: As expected, when adding less *DP* noise in the framework (depending on the Stage at which it is applied), this impacts the effectiveness of a *ConfMI* attacker. In particular, when the inflection points below are reached, the attacker has a non-zero advantage.

Stage 1: Inflection point of ϵ>10∼500

Stage 2: Inflection point of ϵ>0.5

Stage 3: Inflection point of ϵ>1∼100

Additionally, for the same amount of *DP* noise, different ML methods allow the attacker to learn different amounts of private information (i.e., which instances of data are members of the training set). For example, *NB* allows the attacker to learn up to 10x more when the *DP* is applied in S3 than in S1. *LR* however allows similar privacy leaks, regardless of whether the *DP* noise is applied in S1 or S3. In addition, the increase in the attacker's advantage happens at the same inflection point for *LR*, whereas for *NB*, this happens in lower ϵ values for S3 than for S1.

Finally, when *DP* is applied in S2, there is 10x less privacy leakage than S1, regardless of whether it is *RF* or *NN* that is used.

Comparing *ConfMI* and *LossMI* attacks: Next, we look into the differences between *ConfMI* and *LossMI* (for brevity, the results for *LossMI* are shown in Figure D.3 in Appendix D.4). In the *LossMI* attack, the advantage of the attacker is found to be generally higher than in *ConfMI* attack. In particular, during S1, and while training *NB*, the *LossMI* reaches up to 4x higher advantages than *ConfMI*. Similarly, while training *RF* and *NN*, *LossMI* attack can reach up to 2x and 7x higher advantages than with the *ConfMI*, respectively. Interestingly, *LR* leads to similar advantage scores in both MI attacks. Further, in S2, and while training *RF* and *NN*, the *LossMI* attacker can reach up to 12x and 4x higher advantage than *ConfMI*, respectively. Finally, in S3, both *NB* and *LR* lead to similar advantage scores in both attacks.

We also observe that in *LossMI*, the inflection point where the attack is useful (i.e., advantage is non-zero) becomes more clear per Stage, compared to *ConfMI*. Like *ConfMI*, there is also a clear shift to lower levels of ϵ while applying noise to later Stages:

Stage 1: Inflection point of ϵ>100

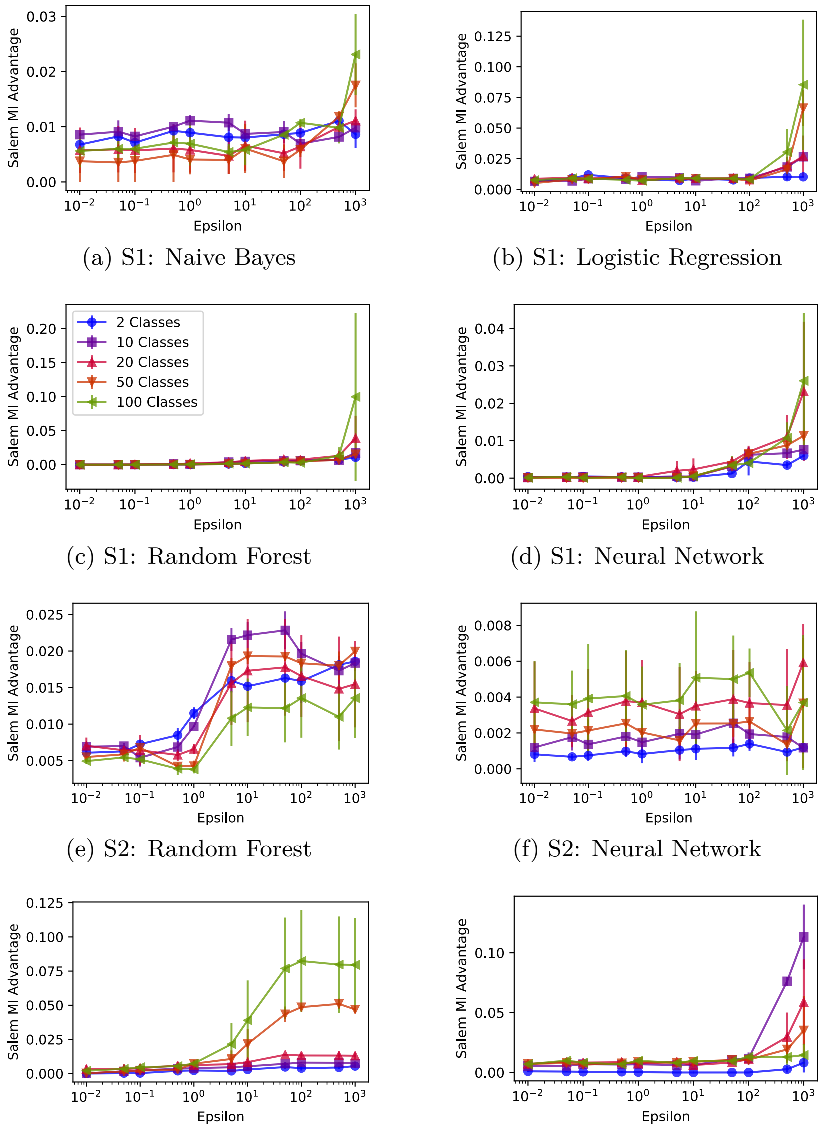
Stage 2: Inflection point of ϵ>1∼100

Stage 3: Inflection point of ϵ>5∼10

6.4.2.3 Attribute Inference Attacks vs. *DP* noise

Next, we analyze the results of *LossAI* attack on real data, in a similar fashion as with the synthetic data, but again, grouping results of all real datasets by class. Again, we discuss the attack's effectiveness on each stage and across stages, and how class complexity is an influencing factor. Finally, we also compare the two AI attacks, *ConfAI* and *LossAI*.

Stage 1 (S1): From Figures 6.7(a-d), we observe that for many of the models, and for the different datasets and class complexities, the *LossAI* advantage is very low or even



(a) S1: Naive Bayes

(b) S1: Logistic Regression

(c) S1: Random Forest

(d) S1: Neural Network

(e) S2: Random Forest

(f) S2: Neural Network

(g) S3: Naive Bayes

(h) S3: Logistic Regression

Figure 6.6: Advantage of *ConfMI* attack for each ML method used, when different amount of *DP* noise is applied at Stage 1, 2 or 3 of the ML framework, and for different datasets used. We summarize the datasets by number of classes used.

negative, which points to failed attack for leaking private information on the attributes of member data vectors in comparison to non-members. Thus, the attacker can achieves greater attack success when a vector has been excluded from the training dataset than when kept within. Therefore, this attack is not very effective when executed on a *DP*-

enabled ML framework that has trained models while injecting *DP* noise at S1, i.e., before any ML training. Out of the four models trained, *NN* would potentially leak the most, when the *DP* noise is low (ϵ>100). Interestingly, the adversary's advantage would still be 6x lower than in *ConfMI* and 50x lower than in *LossMI*.

Stage 2 (S2): When *DP* noise is added at S2, from Figures 6.7(e,f), we observe that *RF* allows the *LossAI* attacker to observe successful leaks when ϵ>1, but the privacy risk is still 10x lower than *ConfMI* and 100x lower than *LossMI*. Furthermore, training the *NN* model with *DP* noise leads the attacker to negative advantage.

Stage 3 (S3): Similarly with S2, in S3, a *LossAI* attacker can leak information about attributes of the data, when *NB* is trained, and above an inflection point of ϵ>5 (Figure 6.7(g,h)). This advantage is 8x lower than *ConfMI* and 10x lower than *LossMI*. Also, training *LR* with *DP* noise added at S3 leads again to negative advantage.

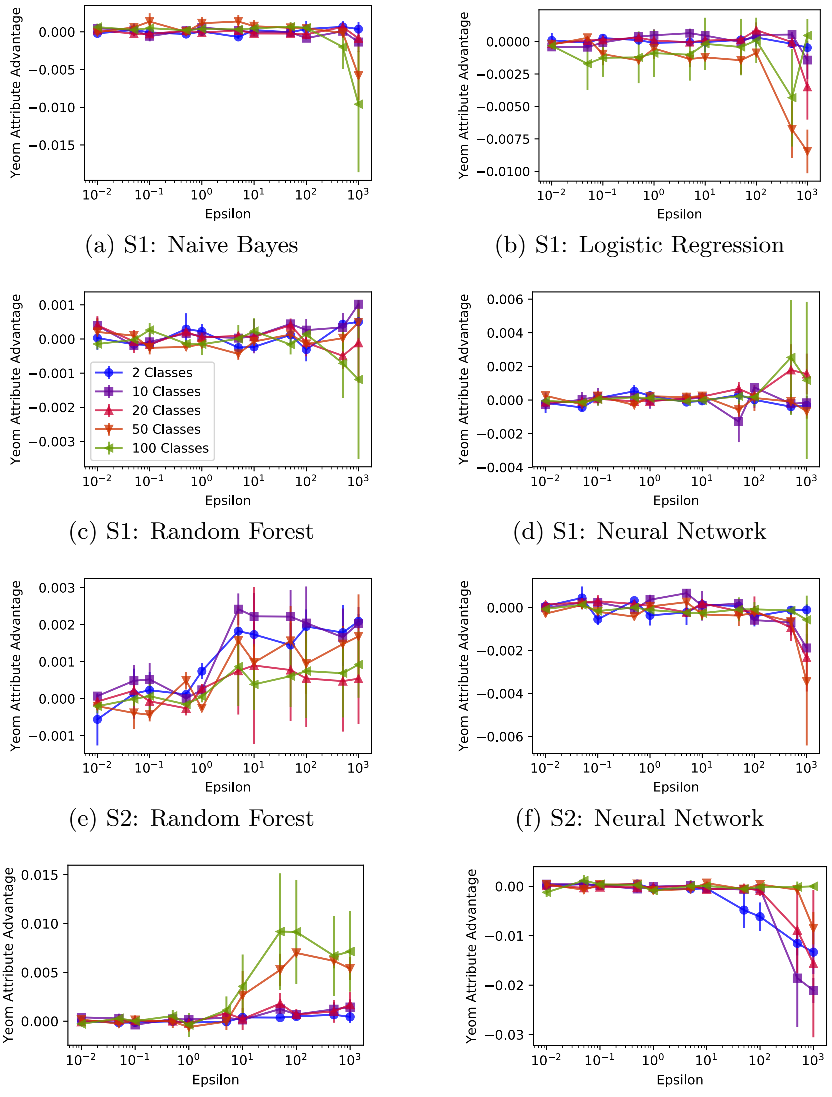
Comparing *LossAI* Across *DP*-ML framework Stages: Examining this attack across Stages, for different ML methods, we observe the following. *NB* allows the attacker a superior ability to leak more private information when *DP* noise is applied in S3. *LR* does not provide the attacker an advantage to leak information in either S1 or S3. In fact, if *DP* is applied at S3, the attacker will make 3x more errors (false positives) than S1, while trying to infer values of attributes. *RF* allows the attacker to leak more information if *DP* is applied at S2 in comparison to S1. Finally, *NN* is more robust against AI attacks when *DP* noise is applied at S2, in comparison to S1 which allows some information on attributes to leak at low amounts of *DP* noise. Again, we notice a clear shift of the inflection point to lower levels of ϵ as previously seen in the MI attacks:

Stage 1: Inflection point of ϵ>100

Stage 2: Inflection point of ϵ>1∼100

Stage 3: Inflection point of ϵ>5

Comparing *LossAI* and *ConfAI* attacks: Now, we highlight differences between *LossAI* and *ConfAI*, an AI attack presented by Zhao et al. [145] (*ConfAI* results are



(a) S1: Naive Bayes

(b) S1: Logistic Regression

(c) S1: Random Forest

(d) S1: Neural Network

(e) S2: Random Forest

(f) S2: Neural Network

(g) S3: Naive Bayes

(h) S3: Logistic Regression

Figure 6.7: Advantage of *LossAI* attack for each ML method used, when different amount of *DP* noise is applied at S1, S2, and S3 of the framework, and for different datasets used. We summarize the datasets by number of classes used.

in Figure D.4 in Appendix D.5). Across all stages, for both *LossAI* and *ConfAI*, the absolute advantage between members and non-members is very small, with *LossAI*, in general, marginally producing larger variances, with similar trends observed across different amounts of *DP* noise. An exception is S3: *NB*, whereby *ConfAI* is able to achieve the largest advantage of 3% for 100 classes (up to 10x better than *LossAI*). It is expected that

LossAI surpasses the performance of *ConfAI* as it has additional access to the training loss, and the prediction confidence of the correct label, whilst *ConfAI* has only access to the maximum prediction confidence irrespective of if the prediction class is correct. Finally, we observe that in both *LossAI* and *ConfAI*, the inflection points (i.e., when advantage is non-zero) are approximately the same across all Stages, with the exception of *ConfAI* on S2, which does not have the same pronounced inflection, as observed in *LossAI*.

Comparing MI and AI attacks: In general, we observe that AI attacks are less successful in leaking information about the data than MI attacks. This is based on the advantages computed in the two MI attacks (*LossMI* and *ConfMI*) that are mostly positive and of higher values than the two AI attacks (*LossAI* and *ConfAI*) values achieved, which were mostly zero or negative. This is to be expected, since an AI attack is an objectively more demanding attack with more potential for producing an incorrect result with the need to predict the exact value, instead of a binary membership/non-membership decision. It is also more difficult to be carried out in practice, due to the prerequisite knowledge the attacker should have of all but 1 attribute values.

6.4.2.4 DP-based ML under Constrained ACL or ϵ

An ML practitioner may wish to apply the most effective ML approach while considering constraints for either the ACL or ϵ.

ACL-bounded recommendations: We now determine which *DP*-based ML algorithm offers the best privacy guarantees (ϵ), when a practitioner's accuracy requirements are constrained. Specifically, we consider when the ACL cannot exceed a pre-determined threshold. To find the corresponding privacy offered (ϵ) and the associated ML technique, we linearly interpolate the empirical trend of ACL and ϵ. Then, we find the value of ϵ closest to the bounded ACL, for all ML methods tested. Finally, we report the lowest ϵ, and the corresponding ML method.

We display results for ACL constraints in Table 6.3. We observe that *NB* with *DP* noise applied in S3 is a prevalent option that can offer good accuracy for datasets with various class complexities. Only *NN* in S1 is a viable option for a binary class dataset, when the ACL requirement is very low (e.g., 0.01).

Table 6.3: Given a constrained ACL, we show best attainable privacy guarantee (ϵ), and the responsible *DP*-ML algorithm.

	2 Classes		10 Classes		20 Classes		50 Classes		100 Classes	
ACL	ϵ	DP-ML	ϵ	DP-ML	ϵ	DP-ML	ϵ	DP-ML	ϵ	DP-ML
0.01	50.00	S1-*NN*	16.52	S3-*NB*	38.11	S3-*NB*	31.71	S3-*NB*	30.17	S3-*NB*
0.02	47.23	S3-*NB*	14.18	S3-*NB*	35.99	S3-*NB*	30.01	S3-*NB*	28.61	S3-*NB*
0.05	37.62	S3-*NB*	9.47	S3-*NB*	29.61	S3-*NB*	24.89	S3-*NB*	23.92	S3-*NB*
0.10	21.61	S3-*NB*	7.31	S3-*NB*	18.99	S3-*NB*	16.37	S3-*NB*	16.10	S3-*NB*
0.20	7.70	S3-*NB*	4.52	S3-*NB*	8.16	S3-*NB*	8.32	S3-*NB*	8.52	S3-*NB*
0.30	0.50	S2-*RF*	3.48	S3-*NB*	4.99	S3-*NB*	5.64	S3-*NB*	6.08	S3-*NB*

ϵ-bounded recommendations: We now determine which *DP*-based ML algorithm offers the least accuracy loss, when a practitioner's privacy guarantee has been mandated. We use a similar interpolation technique.The results in Table 6.4 show that methods such as *RF*, *NN* and *LR* with *DP* noise applied in S1 are better options when high privacy constraints are required.

However, they lead to high ACL, which renders the models useless. When the privacy requirement can be relaxed, and the noise is applied in S2 or S3, then *NB* is a better option for maintaining ML accuracy, this remains true for datasets with low or high class complexity.

6.4.2.5 Summary of Findings

In Figure 6.8, we summarize the findings from different experimental setups, for ACL and for *ConfMI* attack metric (similar plots for the other privacy metrics are in Figure D.5 in Appendix D.6). In this summary figure, the tradeoff between ACL and protection against privacy leaks emerges more clearly. From this figure, and based on all previous explorations with respect to ACL and the four privacy attacks (*ConfMI*, *LossMI*, *LossAI*

Table 6.4: Given a constrained ϵ, we show the smallest compromise in *ACL*, and the responsible *DP*-ML algorithm.

	2 Classes		10 Classes		20 Classes		50 Classes		100 Classes	
ϵ	ACL	DP-ML	ACL	DP-ML	ACL	DP-ML	ACL	DP-ML	ACL	DP-ML
0.01	0.312	S1-*RF*	0.804	S1-*NN*	0.863	S1-*NN*	0.950	S1-*NN*	0.958	S1-*NN*
0.10	0.313	S1-*RF*	0.802	S1-*NN*	0.858	S1-*LR*	0.949	S1-*NN*	0.952	S3-*NB*
1.0	0.224	S2-*RF*	0.540	S3-*NB*	0.634	S3-*NB*	0.717	S3-*NB*	0.727	S3-*NB*
10	0.136	S3-*NB*	0.038	S3-*NB*	0.142	S3-*NB*	0.137	S3-*NB*	0.139	S3-*NB*
100	0.001	S3-*NB*	-0.141	S3-*NB*	-0.055	S3-*NB*	-0.124	S3-*NB*	-0.135	S3-*NB*
1000	-0.001	S3-*NB*	-0.127	S3-*NB*	-0.042	S3-*NB*	-0.109	S3-*NB*	-0.121	S3-*NB*

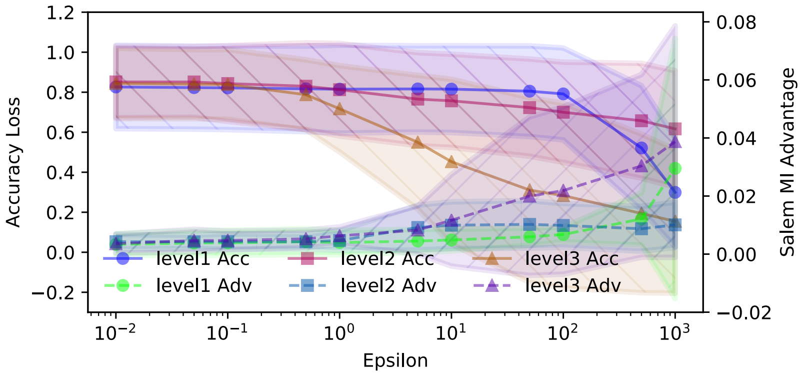


Figure 6.8: Summary plot of *ACL* (y1-axis) and *ConfMI* advantage (y2-axis) vs. ϵ applied (x-axis), for each Stage. Each point, for a line of a given Stage, is the mean across all results for different ML methods and datasets. Shaded colored areas signify 1 st. dev. around each mean.

and *ConfAI*), we summarize our key takeaways:

1. For a given amount of *DP* noise applied, ML models predict better (i.e., have good accuracy and low *ACL*), when the noise is inserted at a later Stage (e.g., S2 or S3 than S1) [Section 6.4.2.1].

2. To achieve reduced privacy leaks (lower attack advantages are better) with least amount of *DP* noise, this must be added in earlier Stages in the framework (S1 > S2 > S3) [Section 6.4.2.2 & 6.4.2.3]. Unfortunately, this comes with a penalty of worse ML prediction (accuracy) [Section 6.4.2.1]. Consequently:

3. The performance of current state-of-art MI and AI attacks is directly related to the prediction accuracy of the DP-ML model used. The inflection points of *ACL*

and privacy advantage for each DP-ML method correspond to approximately the same amount of *DP* noise.

4. The amount of *DP* noise added to a given DP-ML method does not influence the inflection point of a privacy attack (both MI and AI); instead, the inflection of attack success is dependent on the DP-ML method used and framework Stage the noise is applied (as noted in Takeaway 2).

5. The data complexity is unlikely to affect the inflection point of *ACL* [Section 6.4.2.1], or attack advantage [Section 6.4.2.2 and 6.4.2.3]. The inflection observed for each complexity is similar, for a given *DP*-ML method. We do, however, corroborate the known result that higher class complexity (more classes) lead to higher privacy leaks [33, 34], except for S2:*RF*.
6. When investigating the tradeoff over a wide range of *ACL* and ϵ constraints, we observe that S3:*NB* is the superior performing *DP*-ML method.
7. Evaluating the privacy-utility tradeoff with synthetic [Section 6.4.1] and real-world data [Section 6.4.2] yields similarities in trends and takeaways. There is potential for a dataset-agnostic approach to estimate inflection points for similarly classed data.

6.5 Conclusion

Privacy-preserving machine learning (ML) methods come with the inherent tradeoff between model utility achieved and privacy offered by the technique applied to protect the data. Our main contribution in this chapter is the proposal of a practical evaluation framework that enables a privacy ML researcher to study this tradeoff in depth for their data, and make data-driven decisions on where to apply Differentially Private (*DP*) noise inside their ML framework to protect their data and model, while achieving the best possible ML accuracy.

We identify three such Stages in the *DP*-enabled ML framework where *DP*-based noise can be added: 1) directly at the data collection, 2) during model training, or 3) at model finalization. We allow the practitioner to apply different amounts of noise based on the privacy guarantees they have, and at the different stages in the framework, and study the aforementioned tradeoff between the utility of the model trained, and the privacy of the data or model achieved, using four well-known privacy attacks.

We use our framework to comprehensively evaluate various implementations of *DP*-based ML algorithms, and measure their ability to fend off real-world privacy attacks, in addition to measuring their core goal of providing accurate classifications. We evaluate each implementation across a range of privacy budgets, and datasets, each implementation providing the same mathematical privacy guarantees. By measuring the models' resistance to real world attacks of membership and attribute inference, and their classification accuracy, we determine which methods provide the most desirable tradeoff between privacy and utility. Building on our results, we provide recommendations to a privacy ML researcher on how to select appropriate, *DP*-based ML methods, based on the data complexity at hand, and privacy guarantees and utility needs.

Chapter 7

Discussion, Future Works and Conclusion

In this section, we critically analyze the works that have been presented in the Technical chapters. We shall discuss the limitations and potential future works for each of the chapters, in addition to avenues in the general context of security and privacy attacks within this thesis. Finally, we shall provide concluding remarks.

7.1 Behavioral Side Channel Attacks

We begin by highlighting the limitations of Chapter 3. For k-out-of-n ORAS, one way to reduce the efficacy of the proposed side-channel attack is to increase the expected number of secrets present in the challenge as we have previously shown in Section 3.3.2. Unfortunately, this comes at a substantial cost to usability, since the only way to increase the number of secrets (without introducing statistical vulnerabilities [50]) is to either increase the number of secret items k or the window size l. Both increase the cognitive load on the user by requiring more secrets to recognize and more computations to perform. Decreasing the number of total items n is not desirable either as it reduces the password space making it susceptible to a brute-force attack.

As some of our features are directly related to timing information (e.g., total time of authentication), enforcing a minimum time before response submission, as suggested previously [51], may increase the difficulty of detecting the modulus event. This mitigation technique is also applicable to the PassGrids and Mod10 schemes (which are not k-out-of-n ORAS). However, this is ineffective against some of the other features used in our attack to detect the module event, e.g., dwell consistency. Thus, eye movement patterns can still reveal side channel information even with this mitigation technique.

Since we only had a low number of user samples available from the user study, we restricted ourselves to simple classifiers to produce a fair model. We acknowledge that additional user samples would allow more sophisticated classifiers (e.g. neural networks) to be trained, producing an improved side-channel classifier, and thus faster compromise of the user secret. Additionally, a more representative sample of the population may yield more diverse results in both timing and eye-tracking based features, as a majority of our participants were young, research students highly capable of doing basic mental mathematics.

While our attacks do not deem the ORAS considered in Chapter 3 completely insecure, they show that the security of these schemes is greatly reduced under side-channel attacks in terms of the number of rounds a secret can be used before renewal. We have argued that a scheme's claim to being observation resilient should be evaluated against side-channel attacks as well. In the case of BehavioCog and FoxTail, we have less than halved the "safe" number of rounds for these schemes. On other schemes, such as the HB protocol and Mod10 our attacks have shown a stronger result. The former does not have an efficient algebraic/statistical attack, and the latter by definition is immune to any algebraic attacks (being an OTP-based scheme).

For several reasons, we did not pursue more intuitive attacks based on eye movement patterns, such as following the user's gaze and directly labeling items with higher dwell times as possible secret items. The main difficulty is in simulating a given accuracy level of an oracle which predicts an item being a secret or a decoy item. To simulate a given oracle accuracy level, we would need to first determine how it translates to the empty challenge event, i.e., when there are no secret items present in a challenge. Moreover, if it is not an empty challenge event, we need to determine how the oracle accuracy relates to the number of secret items present in the challenge, the dwell times for each of the secret/decoy items in the challenge, and the fact that the user sometimes does not dwell on the secret items at all (we found through our user study that some users would never dwell over any particular item, secret or decoy when computing the response). Thus, while we could assume an oracle that predicts each of the l items present in the challenge as being a secret/decoy item with a given accuracy level, such an oracle would be relying on a lot more assumptions that need to be justified. In comparison, the modulus event is a single binary event tied to the entire challenge. A larger user study would indicate whether such direct eye-gaze attacks are viable or not, by relying on empirical data rather than simulated oracles. Due to our limited user study, we were not able to do so.

Lastly, we remark that another advantage of the modulus-based attack over direct eye-gaze attacks is that it delineates the attack algorithm from the actual side-channel being used. Eye movement patterns may not be the only source of side channel information. It could be possible that third party trackers on a device with access to any one of the many device sensors, may utilize this data and establish additional side-channel to expose the user's secret. This is an interesting avenue for future work.

7.2 Random Input Attacks

Chapter 4 proposes an additional criterion to assess the security of biometric systems, namely their resilience to random inputs. The work has implications for biometric template protection [179], where a target template resides on a remote server and the attacker's goal is to steal the template. In such a setting, obtaining an accepting sample may be enough for an attacker, as it serves as an approximation to the biometric template. Chapter 4 shows that the attacker might be able to find an approximation to the template via random input attacks if the system AR is not tested. Conversely, once the AR is reduced below

FPR (e.g., via adding beta distributed noise), then one can safely use FPR as the baseline probability of success of finding an approximation.

We have assumed that the input to the classifier, in particular, the length of the input is publicly known. In practice, this may not be the case. For instance, in face recognition, a captured image would be of a set size unknown to the attacker. Likewise, the number of features in the (latent) feature space may also be unknown. However, we do not consider this as a serious limitation, as the input length is rarely considered sensitive to be kept secret. In any case, the security of the system should not be reliant on keeping this information secret following Kerckhoffs's well known principle.

We note various detection mechanisms exist to protect the front-end of biometric systems. For example, spoofing detection [180] is an active area in detecting speaker style transfer [181]. Detection of replay attacks is also leveraged to ensure the raw captured biometric is not reused, for example, audio recordings [182]. There is also liveliness detection, which seeks to determine if the biometric that is presented is characteristic of a real person and not a recreation, e.g., face masks remain relatively static and unmoving compared to a real face [183]. Our attack surface applies once the front-end has been bypassed. Our mitigation measures can thus be used in conjunction with these detection mechanisms to thwart random input attacks. Being generic, our mitigation measures also work for systems that do not have defense measures similar to liveness detection.

Once an accepting sample via the feature vector API has been found, it may be possible to obtain an input that results in this sample (after feature extraction), as demonstrated by Garcia et al. with the training of an auto-encoder for both feature extraction and the regeneration of the input image [108].

We have focused on authentication as a binary classification problem, largely because of its widespread use in biometric authentication [2, 22–29, 109]. However, authentication has also been framed as a one-class classification problem [109, 184] or as multi-class classification [109], e.g., in a discrimination model, as noted earlier. In one-class classification, only samples from the target user are used to create the template, and the goal is to detect outliers. If this is achieved like distance-based classifiers, then as we have seen in Section 4.5.3, and as previously indicated in [30], the AR is expected to be small. In the multi-class setting, each of the n users is treated as a different class. This increase in classes is expected to proportionally lower the AR. However, whether this behavior is observed on real world data requires additional experimentation. We remark that as observed in Section 4.5.2, AR is highly dependent on the relative variance of the positive user and the negative user features. This may lead to the possibility of larger AR for some of the users, consequently leading to a higher risk of attack for these users. We leave a thorough investigation of the one-class and multi-class settings as future work.

Our focus was biometric authentication systems, however, this could be extended to other authentication settings. Already it has spurred additional work in Device Authentication [185], Garcia et al. generate a random input as a starting point, and produce adversarial perturbations for the random input to increase the effectiveness of the random attack. Though they did highlight that adversarial training techniques offered provide some mitigation to the random attack in Device Authentication.

7.3 Membership and Attribute Inference Attacks

The three black-box membership inference attacks evaluated in Chapter 5 were proposed by Shokri et al. [33], Salem et al. [34] and Yeom et al. [35]. We have also shown that our results apply in the white-box setting, by evaluating the membership inference attacks from Nasr et al. [1], who proposed passive and active white box attacks targeting both standalone and federated models. Of course, the research on membership inference is not limited to only these approaches or classification models. For instance, in [147] black and white box membership inference attacks are evaluated on generative adversarial networks; in [148] a new membership inference attack is proposed based on the loss-based membership inference attack from Yeom et al., and in [36] the authors show that even if membership inference attacks are ineffective as a whole on a dataset, they have disparate effects on different sub-groups in the dataset. While we have demonstrated that our observations generalize to numerous membership inference attacks and models, since the underlying principle remains the same, i.e., ML models are less susceptible to strong membership inference in the classification setting. It would be valuable to definitively confirm if our observations hold for non-classification models, or standalone attribute inference attacks.

We had observed that strong membership inference, and thus attribute inference could not be effectively achieved using current attack methodologies. Thus in this area, there is work to be done in improving the performance of these attacks. Additionally, the attribute inference attack method remains as attribute inference with membership inference as a sub-routine, it is possible to perform attribute inference directly, as a standalone process without membership inference. The closest to a standalone attribute inference attack is the model inversion attacks [149], though this exploits the learned forward relationships between attributes and the output class, instead of exploiting model memorization.

One open element of the membership and attribute inference attacks that was raised was our ability to defend against such attacks. A defense through differential privacy was evaluated in Chapter 6, however, the privacy-utility tradeoff leaves much to be desired. Nonetheless, in Chapter 5, it was observed that the most dominant class is the least vulnerable to the inference attacks. There is potential in incorporating the proportion of the feature space occupied by each class label as an additional objective in the training process, to force the equal representation of every class in the feature space. This could be interpreted as a means of promoting Fairness [36, 171], however, the existing definitions of fairness, seek to promote equality in the prediction outcomes, instead of the feature space itself.

Defense mechanisms have been developed to train models robust to adversarial examples [157]. Recall that adversarial examples are vectors with applied perturbations close to the original input vector that create large variations in the model's behavior, often manifesting as a mis-prediction [156]. When performing membership and attribute inference, a larger difference between the behavior of the known (member) and adversarial example (non-member) would allow for their distinction. Unfortunately, as Long et al. [152] state, the majority of the neighborhood around the vector would have a minimal difference on the model output; with the adversarial example behaving as an exception, rather than the norm. Due to the infrequency of adversarial examples, We hypothesize that these robust adversarial models will only have a minor positive impact on the mitigation of the inference attacks, as robust models seek to preserve the regular behavior of the model, only mitigating the behavior of the adversarial examples. Though this warrants further investigation.

7.4 Privacy - Utility tradeoffs of Differential Privacy

In Chapter 6, we had presented a comprehensive empirical study on the inherent tradeoff between utility and privacy when applying *DP* on ML algorithms. We investigated four, state of art *DP*-enabled ML and DL algorithms currently available in the literature, and evaluated the aforementioned tradeoff in each ML method, using four privacy inference attacks and one utility metric. We performed this investigation using both synthetic datasets and three commonly used real datasets of varying class and attribute complexity. Finally, we extracted from this experimentation various lessons and offered recommendations to interested privacy ML researchers.

During this evaluation with our framework, we limited the number of experimental configurations, to make the problem tractable with comparable results. However, even with these results, numerous potential experimental variants can still be investigated in the future with our framework.

***DP* variants:** Chapter 6 considers only ϵ-*DP*. However, as previously mentioned in Section 6.2.2, there is an increasing number of *DP* compositions and relaxations, such as (ϵ, δ)-*DP* and (α, ϵ)-*DP*. Interestingly, these *DP* relaxations are relatively recent, and many of the *DP*-enabled ML algorithms available in the literature that we used, are still using the original ϵ-*DP*. Future work should address how to adapt such algorithms to support newer *DP* relaxations, but should also enable ML practitioners to fairly compare these methods. For this, one would need to establish an equivalence between the various *DP* options available. In fact, in the future, even a simple evaluation of how a varying δ in (ϵ, δ)-*DP* impacts the resulting *ACL* and privacy metrics would be highly informative.

Local vs. Global *DP*: The boundary of trusted and non-trusted entities is becoming increasingly blurred. On one hand, ML model holders seek to protect their models' privacy and user data. On the other, privacy advocates argue even the model holders should not be trusted entities. There is a notion of trust in the *DP* ML pipeline: *Local* DP is when *DP* is applied very close to data generation without considering information or context about the entire system. Instead, *Global* DP does not need to tradeoff as much utility for the same mathematical guarantees: with a global system view, it can make more intelligent decisions on how to apply *DP* noise. In our framework, Local *DP* loosely corresponds to inserting *DP* noise in S1, with ML training receiving *DP*-protected data, whereas Global *DP* corresponds to *DP* noise applied in S2 or S3, with the model having unfettered access to unprotected data.

Utility metrics: We focused on the accuracy (loss) of a *DP*-enabled ML model with respect to its non-private counterpart. However, as mentioned in Section 6.2.6, more metrics can be employed to assess the change in utility of a trained *DP*-enabled ML model, such as precision, recall, F1 measure, etc. Furthermore, model Fairness [171] is another metric of particular interest given the increased public scrutiny of ML model fairness in the context of well established anti-discriminatory legal frameworks across the globe.

Privacy metrics: There was a missed opportunity to evaluate the privacy-utility tradeoff with the Approximate Attribute Inference attack defined in Chapter 5. Like [46], the resulting privacy advantage observed from the attribute inference attacks is minute, though still sufficiently large for trends to be observed. Additionally, it would not be advisable to evaluate the strong membership inference attack, as similar to the attribute inference attack, the methods used would produce a very small advantage for analysis. In essence, future evaluations with the framework should consider Membership Inference (Def 5.2.7) and Approximate Attribute Inference (Def 5.2.10).

Computation Cost: Each ML model requires computational resources (CPU, memory) while being trained. We anticipate that when *DP* noise is added at different Stages in the framework, different resources are required to make the data *DP*-enabled, or to train the model in a *DP* fashion, or modify the learned model later in a *DP* fashion. Consequently, another potential aspect of the privacy-utility tradeoff to be studied is the resource overhead and its relation to the amount of *DP* noise, and the framework Stage it is added.

7.5 Security and Privacy Attacks

Beyond the specific examples presented for each of the chapters, we now reflect on these attacks and how they may be applied in the real world. From the deconstruction of cybersecurity attacks, it has been observed that significant breaches are often complex, harnessing multiple attack vectors to realize the attacker's goal. While in this thesis we have focused on individual attacks, it has not yet been realized as to how an attacker may collate and exploit multiple attacks simultaneously. For example, the effectiveness of an active attack, such as model poisoning or backdoor attack could be amplified when combined with additional information obtained from passive inference attacks on the same model.

Finally, in all our investigated attacks, we have attempted to create an analog of real world systems on which we would target our attacks. In every instance, our replica systems are made to mirror their real world counterpart as closely as possible. Unfortunately, due to either the use of these models in the private domain, without disclosure or access to its inner workings; or a lack of use due to the emergent nature of the system, several assumptions about these systems have been made. However, not to discount the value of understanding how and why these attacks are possible, in the future, we would like to extend and evaluate these attacks against real world systems.

7.6 Conclusion

From the numerous works of research collated here in this Thesis, it is clear that emerging technologies, with and without machine learning may be vulnerable to security and privacy attacks. Without extensive consideration of these risks, the forward march of technological innovation would not be built on a strong foundation, creating points of exposure that could lead to irreversible damage to individuals and companies. Conventional cybersecurity attacks, like a data breach, can start with a simple phishing attack, to obtain login credentials and gain unauthorized access to a system. As we have shown in our attacks against alternate authentication schemes, rushing into new technologies like observation resilient schemes, or biometrics, without understanding the full extent of the risks of the human factor, or even a simple random attack, may leave the system vulnerable. In addition to security attacks, our investigation into membership and attribute inference attacks has shown that the existing threat of membership inference may be overstated (in a strong sense), but attribute inference does not account for the possibility of the attacker obtaining an approximate amount of information. We have not yet experienced a full scale attack against commercial machine learning models, but if data breaches in a conventional sense are used as a reference point, there will be a non-negligible financial and reputation cost to the model holders. Not to mention in a time where digital proficiency is on the rise, people's attention is being drawn to data rights and privacy, additional breaches in the privacy of their data could continue to erode trust in these established systems. Finally, with legislation across the world patching deficiencies in data governance, privacy mindful applications that still offer utility will become increasingly important. This thesis, like many others that have preceded and will follow, is not the golden bullet in absolving security and privacy risks, however by revealing attacks, we can work towards defenses and strategies that make these risks more tolerable.

Appendix A

Supplementary material for Chapter 3 - Exploiting Behavioral Side Channels in Observation Resilient Cognitive Authentication Schemes

A.1 Proof of Theorem 3.3.1

Proof.

$$\begin{aligned}
\Pr(Y < d \mid g) &= \sum_{y=0}^{d-1} \Pr(Y = y \mid g), \\
&= \frac{1}{d^g} \sum_{y=0}^{d-1} \sum_{s=0}^{\lfloor y/d \rfloor} (-1)^s \binom{g}{s} \binom{y - sd + g - 1}{g-1},
\end{aligned}$$

Since $0 \leq y < d$, we have $\lfloor y/d \rfloor = 0$. Thus, $s = 0$, and we get

$$\begin{aligned}
\Pr(Y < d \mid g) &= \frac{1}{d^g} \sum_{y=0}^{d-1} \binom{y + g - 1}{g-1}, \\
&= \frac{1}{d^g} \binom{0 + g - 1}{g-1} + \frac{1}{d^g} \binom{1 + g - 1}{g-1} + \cdots \\
&+ \frac{1}{d^g} \binom{d - 1 + g - 1}{g-1} \\
&= \frac{1}{0!d^g} + \frac{g}{1!d^g} + \frac{g(g+1)}{2!d^g} + \cdots \\
&+ \frac{g(g+1)\cdots(g+d+1)}{(d-1)!d^g}
\end{aligned}$$

As $g \to \infty$, we see that each polynomial numerator is $o(d^g)$. Thus, $\Pr(Y < d \mid g) \to 0$. Or equivalently, $\Pr(Y \geq d \mid g) \to 1$. □

A.2 Proof of Lemma 3.3.2

Proof. First let $g = 2$. The proof is by induction on $d \geq 1$. First let $d = 1$. Then since $p(0) > 0$,

$$\begin{aligned}
\frac{1}{2} \sum_{i=0}^{1} ip(i) &= \frac{1}{2} \cdot 0 \cdot p(0) + \frac{1}{2} \cdot 1 \cdot p(1) \\
&< \frac{1}{2} \cdot 1 \cdot p(0) + \frac{1}{2} \cdot 1 \cdot p(1). \\
&= \frac{1}{2} \sum_{i=0}^{1} p(i).
\end{aligned}$$

Thus, the statement is true for $d = 1$. Now assume the statement holds for $d = r$, then

$$\frac{1}{2} \sum_{i=0}^{r+1} ip(i) = \frac{1}{2} \sum_{i=0}^{r} ip(i) + \frac{r+1}{2} p(r+1)$$

$$< \frac{r}{2}\sum_{i=0}^{r} p(i) + \frac{r+1}{2} p(r+1)$$
$$< \frac{r+1}{2}\sum_{i=0}^{r+1} p(i),$$

which completes the proof for $g = 2$. For $g > 2$, observe that

$$\frac{1}{g}\sum_{i=0}^{d} ip(i) < \frac{d}{2}\sum_{i=0}^{d} p(i),$$

and hence the lemma is true for all $g \geq 2$. □

A.3 Proof of Theorem 3.4.1

Proof. Let $i \in [n]$ be a secret item and let $j \in [n], j \neq i$ be a decoy item. Let $\eta(i)$ and $\eta(j)$ denote the number of times the two items appear in m challenges. Let $\eta^{+}(i)$ and $\eta^{-}(i)$ denote the number of times the secret item i appears in the modulus and no-modulus events, respectively. First, for sufficiently large m, we see that both $\eta(i)$ and $\eta(j)$ approach their expected value, and therefore

$$\eta(i) \approx \eta(j) = \eta^{+}(j) + \eta^{-}(j).$$

Next, note that due to step 6 in the algorithm, the secret item never gets penalized in case the oracle correctly identifies the no-modulus event (the secret item if present cannot have weight more than the response r). Therefore, we are looking at the instances where the oracle wrongly labels a modulus challenge as a no-modulus challenge. The probability of a particular response in this case is $1/d$. Since the secret item's weight is randomly generated, the probability that its weight is greater than $r = i$ is given by $(d - 1 - i)/d$. Therefore, the expected points update is given by

$$\frac{1}{d^2}\sum_{i=0}^{d-1}(d-1-i)u_i.$$

Denote the above by u. Then, the expected score of a secret item i in m challenges is given by

$$\begin{aligned}(1-\text{TPR})\cdot\eta^{+}(i)\cdot u &\leq (1-\text{TPR})\cdot\eta(i)\cdot u\\ &\approx (1-\text{TPR})\cdot\eta(j)\cdot u\\ &= (1-\text{TPR})\cdot\eta^{-}(j)\cdot u\\ &+ (1-\text{TPR})\cdot\eta^{+}(j)\cdot u\\ &< \text{TNR}\cdot\eta^{-}(j)\cdot u\\ &+ (1-\text{TPR})\cdot\eta^{+}(j)\cdot u,\end{aligned}$$

which is the expected score of the decoy item j in m challenges. □

A.4 Feature Intuition

Recall that the Dwell is period of user visual intake of a specific item, characterized by lowered rapid eye movement.

A.4.1 Adversary Level 1 Feature Hypotheses

a) **Total Time:** A challenge requiring a modulus operation involves more mental operations (size-effect-problem [99]), and should require more time.

b) **Mean Challenge Weight:** The expected value of individual weights is uniform, however collectively challenge may have a bias in the item weights. E.g. there are more higher weights, potentially providing information about the modulus event.

c) **Challenge Response:** As previously noted, there exists a small bias in the probability of a modulus occurring dependent on the final submitted response. This may be useful for informing the classifier.

A.4.2 Adversary Level 2 Feature Hypotheses

a) **Min Dwell Time:** The shortest time spent viewing an image can be indicative of the user's confidence that a secret image has been located. This value should be shorter when secret items are present. Alternatively this value may be short for when a user retrieves weights from a low number of secret for mental computation (e.g. 1 secret requires no computation). However, a user quickly scanning in the challenge may also exhibit a short min time, which can be managed by considering the 20th percentile.

b) **Max Dwell Time:** The longest time spent viewing an image may be indicative of the time that a user spends stationary to compute the challenge result. A more difficult arithmetic problem should incite a larger cognitive load and hence require more time. Like min dwell time, the longest dwell may reflect instances of user distraction for an extended period of time, hence the consideration of the 80th percentile of dwell times.

c) **Mean Dwell Time:** If there are more secret images, with more math, the user may spend more time processing the challenge (Feature 1.a). But, the verification time of each image may be shorter as they skim over the challenge once again to retrieve item weights for mental computation.

d) **STD Dwell Time:** It is observed that users are more likely to double check the challenge if a low number secrets are present. Spending more time on specific uncertain images, should result in a larger time deviation.

e) **Number of Dwells:** The number of dwell positions should be indicative of the extent of scans and checks for secrets in the challenge. A challenge with more secrets may prompt additional checks, producing more dwells.

f) **Time from longest stationary till end:** A challenge with a secret present, should have the user stop and (mentally) compute a result. After the pause, they will submit their response. This is an attempt to isolate the period of time in which the user should be computing their response, and indicative of the problem difficulty.

g) **Dwell Consistency:** By contrasting the high and low extremes (min/max or 20th/80th percentile), we can obtain a normalized ratio of their differences. Thus, any outlying images such as a secret the user spends additional time on, will be captured by this normalized difference.

h) **Duration of First Fixation:** If a user locates a secret image initially when the challenge is presented, they will remain fixated on their secret image for a longer duration of time as compared to decoys [186].

i) **Duration of Last Fixation:** When a user recovers their secret items from the challenge, their last fixation would also include computation time for the modulus-sum obtain the result. The length of this last fixation is a possible indicator of the difficulty of the computational task, with the inclusion of the modulus hypothesized to take longer.

j) **Longest Dwell Consistency:** As an extension of the previous point, consistently taking a long time traversing multiple items may be indicative of a difficult task like that of the modulus.

A.4.3 Adversary Level 3 Feature Hypotheses

a) **Number of transitions (Halves)** A user scanning through a challenge is likely to traverse the entire challenge, consequently crossing between different areas of the challenge. It is suspected to be larger for challenges with more secrets present. Transitions include: Left-Right (Ignoring center due to odd # of columns), Top-Bottom and (Even), ignoring two center rows.

b) **Time from bottom of the screen to the end:** After a user finds their secrets, they press a button to proceed to a submission page, a user may take additional time to (mentally) compute the response prior to proceeding in a modulus event with many secrets.

A.4.4 Adversary Level 4 Feature Hypotheses

a) **Highest Number of Reentries:** For a given secret image within the challenge, it is likely the user's first pass will view the image to simply recognize it. However, upon completion of a visual search, the user may revisit the image to get the weight for response computation. Potentially, leading to larger values when many secrets are present.

b) **Number of non-entries:** For a given challenge, a user may quickly re-identify their secrets from a rapid search (no dwell), the secrets form salient images. As such, some images may not be viewed at all, thus producing more un-viewed images when less secret images are present.

c) **Length of longest repeating sequence:** During the visual search, a user is may backtrack on the items identified as secrets, either from uncertainty, or a revisit to retrieve weights for response computation. Thus a longer repeating sequence could be related with a larger number of secret items in a challenge, and thus provide modulus event information.

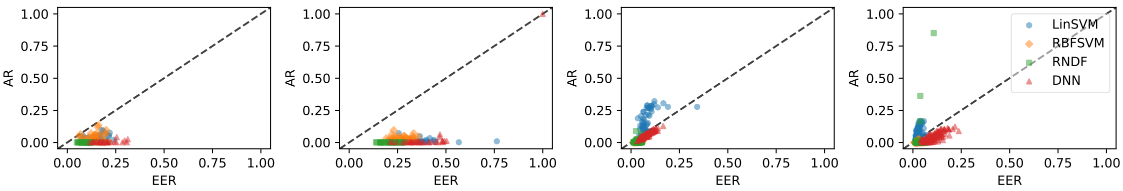
d) **Weight of Longest dwell item (Top 3):** As previously mentioned a user spends more time on secret items. Therefore larger weights on these dwelled items will likely require a modulus operation. The weights of the top 3 largest dwelled items are considered.

Appendix B

Supplementary material for Chapter 4 - On the Resilience of Biometric Authentication Systems against Random Inputs

B.1 Mitigation ROC Plots

This appendix contains plots of the results as discussed in Section 4.6. Figure B.1 contains per-user scatter plots of AR and FPR for all biometric modalities and algorithms. For the same classifiers, Figure B.2 illustrates the ROC curves for classifiers trained with the inclusion of beta distributed noise only. Finally, Figure B.3 displays the ROC curves for all classifiers of touch and face datasets with the inclusion of both beta distributed noise and raw input vectors as an additional mitigation strategy against the the raw inputs, which were unfazed by the beta noise. A summary of changes in FRR, FPR, AR and RAR of both Figure B.2 and B.3 have been provided earlier in Table 4.1 and 4.2 of Section 4.6.

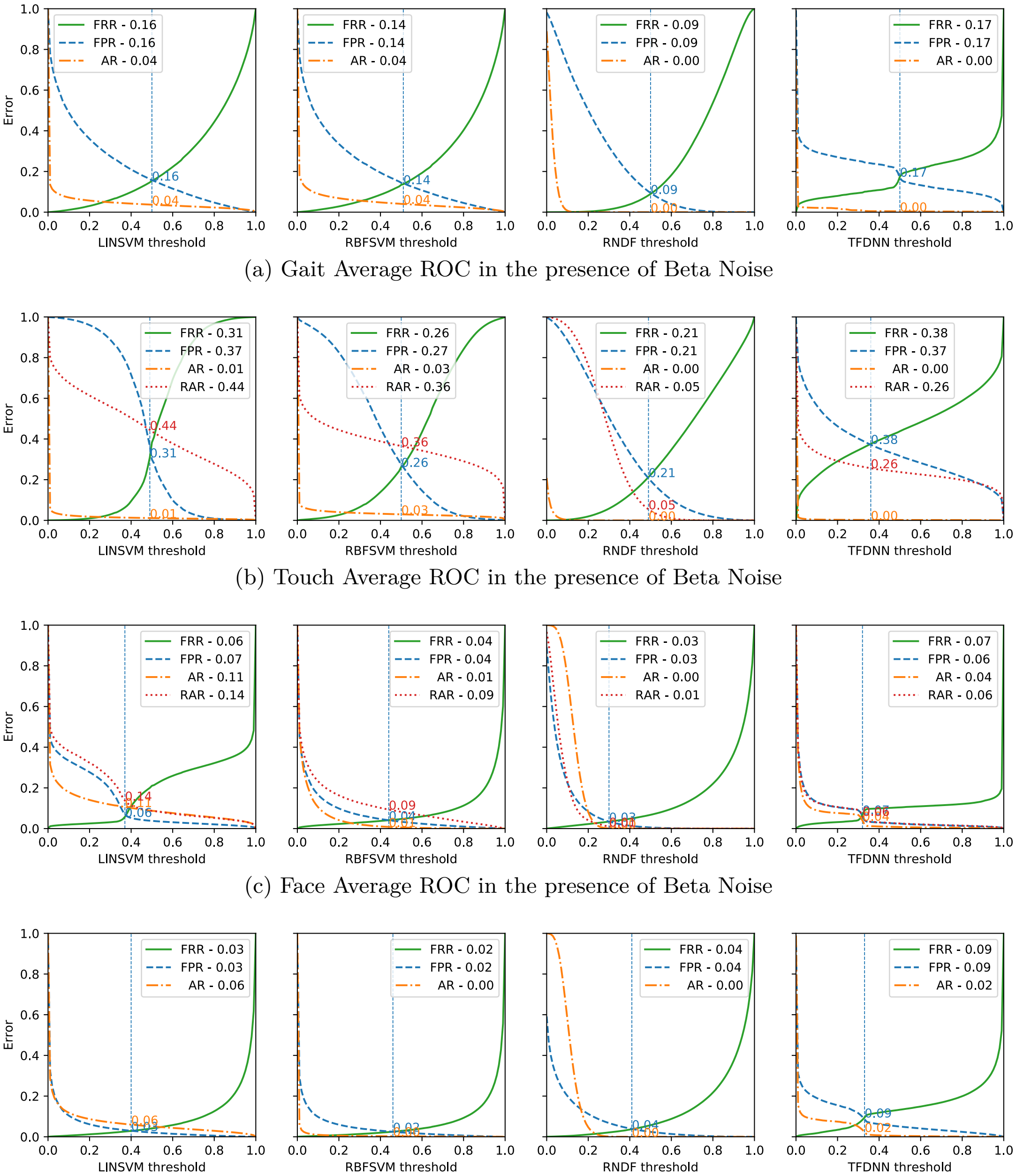


(a) Gait (b) Touch (c) Face (d) Voice

Figure B.1: Individual user scatter of AR and FPR after the addition of beta distributed noise. A substantial proportion of users now exhibit an AR close to zero, or below the AR = FPR. Unfortunately, this defense mechanism did not completely minimize the AR of LINSVM for the Face authenticator. Nor did this defense protect two outlying users in the RNDF voice authenticator.

B.2 DNN Estimator configuration.

All models were trained for 5000 steps, with batch size of 50, with the Adagrad optimizer. The exact layer configuration of the `DNNEstimator` [129] used can be found on our project page (`https://imathatguy.github.io/Acceptance-Region/`).



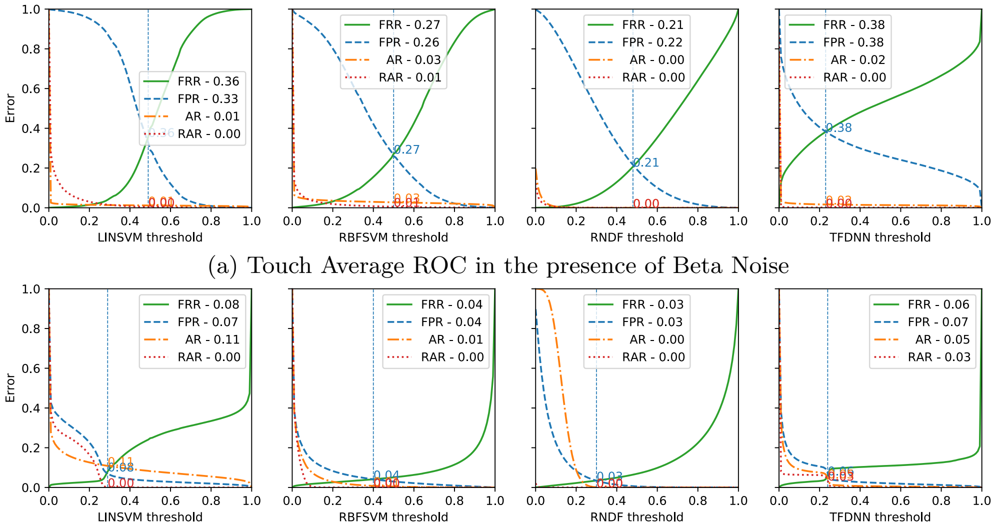
(a) Gait Average ROC in the presence of Beta Noise

(b) Touch Average ROC in the presence of Beta Noise

(c) Face Average ROC in the presence of Beta Noise

(d) Voice Average ROC in the presence of Beta Noise

Figure B.2: Beta-noise mitigation of AR, with additive negative training noise sampled from a symmetric beta distribution around the mean of the user's features. The EER is marked on the diagrams as a vertical line. It is noted the plots with RAR curves the additional Beta-noise is not sufficient in mitigating RAR attacks.



(a) Touch Average ROC in the presence of Beta Noise

(b) Face Average ROC in the presence of Beta Noise

Figure B.3: Beta-noise mitigation of AR, with additional negative samples from the RAR feature set. The EER is marked on the diagrams as a vertical line. Addition RAR vectors were included as it was previously observed that beta noise is sufficient in mitigating AR attacks, but not the RAR attack.

Appendix C

Supplementary material for Chapter 5 - On the (In)Feasibility of Attribute Inference Attacks on Machine Learning Models

C.1 Model Parameters

C.1.1 Target Models

We will first describe the **Neural Network (NN)** based target models used in the bulk of our experiments, followed by the configurations of the classifiers in Section 5.4.2. The training and testing accuracies can be found in Table C.1. **Location:** The model was trained in keras as a fully connected NN with 1 hidden layer of 128 nodes with the "tanh" activation function. We replicate the training and testing accuracy of [33]'s target model. **Purchase:** The target model was trained in keras as a fully connected neural network with 1 hidden layer of [128] nodes with a "tanh" activation function. This architecture replicates the training and testing accuracy for the target model as previously reported in [33]. **CIFAR:** The target model is a multilayer perceptron, consisting of two hidden layers of 256 units, with relu activation layer and a softmax output layer. This is the same architecture used in [46].

Logistic Regression (LR): The parameter C was set at 100 for all datasets, with all other parameters remain at the default values. **Support Vector Machine (SVM)**: We select a linear kernel for all the datasets. We keep parameters at default values. **Random Forest (RF)**: The number of estimators was chosen to be 100 with no depth specified, the remaining parameters were kept as defaults.

The training and testing accuracies for each algorithm, and for each datasets are noted in Table C.1.

C.1.2 MI Attack Configurations

Due to the different data requirements for each attack, the way the data is partitioned differs, we note these differences in this section. The average MI AUC can be found in Table C.1. For the Conf and Loss attacks, we do not require additional data to train an attack model.

Table C.1: Summary of training and testing accuracies, with MI AUC for all machine learning classifiers.

Dataset	Model	Train Acc	Test Acc	MI AUC	Model - MI	Train Acc	Test Acc	MI AUC
Loc-30	LR - Conf	1.000	0.582	0.897	NN - Conf	1.000	0.794	0.705
	SVM - Conf	1.000	0.731	0.916	NN - Loss	1.000	0.794	0.710
	RF - Conf	1.000	0.566	0.975	NN - Shadow	1.000	0.666	0.909
	NN - Local	0.998	0.430	0.891	NN - Global	0.998	0.430	0.886
Pur-100	LR - Conf	1.000	0.484	0.765	NN - Conf	0.999	0.765	0.708
	SVM - Conf	1.000	0.799	0.855	NN - Loss	0.999	0.765	0.720
	RF - Conf	1.000	0.606	0.998	NN - Shadow	1.000	0.700	0.842
	NN - Local	0.538	0.487	0.508	NN - Global	0.538	0.487	0.719
Pur-50	LR - Conf	0.995	0.601	0.614	NN - Conf	0.998	0.832	0.629
	SVM - Conf	1.000	0.857	0.716	NN - Loss	0.998	0.832	0.638
	RF - Conf	1.000	0.724	0.980	NN - Shadow	1.000	0.778	0.763
	NN - Local	0.692	0.657	0.520	NN - Global	0.692	0.657	0.668
Pur-20	LR - Conf	0.973	0.785	0.552	NN - Conf	0.999	0.889	0.577
	SVM - Conf	1.000	0.906	0.584	NN - Loss	0.999	0.889	0.582
	RF - Conf	1.000	0.813	0.917	NN - Shadow	1.000	0.841	0.690
	NN - Local	0.803	0.781	0.505	NN - Global	0.803	0.781	0.626
Pur-10	LR - Conf	0.973	0.878	0.521	NN - Conf	0.999	0.911	0.558
	SVM - Conf	1.000	0.932	0.530	NN - Loss	0.999	0.911	0.561
	RF - Conf	1.000	0.840	0.902	NN - Shadow	1.000	0.868	0.644
	NN - Local	0.836	0.818	0.503	NN - Global	0.836	0.818	0.608
Pur-2	LR - Conf	1.000	0.986	0.499	NN - Conf	0.998	0.959	0.521
	SVM - Conf	1.000	0.987	0.502	NN - Loss	0.998	0.959	0.522
	RF - Conf	1.000	0.921	0.781	NN - Shadow	0.999	0.944	0.580
	NN - Local	0.914	0.906	0.505	NN - Global	0.914	0.906	0.567
CIFAR-20	NN - Conf	0.920	0.322	0.544	NN - Loss	0.920	0.322	0.799
	NN - Shadow	0.999	0.281	0.925	-	-	-	-
CIFAR-100	NN - Conf	0.831	0.214	0.524	NN - Loss	0.831	0.214	0.844
	NN - Shadow	0.999	0.170	0.967	-	-	-	-

C.1.2.1 Conf and Loss attacks

Location: We take the full dataset and divide it into 2 parts. 20% is used for training the target model and remainder 80% is kept for testing purposes. **Purchase:** We sample 20,000 records from the dataset and divide it into 2 parts. The first 80% is used for training the target model and remaining 20% is kept for testing purposes. **CIFAR:** 50,000 records are sampled from the dataset to constitute our experimental dataset, from this 20% is reserves as the training data, and the remaining 80% is use for testing.

C.1.2.2 Shadow MI

Location: We take the full dataset and divide it into 3 parts. The first 20% is used for training the target model, 64% for training the shadow models and the remaining 16% is retained for testing. Our Shadow MI attack is from the open-source library [187]. The training and testing accuracies are found in Table C.1. Our models are as follows:

1. **Shadow Models:** We select 60 attack models for Location dataset, consistent with [33]. The architecture of these shadow models and the size of their training dataset are equivalent to the target model.

2. **Attack Model:** The attack model is multilayer perceptron with a 64-unit hidden layer and a sigmoid output layer. This architecture replicates the precision and recall as previously reported in [33]. For the Location-30 dataset our MI attack obtains a precision of 0.93 and recall of 0.82

Purchase: We sample 40000 records from the dataset and divide it into 3 parts. The first 25% is used for training the target model, 67.5% for training the shadow models and the last 7.5% is kept for testing. The setup for running this attack on the Purchase datasets are as follows:

1. **Shadow Models** We chose the number of shadow models as 20 for Purchase dataset. The architecture of these shadow models and the size of their training dataset are the same as the target model.

2. **Attack Model** The attack model is multilayer perceptron with a 64-unit hidden layer and a sigmoid output layer. This architecture replicates the precision and recall observed in [33]. We obtain precision of 0.66, 0.78, 0.81, 0.85, 0.89 and recalls of 0.54, 0.57, 0.6, 0.67, 0.76 for Purchase-2, 10, 20, 50, 100, respectively.

CIFAR: We sample complete dataset(around 50000 records) from the dataset and divide it into 3 parts. The first 20% is used for training the target model,next 72% for training the shadow model and the rest 8% is kept for testing purposes. The setup for running this attack on this dataset is as follows:

1. **Shadow Models** We chose the number of attack models as 5 for CIFAR dataset which is the same as [46]. The architecture of this shadow model and the size of the training dataset is the same as the target model.

2. **Attack Model** A multilayer perceptron (two 64 unit hidden layer with "tanh" activation layer and a sigmoid output layer). This architecture matches the precision and recall of the attack model previously reported in [33]. We achieve 0.98 precision and 0.9 recall for CIFAR-100.

C.1.3 Local and Global White Box Inference Attacks [1]

As a result of the federated setting, the target models for our datasets differ. The target models and attack model architecture, as well as the training and testing setup, originally described by [1] are utilized in this study.

Target Model Our target model for both datasets consisted of five layers (1024, 512, 256, 128, 100) with "tanh" activation, replicated from [1]. Each party as well as the server is trained on this model across 100 epochs with an Adam optimizer with learning rate of 0.0001 and cross entropy loss.

Attack Model The attack model takes in a number of different inputs from the target model, which are trained on 'submodules' before being combined in a final network. These inputs described below, with c being equal to the number of classes of the dataset:

- Gradient loss of the final layer - One convolutional layer (1000) with kernel size (1, c) and three hidden layers (1024, 512, 128)
- One hot encoded true label - 2 hidden layers (128, 64)
- Predicted labels - 2 hidden layers (100, 64)
- Output for the correct label – 2 hidden layers (c, 64)

The combined input is trained using three hidden layers (256, 126, 64, 1). "ReLu" activation is used throughout the attack model, with an Adam optimiser with learning rate of 0.00001 and mean square error loss.

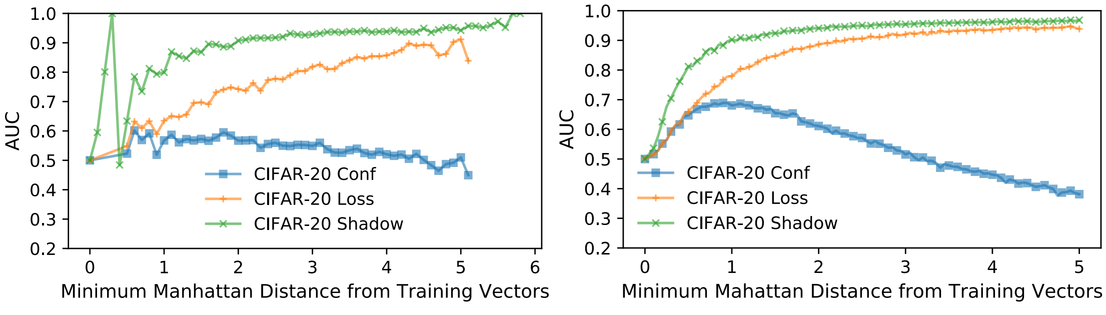
Datasets During target model training the Location and Purchase datasets were both split with 20% (30,000 for Purchase, 1,158 for Location) used for the initial target model training, and 80% (150,000 for Purchase, 5,790 for Location) for testing, as described for the purchase dataset in [1]). The data was further split equally amongst the three parties so that each party had a training and testing set of the same size. The attack model was subsequently trained with half of the original training data and the same amount of the original testing data (representing members and nonmembers, respectively). Each batch was designed to have 50% of members and nonmembers. The remaining samples were used for testing.

C.2 Additional Figures and Experimentation

C.2.1 Additional Plots

CIFAR-20 Plots In Section 5.4, we presented results for CIFAR-100, here we provide accompanying plots in Figure C.1a and C.1b for CIFAR-20, which demonstrates the same trends as those observed in CIFAR-100. We do note that the AUC curves for CIFAR-20 are slightly lower than the respective CIFAR-100 curves. An expected result due to the reduction in the number of class labels.

Per-Label Plots As previously discussed in Section 5.4.1.3, we had only shown the Purchase-20 dataset. We now provide the per-label plots of our remaining binary datasets in Figure C.2.



(a) Original vectors

(b) Generated vectors

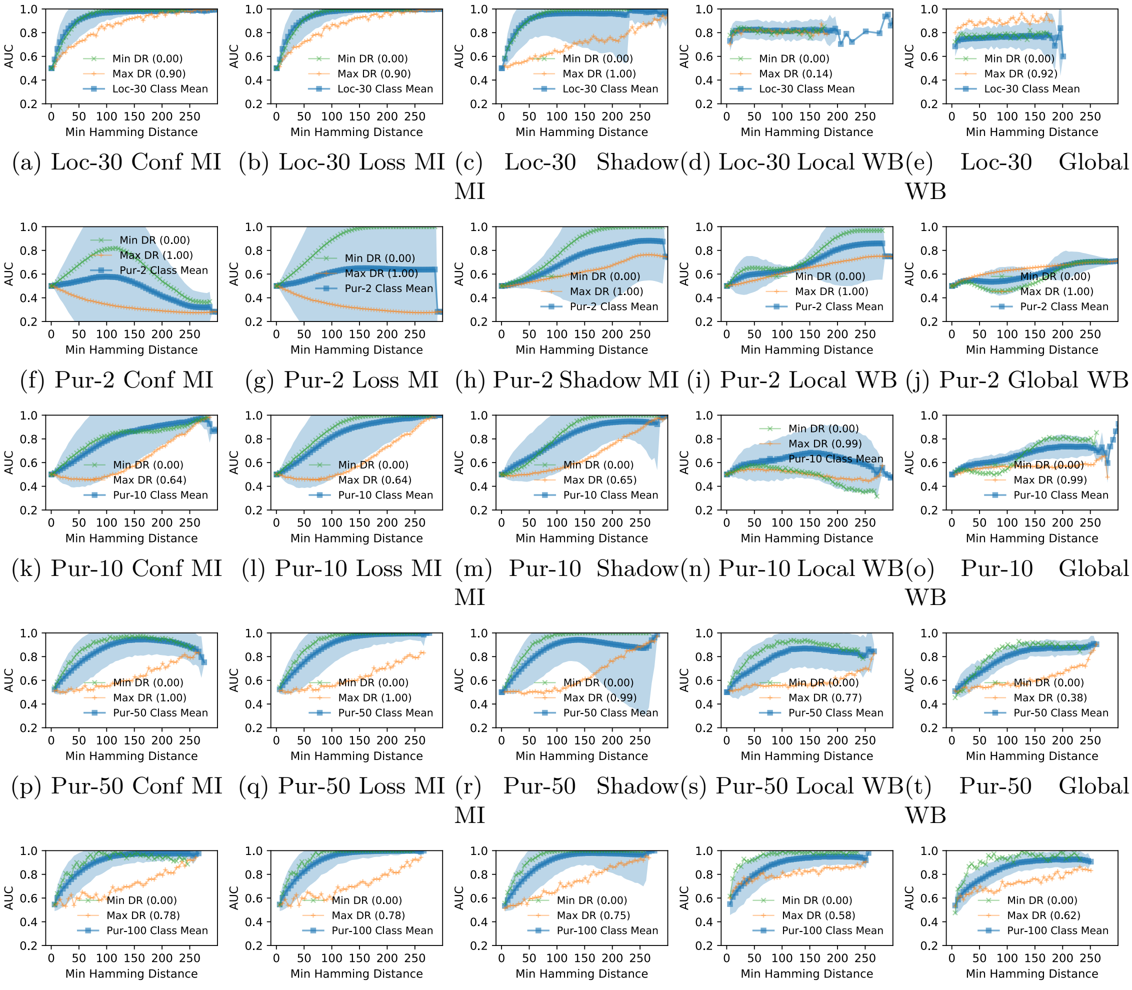
Figure C.1: AUC of MI attacks on original and synthetic non-member vectors of the CIFAR-20 dataset as a function of Manhattan distance.

C.2.2 Validating the Indistinguishable Neighbor Assumption

To demonstrate that the indistinguishable neighbor assumption from Definition 5.2.5 holds for real-world datasets, we train a Generative Adversarial Network (GAN) to produce and discriminate between real and perturbed vectors from the Purchase dataset. We train the GAN over 50 epochs with 90% of the data, and evaluate with the remaining 10%. We use a 100 length noise input to the generator. In Figure C.3, it is clear that at small distances (r-values) there is little advantage in distinguishing between a real vector and a perturbed vector. The advantage increases, and becomes significant, as the distance increases, validating our theoretical assumption.

C.2.3 Exact AI on a Single Missing feature

In this section we present an equivalent AI attack to that in Section 5.5.1, with the exception that only the single most informative feature is to be inferred. Compared to Table 5.1, we see that AI advantages for a single missing feature are better than their counterparts for multiple missing features. This is intuitively clear since with more feature information withheld from an attacker (15 features as in Section 5.5.1), the difficulty of the attack increases, and the likelihood of AI success will decrease. However, when compared to Table C.1, we note that the significant MI performance (in terms of AUC) is not reflected in the AI performance of Table C.2. For a single missing feature, AI is equivalent



(a) Loc-30 Conf MI (b) Loc-30 Loss MI (c) Loc-30 Shadow MI (d) Loc-30 Local WB (e) Loc-30 Global WB

(f) Pur-2 Conf MI (g) Pur-2 Loss MI (h) Pur-2 Shadow MI (i) Pur-2 Local WB (j) Pur-2 Global WB

(k) Pur-10 Conf MI (l) Pur-10 Loss MI (m) Pur-10 Shadow MI (n) Pur-10 Local WB (o) Pur-10 Global WB

(p) Pur-50 Conf MI (q) Pur-50 Loss MI (r) Pur-50 Shadow MI (s) Pur-50 Local WB (t) Pur-50 Global WB

(u) Pur-100 Conf MI (v) Pur-100 Loss MI (w) Pur-100 Shadow MI (x) Pur-100 Local WB (y) Pur-100 Global WB

Figure C.2: Increasing AUC of MIA with increasing distance of synthetic non-members from the training dataset, with a separation of class labels depending on the size of the DR, for the Loc-30, Pur-2, 10, 20, 50, 100 datasets.

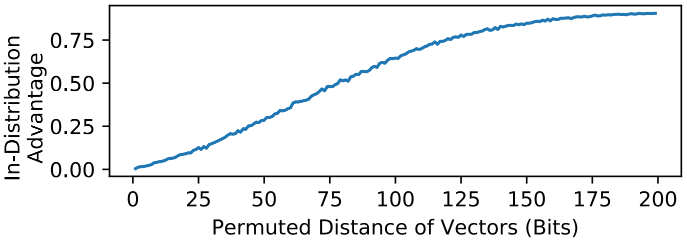


Figure C.3: Advantage of the GAN distinguisher in distinguishing between real and perturbed vectors from the Purchase dataset at increasing distances.

to AAI, since in a binary dataset, with only one missing feature, it is either correct or incorrect. Thus, we only perform AAI for the case of multiple missing features, as is done in Section 5.5.1.

Table C.2: Attribute Inference (Experiment 3) Advantage, where the adversary seeks to infer the exact attribute, when a single most informative feature is missing. The results below are normalized when dealing with ties.

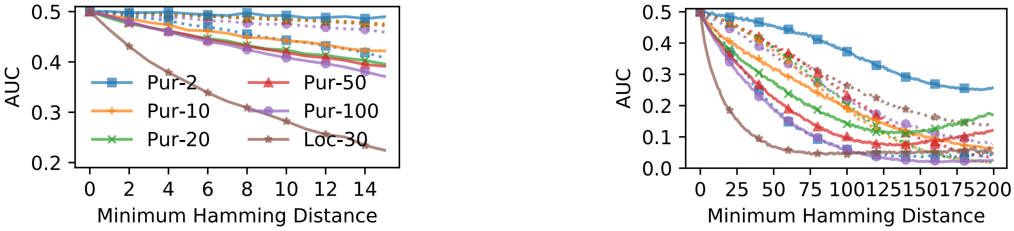
AI	Loc-30	Pur-2	Pur-10	Pur-20	Pur-50	Pur-100
Salem Advantage	0.0700	0.0051	0.0266	0.0396	0.0815	0.0917
Yeom Advantage	0.0581	0.0069	0.0191	0.0294	0.0655	0.0791
Shokri Advantage	0.0377	-0.0057	0.0445	0.0581	0.0318	0.0251

C.2.4 Tuning Attack Models for SMI

It may be argued that these MI attacks are not specifically trained to distinguish between members and nearby (synthetic) non-members, which may explain their poor performance in SMI. To investigate if we can improve their performance of SMI, we tune the training process of these attack models to further include nearby synthetic non-members. This augmented training process is only applicable to the MI attacks that employ an attack model, i.e., Shadow, Local WB, and Global WB. The other two MI attacks, i.e., Conf and Loss MI, directly inspect the outputs of the target model for their MI decision, and hence tuning the decision based on member and nearby synthetic non-member vectors is not applicable.

To perform this experiment we take the same experimental steps as Section 5.4.1.2, select the Shadow MI attack, and augment the tuning step with synthetic non-members generated from both members and non-members of the attack model training set. For each training vector (member or non-member), we generate two synthetic vectors at all Hamming distances up to 10. These synthetic non-members are then used to update the attack model.

From Figure C.4, it can be observed that the AUC of the attack at distances close to the dataset still remains close to 0.5, while at larger distances, the AUC approaches 0,



(a) SMI View

(b) Extended View

Figure C.4: AUC performance on Shadow MI tuned with additional close vectors (dotted lines). The existing Shadow MI results (solid lines) have been mirrored on 0.5 to allow for easier comparison pre and post tuning.

indicating that the attack can distinguish between members and non-members as we move away from the dataset, although with membership label reversed, i.e., more members are now classified as non-members and vice versa. Upon closer inspection, the attack model had no advantage in inferring membership of member vectors (near 0.5 AUC across all datasets). On the other hand, the attack model erred more towards mislabeling non-members (both original and synthetic) as members. We hypothesize this output label “flipping” of the trend is due to the numerous additional close non-members provided to the attack model, which “confuses” the model in distinguishing members from non-members, producing an AUC below 0.5. Regardless, for all datasets tuning the attack model for SMI does not show any improvement in detecting non-members close to the dataset compared to the original attack model. We also carried out an additional repetition of the experiment with one synthetic vector generated per member and non-member, at each Hamming distance up to 50. This demonstrated worse AUC performance over all distances.

We conclude that despite the retraining the attack model with additional nearby non-members, the attack failed to achieve SMI. In fact, MI performance generally decreased, due to the similarity of members and the synthetic nearby non-members.

C.3 Metrics, Balls and Siblings

The results from Section 5.2.2 do not apply to any arbitrary distance metric. For instance, given any distance metric d, the metric $C \cdot d$, where $C > 0$ is a constant is also a distance metric. But this introduces arbitrarily large (artificial) distance between vectors. We, therefore, restrict ourselves to metrics that do not exhibit arbitrarily large deviation given small perturbation in vectors. This leads to the notion of conserving metric [188, §1.6] to be introduced shortly.

Theorem C.3.1 (Metrics)**.** *Let d_1 be a metric on $\mathbb{D}$. Let $\mathbf{x}, \mathbf{x}' \in \mathbb{D}^m$. Then the functions*

1. $d_M(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{m} d_1(x_i, x_i')$,
2. $d_E(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{i=1}^{m} (d_1(x_i, x_i'))^2}$,
3. $d_\infty(\mathbf{x}, \mathbf{x}') = \max_{i \in [m]}(d_1(x_i, x_i'))$,

are metrics on the product space $\mathbb{D}^m$. Moreover, for every $\mathbf{x}, \mathbf{x}' \in \mathbb{D}^m$, we have $d_\infty(\mathbf{x}, \mathbf{x}') \leq d_E(\mathbf{x}, \mathbf{x}') \leq d_M(\mathbf{x}, \mathbf{x}')$ [188, §1.6].

Definition C.3.1 (Conserving metric)**.** *A metric d is called a conserving metric [188, §1.6] on the product space $\mathbb{D}^m$ if for all $\mathbf{x}, \mathbf{x}' \in \mathbb{D}^m$, we have*

$$d_\infty(\mathbf{x}, \mathbf{x}') \leq d(\mathbf{x}, \mathbf{x}') \leq d_M(\mathbf{x}, \mathbf{x}'). \qquad \square$$

Examples of conserving metrics include the Hamming distance over $\mathbb{D}^m = \{0, 1\}^m$, where $d_1(x, x') = |x - x'|$, $x, x' \in \{0, 1\}$, the Euclidean distance over $\mathbb{D}^m = [0, 1]^m$, where $d_1(x, x') = |x - x'|$, $x, x' \in [0, 1]$, and the Manhattan distance (d_M) over $\mathbb{D}^m = [-1, 1]^m$, where $d_1(x, x') = |x - x'|$, $x, x' \in [-1, 1]$. Henceforth we will assume the metric d to be a conserving metric on $\mathbb{D}^m$.

For any subset $X \subseteq \mathbb{D}^m$, the diameter of X, denoted $\mathsf{diam}_d(X)$ is defined as $\max\{d(\mathbf{x}, \mathbf{x}') \mid \mathbf{x}, \mathbf{x}' \in X\}$.

Bounded Feature Space. We assume $\mathbb{D}$ to be bounded, i.e., $\mathsf{diam}_{d_1}(\mathbb{D}) < \infty$. Since d is a conserving metric it follows that $\mathsf{diam}_d(\mathbb{D}^m) < \infty$, and hence the feature space is also bounded. This is equivalent to saying that for any $\mathbf{x} \in \mathbb{D}^m$, there exists an $R > 0$ such that $\mathbb{D}^m = B_d(\mathbf{x}, R)$ [188, §7.1].

Siblings. Overloading notation, we also define

$$\Phi_i(\mathbf{x}) = \bigcup_{\substack{S \subseteq [m] \\ |S| = i}} \Phi_S(\mathbf{x}),$$

where $1 \leq i \leq m - 1$.

Proposition C.3.1. *Let $1 \leq i \leq m - 1$. Let $r \geq i \times \mathsf{diam}_{d_1}(\mathbb{D})$. Then for every feature vector $\mathbf{x} \in \mathbb{D}^m$, we have $\Phi_i(\mathbf{x}) \subseteq B_d(\mathbf{x}, r)$.*

Proof. Consider any $\mathbf{x}' \in \Phi_i(\mathbf{x})$. Then $\mathbf{x}' \in \Phi_S(\mathbf{x})$, for some $S \subseteq [m]$ where $|S| = i$. Then, as d is a conserving metric,

$$\begin{aligned} d(\mathbf{x}, \mathbf{x}') \leq d_M(\mathbf{x}, \mathbf{x}') \leq \sum_{j=1}^{m} d_1(x_j, x_j') &= \sum_{j \in S} d_1(x_j, x_j') \\ \leq \sum_{j \in S} \mathsf{diam}_{d_1}(\mathbb{D}) &= i \times \mathsf{diam}_{d_1}(\mathbb{D}) \leq r. \end{aligned}$$

Hence $\mathbf{x}' \in B(\mathbf{x}, r)$. □

For metrics d_E and d_M, we define d_i to be the restriction of d_E or d_M to i dimensions in a natural way, where $1 \leq i \leq m$.

Proposition C.3.2. *If $\mathsf{diam}_{d_1}(\mathbb{D}) = \delta > 0$, then $\mathsf{diam}_{d_1}(\mathbb{D}) < \mathsf{diam}_{d_2}(\mathbb{D}^2) < \mathsf{diam}_{d_3}(\mathbb{D}^3) < \cdots$.*

Proof. Consider the metric to be d_E. Consider $i = 1$. Then there exist $x, x' \in \mathbb{D}$ such that $\delta = d(x, x')$. Construct the 2-dimensional vectors $\mathbf{x} = (x, x)$ and $\mathbf{x}' = (x', x')$. Then,

$$\begin{aligned} \mathsf{diam}_{d_2}(\mathbb{D}^2) &\geq \sqrt{(d_1(x, x'))^2 + (d_1(x, x'))^2} \\ &= \sqrt{2}\delta > \delta = \mathsf{diam}_{d_1}(\mathbb{D}). \end{aligned}$$

The rest of the proof follows by induction. The case for d_M is similar. □

Proposition C.3.3. *Let $1 \leq i \leq m-1$. Let $\mathsf{diam}_{d_{i+1}}(\mathbb{D}^{i+1}) > r \geq \mathsf{diam}_{d_i}(\mathbb{D}^i)$, where d_j is d_E restricted to j dimensions. Then,*

1. *For any feature vector $\mathbf{x} \in \mathbb{D}^m$, we have $\Phi_i(\mathbf{x}) \subseteq B_{d_E}(\mathbf{x}, r)$.*
2. *There exists a feature vector $\mathbf{x} \in \mathbb{D}^m$, such that $\Phi_{i+1}(\mathbf{x}) \not\subseteq B_{d_E}(\mathbf{x}, r)$.*

Furthermore, the same holds for the metric d_M, and d_j being d_M restricted to j dimensions.

Proof. For part (1), consider any $\mathbf{x}' \in \Phi_i(\mathbf{x})$. Then $\mathbf{x}' \in \Phi_S(\mathbf{x})$, for some $S \subseteq [m]$ where $|S| = i$. Then,

$$\begin{aligned} d_E(\mathbf{x}, \mathbf{x}') &= \sqrt{\sum_{j=1}^{m} (d_1(x_j, x'_j))^2} \\ &= \sqrt{\sum_{j \in S} (d_1(x_j, x'_j))^2} \leq \mathsf{diam}_{d_i}(\mathbb{D}^i) \leq r. \end{aligned}$$

Hence $\mathbf{x}' \in B_{d_E}(\mathbf{x}, r)$. For part (2), let $\delta = \mathsf{diam}_{d_{i+1}}(\mathbb{D}^{i+1})$. Then their exist $(i+1)$-dimensional vectors $\mathbf{x}', \mathbf{x}'' \in \mathbb{D}^{i+1}$ such that $d_{i+1}(\mathbf{x}', \mathbf{x}'') = \delta$. Furthermore, $d_1(x'_j, x''_j) \neq 0$, for all $j \in [i+1]$. Suppose not, and wlog assume that $d_1(x'_{i+1}, x''_{i+1}) = 0$. Then, we can discard the last element from both vectors, and the resulting i-dimensional vectors have distance δ according to d_i, which is greater than $\mathsf{diam}_{d_i}(\mathbb{D}^i)$; a contradiction. Now, sample any $(m-i-1)$-dimensional vector from $\mathbb{D}^{m-i-1}$ and append it to both $\mathbf{x}'$ and $\mathbf{x}''$. Let us call the resulting vectors $\mathbf{x}_1$ and $\mathbf{x}_2$. Let $S = \{1, 2, \ldots, i+1\}$. Then, $|S| = i+1$, and $\mathbf{x}_2 \in \Phi_S(\mathbf{x}_1) \subseteq \Phi_{i+1}(\mathbf{x}_1)$, but

$$\begin{aligned} d_E(\mathbf{x}_1, \mathbf{x}_2) &= \sqrt{\sum_{j=1}^{m} (d_1(x_j, x'_j))^2} \\ &= \sqrt{\sum_{j \in S} (d_1(x_j, x'_j))^2} = \delta > r. \end{aligned}$$

Hence $\mathbf{x}_2 \notin B_{d_E}(\mathbf{x}_1, r)$.

A similar proof holds for the metric d_M. □

Corollary C.3.1.1. *Let i and $\mathbf{x}$ be as in the statement of the previous proposition. Define $d_1(x, x') = |x - x'|$ for $x, x' \in \mathbb{D}$.*

1. *Let d_H be the Hamming distance on $\mathbb{D} = \{0,1\}^m$. Let $r \geq i$. Then $\Phi_i(\mathbf{x}) \subseteq B_{d_H}(\mathbf{x}, r)$.*
2. *Let d_M be the Manhattan distance on $\mathbb{D} = [-1,1]^m$. Let $r \geq 2i$. Then $\Phi_i(\mathbf{x}) \subseteq B_{d_M}(\mathbf{x}, r)$.*
3. *Let d_E be the Euclidean distance on $\mathbb{D} = [-1,1]^m$. Let $r \geq \sqrt{4i}$. Then $\Phi_i(\mathbf{x}) \subseteq B_{d_E}(\mathbf{x}, r)$.*

The above corollary can be used to select an r such that all siblings of a portion are within the r-ball. This is used, for instance, by the AI adversary to employ an SMI attack as a subroutine to infer attributes in Section 5.2.2.

C.4 Relationship between Inference Notions

Proof of Theorem 5.2.1.

Proof. We essentially show that a membership inference (MI) adversary does not imply a strong membership inference (SMI) adversary, i.e., MI $\not\Rightarrow$ SMI. Let $r > 0$ be fixed. Let $k \geq 2$ be a fixed number of labels. Let $N \gg n$. Sample N points from $\mathbb{R}^m$ such that for all pairs of points $\mathbf{x}, \mathbf{x}'$ in this sample, with $\mathbf{x} \neq \mathbf{x}'$, we have $d(\mathbf{x}, \mathbf{x}') > 3r$.[1] Let us call this sample S_1. For each $\mathbf{x} \in S_1$, assign it an arbitrary label from the k labels and set $c(\mathbf{x})$ to this label. Initialize an empty set S_2. Now for each $\mathbf{x} \in S_1$, sample a random point from $B(\mathbf{x}, r) - \{\mathbf{x}\}$, and add to S_2, and assign it the same label as $\mathbf{x}$, i.e., $c(\mathbf{x})$. Let $S = S_1 \cup S_2$. Notice that every vector in S has precisely one r-neighbor in S. To see this, first note that every vector in S_1 is not an r-neighbor of any other vector in S_1 by construction. Next, we take a vector $\mathbf{x}$ in S_1, and see if it has more than one r-neighbors in S_2. Let $\mathbf{y}$

[1] There can be many such vectors, which can be found using a greedy algorithm [189]. For instance, if $\mathbb{D} = \{0,1\}$, $r = 1$, and d is the Hamming distance, then the Gilbert-Varshamov bound states that there are at least $2^m / \sum_{i=0}^{3} \binom{m}{i}$, vectors with minimum Hamming distance $> 3r = 3$ [189, 190].

be the r-neighbor guaranteed by construction. Assume now that $\mathbf{w} \in S_2$ different from $\mathbf{y}$ is another r-neighbor of $\mathbf{x}$. Let $\mathbf{z} \in S_1$ be the r-neighbor of $\mathbf{w}$ in S_1 guaranteed by construction. Then,

$$d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{w}) + d(\mathbf{w}, \mathbf{z}) \Rightarrow d(\mathbf{x}, \mathbf{z}) \leq r + r = 2r,$$

a contradiction. Next, we will look at vectors in S_2. We will check if any vector from S_2 has more than one r-neighbor in S_1. Then, we will check if the vectors in S_2 have any r-neighbors in S_2. This exhausts the cases.

Let $\mathbf{y}$ be the r-neighbor in S_2 of some $\mathbf{x} \in S_1$. This is true by construction. Let $\mathbf{z}$ be some other vector in S_1. Then, $d(\mathbf{x}, \mathbf{y}) \leq r$, and $d(\mathbf{x}, \mathbf{z}) > 3r$. Therefore,

$$\begin{aligned} d(\mathbf{x}, \mathbf{z}) &\leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \\ &\Rightarrow 3r < d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \\ &\Rightarrow 3r < r + d(\mathbf{y}, \mathbf{z}) \Rightarrow 2r < d(\mathbf{y}, \mathbf{z}), \end{aligned}$$

hence $\mathbf{y}$ is not an r-neighbor of any other $\mathbf{z}$ in S_1. Now consider some $\mathbf{w} \in S_2$ not equal to $\mathbf{y}$. Assume to the contrary that $d(\mathbf{y}, \mathbf{w}) \leq r$. Let $\mathbf{z}$ be the r-neighbor of $\mathbf{w}$ in S_1 (again by construction, it should exist). Then,

$$\begin{aligned} d(\mathbf{x}, \mathbf{z}) &\leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{w}) + d(\mathbf{y}, \mathbf{z}) \\ \Rightarrow d(\mathbf{x}, \mathbf{z}) &\leq r + r + r = 3r, \end{aligned}$$

which is a contradiction.

Let $\mathbb{D}^m = S$. Define the distribution $\mathcal{D}$ as the uniform distribution over S. Sample a dataset $X \leftarrow \mathcal{D}^n$. Define a classifier h_X which given a point $\mathbf{x}$ in X, assigns its label $c(\mathbf{x})$ to all vectors within the ball $B(\mathbf{x}, r)$, i.e., all r-neighbors of $\mathbf{x}$ have the constant label. The classifier h_X, when queried for a point $\mathbf{x} \in X$, simply outputs the label $c(\mathbf{x})$. For any point $\mathbf{x} \notin X$, it checks if there is some $\mathbf{x}' \in X$ such that $d(\mathbf{x}', \mathbf{x}) \leq r$. If yes, it returns the label $c(\mathbf{x}')$. Otherwise, it returns an arbitrary label from the k labels.

Now consider an MI adversary $\mathcal{A}$ which given $(\mathbf{x}, c(\mathbf{x}))$, queries h_X with $\mathbf{x}$, and outputs 1 (member) if $h_X(\mathbf{x}) = c(\mathbf{x})$ and 0 (non-member) otherwise. Let us calculate the probabilities in:

$$\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0],$$

which define the adversary's advantage (Definition 5.2.7). If $\mathbf{x}$ is a member, then the adversary does not make a mistake, as the label returned by h_X is exactly the label $c(\mathbf{x})$ by construction. Therefore,

$$\Pr[b' = 1 \mid b = 1] = 1.$$

Now consider the other probability, i.e., $\Pr[b' = 1 \mid b = 0]$. The adversary could erroneously output $\mathbf{x}$ as a member either if its r-neighbor was in X, or if its r-neighbor was not part of X, but the classifier gives it the correct label by chance. Thus

$$\begin{aligned}\Pr[b' = 1 \mid b = 0] &= \left(1 - \left(\frac{2N-2}{2N-1}\right)^n\right) \\ &+ \left(\frac{2N-2}{2N-1}\right)^n \left(\frac{1}{k}\right) \\ &= 1 - \left(1 - \frac{1}{2N-1}\right)^n \left(\frac{k-1}{k}\right)\end{aligned}$$

Subtracting this from the above, we see that the advantage is

$$\left(1 - \frac{1}{2N-1}\right)^n \left(\frac{k-1}{k}\right)$$

By Bernoulli's inequality [191], we have

$$\left(1 - \frac{1}{2N-1}\right)^n \geq 1 - \frac{n}{2N-1},$$

and noting that $N > n$, we get $2N - 1 \geq 2n$. And therefore,

$$1 - \frac{n}{2N-1} \geq 1 - \frac{n}{2n} = \frac{1}{2}.$$

Finally, we get the advantage of at least $\frac{1}{2}\frac{k-1}{k}$, which is a constant.[2] However, the same adversary if used as a subroutine in Experiment 2, will always output 1 if queried on $\mathbf{x}$ and its r-neighbor, since every r-neighbor of a member $\mathbf{x} \in X$, is assigned the true label (even if it is not in X, by construction). Hence, the resulting adversary has no advantage in the sense of SMI. □

[2] Note that if the adversary just guesses randomly, the advantage is 0. This is significantly greater than 0.

C.5 Miscellaneous Results

Relationship between AUC and Advantage. The MI advantage from Definition 5.2.7 denoted $\text{Adv}_{\text{MI}}(\mathcal{A}, h_X, n, \mathcal{D})$ can be empirically estimated as $\text{TPR}(\tau) - \text{FPR}(\tau)$[3] with τ denoting the threshold parameter of the given classifier h_X and $\text{TPR}(\tau)$ and $\text{FPR}(\tau)$ denoting the True Positive Rate and False Positive Rate respectively at τ. The AUC-ROC statistic captures the aggregate performance of the classifier h_X for all possible values of the threshold τ and is computed as $\text{AUC} = \int_{\text{FPR}(\tau)=0}^{1} \text{TPR}(\tau) d(\text{FPR}(\tau)) = \int_{x=0}^{1} \text{TPR}(\text{FPR}^{-1}(x)) dx$.

When $\text{Adv}_{\text{MI}}(\mathcal{A}, h_X, n, \mathcal{D})) = \text{Adv}_m$ for all possible values of τ (*i.e.* Advantage is same for all values of the threshold parameter), the AUC is computed as $\int_{x=0}^{1}(\text{FPR}(\text{FPR}^{-1}(x)) + \text{Adv}_m) dx = \frac{1}{2} + \text{Adv}_m$. Thus, $\text{AUC} - \frac{1}{2}$ equals the advantage from Definition 5.2.7. Even when the advantages vary with τ, $\text{AUC} - \frac{1}{2}$ is a good approximation for the average advantage.

Similarly, the Advantage in the strong membership inference definition, $\text{Adv}_{\text{SMI}}(\mathcal{A}, h_X, r, n, \mathcal{D})$ can be empirically estimated as $\text{TPR}(\tau) - \text{FPR}(\tau)$ as long as $B_d(\mathbf{x}_0, r)$ is assumed to have a small number of samples from X, i.e., in general $B_d(\mathbf{x}_0, r)$ would contain more elements outside of X.

Average Manhattan Distance. Let $\mathbb{D}^m = [-1, 1]^m$. Given a vector $\mathbf{x} \in \mathbb{D}^m$, we want to find the Manhattan distance d_M between $\mathbf{x}$ and a vector $\mathbf{y} \in \mathbb{D}^m$, each of whose elements is sampled uniformly at random from the set $\mathbb{D} = [-1, 1]$. Define the distance as α_m. Consider first $m = 1$. Then, α_1, the expected Manhattan distance between x and y, can be defined as

$$\alpha_1 = \frac{1}{R} \int_{-1}^{+1} \int_{-1}^{+1} |x - y| \, dx \, dy,$$

[3] i.e., $\Pr[b' = 1 \mid b = 1] = \frac{\Pr[b'=1 \wedge b=1]}{\Pr[b=1]} = \text{TPR}$ and $\Pr[b' = 1 \mid b = 0] = \frac{\Pr[b'=1 \wedge b=0]}{\Pr[b=0]} = \text{FPR}$

where $R = 4$ is the area of the square $[-1, 1] \times [-1, 1]$. Integrating the above we get,

$$\begin{aligned}\alpha_1 &= \frac{1}{4}\int_{-1}^{+1}\left(\int_{-1}^{y}(y-x)\,dx + \int_{y}^{+1}(x-y)\,dx\right)\,dy\\ &= \frac{1}{4}\int_{-1}^{+1}(y^2+1)\,dy = \frac{1}{4}\cdot\frac{8}{3} = \frac{2}{3}.\end{aligned}$$

By independence, we get $\alpha_m = m\alpha_1 = 2m/3$. For $m = 5$, we get $\alpha_5 = 10/3 \approx 3.33$. Thus, we set $\alpha = 3.33$ as the benchmark for a random guess with 5 missing features in the CIFAR dataset.

Appendix D

Supplementary material for Chapter 6 - Not one but many Tradeoffs: Privacy Vs. Utility in Differentially Private Machine Learning

D.1 Machine Learning algorithm training configurations

The hyper-parameters of the *NN* models were replicated from [46]. For *RF*, the number of trees was fixed at 100, and the maximum tree depth was capped at 15 (due to memory limits in [161]). For *LR*, the solver was 'lbfgs'; multi-class classification problems were handled in a one vs. rest manner. All other models' parameters are kept at library defaults.

D.2 Experimental Results for *LossMI* on Synthetic Data

This section contains privacy advantage plots for the *LossMI* attack when applied on synthetic data with various ML methods. From Figure D.1, we can observe that like *ConfAI*, the inflection point for S1 DP-ML models occurs at $\epsilon \approx 100$, and for S3 at $\epsilon \approx 1$-10. However unlike *ConfAI*, the inflection point on S2 is much more pronounced, clearly occurring at $\epsilon = 1$ and $\epsilon = 100$ for *RF* and *NN* respectively. Across all DP-ML methods, the absolute advantage (when the attack is effective) tends to be higher than that of *ConfMI*, an expected result given access to ground truth information about the class label.

D.3 Experimental Results for *ConfAI* on Synthetic Data

This section contains privacy advantage plots for the *ConfAI* attack when applied on synthetic data with various ML methods. Figure D.2 allows us to view trends similar to those mentioned when analyzing *LossAI*. Interestingly, it appears that both *ConfMI* and *ConfAI* obtain a poor advantage for an attacker for the models in S2, due to the *ConfAI* attack relying on *ConfMI* as a subroutine.

D.4 Experimental Results for *LossMI* on Real-world Data

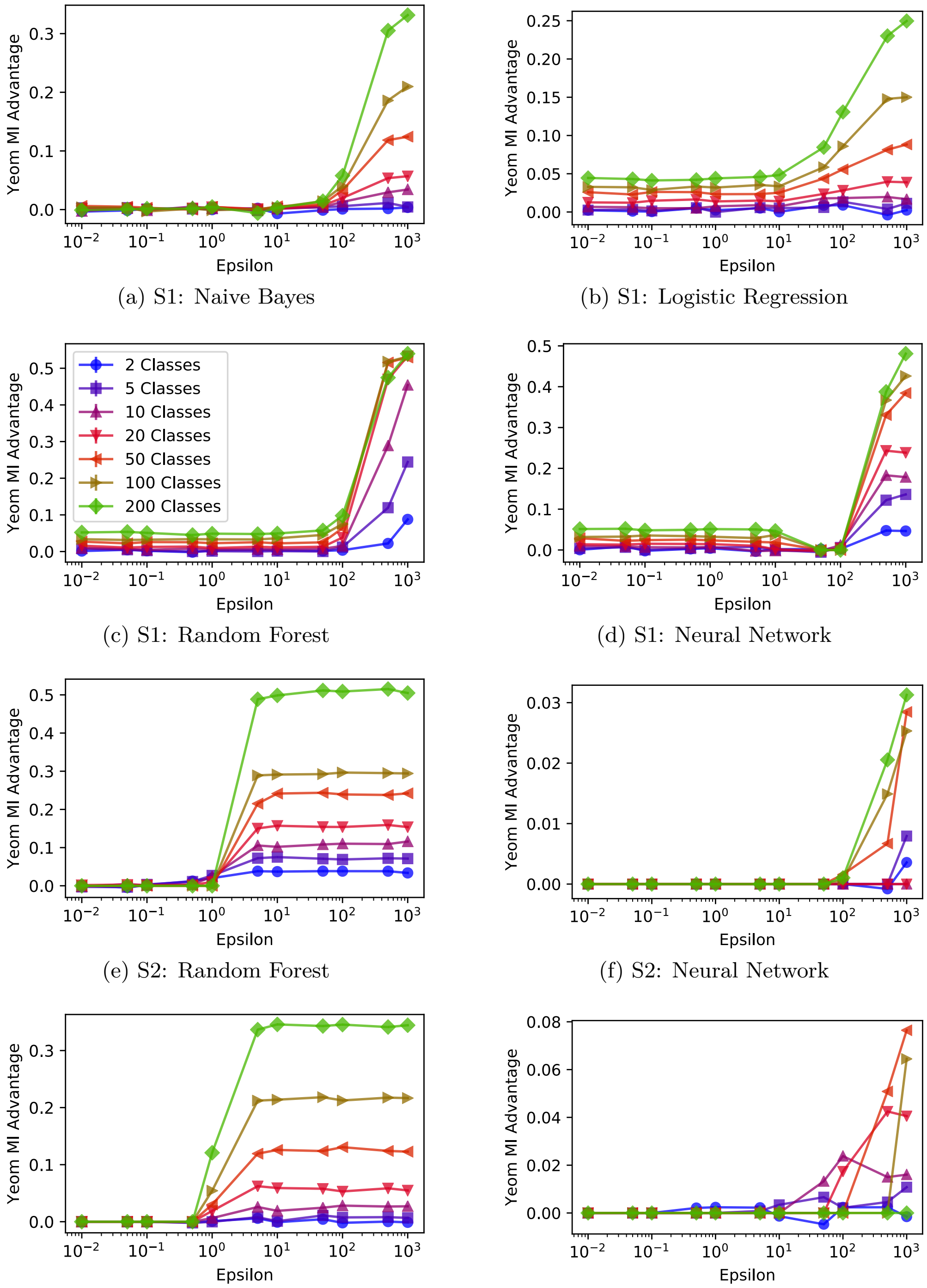
This section contains privacy advantage plots for the *LossMI* attack when applied on real-world data with various ML methods. The trends we observe in Figure D.3 are reminiscent of the trends observed in the synthetic results seen in Figure D.1. Again we can observe that like *ConfAI*, the inflection point for S1 DP-ML models occurs at $\epsilon\approx100$, and for S3 at $\epsilon\approx1$-10. However, unlike *ConfAI* on the real world data, the inflection point on S2:*NN* is much more pronounced, clearly occurring at $\epsilon=100$. As expected, across all DP-ML methods, the absolute advantage (when the attack is effective) tends to be higher than that of *ConfMI*.

D.5 Experimental Results for *ConfAI* on Real-world Data

This section contains privacy advantage plots for the *ConfAI* attack when applied on real-world data with various ML methods. Figure D.4 contains our results. The *ConfAI* attack observes inflection points for S1 at $\epsilon\approx1$. For S3, these inflection points occur at $\epsilon=1$ and 10, for *NB* and *LR*, respectively. Similar to what we observed in the synthetic results, we see that the *ConfAI* advantage on S2 models is near zero, which is not the case in *LossAI*.

D.6 Summaries of Experimental Results for *LossMI*, *LossAI*, *ConfAI*

In this Section, and as previously observed in Section 6.4.2.5, we present the additional summary figures for *LossMI*, *LossAI* and *ConfAI* (Figure D.5).



(a) S1: Naive Bayes

(b) S1: Logistic Regression

(c) S1: Random Forest

(d) S1: Neural Network

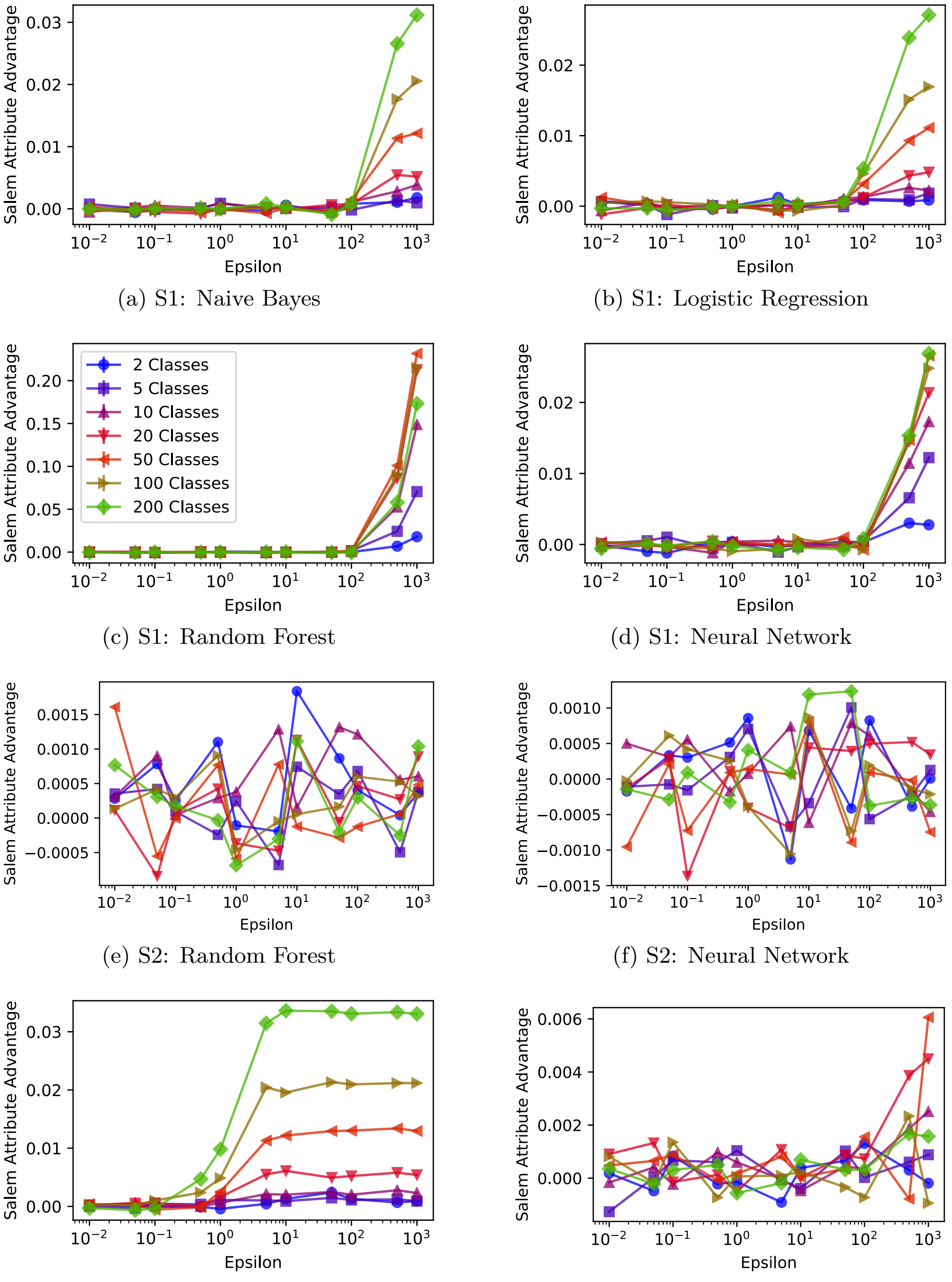
(e) S2: Random Forest

(f) S2: Neural Network

(g) S3: Naive Bayes

(h) S3: Logistic Regression

Figure D.1: Advantage of *LossMI* attack for each of the ML methods used, when different amount of *DP* noise is applied at Stage 1, 2, 3 of the ML pipeline, and for different synthetic datasets.



(a) S1: Naive Bayes

(b) S1: Logistic Regression

(c) S1: Random Forest

(d) S1: Neural Network

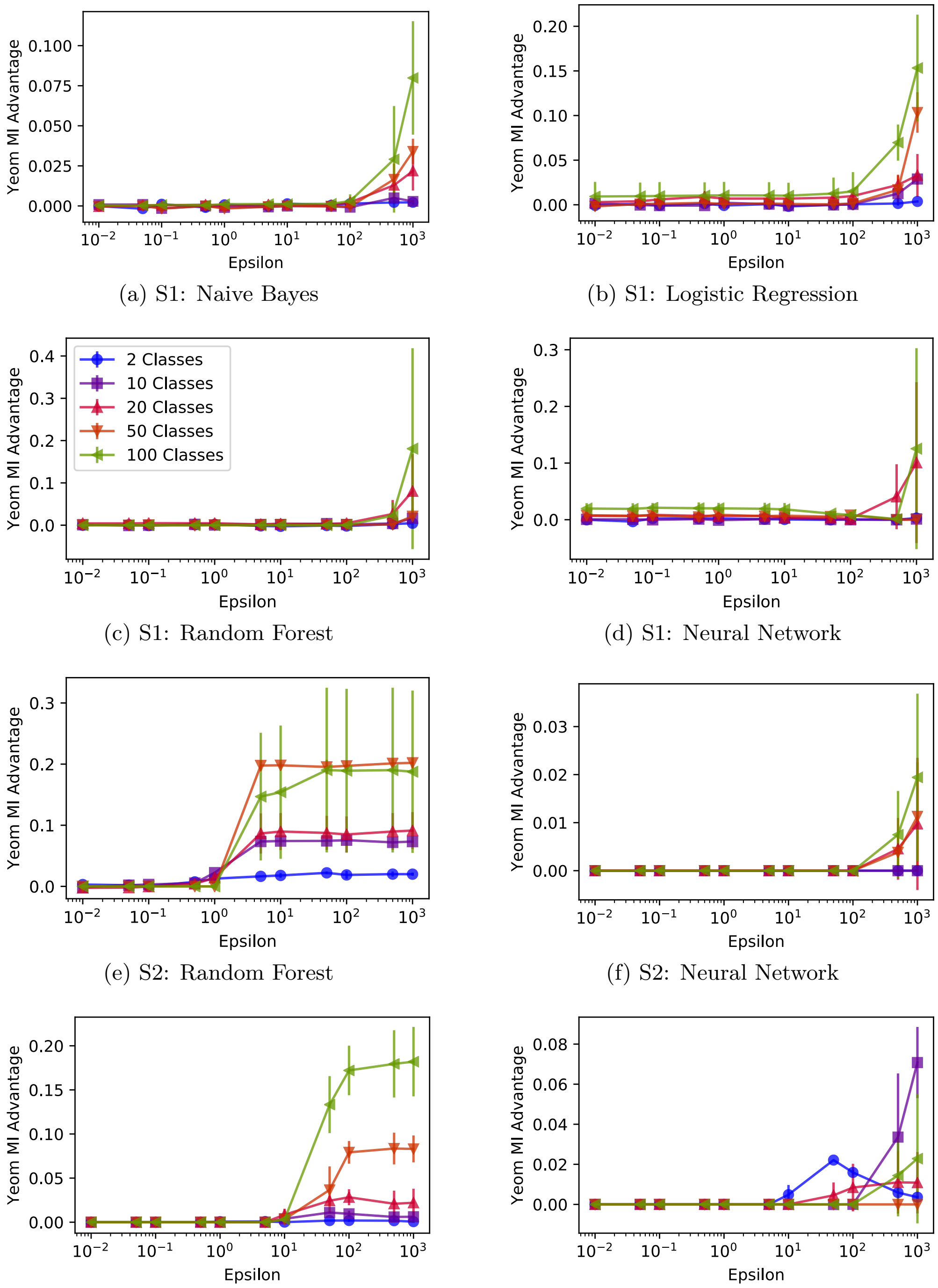
(e) S2: Random Forest

(f) S2: Neural Network

(g) S3: Naive Bayes

(h) S3: Logistic Regression

Figure D.2: Advantage of Zhao et al.'s Attribute inference attack (*ConfAI*) for each of the ML methods used, when different amount of *DP* noise is applied at Stage 1, 2 and 3 of the ML pipeline, and for synthetic datasets used.



(a) S1: Naive Bayes

(b) S1: Logistic Regression

(c) S1: Random Forest

(d) S1: Neural Network

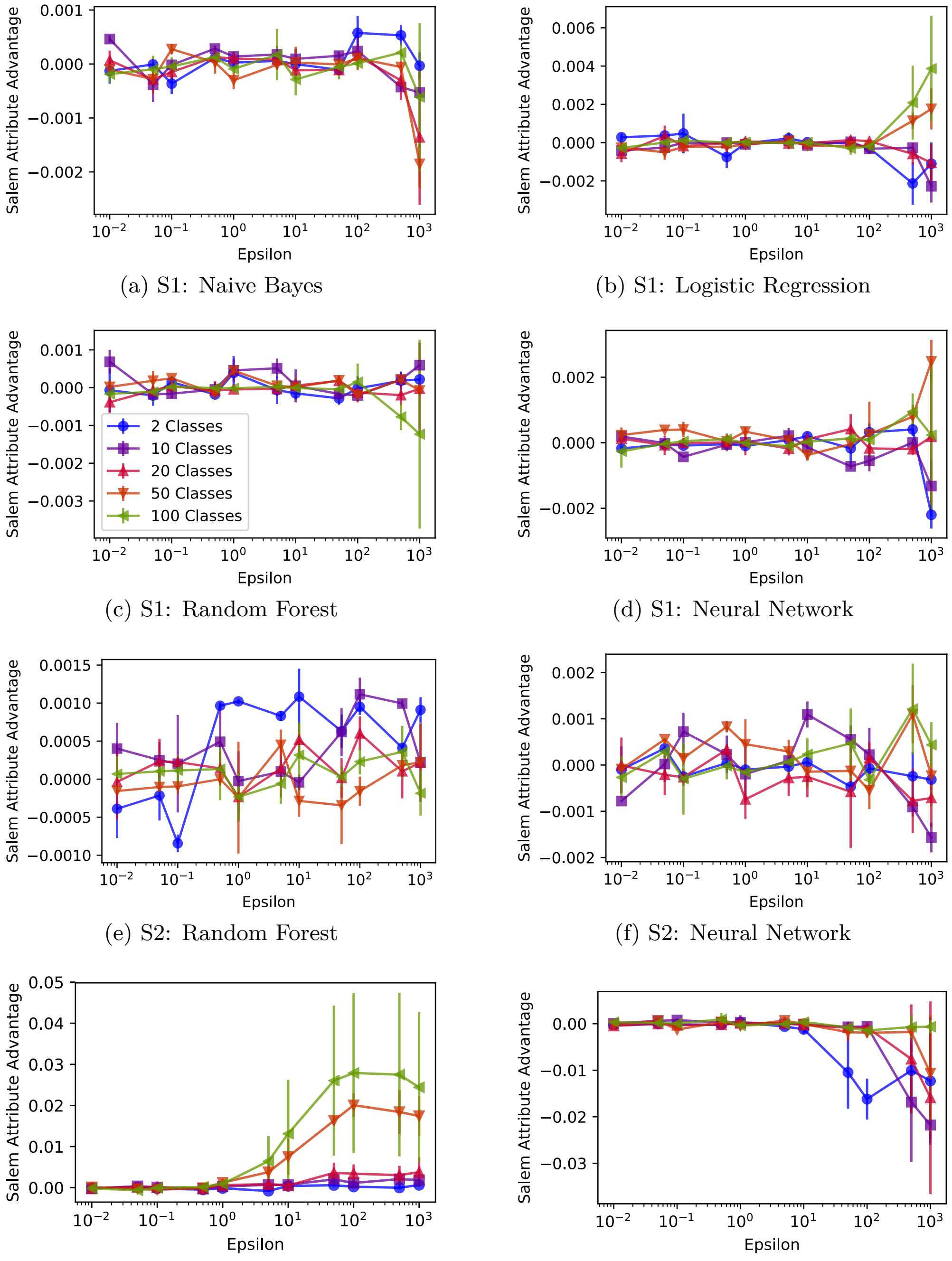
(e) S2: Random Forest

(f) S2: Neural Network

(g) S3: Naive Bayes

(h) S3: Logistic Regression

Figure D.3: Advantage of *LossMI* attack for each of the ML methods, when different amount of *DP* noise is applied at Stage 1, 2, 3 of the pipeline, and for different real datasets. We summarize the datasets by the number of classes used.



(a) S1: Naive Bayes

(b) S1: Logistic Regression

(c) S1: Random Forest

(d) S1: Neural Network

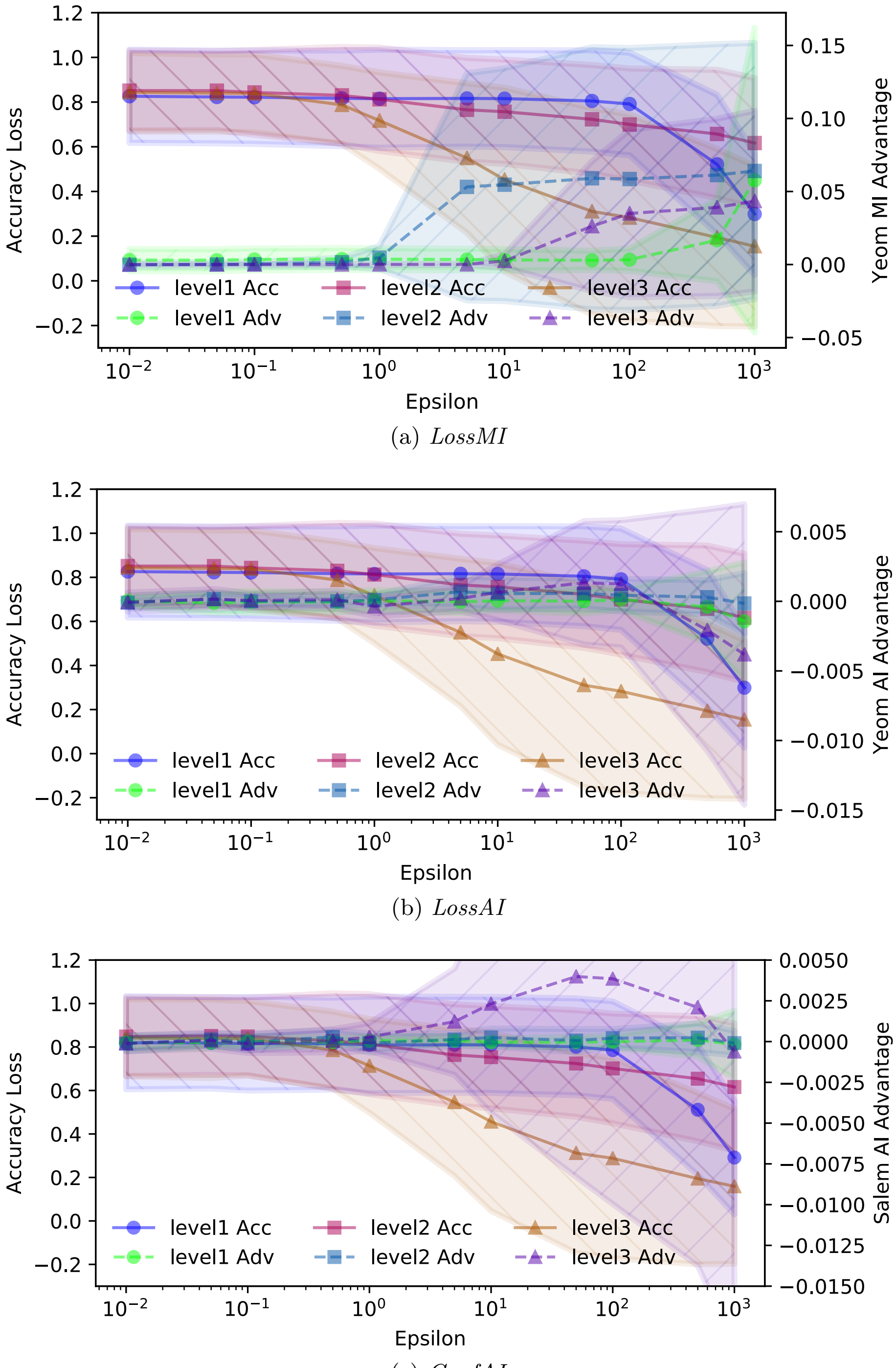
(e) S2: Random Forest

(f) S2: Neural Network

(g) S3: Naive Bayes

(h) S3: Logistic Regression

Figure D.4: Advantage of Zhao et al.'s AI attack (*ConfAI*) for each ML method used, when different amount of *DP* noise is applied at Stage 1, 2 and 3 of the pipeline, for different real datasets. We summarize the datasets by the number of classes used.



(a) *LossMI*

(b) *LossAI*

(c) *ConfAI*

Figure D.5: Summary plot of accuracy loss (y1-axis) and privacy advantage (y2-axis) vs. ϵ applied (x-axis), for each pipeline Stage. Each point, for a line of a given Stage, is the mean across all results we have for different ML methods and real-world datasets. Shaded colored areas signify 1 standard deviation around each mean.

References

[1] M. Nasr, R. Shokri, and A. Houmansadr, “Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks,” *arXiv preprint arXiv:1812.00910*, 2018.

[2] J. Chauhan, B. Z. H. Zhao, H. J. Asghar, J. Chan, and M. A. Kaafar, “Behaviocog: An observation resistant authentication scheme,” in *International Conference on Financial Cryptography and Data Security.* Springer, 2017, pp. 39–58.

[3] H. J. Asghar, R. Steinfeld, S. Li, M. A. Kaafar, and J. Pieprzyk, “On the linearization of human identification protocols: Attacks based on linear algebra, coding theory, and lattices,” *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 8, pp. 1643–1655, 2015.

[4] “The 2019 official annual cybercrime report,” Dec. 2020. [Online]. Available: https://www.herjavecgroup.com/the-2019-official-annual-cybercrime-report/

[5] K. Security, “Hidden cameras on automated teller machines (atms),” https://krebsonsecurity.com/tag/atm-skimmer/, accessed: 2018-11-30.

[6] G. T. Wilfong, “Method and apparatus for secure pin entry,” U.S. Patent 5 940 511, Aug., 1999.

[7] N. J. Hopper and M. Blum, “Secure human identification protocols,” in *International conference on the theory and application of cryptology and information security.* Springer, 2001, pp. 52–66.

[8] S. Li and H. Y. Shum, "Sechci: Secure human-computer identification (interface) systems against peeping attacks," *Computer Science Preprint Archive*, vol. 2004. Issue 3, pp. 21–69, 2004.

[9] H. J. Asghar, S. Li, R. Steinfeld, and J. Pieprzyk, "Does counting still count? revisiting the security of counting based user authentication protocols against statistical attacks," in *Proceedings of the 20th Annual Network and Distributed System Security Symposium (NDSS 2013).* The Internet Society 2013, 2013, pp. 1–18.

[10] T. Matsumoto, "Human–computer cryptography: An attempt," *Journal of Computer Security*, vol. 6, no. 3, pp. 129–149, 1998.

[11] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, L. Bauer, N. Christin, and L. F. Cranor, "The impact of pattern length, pattern compactness, and mathematical operators on the usability and security of system-assigned graphical one-time pins," in *International Conference on Financial Cryptography and Data Security*, 2013, pp. 34–51.

[12] H. J. Asghar, J. Pieprzyk, and H. Wang, "A new human identification protocol and coppersmith's baby-step giant-step algorithm," in *International Conference on Applied Cryptography and Network Security.* Springer, 2010, pp. 349–366.

[13] S. Wiedenbeck, J. Waters, L. Sobrado, and J.-C. Birget, "Design and evaluation of a shoulder-surfing resistant graphical password scheme," in *The working conference on Advanced visual interfaces.* ACM, 2006, pp. 177–184.

[14] D. LeBlanc, A. Forget, and R. Biddle, "Guessing click-based graphical passwords by eye tracking," in *Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on.* IEEE, 2010, pp. 197–204.

[15] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of fingerprint recognition.* Springer Science & Business Media, 2009.

[16] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," *arXiv preprint arXiv:1411.7923*, 2014.

[17] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[18] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," *arXiv preprint arXiv:1706.08612*, 2017.

[19] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," *arXiv preprint arXiv:1806.05622*, 2018.

[20] U. Mahbub, S. Sarkar, V. M. Patel, and R. Chellappa, "Active user authentication for smartphones: A challenge data set and benchmark results," in *Biometrics Theory, Applications and Systems (BTAS), 2016 IEEE 8th International Conference on.* IEEE, 2016, pp. 1–8.

[21] W. Xu, G. Lan, Q. Lin, S. Khalifa, N. Bergmann, M. Hassan, and W. Hu, "Keh-gait: Towards a mobile healthcare user authentication system by kinetic energy harvesting." in *NDSS*, 2017.

[22] M. T. Curran, N. Merrill, J. Chuang, and S. Gandhi, "One-step, three-factor authentication in a single earpiece," in *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers.* ACM, 2017, pp. 21–24.

[23] C. Huang, H. Chen, L. Yang, and Q. Zhang, "Breathlive: Liveness detection for heart sound authentication with deep breathing," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 1, p. 12, 2018.

[24] R. Liu, C. Cornelius, R. Rawassizadeh, R. Peterson, and D. Kotz, "Vocal resonance: Using internal body voice for wearable authentication," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 1, p. 19, 2018.

[25] Y. Chen, J. Sun, X. Jin, T. Li, R. Zhang, and Y. Zhang, "Your face your heart: Secure mobile face authentication with photoplethysmograms," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications.* IEEE, 2017, pp. 1–9.

[26] C. Song, A. Wang, K. Ren, and W. Xu, “Eyeveri: A secure and usable approach for smartphone user authentication,” in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications.* IEEE, 2016, pp. 1–9.

[27] J. Chauhan, Y. Hu, S. Seneviratne, A. Misra, A. Seneviratne, and Y. Lee, “Breathprint: Breathing acoustics-based user authentication,” in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services.* ACM, 2017, pp. 278–291.

[28] J. Ho and D.-K. Kang, “Mini-batch bagging and attribute ranking for accurate user authentication in keystroke dynamics,” *Pattern Recognition*, vol. 70, pp. 139–151, 2017.

[29] H. Crawford and E. Ahmadzadeh, “Authentication on the go: assessing the effect of movement on mobile device keystroke dynamics,” in *Thirteenth Symposium on Usable Privacy and Security ({SOUPS} 2017)*, 2017, pp. 163–173.

[30] E. Pagnin, C. Dimitrakakis, A. Abidin, and A. Mitrokotsa, “On the leakage of information in biometric authentication,” in *International Conference in Cryptology in India.* Springer, 2014, pp. 265–280.

[31] M. Martinez-Diaz, J. Fierrez-Aguilar, F. Alonso-Fernandez, J. Ortega-García, and J. Siguenza, “Hill-climbing and brute-force attacks on biometric systems: A case study in match-on-card fingerprint verification,” in *Proceedings 40th Annual 2006 International Carnahan Conference on Security Technology.* IEEE, 2006, pp. 151–159.

[32] J. Galbally, C. McCool, J. Fierrez, S. Marcel, and J. Ortega-Garcia, “On the vulnerability of face verification systems to hill-climbing attacks,” *Pattern Recognition*, vol. 43, no. 3, pp. 1027–1038, 2010.

[33] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in *2017 IEEE Symposium on Security and Privacy (SP).* IEEE, 2017, pp. 3–18.

[34] A. Salem, Y. Zhang, M. Humbert, M. Fritz, and M. Backes, “Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models,” *NDSS*, 2019.

[35] 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)*. IEEE, 2018, pp. 268–282.

[36] M. Yaghini, B. Kulynych, and C. Troncoso, “Disparate vulnerability: on the unfairness of privacy attacks against machine learning,” *arXiv preprint arXiv:1906.00389*, 2019.

[37] M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1322–1333.

[38] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, “Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing,” in *23rd USENIX Security Symposium*, 2014, pp. 17–32.

[39] “Children’s Online Privacy Protection Rule (COPPA),” https://www.ftc.gov/enforcement/rules/rulemaking-regulatory-reform-proceedings/childrens-online-privacy-protection-rule, 1998, accessed: 2020-02-14.

[40] “General Data Protection Regulation (GDPR),” https://gdpr-info.eu/, 2018, accessed: 2020-02-14.

[41] “Proposal for an ePrivacy Regulation,” https://ec.europa.eu/digital-single-market/en/proposal-eprivacy-regulation, 2019, accessed: 2020-02-14.

[42] “California Consumer Privacy Act (CCPA),” https://oag.ca.gov/privacy/ccpa, 2020, accessed: 2020-02-14.

[43] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, “Our data, ourselves: Privacy via distributed noise generation,” in *Annual International Conference*

on the Theory and Applications of Cryptographic Techniques. Springer, 2006, pp. 486–503.

[44] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[45] J. Konecný, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *CoRR*, vol. abs/1511.03575, 2015.

[46] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *USENIX*, 2019.

[47] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[48] "Acquire valued shoppers challenge - kaggle," https://www.kaggle.com/c/acquire-valued-shoppers-challenge/data, 2014, accessed: 2020-01-30.

[49] "Netflix prize dataset - kaggle," https://www.kaggle.com/netflix-inc/netflix-prize-data, 2006, accessed: 2020-01-30.

[50] Q. Yan, J. Han, Y. Li, and R. H. Deng, "On limitations of designing leakage-resilient password systems: Attacks, principles and usability," in *The 19th Annual Network and Distributed System Security Symposium.* Citeseer, 2012.

[51] M. Čagalj, T. Perković, and M. Bugarić, "Timing attacks on cognitive authentication schemes," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 584–596, 2015.

[52] S. Li and H.-Y. Shum, "Secure human-computer identification (interface) systems against peeping attacks: Sechci," 2005.

[53] S. Y. Ooi, A. B. J. Teoh, Y. H. Pang, and B. Y. Hiew, "Image-based handwritten signature verification using hybrid methods of discrete radon transform, principal component analysis and probabilistic neural network," *Applied Soft Computing*, vol. 40, pp. 274–282, 2016.

[54] M. Diaz, A. Fischer, M. A. Ferrer, and R. Plamondon, “Dynamic signature verification system based on one real signature,” *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 228–239, 2018.

[55] A. Hadid, N. Evans, S. Marcel, and J. Fierrez, “Biometrics systems under spoofing attack: an evaluation methodology and lessons learned,” *IEEE Signal Processing Magazine*, vol. 32, no. 5, pp. 20–30, 2015.

[56] M. Yang, L. Zhang, J. Yang, and D. Zhang, “Metaface learning for sparse representation based face recognition,” in *Image Processing (ICIP), 2010 17th IEEE International Conference on.* IEEE, 2010, pp. 1601–1604.

[57] S. Li, A. Ashok, Y. Zhang, C. Xu, J. Lindqvist, and M. Gruteser, “Whose move is it anyway? authenticating smart wearable devices using unique head movement patterns,” in *Pervasive Computing and Communications (PerCom), 2016 IEEE International Conference on.* IEEE, 2016, pp. 1–9.

[58] R. Kumar, V. V. Phoha, and A. Serwadda, “Continuous authentication of smartphone users by fusing typing, swiping, and phone movement patterns,” in *Biometrics Theory, Applications and Systems (BTAS), 2016 IEEE 8th International Conference on.* IEEE, 2016, pp. 1–8.

[59] T. Feng, Z. Liu, K.-A. Kwon, W. Shi, B. Carbunar, Y. Jiang, and N. Nguyen, “Continuous mobile authentication using touchscreen gestures,” in *Homeland Security (HST), 2012 IEEE Conference on Technologies for.* Citeseer, 2012, pp. 451–456.

[60] S. Krishnamoorthy, L. Rueda, S. Saad, and H. Elmiligi, “Identification of user behavioral biometrics for authentication using keystroke dynamics and machine learning,” in *Proceedings of the 2018 2nd International Conference on Biometric Engineering and Applications.* ACM, 2018, pp. 50–57.

[61] A. Serwadda and V. V. Phoha, “When kids’ toys breach mobile phone security,” in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security.* ACM, 2013, pp. 599–610.

[62] T. Eude and C. Chang, "One-class svm for biometric authentication by keystroke dynamics for remote evaluation," *Computational Intelligence*, vol. 34, no. 1, pp. 145–160, 2018.

[63] P. Abeni, M. Baltatu, and R. D'Alessandro, "Nis03-4: Implementing biometrics-based authentication for mobile devices," in *Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE.* IEEE, 2006, pp. 1–5.

[64] B. Yegnanarayana, *Artificial neural networks.* PHI Learning Pvt. Ltd., 2009.

[65] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[66] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning.* MIT press Cambridge, 2016, vol. 1, no. 2.

[67] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[68] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[69] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[70] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, 2012, pp. 1467–1474.

[71] S. Li, M. Xue, B. Zhao, H. Zhu, and X. Zhang, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[72] S. Li, H. Liu, T. Dong, B. Z. H. Zhao, M. Xue, H. Zhu, and J. Lu, "Hidden backdoors in human-centric language models," *arXiv e-prints*, pp. arXiv–2105, 2021.

[73] X. Yuan, P. He, Q. Zhu, R. R. Bhat, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *arXiv preprint arXiv:1712.07107*, 2017.

[74] J. Gilmer, R. P. Adams, I. Goodfellow, D. Andersen, and G. E. Dahl, "Motivating the rules of the game for adversarial example research," *arXiv preprint arXiv:1807.06732*, 2018.

[75] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia conference on computer and communications security.* ACM, 2017, pp. 506–519.

[76] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[77] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Adversarial generative nets: Neural network attacks on state-of-the-art face recognition," *arXiv preprint arXiv:1801.00349*, 2017.

[78] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *arXiv preprint arXiv:1611.02770*, 2016.

[79] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[80] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," *arXiv preprint arXiv:1611.03814*, 2016.

[81] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *International Journal of Security and Networks*, vol. 10, no. 3, pp. 137–150, 2015.

[82] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *USENIX Security Symposium*, 2016, pp. 601–618.

[83] C. A. C. Choo, F. Tramer, N. Carlini, and N. Papernot, "Label-only membership inference attacks," *arXiv preprint arXiv:2007.14321*, 2020.

[84] Z. Li and Y. Zhang, "Label-leaks: Membership inference attack with label," *arXiv preprint arXiv:2007.15528*, 2020.

[85] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[86] C. Song and V. Shmatikov, "Auditing data provenance in text-generation models," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 196–206.

[87] Y. Miao, B. Z. H. Zhao, M. Xue, C. Chen, L. Pan, J. Zhang, D. Kaafar, and Y. Xiang, "The audio auditor: Participant-level membership inference in voice-based iot," *arXiv preprint arXiv:1905.07082*, 2019.

[88] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, "Eye tracking for everyone," in *IEEE conference on computer vision and pattern recognition*, 2016, pp. 2176–2184.

[89] H. Sasamoto, N. Christin, and E. Hayashi, "Undercover: authentication usable in front of prying eyes," in *SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2008, pp. 183–192.

[90] R. Dhamija, A. Perrig *et al.*, "Deja vu-a user study: Using images for authentication." in *USENIX Security Symposium*, vol. 9, Aug. 2000.

[91] S. Brostoff and M. A. Sasse, "Are passfaces more usable than passwords? a field trial investigation," in *People and Computers XIV-Usability or Else!* Springer, 2000, pp. 405–424.

[92] J. V. Uspensky, *Introduction to mathematical probability.* McGraw-Hill Book Company, New York, 1937, pp. 23–24.

[93] D. W. Hansen and Q. Ji, “In the eye of the beholder: A survey of models for eyes and gaze,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 3, pp. 478–500, 2010.

[94] T. Baltrušaitis, P. Robinson, and L.-P. Morency, “Openface: an open source facial behavior analysis toolkit,” in *Applications of Computer Vision (WACV).* IEEE, 2016, pp. 1–10.

[95] E. S. Dalmaijer, S. Mathôt, and S. Van der Stigchel, “Pygaze: An open-source, cross-platform toolbox for minimal-effort programming of eyetracking experiments,” *Behavior research methods*, 2014.

[96] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning.* MIT press Cambridge, 2016, vol. 1.

[97] H. Peng, F. Long, and C. Ding, “Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[98] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[99] J.-A. LeFevre, G. S. Sadesky, and J. Bisanz, “Selection of procedures in mental addition: Reassessing the problem size effect in adults.” *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 1996.

[100] D. Foo Kune and Y. Kim, “Timing attacks on pin input devices,” in *The 17th ACM Conference on Computer and Communications Security*, ser. CCS ’10. New York, NY, USA: ACM, 2010, pp. 678–680.

[101] L. Cai and H. Chen, "Touchlogger: inferring keystrokes on touch screen from smartphone motion," in *The 6th USENIX conference on Hot topics in security.* USENIX, 2011.

[102] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge attacks on smartphone touch screens," in *The 4th USENIX conference on Offensive technologies.* USENIX, 2010, pp. 1–7.

[103] Q. Yue, Z. Ling, X. Fu, B. Liu, K. Ren, and W. Zhao, "Blind recognition of touched keys on mobile devices," in *The 2014 ACM SIGSAC Conference on Computer and Communications Security.* ACM, 2014, pp. 1403–1414.

[104] L. Simon and R. Anderson, "Pin skimmer: Inferring pins through the camera and microphone," in *The Third ACM workshop on Security and privacy in smartphones & mobile devices.* ACM, 2013, pp. 67–78.

[105] A. Blum, J. Hopcroft, and R. Kannan, "Foundations of data science," *Vorabversion eines Lehrbuchs*, 2016.

[106] D. Gafurov, K. Helkala, and T. Søndrol, "Biometric gait authentication using accelerometer sensor." *JCP*, vol. 1, no. 7, pp. 51–59, 2006.

[107] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication," *IEEE transactions on information forensics and security*, vol. 8, no. 1, pp. 136–148, 2012.

[108] W. Garcia, J. I. Choi, S. K. Adari, S. Jha, and K. R. Butler, "Explainable blackbox attacks against model-based authentication," *arXiv preprint arXiv:1810.00024*, 2018.

[109] H. Xu, Y. Zhou, and M. R. Lyu, "Towards continuous and passive authentication via touch biometrics: An experimental study on smartphones," in *10th Symposium On Usable Privacy and Security ({SOUPS} 2014)*, 2014, pp. 187–198.

[110] C.-C. Han, H.-L. Cheng, C.-L. Lin, and K.-C. Fan, “Personal authentication using palm-print features,” *Pattern recognition*, vol. 36, no. 2, pp. 371–381, 2003.

[111] L. Li, X. Zhao, and G. Xue, “Unobservable re-authentication for smartphones.” in *NDSS*, vol. 56, 2013, pp. 57–59.

[112] J. Chauhan, H. J. Asghar, A. Mahanti, and M. A. Kaafar, “Gesture-based continuous authentication for wearable devices: The smart glasses use case,” in *International Conference on Applied Cryptography and Network Security.* Springer, 2016, pp. 648–665.

[113] A. Das, N. Borisov, and M. Caesar, “Tracking mobile web users through motion sensors: Attacks and defenses,” in *NDSS*, 2016.

[114] “Android developers - monkeyrunner,” https://developer.android.com/studio/test/monkeyrunner, accessed: 2019-10-13.

[115] O. Shwartz, A. Cohen, A. Shabtai, and Y. Oren, “Shattered trust: When replacement smartphone components attack,” in *11th {USENIX} Workshop on Offensive Technologies ({WOOT} 17)*, 2017.

[116] C. Son, W. Chang, K. Deoksang, D.-K. Shin, B. Yoo, H. SeungJu, H. JaeJoon, S. Jinwoo, and C. K. Choi, “Face verification method and apparatus,” Oct. 2018, uS Patent App. 15/833,292.

[117] “Clarifai - face embedding model,” https://www.clarifai.com/models/face-embedding-image-recognition-model-d02b4508df58432fbb84e800597b8959, accessed: 2019-10-13.

[118] C. M. Bishop, *Pattern recognition and machine learning.* Springer, 2006.

[119] F. Rosenblatt, “Principles of neurodynamics. perceptrons and the theory of brain mechanisms,” Cornell Aeronautical Lab Inc Buffalo NY, Tech. Rep., 1961.

[120] V. M. Patel, R. Chellappa, D. Chandra, and B. Barbello, “Continuous user authentication on mobile devices: Recent progress and remaining challenges,” *IEEE Signal Processing Magazine*, vol. 33, no. 4, pp. 49–61, 2016.

[121] D. Anguita, A. Ghio, L. Oneto, X. Parra Perez, and J. L. Reyes Ortiz, “A public domain dataset for human activity recognition using smartphones,” in *Proceedings of the 21th International European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2013, pp. 437–442.

[122] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, “Vggface2: A dataset for recognising faces across pose and age,” in *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018).* IEEE, 2018, pp. 67–74.

[123] G. B. H. E. Learned-Miller, “Labeled faces in the wild: Updates and new reporting procedures,” University of Massachusetts, Amherst, Tech. Rep. UM-CS-2014-003, May 2014.

[124] “Github - facenet source repository,” https://github.com/davidsandberg/facenet, accessed: 2019-10-13.

[125] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, “Signature verification using a" siamese" time delay neural network,” in *Advances in neural information processing systems*, 1994, pp. 737–744.

[126] A. Fawzi, S.-M. Moosavi-Dezfooli, P. Frossard, and S. Soatto, “Classification regions of deep neural networks,” *arXiv preprint arXiv:1705.09552*, 2017.

[127] S. Eberz, K. B. Rasmussen, V. Lenders, and I. Martinovic, “Evaluating behavioral biometrics for continuous authentication: Challenges and metrics,” in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security.* ACM, 2017, pp. 386–399.

[128] S. Sugrim, C. Liu, M. McLean, and J. Lindqvist, “Robust performance metrics for authentication systems,” in *Network and Distributed Systems Security (NDSS) Symposium*, 2019.

[129] “Github - timesynth source repository,” https://www.tensorflow.org/guide/estimators, accessed: 2019-10-13.

[130] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining.* ACM, 2016, pp. 1135–1144.

[131] S. Sprager and M. Juric, "Inertial sensor-based gait recognition: A review," *Sensors*, vol. 15, no. 9, pp. 22 089–22 127, 2015.

[132] "Tensorflow - estimators," https://github.com/TimeSynth/TimeSynth, accessed: 2019-10-13.

[133] C. Soutar, "Biometric system performance and security," *IEEE Auto. Identification Advanced Technol.*, 1999.

[134] E. Chan-Tin, V. Heorhiadi, N. Hopper, and Y. Kim, "The frog-boiling attack: Limitations of secure network coordinate systems," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 3, p. 27, 2011.

[135] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM workshop on Security and artificial intelligence.* ACM, 2011.

[136] D. Lowd and C. Meek, "Adversarial learning," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining.* ACM, 2005, pp. 641–647.

[137] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[138] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, "Certified adversarial robustness via randomized smoothing," *arXiv preprint arXiv:1902.02918*, 2019.

[139] X. Cao and N. Z. Gong, "Mitigating evasion attacks to deep neural networks via region-based classification," in *Proceedings of the 33rd Annual Computer Security Applications Conference.* ACM, 2017, pp. 278–287.

[140] S. Sugrim, C. Liu, and J. Lindqvist, "Recruit until it fails: Exploring performance limits for identification systems," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 3, p. 104, 2019.

[141] M. J. Kearns and U. V. Vazirani, *An introduction to computational learning theory.* MIT press, 1994.

[142] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms.* Cambridge university press, 2014.

[143] B. Z. H. Zhao, H. J. Asghar, and M. A. Kaafar, "On the resilience of biometric authentication systems against random inputs," in *Network and Distributed System Security Symposium (NDSS)*, 2020.

[144] D. Yang, D. Zhang, and B. Qu, "Participatory cultural mapping based on collective behavior data in location-based social networks," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 7, no. 3, p. 30, 2016.

[145] B. Z. H. Zhao, H. J. Asghar, R. Bhaskar, and M. A. Kaafar, "On inferring training data attributes in machine learning models," *arXiv preprint arXiv:1908.10558*, 2019.

[146] X. Wu, M. Fredrikson, S. Jha, and J. F. Naughton, "A methodology for formalizing model-inversion attacks," in *2016 IEEE 29th Computer Security Foundations Symposium (CSF).* IEEE, 2016, pp. 355–370.

[147] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro, "Logan: Membership inference attacks against generative models," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 1, pp. 133–152, 2019.

[148] B. Jayaraman, L. Wang, D. Evans, and Q. Gu, "Revisiting membership inference under realistic assumptions," *arXiv preprint arXiv:2005.10881*, 2020.

[149] Z. Yang, J. Zhang, E.-C. Chang, and Z. Liang, "Neural network inversion in adversarial setting via background knowledge alignment," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 225–240.

[150] H. Zhao, J. Chi, Y. Tian, and G. J. Gordon, "Adversarial privacy preservation under attribute inference attack," *arXiv preprint arXiv:1906.07902*, 2019.

[151] A. Sablayrolles, M. Douze, C. Schmid, Y. Ollivier, and H. Jégou, "White-box vs black-box: Bayes optimal strategies for membership inference," in *International Conference on Machine Learning*, 2019, pp. 5558–5567.

[152] Y. Long, L. Wang, D. Bu, V. Bindschaedler, X. Wang, H. Tang, C. A. Gunter, and K. Chen, "A pragmatic approach to membership inferences on machine learning models," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2020, pp. 521–534.

[153] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 691–706.

[154] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 308–318.

[155] F. Farokhi and M. A. Kaafar, "Modelling and quantifying membership information leakage in machine learning," *arXiv preprint arXiv:2001.10648*, 2020.

[156] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[157] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *arXiv preprint arXiv:1412.5068*, 2014.

[158] I. Mironov, "Rényi differential privacy," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, 2017, pp. 263–275.

[159] R. Bassily, A. Smith, and A. Thakurta, "Private empirical risk minimization: Efficient algorithms and tight error bounds," in *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE, 2014, pp. 464–473.

[160] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[161] S. Fletcher and M. Z. Islam, "Differentially private random decision forests using smooth sensitivity," *Expert Systems with Applications*, vol. 78, pp. 16–31, 2017.

[162] K. Gurney, *An introduction to neural networks.* CRC press, 2014.

[163] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, 2001, pp. 41–46.

[164] N. Holohan, S. Braghin, P. Mac Aonghusa, and K. Levacher, "Diffprivlib: The ibm differential privacy library," *arXiv preprint arXiv:1907.02444*, 2019.

[165] J. Vaidya, B. Shafiq, A. Basu, and Y. Hong, "Differentially private naive bayes classification," in *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, vol. 1. IEEE, 2013, pp. 571–576.

[166] P. McCullagh, *Generalized linear models.* Routledge, 2019.

[167] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *Journal of Machine Learning Research*, vol. 12, no. Mar, pp. 1069–1109, 2011.

[168] S. A. M'rio, K. Chatzikokolakis, C. Palamidessi, and G. Smith, "Measuring information leakage using generalized gain functions," in *2012 IEEE 25th Computer Security Foundations Symposium.* IEEE, 2012, pp. 265–279.

[169] I. Issa, S. Kamath, and A. B. Wagner, "An operational measure of information leakage," in *2016 Annual Conference on Information Science and Systems (CISS).* IEEE, 2016, pp. 234–239.

[170] W. Wang, L. Ying, and J. Zhang, "On the relation between identifiability, differential privacy, and mutual-information privacy," *IEEE Transactions on Information Theory*, vol. 62, no. 9, pp. 5018–5029, 2016.

[171] S. Corbett-Davies and S. Goel, "The measure and mismeasure of fairness: A critical review of fair machine learning," *arXiv preprint arXiv:1808.00023*, 2018.

[172] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010.* Springer, 2010, pp. 177–186.

[173] J. Chauhan, S. Seneviratne, Y. Hu, A. Misra, A. Seneviratne, and Y. Lee, "Breathing-based authentication on resource-constrained iot devices using recurrent neural networks," *Computer*, vol. 51, no. 5, pp. 60–67, 2018.

[174] B. Jayaraman and D. Evans, "Evaluating Differentially Private Machine Learning in Practice," https://github.com/bargavj/EvaluatingDPML, 2019, accessed: 2020-02-12.

[175] Google, "TensorFlow Privacy," https://github.com/tensorflow/privacy, 2019, accessed: 2020-02-12.

[176] S. Fletcher, "A Differentially-Private Random Decision Forest using Smooth Sensitivity," https://github.com/sam-fletcher/Smooth_Random_Trees/, 2016, accessed: 2020-02-12.

[177] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[178] IBM, "IBM Differential Privacy Library," https://github.com/IBM/differential-privacy-library, 2019, accessed: 2020-02-12.

[179] A. K. Jain, K. Nandakumar, and A. Nagar, "Biometric template security," *EURASIP Journal on advances in signal processing*, vol. 2008, p. 113, 2008.

[180] S. Marcel, M. S. Nixon, and S. Z. Li, *Handbook of biometric anti-spoofing.* Springer, 2014, vol. 1.

[181] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: Towards end-to-end speech synthesis," *arXiv preprint arXiv:1703.10135*, 2017.

[182] T. Kinnunen, M. Sahidullah, H. Delgado, M. Todisco, N. Evans, J. Yamagishi, and K. A. Lee, "The asvspoof 2017 challenge: Assessing the limits of replay spoofing attack detection," *ISCA (the International Speech Communication Association)*, 2017.

[183] X. Tan, Y. Li, J. Liu, and L. Jiang, "Face liveness detection from a single image with sparse low rank bilinear discriminative model," in *European Conference on Computer Vision.* Springer, 2010, pp. 504–517.

[184] M. Bicego, E. Grosso, and M. Tistarelli, "Face authentication using one-class support vector machines," in *International Workshop on Biometric Person Authentication.* Springer, 2005, pp. 15–22.

[185] W. Garcia, A. Chhotaray, J. I. Choi, S. K. Adari, K. R. Butler, and S. Jha, "Brittle features of device authentication," in *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, 2021, pp. 53–64.

[186] L. Elazary and L. Itti, "A bayesian model for efficient visual search and recognition," *Vision research*, vol. 50, no. 14, pp. 1338–1352, 2010.

[187] B. Kulynych and M. Yaghini, "mia: A library for running membership inference attacks against ML models," 2018. [Online]. Available: https://doi.org/10.5281/zenodo.1433744

[188] M. O'Searcoid, *Metric spaces.* Springer Science & Business Media, 2006.

[189] V. Guruswami, "Gilbert-varshamov bound," Lecture Notes, Introduction to Coding Theory, 2010.

[190] P. Gaborit and G. Zemor, “Asymptotic improvement of the gilbert–varshamov bound for linear codes,” *IEEE Transactions on Information Theory*, vol. 54, no. 9, pp. 3865–3872, 2008.

[191] D. S. Mitrinović, J. E. Pečarić, and A. M. Fink, *Bernoulli’s Inequality.* Springer Netherlands, 1993, pp. 65–81.