2018

Towards privacy-aware mobile-based continuous authentication systems

Mohammad Al-Rubaie
Iowa State University

Towards privacy-aware mobile-based continuous authentication systems

by

Mohammad Al-Rubaie

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Program of Study Committee:
J. Morris Chang, Co-major Professor
Ying Cai, Co-major Professor
Thomas E. Daniels
Neil Zhenqiang Gong
Chinmay Hegde
Ahmed Kamal

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2018

DEDICATION

To my parents, Tarik and Zainab,

for without their constant support,

it would have been impossible for me to come this far.

TABLE OF CONTENTS

Page

LIST OF TABLES vii

LIST OF FIGURES viii

ACKNOWLEDGEMENTS ix

ABSTRACT xi

CHAPTER 1. GENERAL INTRODUCTION 1

CHAPTER 2. PRIVACY PRESERVING MACHINE LEARNING: THREATS AND SOLUTIONS 6

2.1 Abstract 6

2.2 Introduction 6

2.3 Machine Learning 7

2.4 Threats 10

2.5 Privacy-Preserving Machine Learning (PPML) 13

2.5.1 Cryptographic Approaches 14

2.5.2 Perturbation Approaches 17

2.6 Challenges and outlook 21

CHAPTER 3. RECONSTRUCTION ATTACKS AGAINST MOBILE-BASED CONTINUOUS AUTHENTICATION SYSTEMS IN THE CLOUD 24
3.1 Abstract 24
3.2 Introduction 25
3.3 Related Work 28
3.4 Preliminaries and Attacks Overview 31
3.4.1 SVM Classification and System Evaluation 32
3.4.2 System Model and Design Goals 33
3.4.3 Problem Setting and Threats 35
3.4.4 Adversary Model 35
3.4.5 Adversary Knowledge 37
3.4.6 Choice of Systems and their features 39
3.5 The Reconstruction Approach 41
3.5.1 The Numerical Approach 42
3.5.2 The Randomization Approach 47
3.6 Experimental Results and Discussion 51
3.6.1 Experimental Setup and Remarks 51
3.6.2 Results and Analysis 55
3.6.3 Discussion and Recommendations 60
3.7 Conclusions 64

CHAPTER 4. PRIVACY-AWARE GESTURE-BASED CONTINUOUS AUTHENTICATION BY REDUCING DIMENSIONS 66
4.1 Abstract 66
4.2 Introduction 67
4.3 Related Work 70
4.4 Compressive Privacy 72

4.5 Proposed Framework . . . 73
4.5.1 Problem Description . . . 73
4.5.2 Design Goals and Threat Model . . . 74
4.5.3 Gesture-based Continuous Authentication . . . 74
4.5.4 Privacy Enhancement . . . 75
4.6 Experimental Evaluation . . . 78
4.6.1 Case Study: Gesture-based Continuous Authentication . . . 78
4.6.2 Evaluation Criteria . . . 79
4.6.3 Experimental Results . . . 81
4.7 Compatibility with Cancelable Biometrics Approaches . . . 82
4.8 Conclusion . . . 83

CHAPTER 5. PRIVACY-PRESERVING PCA/DCA ON HORIZONTALLY-PARTITIONED DATA . . . 85
5.1 Abstract . . . 85
5.2 Introduction . . . 86
5.3 Related Work . . . 89
5.4 Preliminaries . . . 92
5.4.1 Dimensionality Reduction . . . 92
5.4.2 Cryptographic Background . . . 96
5.5 Problem Statement . . . 96
5.5.1 Overview . . . 96
5.5.2 Threat Model . . . 97
5.6 Privacy-Preserving PCA/DCA . . . 98
5.6.1 Privacy-Preserving PCA . . . 100
5.6.2 Privacy-Preserving DCA . . . 102
5.6.3 Security Analysis . . . 106

5.7 Eigenvalue Decomposition Garbled Circuit 109
5.7.1 Eigenvalue Decomposition (EVD) 109
5.7.2 Generalized Eigenvalue Decomposition (GEVD) 111
5.7.3 Implementation 112
5.8 Experiments 114
5.8.1 Efficiency 115
5.8.2 Accuracy 116
5.9 Conclusion 117

CHAPTER 6. GENERAL CONCLUSIONS AND FUTURE WORK 119

BIBLIOGRAPHY 121

LIST OF TABLES

Page

Table 3.1 Similar and different features to *SysC* [35] 40

Table 3.2 Distribution of features for different systems 40

Table 3.3 EERs of different attacked systems 52

Table 3.4 Pairwise comparisons of attacks and algorithms using paired t-test . . 59

Table 3.5 Average reconstruction attack times per user 59

Table 4.1 Features categories of different papers 76

Table 5.1 Distributed PCA Efficiency 114

Table 5.2 Distributed DCA Efficiency 114

Table 5.3 Privacy-Preserving DCA Accuracy 115

LIST OF FIGURES

Page

Figure 2.1 Machine Learning Tasks Overview 9
Figure 2.2 Machine Learning and threats 11

Figure 3.1 Attack points in AA systems 34
Figure 3.2 Overview of attack procedures 34
Figure 3.3 Reconstruct X & Y 44
Figure 3.4 Touch pressure 47
Figure 3.5 Parameter Selection 51
Figure 3.6 Results for left-to-right swipe classifier (LTR) 54
Figure 3.7 Results for right-to-left swipe classifier (RTL) 58
Figure 3.8 System Architecture 62
Figure 3.9 Results of the binary result attack 63

Figure 4.1 Data Projection 72
Figure 4.2 Dimensionality Reduction 76
Figure 4.3 Utility/Privacy/Security vs number of reduced dimensions 79
Figure 4.4 Security/Privacy vs. Utility ROC Curves 80
Figure 4.5 Enhancement to Cancelable Biometrics 83

Figure 5.1 Dimensionality Reduction Techniques 87
Figure 5.2 Data Projection 93
Figure 5.3 System architecture and protocols 99
Figure 5.4 Privacy-Preserving PCA Accuracy 118

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my gratitude to those who have supported, guided, inspired, and motivated me during my years as a doctoral student at ISU.

First and foremost, I would like to convey my sincerest appreciation and gratitude to Prof. J. Morris Chang, my major professor, for his support, patience, and continuous guidance during my doctoral studies. His support started even before departing Iraq, and when I arrived to Ames, he helped me whenever I approached him. Immediately after arriving to Ames, Prof. Chang provided me with an opportunity to work on his most recent DARPA project, Active Authentication, which helped steer my research interests into a very interesting combination: Security, Privacy, and Machine Learning. I am also grateful for the broad exposure to further aspects of academic and professional work such as developing competitive research proposals, presentation skills, and paper writing. I will always fondly remember the long discussions we had about research and proposals, his patience during our disagreements, and how comforting it was when reaching an agreement on a single viewpoint after long hours/days of discussions. One of the most valuable lessons I learned from Prof. Chang is trying to understand the background of other people, their terminology, and communicating with them in a language they understand.

Second, I would like to thank my co-major professor Prof. Ying Cai for his support, and willingness to help in whatever way possible. I would also like to thank my committee members for their constructive comments and suggestions to this work: Professors Ahmed Kamal, Chinmay Hegde, Neil Zhenqiang Gong, and Thomas E. Daniels. The knowledge that I have gained from them during coursework or discussions helped shape my research directions.

Third, I would like to thank all my collaborators, specifically, Prof. Sun-Yuan Kung, Prof. Pei-Yuan Wu, and Dr. Emre Yilmaz. I have learned a lot from them both in machine learning and security/privacy areas. Furthermore, I would like to thank my fellow graduate students at ISU:

Di Zhuang, Terry Fang, Sen Wang, Danny Shih, Yu-Wen Chen, Motassem Al-Tarazi, Chan-Ching Hsu, Wenhao Chen and Abdullah Almasoud.

I would like to thank my family for their unconditional love and support throughout the years. Special thanks go to my father, Tarik, and aunt Saiba, for supporting my decision to follow my dream to become an engineer. I apologize to my mother, Zainab, for not following her dream for me to become a medical doctor, and I hope that this PhD is a good enough replacement. I also want to thank my parents for paving the road to any success I achieved in my life. As my late friend, Sven Fecht, said: "your father must really love you to spend all that money on equipment to help you succeed", and that was during the difficult times in Iraq. I also want to thank my three wonderful aunts for their constant love and support. Finally, my brothers: Hussein, Ali, Ibrahim, and Yahia: thank you for your support throughout the years, and for your faith in me. More importantly, thank you for being better people than I could ever hope to be when caring for my father during his continuing illness for the last two years. I am only able to sleep at night because I know you are there for him. Last but not least, I would like to thank the two newest members of my family: Judy and Hasan.

Finally, I would like to thank my wife, Andrea Olson, for being there for me through thick and thin. Only she knows what I had to go through to get to this point. Her love, encouragement, patience, and company kept me going through all these years. It was a nice experience to be attending the same university, and her company during our daily lunch (in Carver Hall) helped recharge my batteries for the rest of the day.

Last but not least, I would like to thank the Defense Advanced Research Projects Agency (DARPA) for supporting the majority of the work in this dissertation.

ABSTRACT

User authentication is used to verify the identify of individuals attempting to gain access to a certain system. It traditionally refers to the initial authentication using knowledge factors (e.g. passwords), or ownership factors (e.g. smart cards). However, initial authentication cannot protect the computer (or smartphone), if left unattended, after the initial login. Thus, continuous authentication was proposed to complement initial authentication by transparently and continuously testing the users' behavior against the stored profile (machine learning model).

Since continuous authentication utilizes users' behavioral data to build machine learning models, certain privacy and security concerns have to be addressed before these systems can be widely deployed. In almost all of the continuous authentication research, non-privacy-preserving classification methods were used (such as SVM or KNN). The motivation of this work is twofold: (1) studying the implications of such assumption on continuous authentication security, and users' privacy, and (2) proposing privacy-aware solutions to address the threats introduced by these assumptions.

First, we study and propose reconstruction attacks and model inversion attacks in relation to continuous authentication systems, and we implement solutions that can be effective against our proposed attacks. We conduct this research assuming that a certain cloud service (which rely on continuous authentication) was compromised, and that the adversary is trying to utilize this compromised system to access a user's account on another cloud service. We identify two types of adversaries based on how their knowledge is obtained: (1) full-profile adversary that has access to the victim's profile, and (2) decision value adversary who is an active adversary that only has access to the cloud service mobile app (which is used to obtain a feature vector). Eventually, both adversaries use the user's compromised feature vectors to generate raw data based on our proposed reconstruction methods: a numerical method that is tied to a single attacked system (set of features), and a randomized algorithm that is not restricted to a single set of features. We

conducted experiments using a public data set where we evaluated the attacks performed by our two types of adversaries and two types of reconstruction algorithms, and we have shown that our attacks are feasible. Finally, we analyzed the results, and provided recommendations to resist our attacks. Our remedies directly limit the effectiveness of model inversion attacks; thus, dealing with decision value adversaries.

Second, we study privacy-enhancing technologies for machine learning that can potentially prevent full-profile adversaries from utilizing the stored profiles to obtain the original feature vectors. We also study the problem of restricting undesired inference on users' private data within the context of continuous authentication. We propose a gesture-based continuous authentication framework that utilizes supervised dimensionality reduction (S-DR) techniques to protect against undesired inference attacks, and meets the non-invertibility (security) requirement of cancelable biometrics. These S-DR methods are Discriminant Component Analysis (DCA), and Multiclass Discriminant Ratio (MDR). Using experiments on a public data set, our results show that DCA and MDR provide better privacy/utility performance than random projection, which was extensively utilized in cancelable biometrics.

Third, since using DCA (or MDR) requires computing the projection matrix from data distributed across multiple data owners, we proposed privacy-preserving PCA/DCA protocols that enable a data user (cloud server) to compute the projection matrices without compromising the privacy of the individual data owners. To achieve this, we propose new protocols for computing the scatter matrices using additive homomorphic encryption, and performing the Eigen decomposition using Garbled circuits. We implemented our protocols using Java and Obliv-C, and conducted experiments on public datasets. We show that our protocols are efficient, and preserve the privacy while maintaining the accuracy.

CHAPTER 1. GENERAL INTRODUCTION

User authentication verifies the identity of an individual attempting to access a certain system. It can be categorized into initial authentication and continuous authentication. Initial authentication is a mature field that relies on things we know (knowledge factors such as passwords or lock-screen patterns), things we have (ownership factors such as Smart cards, smartphones, and hardware or software tokens), and things we are (Inherent factors such as fingerprint, retinal pattern, face or voice). While initial authentication was mostly done with a single factor authentication (knowledge factors), it has progressed to include multi-factors where a combination of factors has to be used. A common example would be using a regular password, in addition to an ephemeral pass-code sent to the user's smartphone. Initial authentication does not address the threat of leaving the computer (or smartphone) unattended for a period of time after the initial login. A usual remedy would be locking the user's device after a period of inactivity, and requiring re-login, which might lead to inconvenience for users causing them to disable this feature, or increase the inactivity timeout value, which in turn limits the effectiveness of this method.

Continuous authentication was proposed to complement initial authentication by continuously and transparently looking for anomalies in the user's behavior after the initial login. If such anomaly was detected, the system might require re-entering the user's credentials. Continuous authentication has concentrated on behavioral biometrics to ensure transparency as physical biometrics are more intrusive. Behavioral biometrics include keystrokes' dynamics, mouse and web usage for computers, and touch gestures and keystrokes in addition to gait (using mobile device sensors) for mobile devices. Almost all continuous authentication systems rely on machine learning algorithms.

As with other machine learning based systems, continuous authentication needs an enrollment phase. During this phase, training data is collected from each user (the positive class), and in most cases, it is used alongside training data from other users (the negative class) to build a user profile

using a classification algorithm such as Support Vector Machines (SVM) or K-Nearest Neighbors (KNN). This profile can be stored on the device itself, or in the cloud. The latter case is meant to protect against intruders to online services. In almost all of the continuous authentication literature, it was assumed that non-privacy-preserving classification methods were to be used (such as SVM or KNN), and that negative samples from other users will be readily available for the enrollment phase, regardless of whether the profile will eventually be stored on the device itself or in the cloud. In this work, we study the effects of such assumptions on the security of continuous authentication systems, and the implications on user's privacy. Furthermore, we propose potential remedies to current problems with continuous authentication systems by proposing privacy-preserving machine learning algorithms and protocols.

We start by surveying the recent literature for the potential threats against machine learning (ML) systems (Chapter 2). These are broadly categorized into: (1) performing the training on plain-text raw data, (2) reconstruction attacks that aim to reconstruct an approximation of the raw data (given the associated feature vectors), (3) model inversion attacks that attempt to retrieve feature vectors similar to the ones used to build a ML model output given that model's output, and (4) membership inference attacks that aim to determine whether a certain record was used in training an ML model. Furthermore, in Chapter 2, we provide a broad overview of the techniques used to perform privacy-preserving machine learning. These can be categorized into cryptographic approaches (e.g. homomorphic encryption and garbled circuits) and perturbation approaches (e.g. differential privacy and dimensionality reduction).

In Chapter 3, we study reconstruction attacks and model inversion attacks in relation to continuous authentication systems. We further study the possibility of using dimensionality reduction (DR) to perturb the feature vectors before sending them to the cloud (to preserve the data owner's privacy) in Chapter 4. Since these DR methods need the projection matrix to be computed beforehand, we propose privacy-preserving PCA/DCA protocols in Chapter 5.

In chapter 3, we study the problem of reconstructing behavioral biometrics raw data on mobile devices, and we concentrate on touch gestures since it received most of the attention in the recent

literature. By reconstruction attacks, we refer to an adversary's attempt to reconstruct the raw data that generated a given feature vector. Since most of the recent literature in the area of mobile-based continuous authentication utilized SVM, we concentrated on this classification algorithm in our study. The case of protecting an online resource by relying on continuous authentication using mobile gestures is considered. We identify two types of adversaries based on how their knowledge is obtained: (1) full-profile adversary that has access to the victim's profile that includes user's feature vectors (and potentially other users' feature vectors), and (2) decision value adversary who is an active adversary that only has access to a mobile app that is used to obtain a feature vector that can help the adversary impersonate the victim. The adversaries differ in the way they obtain feature vectors, but they can share our two proposed reconstruction methods: a numerical method that is tied to a single attacked system (set of features), and a randomized algorithm that is not restricted to a single set of features. We chose five sets of features from the recent literature. One of them was selected to be a compromised system, while that other four were assumed to be attacked by an adversary that gained knowledge from the compromised system. We conducted experiments using a public data set where we evaluated the attacks performed by our two types of adversaries and two types of reconstruction algorithms. The experiments showed that our attacks were feasible, with a median ranging from 80% to 100% against one attacked system (using all types of attacks), and a median ranging from 73% to 100% against all attacked systems using the randomization-based algorithm and the negative support vector attack (full-profile adversary). Finally, we analyzed the results, and provided recommendations to resist our attacks.

Chapter 4 addresses the problem of restricting undesired inference on users' private data within the context of continuous authentication. To protect physical biometrics profiles, cancelable biometrics techniques were proposed to transform the biometrics model (profile) in a revocable and non-invertible way. Traditionally, cancelable biometrics requirements included security (non-invertibility), performance, reovacability and diversity. Many of these approaches utilized random projection to reduce the data dimensions to meet the security (non-invertibility) requirement. The mapping from the original feature vectors to a lower dimensional subspace is a many-to-one

mapping; hence, it is more resilient to reconstruction attacks than a dimensionality-preserving transformation [1]. However, we observe that the reduced dimensions data might still be used to infer privacy-intrusive information about the users. Recognizing the possibility of performing undesired inference on behavioral biometrics profiles, we introduce a new cancelable biometrics requirement: Privacy (inference restriction). We propose a gesture-based continuous authentication framework that utilizes supervised dimensionality reduction (S-DR) techniques to protect against undesired inference attacks. These S-DR methods are Discriminant Component Analysis (DCA), and Multiclass Discriminant Ratio (MDR). Using experiments on a public data set, our results show that DCA and MDR provide better privacy/utility performance than random projection, which was extensively utilized in cancelable biometrics. Finally, we outline how these techniques can be used to enhance previous cancelable biometrics approaches in terms of privacy (inference restriction).

Since using DCA (or MDR) requires computing the projection matrix from data distributed across multiple data owners, we proposed privacy-preserving PCA/DCA protocols. Chapter 5 addresses the problem of performing collaborative learning (privacy-preserving PCA/DCA) by a data user (cloud server) that utilizes a joint data set formed of samples gathered from multiple data owners. The samples held by the different data owners contain the same attributes (features) for different data objects. This case is called collaborative learning on horizontally-partitioned data. The computation of the projection matrices for PCA (and DCA) require computing the scatter matrices followed by performing eigenvalue decomposition. In Chapter 5, we propose methods that would enable the PCA/DCA computation to be performed on horizontally-partitioned data among multiple data owners without exposing the private data of any of the participating data owners, or requiring them to stay online for the execution of the protocol. To address this problem, we propose new protocols for computing the scatter matrices using additive homomorphic encryption, and performing the Eigen decomposition using Garbled circuits. We implemented our protocols using Java and Obliv-C, and conducted experiments on public datasets. We show that our protocols are efficient, and preserve the privacy while maintaining the accuracy.

Chapter 6 concludes the dissertation's contributions, and outlines the future work that needs to be done to extend the work presented in Chapter 5 to Kernel methods, in addition to new research directions related to local differential privacy and machine learning.

CHAPTER 2. PRIVACY PRESERVING MACHINE LEARNING: THREATS AND SOLUTIONS

Modified from a paper accepted for publication in the IEEE Security and Privacy Magazine, Jan. 2018.

Mohammad Al-Rubaie [1] and J. Morris Chang

2.1 Abstract

For privacy concerns to be addressed adequately in todays machine learning systems, the knowledge gap between the machine learning and privacy communities must be bridged. This article aims to provide an introduction to the intersection of both fields with special emphasis on the techniques used to protect the data.

2.2 Introduction

Our search queries, browsing history, purchase transactions, the videos we watch, and our movies' preferences are but few types of information that are being collected and stored on daily basis. This data collection happens within our mobile devices and computers, on the streets, and even in our own offices and homes. Such private data is being used for a variety of machine learning applications.

Machine learning (ML) is being increasingly utilized for a variety of applications from intrusion detection to recommending new movies. Some ML applications require private individuals data. Such private data is uploaded to centralized locations in clear text for ML algorithms to extract patterns, and build models from them. The problem is not limited to the threats associated with

[1] primary researcher and author. Mohammad is the primary author of this paper. Mohammad wrote the paper under the supervision and advice of Prof. J. Morris Chang

having all this private data exposed to insider threat at these companies, or outsider threat if the companies holding these data sets were hacked. In addition, it is possible to glean extra information about the private data sets even if the data was anonymized [2], or the data itself and the ML models were inaccessible and only the testing results were revealed [3].

In this article, we will describe machine learning tasks and applications, and the potential threats associated with current methods of collecting data or building ML systems. We will further elaborate on the techniques proposed to protect the privacy of individuals or corporates. Our intention is to bridge the gap between machine learning and privacy and security technologies by helping professionals of either fields to be more acquainted with machine learning, the potential threats to privacy, the proposed solutions, and the challenges that lie ahead.

2.3 Machine Learning

Arthur Samuel, a pioneer in the fields of computer gaming and artificial intelligence, described machine learning as a field of study that gives computers the ability to learn without being explicitly programmed. The aim of machine learning algorithms is to learn how to perform certain tasks by generalizing from data. Such tasks might include giving accurate predictions or finding structures in data.

The input data to a ML algorithm is usually represented as a set of samples. Each sample would contain a set of feature values. For example, consider a 100x100 pixels photo, where each pixel is represented by a single number (0-255 grayscale). We can use these pixel values to form a vector of length 10,000, which is normally called a feature vector. Each photo, represented as a feature vector, can be associated with a label (e.g., name of the person in the photo). An ML algorithm would use a training set formed of multiple feature vectors, and their associated labels, to build an ML model. This process is called the training or learning phase. When presented with a new test sample, this ML model should give the predicted label (persons name or identifier in face recognition applications). The ability of such an ML model to accurately predict the label is a measure of how well this ML model generalizes to unseen data. It is measured empirically by

the test error (generalization error), and it can depend on the quality and quantity of the data used for training the model, what ML algorithm was used to build the model, the selection of ML algorithm hyper parameters (e.g., using cross validation), and even the features extraction method (if any was required).

In general, some feature extraction method might be needed to produce useful features from the raw data, such as the preprocessing step for pictures (as raw data) which might involve face detection, followed by cropping and resizing to 100x100 pixels to match the feature vector length [4], or projecting the data to lower dimensions using PCA [5]. Feature engineering utilizes domain knowledge to produce features from raw data, and it is important for many applications, however, in many modern applications it has been reduced to mere preprocessing steps. Data sets are generally formed of feature vectors, regardless of whether this data is labeled or not, which depends on the application or learning style.

In general, we can categorize ML algorithms based on their learning style into supervised or unsupervised learning (or a combination of both):

Supervised Learning: that utilizes labeled data where each feature vector is associated with an output value that might be a class label (classification), or a continuous value (regression). This labeled data is used to build models (training phase) that can predict the label of new feature vectors (testing phase).

With ***classification***, the samples (feature vectors) belong to two or more classes, and the objective of the ML algorithm is to determine the class to which the new sample belongs. Some algorithm might achieve that by finding a separating hyperplane between the different classes as can be seen in Fig. 2.1a. An example application is face recognition where a face image can be tested to ascertain that it belongs to a certain person. Multiple classification algorithms can be used for each of the above applications such as Support Vector Machines (SVM), Neural Networks or Logistic Regression.

When the label of a sample is a continuous value (also called the dependent or response variable) rather than a discrete one, the task is called ***regression***. The samples are formed of features that

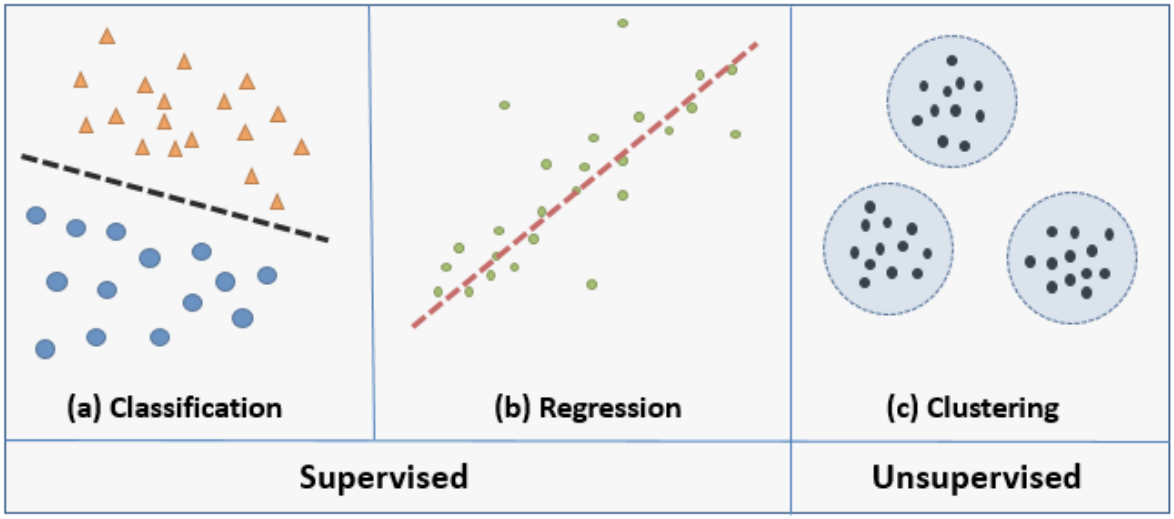


Figure 2.1: Machine Learning Tasks Overview. (a) Classification: finding a separating dashed line between the two classes, circles and triangles classes, (b) Regression: fitting a predictive model (dashed line to the observed data points, (c) Clustering: grouping a set of samples into a number of clusters.

are also called independent variables. The target of regression would be fitting a predictive model (such as a line) to an observed data set such that the distances between the observed data points and this line are minimized (Fig. 2.1b). A simple example would be estimating the price of a house based on its location, area, and number of rooms.

Unsupervised Learning: With this type of learning, data is not labeled as feature vectors do not come with a class label or a response variable. The target in this case would be to find structure in the data. ***Clustering*** is probably the most common unsupervised learning technique, and its aim is to group a set of samples into different clusters (Fig. 2.1c). Samples in the same cluster are supposed to be relatively similar to each other, and different from samples in other clusters (the similarity measure could be the Euclidean distance). K-means is one of the most popular clustering methods.

Semi-supervised Learning: Labeling the data can be expensive as it requires human experts or special devices; hence, only some of data gets labeled sometimes, while the vast majority remains unlabeled. Researchers found that even having a small portion of labeled data can considerably improve the learning process.

Other Applications: Some ML tasks and applications do not strictly fall into one of the categories above as they can be performed either in a supervised or unsupervised way. Examples include dimensionality reduction and recommender systems.

2.4 Threats

In each of the ML tasks mentioned above, three different roles are possible: the input party (data owners or contributors), the computation party and the results party [6]. In such systems, the data owner(s) send their data to the computation party that performs the required ML task and delivers the output to the results party. Such output could be an ML model that the results party can utilize for testing new samples. In other cases, the computation party might keep the ML model, and performs the task of testing new samples submitted by the results party, and returning the testing results to the results party. If all three roles are assumed by the same entity, then privacy is naturally preserved; however, when these roles are distributed across two or more entities, then privacy enhancing technologies are needed.

It is common to have the same entity be both the computation and the results parties, and this entity is mostly separate from the data owners. In fact, with all of the data that is collected from individuals around the world on daily basis, data owners might not be aware of how the data collected from them is being used (or misused), and in many cases, not even aware that some data types are being collected.

There are multiple levels of threats depending on the privacy leaks associated with the data sharing process as can be seen from Fig. 2.2, and the following paragraphs:

Private Data in the Clear: If the data owner(s) are separate from the computation party, then the private data would be transferred to the computation party, possibly over a secure channel. However, it would most likely reside in the computation server(s) in its original form, i.e. not encrypted or transformed in any way. This is the biggest type of threat as the private data would be susceptible to both insider and outsider attacks. Such private data could be stored as raw data,

or as features extracted from the raw data. Naturally, storing it in a raw form imposes greater threat as the data is ready to be processed in any way possible.

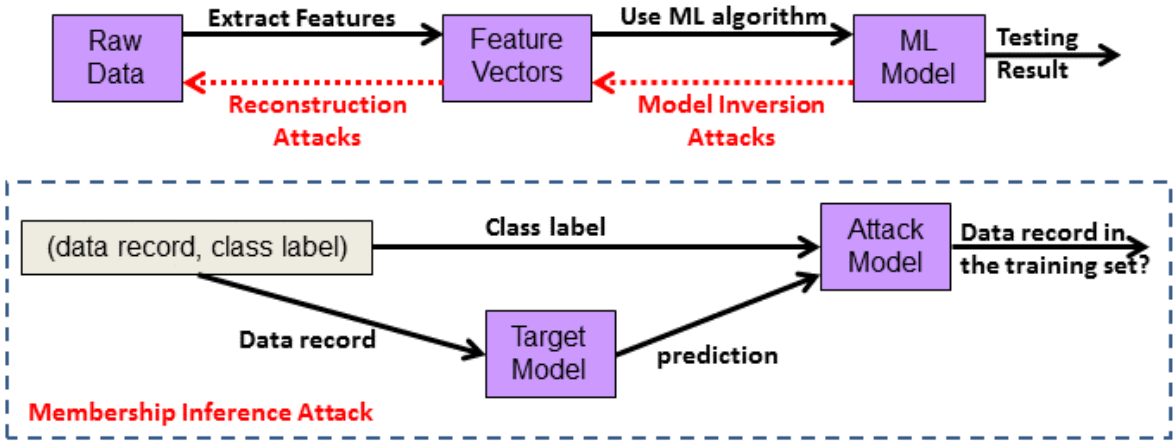


Figure 2.2: Machine Learning and threats

Reconstruction Attacks: Even when only the features (extracted from the raw data) are transferred to, and stored in the computation party server(s), there is a threat imposed by reconstruction attacks. In this case, the ***adversarys goal*** is reconstructing the raw private data by using their ***knowledge*** of the feature vectors. Reconstruction attacks require ***white-box*** access to the ML model, i.e. the feature vectors in a model must be known. Such attacks could be possible when the feature vectors used for the ML training phase were not deleted after building the desired ML model. In fact, some ML algorithms such as SVM or kNN store feature vectors in the model itself. Examples of successful cases of reconstruction attacks include: fingerprint reconstruction [7] where a fingerprint image (raw data) could be reconstructed from a minutiae template (features), and mobile device touch gesture reconstruction [8] where touch events (raw data) were reconstructed from gesture features such as velocity and direction. In both cases, a privacy threat (caused by not protecting the private data in its feature form) resulted in a security threat to authentication systems that in turn results in failure to protect the data owners privacy (since attackers might gain access to the data owners devices). While the aim of these cases was misguiding a ML system into thinking the reconstructed raw data belonged to a certain data owner, other reconstruction attacks might reveal private data directly such as location or age.

To resist reconstruction attacks, ML models that store explicit feature vectors (e.g. SVM) should be avoided, or, if used, they should not be provided to the results party. Moreover, protection against model inversion attacks should be in place to prevent synthesizing feature vectors.

Model Inversion Attacks: Some ML algorithms produce models where explicit feature vectors are not stored in the ML model, e.g. ridge regression or neural networks. Hence, the ***adversarys knowledge*** would be limited to either: (a) an ML model with no stored feature vectors (white-box access), or (b) only the responses returned by the computation party when the results party submits new testing samples (black-box access). Here, the ***adversarys target*** is creating feature vectors that resemble those used to create an ML model by utilizing the responses received from that ML model. Such attacks utilize the confidence information (e.g. probability or SVM decision value) that are sent back as a response for testing samples submitted by the results party. These attacks produce an average that represents a certain class; hence, they would be most threatening to privacy when a certain class represents a single individual, as in face recognition. It should be noted that Fredrikson *et al.* [3] demonstration of their model inversion attack also included reconstruction in the same step. This was because the features in their case matched the raw data (face images).

To resist such attacks, the results party should be limited to black-box access, and the output should be limited; thereby decreasing the black-box adversarys knowledge. For example, the attack success rate decreased when classification algorithms reported rounded confidence values [3] or just the predicted class labels [8]. A step further could be aggregating the result of testing multiple samples [8], but this approach would not be appropriate for all applications.

Membership Inference Attacks: While model inversion attacks do not produce an actual sample from the training set, nor do they infer whether a sample was in the training set based on the ML model output, membership inference attacks do the latter. Given an ML model and a sample (***adversarys knowledge***), membership inference attacks aim to determine if the sample was a member of the training set used to build this ML model (***adversarys target***). This attack could be used by an adversary to learn whether a certain individuals record was used to train an

ML model associated with a specific disease. Such attacks utilize the differences in the ML model predictions on samples that were used in the training set versus those that were not included. Shokri *et al.* [9] investigated these attacks, and trained attack models that take a samples correct label and the target ML model prediction as inputs, and determines whether the sample was in the training set or not. These attack models were trained using shadow models built from data generated using three methods: model inversion attack, statistics-based synthesis, or noisy real data. While training the attack models utilized a black or white-box access, performing the attacks using these models only required a black-box adversary.

Shokri *et al.* [9] tried different mitigation techniques such as regularization and coarse precision of prediction vectors, and found that limiting the output to the class label was the most effective, albeit not enough to thwart the attack completely. By definition, differential privacy can resist membership inference attacks (as will be seen in the next section).

De-anonymization (Re-Identification): Anonymization by removing personal identifiers before releasing the data to the public may seem like a natural approach for protecting the privacy of individuals. Indeed, some companies attempted to protect their users privacy by only releasing anonymized versions of their datasets, as was the case with the anonymous movie ratings released by Netflix to aid contestants for its 1M$ prize to build better recommender systems (for movies). Despite the anonymization, researchers were able to utilize this dataset along with IMDB background knowledge to identify the Netflix records of known users, and were further able to deduce the users apparent political preferences [2]. This incident demonstrates that anonymization cannot reliably protect the privacy of individuals in the face of strong adversaries.

2.5 Privacy-Preserving Machine Learning (PPML)

Many privacy-enhancing techniques concentrated on allowing multiple input parties to collaboratively train ML models without releasing their private data in its original form. This was mainly performed by utilizing cryptographic approaches, or differentially-private data release (perturbation techniques). Differential privacy is especially effective in preventing membership inference attacks.

Finally, as discussed above, the success of model inversion and membership inference attacks can be decreased by limiting the model prediction output (e.g. class labels only).

2.5.1 Cryptographic Approaches

When a certain ML application requires data from multiple input parties, cryptographic protocols could be utilized to perform ML training/testing on encrypted data. In many of these techniques, achieving better efficiency involved having data owners contribute their encrypted data to the computation servers, which would reduce the problem to a secure two/three party computation setting. In addition to increased efficiency, such approaches have the benefit of not requiring the input parties to remain online.

Most of these approaches address the case of horizontally-partitioned data: Each data owner has collected the same set of features for different data objects. Face recognition is one example since each person that wants an ML model trained for her face can submit multiple feature vectors extracted from their own photos. In each of these cases, the same set of features is extracted by each data owner.

Homomorphic encryption, garbled circuits, secret sharing and secure processors are the most widely used cryptographic techniques to achieve PPML:

Homomorphic Encryption: Fully homomorphic encryption enables the computation on encrypted data, with operations such as addition and multiplication that can be used as basis for more complex arbitrary functions. Due to the high cost associated with frequently bootstrapping the cipher text (refreshing the cipher text because of the accumulated noise), additive homomorphic encryption schemes were mostly used in PPML approaches. Such schemes only enable addition operations on encrypted data, and multiplication by a plaintext. A popular example is Paillier cryptosystem.

To extend additive homomorphic encryption functionality, protocols were developed to enable comparison of two encrypted values, or to perform secure multiplication and decryption operations, mostly by blinding the cipher text through adding an encrypted random value to the encrypted value

that needs to be protected. To increase the efficiency of using additive homomorphic encryption, data packing techniques were developed to enable more than one plain text value to be encrypted by the same cipher text. Such techniques were employed by some PPML approaches to enable efficient and secure PPML systems such as the collaborative filtering system [10] proposed by Erkin *et al.* which used all of the previous techniques. In this system, data owners contribute data encrypted with the public key of a privacy service provider (PSP), but send the encrypted data to the service provider (SP). The PSP provides privacy and computation services, while the SP provides storage and computation services with the intention of generating private recommendations for its customers (the data owners). For such system to be secure, the SP and the PSP must not collude. Since they provide different services, it is understandable that the SP and the PSP could be different companies; hence, the non-collusion assumption is plausible. In this system, the data owners are both input and results parties while the SP and PSP are the computation parties.

Garbled Circuits: Assuming a two-party setup with Alice and Bob wanting to obtain the result of a function computed on their private inputs, Alice can convert the function into a garbled circuit, and send this circuit along with her garbled input. Bob obtains the garbled version of his input from Alice without her learning anything about Bobs private input (e.g., using oblivious transfer). Bob can now use his garbled input with the garbled circuit to obtain the result of the required function (and can optionally share it with Alice). Some PPML approaches combined additive homomorphic encryption with Garbled circuits. Nikolaenko *et al.* [11] developed a PP ridge regression system that utilized both techniques, where an evaluator (similar to the SP in Erkin *et al.* [10]) adds the encrypted shares submitted by multiple data owners to obtain encrypted intermediate values. These shares are encrypted using additive homomorphic encryption with the public key of the crypto service provider CSP (similar to the PSP in Erkin *et al.* [10]). The CSP then creates a garbled circuit and sends it to the evaluator that also obtains the garbled version of the intermediate shares from the CSP. The evaluator can proceed with using the garbled circuit and its garbled input to create the ML model(s) it needs.

Some PPML approaches concentrated on the classification task alone (testing phase rather than training/testing phases). Bost *et al.* [12] developed cryptographic building blocks using homomorphic encryption and garbled circuits that enabled them to construct three popular classification protocols: hyperplane decision, Nave Bayes, and decision trees. The aim was to enable testing new samples while protecting both the ML model and the submitted samples.

Secret Sharing is a method for distributing a secret among multiple parties, with each one holding a share of the secret. Individual shares are of no use on their own; however, when the shares are combined, the secret can be reconstructed. With threshold secret sharing, not all the shares are required to reconstruct the secret; but only t of them (t refers to threshold).

In one setting, multiple input parties can generate shares of their private data, and send these shares to a set of non-colluding computation servers. Each server could compute a partial result from the shares it received. Finally, a results party (or a proxy) can receive these partial results, and combine them to find the final result. Since these computation servers have similar functionalities (unlike SP and PSP mentioned above), special attention should be paid to where such servers are hosted, and which entities control them, in order to convince data owners that the computation servers would not collude. In general, secret sharing approaches might be more efficient than the other cryptographic approaches, and this resulted in having multiple commercial products that utilize secret sharing. An example is ShareMind developed by Cybernetica which was used to develop a privacy-preserving system for performing PCA computation by adapting parallelized PCA computation method to the secret sharing paradigm [13].

In another settings [14], the shares are distributed among other input parties (users), rather than with the computation party (server). Bonawitz [14] developed a protocol for securely computing sums of vectors, to aggregate user-provided model updates for training a neural network model. Each user applies double-masking (blinding) on its private update vector using: a user-specific secret value, and secret values shared with other users (generated using Diffie-Hellman key agreement). To account for users dropping out before finishing the protocol, each user distributes shares of its user-specific secret and its Diffie-Hellman private key to other users. The server is tasked with

routing messages between the users, and computing the final result if at least t of the users survive until the last round (threshold secret sharing). This protocol is communication-efficient (no more than twice the plain text counterpart); which makes it suitable for high-dimensional vectors.

Secure Processors: While initially introduced to ensure the confidentiality and integrity of sensitive code from unauthorized access by rogue software at higher privilege levels, Intel SGX-processor are being utilized in privacy-preserving computation. Ohrimenko *et al.* [15] developed a data oblivious ML algorithms for neural networks, SVM, k-means clustering, decision trees and matrix factorization that are based on SGX-processors. The main idea involves having multiple data owners collaborate to perform one of the above mentioned ML tasks with the computation party running the ML task on an SGX-enabled data center. An adversary can control all the hardware and software in the data center except for the SGX-processors used for computation. In this system, each data owner independently establishes a secure channel with the enclave (containing the code and data), authenticates themselves, verifies the integrity of the ML code in the cloud, and securely uploads its private data to the enclave. After all the data is uploaded, the ML task is run by the secure processor, and the output is sent to the results parties over secure authenticated channels.

2.5.2 Perturbation Approaches

Differential privacy (DP) techniques resist membership inference attacks by adding random noise to the input data, to iterations in a certain algorithm, or to the algorithm output. While most DP approaches assume a trusted aggregator of the data, local differential privacy allows each input party to add the noise locally; thus, requiring no trusted server. Finally, dimensionally reduction perturbs the data by projecting it to a lower dimensional hyperplane to prevent reconstructing the original data, and/or to restrict inference of sensitive information.

Differential Privacy (DP): A randomized algorithm M is ϵ-differentially private if for all S in the range of M, and for all data sets D and D' differing in one record:

$$Pr[M(D) \in S] \leq exp(\epsilon) Pr[M(D') \in S] \tag{2.1}$$

Hence, DP ensures that any sequence of outputs (response to queries) is essentially equally likely to happen, whether a certain record was included in the data set or not [16] (essentially is captured by the parameter ϵ). In recent practical publications [5, 17], the parameter ϵ was set to be a single digit (smaller values indicate better privacy).

Composition is an important property of DP which enables the design and analysis of complex DP algorithms from simpler DP building blocks. The composition of a sequence of k mechanisms, where the i^{th} mechanism provides ϵ_i-DP, is $(\sum_{i=1}^{k} \epsilon_i)$-DP (refer to Dwork and Roth [16] for proofs and more advanced composition theorems). It is not unnatural that the strength of the privacy guarantee would degrade with repeated use of the mechanism, but DP provides a way to quantify the privacy loss.

Adding δ to the right-hand-side of the above equation yields (ϵ, δ)-differential privacy (a relaxation with weaker privacy guarantees than pure differential privacy, i.e. ϵ-DP). The (ϵ, δ)-DP definition was proposed to capture the privacy guarantees of the Gaussian mechanism, and due to the applications of advanced composition theorems. To avoid compromising the privacy, the value of δ has to be less than the inverse of any polynomial in the size of the database [16]. Recently, alternative relaxations of ϵ-DP were proposed, e.g. Rnyi Differential Privacy (RDP) [18]. While accommodating the analysis of the Gaussian mechanism and advanced composition theorems, RDP could be preferred over (ϵ, δ)-DP since RDP does not allow a total breach of privacy with no residual uncertainty [18].

By definition, differential privacy (DP) can deter membership inference attacks. DP is also utilized by some distributed learning approaches to enable protection of the original data in case of multiple input parties. In addition, DP is immune to post-processing; meaning that even in the presence of auxiliary information, an adversary cannot increase the privacy loss. Thus, DP neutralizes linkage attacks used for de-anonymization.

From the definition above, its clear that DP algorithms are randomized. They can be categorized according to where, and how, the randomness is applied:

- ***Input Perturbation***: In this case, noise is added to the data itself, and after the desired non-private-computation is performed on the noisy input, the output would be differentially-private. Taking PCA as an example where Eigen-decomposition is performed on the covariance matrix, Dwork *et al.* [19] adds symmetric Gaussian noise matrix to the covariance matrix before performing Eigen-decomposition. The output would be a DP-projection matrix (the target here is not releasing the projected data, but the DP-projection matrix).

- ***Algorithm Perturbation***: Another approach is perturbing intermediate values in iterative algorithms. For example, Eigen-decomposition for PCA can be performed using the power method, which is an iterative algorithm. Hardt and Price [20] proposed adding Gaussian noise in each iteration of the algorithm which operates on the non-perturbed covariance matrix, leading to DP-PCA. Similarly, Abadi *et al.* [5] proposed a DP-Deep learning system by modifying the stochastic gradient descent algorithm to have Gaussian noise added in each of its iterations.

- ***Output Perturbation*** involves running the non-private-learning algorithm, and then adding noise to the generated model. For cases where adding noise to the computed output would destroy its value, the Exponential mechanism could be used. Given some utility function $u(D, r)$ that tells us how good an output r is on database D, the exponential mechanism selects an output r with probability proportional to this utility function (i.e., the output would be biased towards ones with higher quality). This method was used for achieving DP-PCA by sampling a random k-dimensional subspace that approximates the top-k PCA subspace [21].

- Finally, ***objective perturbation*** entails adding noise to the objective function for learning algorithms such as Empirical Risk Minimization [22].

The approaches mentioned above work on data hosted by a single server (a trusted server). To enable training on disjoint datasets held by multiple input parties, Papernot *et al.* [17] proposed to first learn an ensemble of teacher models from the disjoint datasets, use these teachers to make noisy predictions on public data, which can be used in turn to build a student model. The privacy loss is determined by the number of queries made to the teachers during the student training, and do not increase as end-users query the deployed student model [17]. This approach is not limited to a single ML algorithm, but it requires adequate data quantity at each location.

Local Differential Privacy: When the input parties do not have enough information to train a ML model, it might be better to utilize approaches that rely on ***local*** differential privacy (LDP). With LDP, each input party would perturb their data, and only release this obscure view of the data. An old, and well-known version of local privacy is randomized response (Warner 1965), which provided plausible deniability for respondents to sensitive queries. For example, a respondent would flip a fair coin: (a) if tails, the respondent answers truthfully, and (b) if heads, then flip a second coin, and respond Yes if heads, and No if tails. This version of randomized response (RR) is $(ln3)$-differentially private [16].

RAPPOR [23] is a technology for crowd-sourcing statistics from end-user client software by applying RR to Bloom filters with strong -DP guarantees. RAPPOR is deployed in Google Chrome web browser, and it permits collecting statistics on client-side values and strings, such as their categories, frequencies, and histograms. By performing RR twice with a memoization step in between, privacy protection is maintained even when multiple responses are collected from the same participant over time [23].

A ML oriented work, AnonML [24], utilized the ideas of RR for generating histograms from multiple input parties. AnonML utilizes these histograms to generate synthetic data on which a ML model can be trained. Like other local DP approaches, AnonML is a good option when no input party has enough data to build a ML model on their own (and there is no trusted aggregator).

Dimensionality Reduction (DR) perturbs the data by projecting it to a lower dimensional hyperplane. Such transformation is lossy, and it was suggested by Liu *et al.* [1] that it would en-

hance the privacy, since retrieving the exact original data from a reduced dimension version would not be possible (the possible solutions are infinite as the number of equations is less than the number of unknowns). Hence, Liu *et al.* [1] proposed to use a random matrix to reduce the dimensions of the input data. Since a random matrix might decrease the utility, other approaches used both unsupervised and supervised DR techniques such as principal component analysis (PCA), discriminant component analysis (DCA), and multidimensional scaling (MDS). These approaches try to find the best projection matrix for utility purposes, while relying on the reduced dimensionality aspect to enhance the privacy.

Since an approximation of the original data can still be obtained from the reduced dimensions, some approaches, e.g. Jiang *et al.* [25], combined dimensionality reduction with DP to achieve differentially-private data publishing. While some entities might seek total hiding of their data, DR has another benefit for privacy. For datasets that have samples with two labels: a utility label and a privacy label, Kung [26] proposes a DR method to enable the data owner to project her data in a way that enables maximizing the accuracy of learning for the utility labels, while decreasing the accuracy for learning the privacy labels. Although this method does not eliminate all privacy risks of the data, it enables controlling the misuse of the data when the privacy target is known.

2.6 Challenges and outlook

Despite the aforementioned techniques to protect the private data while performing ML training and/or testing, non-privacy-aware ML algorithms are still being widely used, and private data is still being uploaded to the cloud on daily basis. Current laws might force companies to declare that they are collecting data, and might even give the user the option to opt out of such data collection, but it seems like a zero or one decision. There should be an alternative option that utilizes some of privacy-preserving ML techniques already proposed by the research community, however, some issues might hinder achieving that.

The first issue is flexibility. Many of the aforementioned PPML techniques are tied to a certain ML algorithm. This would pose a problem especially considering the fact that new ideas and

advances in ML are being proposed on regular basis. Thus, some of these PPML techniques might need to be re-purposed regularly to cope up with the new advances. In such cases, transformed data release, the distributed approach by Papernot *et al.* [17], or local differential privacy might be favorable as they provide the opportunity to apply new advances in ML without extensive customization.

Another issue is scalability, both in terms of processing and communication costs. A similar problem happens with algorithms design where higher processing powers was accompanied by greater amounts of data; hence, new techniques were required. New advances in ML concentrated on efficiency, parallelism and reducing the communication cost. However, many PPML techniques impose additional processing and communication costs that might limit the ability to utilize the huge amounts of data available today. Promising PPML techniques could be those that are already built around distributed processing, and only exchanging summary statistics or model parameters.

In addition, security assumptions for some PPML systems need to be addressed properly. An example is non-collusion assumptions where two or more of the computation parties might be assumed not to collude. Such assumptions might be easier to justify when the different parties perform different roles; thus implying they could be different companies (such as the service provider and the PSP in Erkin *et al.* [10]). the question remains: who will take the role of the privacy service provider? And how will they make their profit to sustain their business?

Policies are yet another issue. Privacy policies can help the data owners specify which data is being shared, according to what rules or privacy guarantees, who will use the data, and for what purpose? It is important to determine if the policy will be enforced at the client side, meaning that it would be transformed to a form that limits all other uses, or at least eliminates some uses with known privacy threats. Another option would be sending the policy with the data to the computation servers where the servers have to be trusted to honor the policy rules. If the computation party and the results party are controlled by the same entity, trust would be an issue.

To summarize and present other issues, we can take the human data interaction (HDI) principles into consideration. While HDI applies to any type of data sharing, we concentrate here on

the specific case of Machine Learning. Mortier *et al.* identify three core themes of HDI [27]: Legibility, Agency and Negotiability. ***Legibility*** means informing data owners that their data is being collected, what data is being collected, how it is being used (including what inferences might be made), and potentially providing legible explanation of how their privacy is being preserved. While some companies, e.g. Google and Apple, started using local differential privacy for data collection, it might still be a challenge to explain the technology to the public, and the implications of their choice of specific on the users privacy.

Agency is concerned with enabling data owners to have control of their data in ML systems. Enabling data owners to opt out of data collection, or set rules and policies about how their data is being used, and even adjust some incorrect inferences that were made about them. One example would be machine unlearning [28] where the target was adjusting an ML model, that was built using wrong data about a data owner, and enable incremental unlearning without having to start the training process from scratch. Such techniques could be beneficial to help realize the right to be forgotten that was recently enforced by the European Union.

Since the definition of privacy (and associated threats) might evolve overtime, Negotiability in PPML systems is essential. ***Negotiability*** enables the data owners to re-evaluate their data sharing decisions (e.g. enabling them to withdraw from ML systems completely or partially, or to simply change the policies of using their data). As we design current systems, how can we make the policy definitions extensible to enable addressing unforeseen concerns in policies that were created to address todays concerns?

CHAPTER 3. RECONSTRUCTION ATTACKS AGAINST MOBILE-BASED CONTINUOUS AUTHENTICATION SYSTEMS IN THE CLOUD

Modified from a paper published in the IEEE Transactions on Information Forensics and Security, vol. 11, no. 12, pp. 2648-2663, Dec. 2016.

Mohammad Al-Rubaie [1] and J. Morris Chang

3.1 Abstract

Continuous authentication for mobile devices using behavioral biometrics is being suggested to complement initial authentication for securing mobile devices, and the cloud services accessed through them. This area has been studied over the past few years, and low error rates were achieved; however, it was based on training and testing using SVM and other non-privacy-preserving machine learning algorithms. To stress the importance of carefully designed privacy-preserving systems, we investigate the possibility of reconstructing gestures raw data from users' authentication profiles or samples testing results. We propose two types of reconstruction attacks based on whether actual user samples are available to the adversary (as in SVM profiles) or not. We also propose two algorithms to reconstruct raw data: a numerical-based algorithm that is specific to one compromised system, and a randomization-based algorithm that can work against almost any compromised system. For our experiments, we selected one compromised and four attacked gesture-based continuous authentication systems from the recent literature. The experiments, performed using a public data set, showed that the attacks were feasible, with a median ranging from 80% to 100% against one attacked system using all types of attacks and algorithms, and a median ranging from 73% to 100% against all attacked systems using the randomization-based algorithm and the negative support

[1] primary researcher and author. Mohammad is the primary author of this paper that conducted the research, performed the experiments, and wrote the paper under the supervision and advice of Prof. J. Morris Chang

vector attack. Finally, we analyze the results, and provide recommendations for building active authentication systems that could resist reconstruction attacks.

3.2 Introduction

Digital media access in the United States has been dominated by mobile devices since 2014. The time spent on mobile apps has increased to 52% of the total U.S. digital media time, compared to only 40% for desktop access [29]. Mobile browser usage formed the remaining 8%, adding to the total mobile device usage. Such reliance on mobile devices is coupled with an increase in using cloud storage services for storing personal data to enable easy backup, and easy access across multiple devices. However, such trends were not accompanied by an increase in the security of authentication to these services. This was clearly demonstrated by the iCloud hacking incident in 2014 that exposed victims' private data [30, 31]. It was not done from the victims' own mobile devices, and was a targeted attack on user names, passwords and security questions [30]. While two-factor authentication is normally suggested, it would be tedious for users to use these lengthy steps every time they need to access an online service. Hence, continuous authentication (also called active or implicit authentication, and used interchangeably hereinafter) is a possible solution to this issue as it is transparent to users while providing security based on their behavioral biometrics.

Active authentication (AA) on mobile devices has received significant attention over the past few years with solutions utilizing touchscreen gestures ([32, 33, 34, 35]), keystrokes ([36]), sensors ([37]) or more than one input type (e.g., [38]). With the exception of papers like [39], almost all active authentication research on mobile devices utilized regular machine learning methods that didn't preserve the privacy of users. Almost all of them used Support Vector Machines (SVM) for classification, either as the only algorithm, or one of few. Systems that used SVM achieved lower error rates than the few ones that used privacy-preserving methods.

Continuing to rely on SVM, or similar algorithms, without studying their effect on users' privacy and security might give a premature expectation that AA systems are ready to be widely deployed. It is important to notice that SVM (and k-NN) stores actual feature vector samples in

user authentication profiles. In each user's profile, some of these samples belong to the user herself: positive support vectors (PSVs). The rest of the samples belong to other users: negative support vectors (NSVs). Each feature vector contains information that represents an object's important characteristics. For gestures, this information (features) might include average velocity and trajectory length. Using such features to forge raw data has not been addressed in the literature and, up till now, it was unclear how to perform this reconstruction (as was also reported by [40]). It is worth noting that reconstructing fingerprint images from minutiae templates was proven to be possible although these templates are very compact, and many did not think it was possible at first [41, 7, 42, 43]. We believe that it is important to understand the possibility of reconstructing raw data from users' authentication profiles before designing secure and privacy-preserving AA systems.

In this paper, we investigate the problem of reconstructing behavioral biometrics on mobile devices. Since gestures received the most attention in the recent literature ([32, 33, 34, 35, 44, 45, 46, 47, 48, 49, 50, 40, 51, 52, 53]), it was chosen to be studied in this paper. We foresee that cloud services would start adopting behavioral-based AA systems for mobile devices to prevent attacks like the iCloud hacking incident mentioned earlier [30, 31]. Such services might be possible targets for successful attacks. In fact, even large companies were targeted by successful attacks that revealed both personal and corporate private data [54]. If one of these cloud web services that uses behavioral biometrics for authentication was hacked, a user with an account on these systems might have their behavioral biometrics-based profiles compromised. If these profiles were not privacy-preserved, a reconstruction attack could potentially be launched to reconstruct the raw data that could be used afterwards to hack into that user's accounts on other cloud services. Furthermore, we investigate the possibility of reconstructing raw gesture data even in the absence of user samples in their profiles. This could be done by utilizing the output of user samples' testing such as the SVM decision value or the logistic regression probability. Such type of attack can even be launched by an adversary with no access to the AA server.

Unlike passwords or digital certificates, behavioral biometrics cannot be revoked once compromised, and the users' behavior might not change quickly enough overtime for the compromised

profiles to be obsolete [55]. Hence, we intend to highlight the importance of using carefully designed privacy-preserving AA systems by studying reconstruction attacks.

To the best of our knowledge, this is the first work that proposes reconstruction attacks against gesture-based AA systems. The contributions of this paper are as follows:

First. We propose two types of reconstructions attacks depending on the amount of private information leaked to the adversary. The first one (full profile attack) closely resembles the current usage of SVM in AA systems as it assumes that both PSVs and NSVs are stored in the clear within the authentication profile. These are the actual feature vectors that belong to the profile owner herself, and other users, and can possibly be utilized to reconstruct raw data that is used to hack into user accounts on other systems.

On the other hand, the decision value attack assumes that none of the user samples were available for the adversary (since he has no access to the AA server or the victim's own device). Alternatively, this adversary could submit a user sample to the AA server for testing and retrieve a decision value. To perform the decision value attack, we developed an algorithm that starts with a generic feature vector, and randomizes it while monitoring the change in the SVM decision value. This allows the algorithm to finally obtain feature vectors that can be used to reconstruct raw data using our reconstruction algorithms. The decision value attack shares the same target of recovering feature vectors as the model inversion attack proposed in [3] (section 5.3). However, their algorithm is based on gradient descent while ours is based on randomization.

Second. We developed two reconstruction algorithms for mobile gestures in order to assess the vulnerabilities of AA systems in both proposed attack scenarios. Both of these algorithms take feature vectors as input and output raw gesture data. The first algorithm is numerical, and we use numerical estimation and procedures on the summary statistics contained in the feature vector to reconstruct the raw data. We further test our reconstructed raw data against the user's profile, and only use the raw gesture data that passes as user's samples.

When designing a numerical algorithm, it has to be tailored to a single compromised system. Hence, we developed a randomization algorithm that can reconstruct raw data from almost any

compromised system. This algorithm has the added benefit of reconstructing raw data that can yield the closest feature vector possible to the original feature vector. The randomization algorithm starts with generic raw data, and continuously randomizes it while monitoring how close the extracted feature vector is to the original one. It stops when no further randomization can help reduce the mean square error (MSE) between the reconstructed feature vector and original feature vector.

Third. We tested both algorithms and the two types of attacks using two mobile gesture classifiers: left-to-right and right-to-left swipes. For this purpose, we used a public data set that was provided by Frank et al [35]. We reviewed the literature to select one AA system as a compromised system, and four others to be the attacked systems against which the reconstructed raw data would be used. Our tests revealed the feasibility of reconstruction attacks using the proposed algorithms. The experiments showed high attack success rates with a median ranging from 73% to 100% against all attacked systems using the randomization algorithm and the full-profile attack (with negative support vectors).

Fourth. We further analyze the results, and provide recommendations for future design attempts of AA systems that could resist our proposed reconstruction attacks. In addition, we show the effectiveness of these recommendations through experiments.

The remainder of this paper is organized as follows. We review the related work in Section 5.3. In Section 3.4, we introduce the threat model and the two reconstruction attacks. The numerical and randomization reconstruction algorithms are presented in section 3.5. Section 3.6 presents the experimental results and their analysis while section 3.7 concludes this paper.

3.3 Related Work

Attacks on behavioral biometrics Non-zero effort attacks on behavioral biometrics did not receive as much attention as it should have. On gesture-based AA systems, Phoha et al [56] used a simple "lego" robotic arm to forge gestures based on general population statistics. Their results indicated a noticeable increase in EER when considering their non-zero effort attack instead of

the zero-effort attacks normally considered in the literature[2]. However, we have a different threat model, as we target online services protected by a gesture-based AA system, while they targeted information stored on owners' devices. Moreover, they used their own feature vector while we select five from the literature: one to be a target of our reconstruction attack, and four others for testing. More recently, Gong et al [40] proposed a forgery-resistant method that works especially well against robotic arm forgeries, by utilizing the impact of screen settings on users' behavior. However, it does not address our threat model.

Model Inversion Attacks Fredrikson et al [3] proposed model inversion attacks that used the confidence value revealed along with a classification result to recover feature vectors that were used to build the model. Their case studies used the raw data as features. In their face recognition case study, they recovered feature vectors which were the face images themselves that were used for training the models. Whereas in gesture-based active authentication, gesture raw data cannot be used directly as feature vectors. Therefore, we develop reconstruction algorithms to reconstruct the raw gesture data from feature vectors. This step was not performed in [3] since the feature vectors matched the raw data in their case studies.

Reconstruction Attacks While being in another biometrics-research area, the closest work to ours is that related to reconstructing fingerprints using minutiae-based templates. Minutiae-based templates are very compact, and for a long time, it was assumed that reconstructing the original fingerprint image from them was not possible. That was challenged by [42] and [43], and was later followed by other successful attempts like [41]. The closest of these attempts to our work are likely those similar to [7] where a minutiae-based template is used to reconstruct another type of representation, namely phase image in [7]. This template is later used to reconstruct the fingerprint greyscale image. Its similarity to our work stems from its use of one type of representation to create another type of representation. We might think of different feature vectors as different representations of the raw gesture data. To the best of our knowledge, ours is the first research

[2]Zero-effort attacks assume that adversary has not made any effort to obtain any auxiliary information about the victim. In contrast non-zero effort attacks assume the adversary obtained some prior information about the victim, e.g. watched them swipe to unlock their phone; thus, learning the swiping pattern of the victim.

that addresses the issue of reconstructing the raw gesture data from feature vectors. We add the extra challenge of using the reconstructed raw gesture data to attack additional AA systems that utilize different feature vectors than the compromised one.

Gesture-based active authentication. Gesture-based AA research is presented in this subsection. Our aim is to select a group of systems that can be utilized for testing our hypothesis. For papers to be selected, they had to clearly define a feature vector which is required by SVM (and other machine learning algorithms like logistic regression, k-NN and neural networks) since we are studying the privacy vulnerabilities of such algorithms. Moreover, the features in the selected systems had to be diverse to be able to study the effect of having different features on our reconstruction attack algorithms. After examining the recent literature, we were able to find a group of papers that matched the previous criteria and had other desirable characteristics: (1) they were less susceptible to over-fitting as their feature vectors were not too long; (2) they were flexible as they enabled testing a single sample, as well as aggregate testing of more than one (as opposed to some papers that only allowed aggregation of testing samples), and (3) they did not use special equipment or testing conditions.

We start by presenting the papers that matched our criteria mentioned above. In [35], the authors selected features that represented location, shape, velocity, and acceleration of strokes. Pressure and area only had one feature each. They used SVM and k-NN for classification to obtain approximately 13% EER for one-stroke, and 0-4% median EER when bundling 11 strokes together for testing. Li et al [34] did a study on gestures and taps. After feature selection, they chose 10 gesture features that represented location, shape, duration, and area of strokes. They used SVM for classification. In [32], an evaluation of 10 classification algorithms was done. Logistic regression and SVM were found to be the two best performing classifiers. They used 28 features that described location, shape, velocity, acceleration, area and pressure of strokes. Xu et al [38] performed a study on gestures, taps, pinches, and handwriting on touch screens. As for gestures, they chose 37 features representing location, shape, velocity, area and pressure of strokes, and used SVM for classification. In [57] and [33], the authors chose 15 features that were a subset of the features in [35]; and used

k-NN, SVM and Random Forest for classification. The previously mentioned papers were selected for our testing purposes since they all mentioned their feature vectors clearly. They also matched our selection criteria including diverse selection of features.

The papers that were not selected include [44] and [45] which relied on graphical techniques to create a graphic touch gesture feature (GTGF) in [44], and improved the performance on mobile devices in [45]. It was not selected as it did not use a regular feature vector. There was a similar case with [46] that used HMM-based classification. Velten et al [47] had a much longer feature vector than the number of samples per user that we have in the public data set from [35], so we couldn't use it as it would lead to overfitting. In [48], 10 minutes worth of samples were needed for a decision, allowing intruders long time to carry their attacks. The features from [35] were used in both [49] and [50], so we excluded them. The features were used in [49] to test new classification algorithms, and in [50] to test application-centric active authentication. The features in [40] were also the same as in [35] but were used with different screen settings to resist forgery attacks; however, we did not have the appropriate data set to do the testing. A similar issue with the data set happened in [51] and [52]. In [51], a glove was used to complement the features collected from the touch screen, and in [52], the authors proposed analyzing touch interactions with common user interface (UI) elements (e.g., button, checkboxes and sliders). Finally, [53] was not selected because of the absence of a detailed list of features.

3.4 Preliminaries and Attacks Overview

In this section, we start with a brief description of classification using SVM, since we are targeting systems that use SVM and similar machine learning algorithms. We follow that by describing the system model and attack scenarios. Finally, we present our choice of the compromised and attacked systems.

3.4.1 SVM Classification and System Evaluation

To perform classification, *raw* data belonging to multiple classes (e.g., different users) need to have its features extracted to create *feature vectors*. For mobile gestures, such features may include trajectory length and average velocity. Before using these feature vectors to train an AA profile, normalization (or standardization) can be applied, which might include normalizing the feature values to be between -1 and +1, for example. In a binary classification problem, we want to know if the new sample $\mathbf{x}$ (feature vector) belongs to the user k (positive class) or not (negative class). Thus, when training profiles, for each user, all their training data is placed in the positive class, and all other users' training data is placed in the negative class (one-versus-all classification).

Support Vector Machines (SVM) [58] is a maximal margin classifier, meaning that it tries to maximize the margin (distance) between classes, rather than merely choosing any separating hyperplane. After solving the SVM problem in its dual form, the support vectors are identified. Some of these support vectors belong to the positive class while others belong to the negative class (formed from many other classes). Thus, any user's profile is in reality formed of actual user samples. In fact, even if some user leaves the system, and her profile is deleted, some of her samples might still be left as NSVs in other users' profiles. Using these support vectors, any new sample $\mathbf{x}$ can be tested with the following equation:

$$f(\mathbf{x}) = \sum_{i:\alpha_i>0} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \tag{3.1}$$

The sample $\mathbf{x}$ is then classified based on its SVM decision value (returned by eq. 3.1) either through varying a threshold, or based on the sign of the SVM decision function $f(\mathbf{x})$, which is equivalent to setting the threshold to zero.

AA systems are normally evaluated based on false acceptance rate (FAR) and false rejection rate (FRR). There's a tradeoff between FRR and FAR, and by varying the threshold used with the decision value from eq. 3.1, one can obtain different values for FAR and FRR. Some systems, or individual users, might want to have a smoother user experience by decreasing FRR at the expense of FAR, while others might want more security than convenience by decreasing FAR at the expense

of FRR. In order to compare different systems more easily, equal error rate (EER) is used, and it is the value where FAR equals FRR. The value of EER can be determined by varying the SVM decision value threshold.

3.4.2 System Model and Design Goals

3.4.2.1 System Model

We consider a system with two main entities: a mobile device and a cloud service provider. The cloud service provider has an authentication server and an application server. A mobile device user wants to gain access to the application server, while ensuring that no intruders can obtain access to her account. Hence, the cloud service provider is using an AA server to continuously and transparently authenticate the user. An active authentication system consists of the following entities (fig. 3.1):

AA Server: provides a transparent way to continuously authenticate the user based on touch gestures. Since these systems protect cloud services, and might be accessed using different devices, the user profiles have to be stored on the authentication server itself. The user profiles were built using SVM since most papers report the efficacy of their systems based on SVM.

Client App: The user uses this app to gain access to the cloud services. This app collects raw gesture data R, extract the features from R, and sends the feature vector F to the AA server to be matched against the stored profile P. The decision value $decVal$ is returned to the app that would allow or prevent the app user based on that decision value.

3.4.2.2 Design Requirements

Extending the design goals given in [59], we outline the following four requirements:

Biometric Viability: A biometric modality that is used for continuous authentication has to satisfy several requirements including being universal (i.e. it applies to everyone), unique, *permanent* (i.e. it lasts at least through the lifetime of the system), unobtrusive, difficult to circumvent,

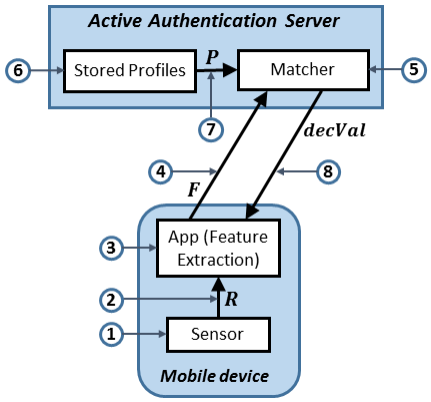


Figure 3.1: Attack points in AA systems

cost effectiveness, and collectible [60]. The viability of using gestures as a biometric was shown in many studies including [35].

Security and Privacy Requirements: The primary requirement of an AA system is to prevent illegitimate users from continuing to use the app. Thus, it protects the original user's private data in the cloud, which makes it a security and privacy requirement at the same time. Another privacy requirement is ensuring that none of the raw gesture data is sent to the AA server. Thus, if the AA server is compromised, hackers would not have access to such data [40]. Instead, only the feature vector is sent to the AA server (fig. 3.1).

Usability Requirement: Attempting to make the system more resistant to adversaries would require tuning the threshold to decrease FAR. However, decreasing FAR could potentially increase the FRR which would be obtrusive to usability. Hence, it would make sense to allow the user to tune her threshold to allow for more usability or security.

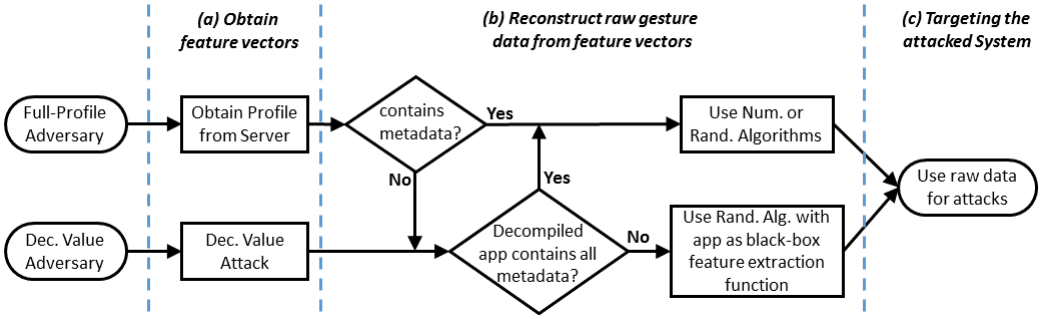


Figure 3.2: Overview of attack procedures

3.4.3 Problem Setting and Threats

When password databases are compromised [61, 62], the damage can be limited to the related service. However, the permanence property of biometrics could jeopardize user accounts on other services, and might even prevent that user from using said biometrics in the future.

We foresee the existence of multiple online services that implement gesture-based AA systems. Each one of these services would implement their own AA system with a separate set of features. A user u can have accounts on more than one online service (e.g., Amazon, Google and eBay). She might also have accounts for other online services that also implement AA systems, but might have less secure infrastructure, which are called here the compromised systems $SysC$.

An adversary could use information leaked from $SysC$ to obtain raw gesture data R, which in turn can be used to access the victim's account on another online service provider, e.g. eBay or iCloud. The adversary might want to access the victim's account to perform purchases (eBay or Amazon), or to to gain access to their private data stored online (iCloud, Dropbox, or Gmail). We call these the attacked systems $SysA$.

As stated in Ratha et al [63], there are eight attack points on pattern recognition systems. We show them in fig. 3.1 as they apply to a mobile-based AA system. It is worth noting that our attack regarding $SysA$ are limited to *type* 1 (fig. 3.1), i.e. injecting forged raw data to the sensor, which is possible through replaying events directly within the mobile device (see [64]). We do not require any vulnerabilities in $SysA$ for our attacks to happen. As for $SysC$, we defer the discussion to the next section.

3.4.4 Adversary Model

The goal of the adversary is to utilize the biometric ***permanence*** property to gain access to a victim's cloud stored data on $SysA$ by using the victim's leaked biometric information from $SysC$.

We present two types of adversaries associated with two types of attacks: the full-profile attack and the decision value attack. Each one of these attacks has two stages: the first stage is where the adversary obtains his knowledge, uses different methods to do so for each attack. It basically

comprises (a) obtaining feature vectors, and (b) reconstructing raw gesture data from these vectors (fig. 3.2). The second stage, which corresponds to step (c) in fig. 3.2, is similar for both adversary types. It includes injecting the raw data into the sensor while using the attacked system app. As stated before, this is the *type* 1 attack point according to [63], and does not require the attacked system to have any vulnerabilities.

Before describing each adversary type, it should be noted that the numerical reconstruction algorithm (section 3.5.1) requires having the features metadata beforehand. This metadata contains the features' names and normalization parameters. On the other hand, the randomization algorithm can accept a black-box feature extraction function. Such black-box functionality can be achieved by the app itself. The adversary, using its account, can inject randomized raw data into the sensor and monitor the output feature vectors sent to the cloud. Even if the traffic is encrypted, a Man-in-the-middle HTTPS proxy could be utilized, and such software is readily available such as Fiddler, Charles or mitmproxy.

We characterize each of the two adversaries by describing how their knowledge is obtained (fig. 3.2):

- Full-profile (FP) adversary: This could be an internal entity with respect to the system with a legitimate system role (e.g., system admin, an employee). It can also be an external entity that hacked into the AA server (similar to the Sony hack [54], or the Rockyou [61] and CSDN [62] password database leaks). This adversary targets the privacy of the victim by overcoming the privacy safeguard of not uploading the raw gesture data to the AA server. If the features' metadata was available with the profile, or can be obtained through decompiling the app, this would be a passive adversary with respect to $SysC$; otherwise, it would be active adversary.

- Decision value (DV) adversary: This is a weaker adversary, and a relaxation from the previous one. We assume the adversary has no access to the victims mobile device or the AA server. The adversary operates a mobile app, impersonates a victim, and continuously sends feature vector samples to the AA server; thus being an active adversary, and monitors the returned decision value (e.g., by using MITM HTTPS proxy). This adversary targets the victims

privacy by utilizing the decision value which is used mainly used for ***usability*** reasons. To obtain the metadata, this adversary can decompile the app (e.g., using a website for Android apps [65]). The adversary can also resort to using the app as a black-box feature extraction function as mentioned earlier.

An adversary is deemed successful in their attack when they can impersonate the victim. It is unrealistic to assume that an adversary can obtain success rates greater than $(1 - EER)$. Hence, if the success rate is equal to, or greater than, $(1 - EER)$, then we call the attack "successful". In order to decrease the EER, many studies aggregated results in the form of taking a majority vote of a group of gesture strokes. Since a majority can be obtained using 3 out of 5 gesture strokes (e.g. as was done in [38]), we call an attack "conditionally successful" if it had a success rate ranging from 0.6 to $(1 - EER)$.

3.4.5 Adversary Knowledge

This section corresponds to step (a) in fig. 3.2. We elaborate more on how each adversary obtains the feature vectors necessary for reconstructing the raw gesture data of the victim.

Full Profile Adversary: As stated earlier, this adversary gained access to the victim's profile which are regular SVM profiles as described in the system model (section 3.4.2.1). Such profiles contain both positive (PSVs) and negative support vectors (NSVs), and any other values needed for the SVM decision function in eq. 3.1.

For each user's profile, ***Positive support vectors*** are feature vector samples that belong to that specific user. If any of them is used with the SVM decision function (eq. 3.1), the decision value would most likely equal +1 (or smaller with soft margin classifiers). To perform full profile attacks using PSVs, these samples can be directly extracted from these profiles, and used as input for our reconstruction algorithms. Other user samples that weren't selected to be PSVs might be more representative of user's behavior as they can give decision values that exceed +1. Some of these samples might be used as NSVs for other users' models.

Each user's model also contains ***Negative support vectors***. These are feature vector samples from other users that are used to help set the boundary of the negative class. They would most likely have decision values that equal -1 (or larger with soft margin classifiers). Our method involves: (1) collecting all the negative samples from all available profiles; (2) removing any samples that already exist as PSVs for any profile (To isolate the effect of only using NSVs); and finally (3) testing the remaining NSVs against each user's profile using the SVM decision function (eq. 3.1). The NSVs that yield decision values greater than +1 for a particular user are selected to be used in our attacks. We anticipate that these samples would give better attack results than the PSVs as they are more representative of user's behavior, especially that their decision values always exceed those of the PSVs for a particular user.

Decision Value Adversary: To synthesize (generate) a victim's feature vectors, this adversary can use algorithm 1. This adversary would basically inject randomized feature vectors into the upstream at attack point 4 (fig. 3.1), and monitor the returned decision value at attack point 8.

Algorithm 1 starts with any feature vector (all zeros or extracted from a generic gesture by the attacker), randomizes the features, and tests the feature vector against the user profile to see if the current decision value is greater than the previous best decision value. If that is the case, the algorithm updates the feature vector to the new randomized one. This process continues until the algorithm reaches the target decision value V_{th} (set in our algorithm to be greater than +1 to obtain more representative samples of user behavior than PSVs).

The randomization of the feature vector is done by adding Gaussian noise to each feature independently. The additive noise has zero mean and a standard deviation, σ, controlled by the value *sRatio* (line 7 in algorithm 1). σ is dependent on the features standard deviation, and we set it to 1 in our case, since that is the standard deviation of our standardized feature vectors. $sRatio \in (0, 1]$ is a design parameter that scales the value of σ; thus controlling the added noise which would affect how quickly and effectively the algorithm reaches its target. Setting *sRatio* will be discussed in section 3.6. Furthermore, to limit the amount of randomization as the algorithm

approaches its target value, $sRatio$ is decreased after each successful update since it means that the algorithm is one step closer to reaching its target. Hence, the feature values do not need to be changed as much as the previous steps to reach its final target.

With this method, as many samples as needed can be created. Therefore, additional steps were taken to ensure having reconstructed raw data that can be useful for performing the attack (steps 13-16 in algorithm 1). The synthesized feature vector is passed into any of our two reconstruction algorithms (section 3.5). The resulting raw data would then have its features extracted, and tested against the user's profile. We only use reconstructed raw data that passes this test.

Algorithm 1 Generate synthetic feature vectors for user u

1: **function genSyntheticFV**($model_u$, $initF$, V_{th}, $initR$, σ)
Input: The classification model $model_u$ of user u
Input: Initial feature vector $initF$
Input: The SVM decision value threshold we want to reach V_{th}
Input: The initial ratio $initR$ controlling the additive noise
Input: The standard deviation σ of the added noise
Output: The synthetic feature vector sF_u generated for user u
2: **do**
3: $sRatio = initR$;
4: $sF_u = initF$;
5: $decVal = testFV(model_u, sF_u)$;
6: **while** $decVal < V_{th}$ **do**
7: $rndF = sF_u + normRand(0, sRatio * \sigma)$;
8: $tmpDecVal = testFV(model_u, rndF)$;
9: **if** $tmpDecVal > decVal$ **then**
10: $sF_u = rndF$;
11: $decVal = tmpDecVal$;
12: $sRatio = 0.999 * sRatio$;
13: $recGesture = reconstructRawData(sF_u)$;
14: $recF = extractFeatures(recGesture)$;
15: $recDecVal = testFV(model_u, recF)$;
16: **while** $recDecVal < V_{th}$
17: return sF_u;

3.4.6 Choice of Systems and their features

Out of the five gesture-based AA systems ([35, 32, 33, 34, 38]) that we selected in section 5.3, we chose the features from Frank et al[35] to represent that of the compromised system (called hereinafter $SysC$). This system has achieved low error rates which might make it a good candidate

for deployment. It was also previously chosen by other researchers when testing new idea or machine learning algorithms [49, 50, 40]. An important reason behind our choice was that they only had one feature each for both pressure and area (table 4.1). This would enable us to study the effect of using the reconstructed raw data to attack other systems that have more reliance on pressure and area.

The remaining four: Serwadda et al [32], Antal et al [33], Li et al [34] and Xu et al [38] were selected to be the attacked systems, and will be called $SysA1$, $SysA2$, $SysA3$ and $SysA4$ respectively. In table 3.1, we show the number of features of the attacked systems which are similar or different to the compromised system. Table 4.1 shows all five systems and the distribution of their features according to 8 categories. It can be seen from tables 3.1 and 4.1 that the selected attacked systems ([32, 33, 34, 38]) have varying levels of differences with our chosen compromised system [35]. These differences would enable us to study the effect of having different features on the proposed attacks. For example, $SysA1$ and $SysC$ are different in more features than they share. Moreover, $SysA4$ only shares 7 features with $SysC$, and differs in 30 features.

Table 3.1: Similar and different features to $SysC$ [35]

	Similar	Different
$SysA1$ [32]	11	17
$SysA2$ [33]	15	0
$SysA3$ [34]	5	5
$SysA4$ [38]	7	30

Table 3.2: Distribution of features for different systems

	$SysC$	$SysA1$	$SysA2$	$SysA3$	$SysA4$
Location	4	4	4	2	6
Shape	9	3	7	4	13
Velocity	5	5	1	-	4
Acceleration	4	5	-	-	-
Pressure	1	5	1	-	5
Area	1	5	1	3	5
Time	2	1	1	1	3
Others	3	-	-	-	1

3.5 The Reconstruction Approach

Raw data of each touch gesture stroke is composed of a series of vectors $(S_1, S_2, ..., S_N)$ with N being the number of touch events. Each touch event produces a vector, S_i, of the following values: location of the point (represented by X and Y coordinates), time stamp, pressure, area, phone and finger orientation [35]. The number of touch events (N) varies for different gesture strokes based on multiple factors including the length and duration of strokes and the type of the mobile device. Out of the raw data, feature extraction has to be done to create a feature vector. Normally, feature vectors do not contain any indication of the original number of touch events (N) from which it was extracted. Hence, in our algorithms, we allow passing this number as an argument, and we denote it as *length* as shown in algorithms 2-4.

The chosen compromised system [35] has 34 elements in the feature vector. Out of these features, there are five that will not be used in the feature vector, and they are PhoneID, UserID, Document ID, direction flag, and direction of end-to-end line. The direction flag is used in our case as a classifier as we distinguish between left-to-right and right-to-left strokes; as opposed to [35] that treated both as a single classifier called horizontal strokes. The remaining 29 features from [35] are listed below (along with their abbreviated names that will be used in the algorithms below):

- Location and Shape: Start X and Start Y ($startP$), End X and End Y ($endP$), Direct End-to-End Distance, Mean Resultant Length, largest deviation from straight line (LD), 20%, 50% and 80% percentiles of deviation from end-to-end line ($prctD$), average direction, length of trajectory, and ratio of end-to-end distance.

- Velocity and Acceleration: 20%, 50% and 80% percentiles of pairwise velocity ($prctV$) and acceleration, median velocity at last 3 points, average velocity ($avgV$), median acceleration at first 5 points.

- Pressure: Mid-stroke pressure ($midP$).

- Area: Mid-stroke area covered ($midA$).

- Time stamps: Stroke Duration, and inter-stroke time.
- Others: phone Orientation (seems to be fixed at 1, and mid-stroke finger orientation and change of finger orientation (both seem to be fixed at 0).

In the listing above, the features with no abbreviations within the parentheses were not used in our numerical approach (section 3.5.1). Some features were not included for being redundant like the direct end-to-end distance that can be calculated from other features, or the percentiles of pairwise acceleration as we already use those of the velocity.

In the next two subsections, two approaches will be presented. The first would be based on starting with the features, and using numerical estimation and procedures to reconstruct the raw data (section 3.5.1). Whereas the second would be based on randomizing the raw data until we get a feature vector that is close enough to our target feature vector (section 3.5.2).

In both approaches, we start with a feature vector F, the required gesture events *length*, and we finally compute the reconstructed raw data matrix *recGesture*. For this to happen, the following four sub-tasks are needed:

- ReconstructXY: reconstruct X and Y coordinates based on location and shape features (sections 3.5.1.2 and 3.5.2.1).
- ReconstructTime: reconstruct time based on time, velocity and acceleration features (sections 3.5.1.3 and 3.5.2.2).
- ReconstructPA: reconstruct pressure and area based on pressure and area features (section 3.5.1.4).
- ReconstructOthers: reconstruct other raw data like finger and phone orientation (section 3.5.1.4).

3.5.1 The Numerical Approach

With this approach, we use the summary statistics contained in the feature vector to estimate the original raw data. This estimation is done using procedures which aim to generate raw data that

has the same characteristics as the original, so that this reconstructed raw data could produce a feature vector that would resemble the user's behavior. This is particularly true for reconstructing time and location vectors. Since the feature vector does not include sufficient information for pressure and area (with one feature only each), we rely on the general population touch behavior to estimate the gesture's pressure and area data.

In the following subsections, we start by describing how to estimate values from percentiles (section 3.5.1.1) which is needed for both $reconstructXY()$ and $reconstructTime()$ functions (sections 3.5.1.2 and 3.5.1.3). We finally describe reconstructing pressure, area and other values in section 3.5.1.4.

3.5.1.1 Estimating values from percentiles

There are 3 sets of percentiles (largest deviation from straight line, velocity and acceleration) in the compromised system feature vector. These can provide information about the values from which these percentiles were calculated. Hence, a function is needed to estimate all the values from 3 percentiles only. For this purpose, we chose to estimate the points using shape-preserving piece-wise hermite cubic interpolation (PCHIP). It was found to perform better than other options like linear interpolation especially since linear interpolation cannot extrapolate the points outside its range. PCHIP also preserves monotonicity and the shape of the data [66]. Since we are using MATLAB, we use its implementation of this function (denoted as $estimateValues()$ in our algorithms). It takes the three percentiles and the required number of values as inputs, and returns the estimated values.

3.5.1.2 Reconstructing X and Y coordinates

We mainly utilize the deviation from straight line features to reconstruct X and Y. This is the perpendicular distance between the end-to-end straight line and the trajectory. It can have a positive or a negative sign based on whether it is above or below the straight line. In fig. 3.3, the largest absolute deviation LD is shown for the gesture denoted "Original Gesture". In the same

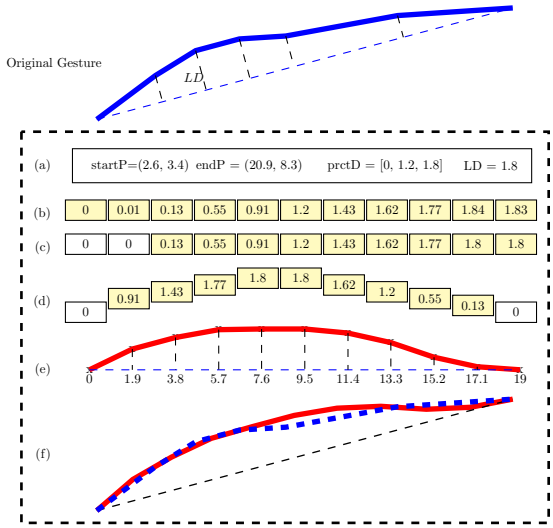


Figure 3.3: Reconstruct X & Y

gesture, it can be seen that all deviation values are positive. In other gestures, they can be all negative, or a mix of both.

Observing the red gestures in e and f of fig. 3.3, it can be noticed that they have the same shape, and that one of them is the rotated version of the other. The deviation values would be the same for both, and they can represent the Y values for the horizontal version, e. Hence, we start with a horizontally laid gesture that starts at the origin $(0, 0)$. After reconstructing all the X and Y values for it, we rotate and shift it back to match the original gesture.

We demonstrate our algorithm using the graphical example shown in fig. 3.3: (a) we only start with the features shown in the fig. 3.3; (b) the function $estimateValues()$ (section 3.5.1.1) is used to estimate the deviation values from the three deviation percentiles; (c) since the start and end points of any gesture have zero deviation, the two smallest absolute deviations are set to zeros. Also, any values exceeding the largest possible deviation are set to LD; (d) the values that were originally in ascending order have to be arranged so that the gesture would approximately have a bell shape to resemble normal gestures; (e) now that the deviations values are arranged appropriately, they need to be matched to X values. These are selected to be equidistant values from 0 to the total gesture straight line distance from $startP$ to $endP$; (f) finally, the horizontal gesture from (e) is

rotated and shifted to match the original gesture based on its starting and ending points ($startP$ and $endP$) given in the feature vector.

3.5.1.3 Reconstructing time vector T

Having the pairwise distances between all points, and the pairwise velocities, makes it possible to calculate the pairwise timestamps. Algorithm 2 starts by computing the pairwise distances from the X and Y values previously generated. The pairwise velocities can be estimated using $estimateValues()$ but the returned values are sorted in an ascending order, and need to be organized according to velocity rate of change characteristics before being used to find the pairwise timestamps.

Unfortunately, features like the median acceleration at the first 5 points does not contain helpful information about velocities at the beginning, middle or end of the stroke. It would mean different thing if it were extracted from a 5-points gesture as opposed to a 50-points gesture. Hence, we rely on general population statistics (from public data sets) in the function $arrangeVelocities()$ to arrange the estimated pairwise velocities. Based on aggregate relative velocity figures, we found that the smallest velocity comes first followed by the rest of the velocity values sorted in descending order. The function $arrangeVelocities()$ sorts the pairwise velocities this way.

In order to have the average velocity of the reconstructed raw data match that of the original feature vector, we adjust all the pairwise velocities in line 5 of algorithm 2. After that, calculating the pairwise time is followed by setting all the elements of the time vector T starting from zero with increments that equal the pairwise timestamps.

Algorithm 2 Reconstruct time vector T

1: **function reconstructTime**$(prctV, avgV, X, Y, length)$
2: $distances = getPairwiseDistances(X, Y)$;
3: $valuesV = estimateValues(prctV, length - 1)$;
4: $valuesV = arrangeVelocities(valuesV)$;
5: $valuesV = valuesV + (avgV - mean(valuesV))$;
6: $pairwiseT = distances./valuesV$;
7: $T = setTime(pairwiseT)$;
8: $returnT$;

3.5.1.4 Reconstructing other values

Pressure and Area were grouped together in the function $reconstructPA()$ because we use similar methods to reconstruct them (as there is only one feature each in the compromised feature vector). These features are mid-stroke pressure $midP$ and area $midA$. Whereas we discuss the pressure data reconstruction here, the same discussion and operations are applicable to reconstructing the area raw data A from the mid-stroke area $midA$.

Since only the mid-stroke pressure $midP$ is available in the feature vector, and the user pressure behavior is unknown, we take a look at general population statistics to find out how gesture pressure changes. In fig. 3.4, we show some gesture pressure data taken from public datasets. Since gesture strokes do not have a fixed number of points, the pressure data for all strokes was interpolated to have the same number of points, which was 11. The interpolated pressure values were then scaled from 0 to 10. Figure 3.4 shows the average overall relative pressure as well as the average of one of the users. It also shows two samples of gestures. It can be noticed that "Avg. Overall", "Avg. for User 4" and "sample 2" all have a similar rate of change relative to the pressure in their mid-points. Hence, we conclude that knowing the overall pressure rate of change would help find the other pressure values from $midP$. "Sample 1" on the other hand cannot be predicted using $midP$ but we can never accommodate all exceptions to the rule as this would require more data than a single point pressure value.

Therefore, we rely on the rate of change curve of "Avg. Overall" pressure in fig. 3.4, and scale it using the maximum and minimum possible values to be between 0 and 10 to obtain the vector ($avgPressV$). Having only one value ($midP$) the pressure vector P is found using the equation:

$$P(i) = \frac{midP}{avgPressV(\lfloor \frac{N}{2} \rfloor)} * avgPressV(i) \tag{3.2}$$

where $i \in [1, 2, ..., N]$ and N is the gesture length.

As for the other values in the raw data matrix, we noticed that they do not change. These include phone orientation and finger orientation. Therefore, the function $reconstructOthers()$ only returns vectors of the $length$ required that hold the values found in the feature vector F.

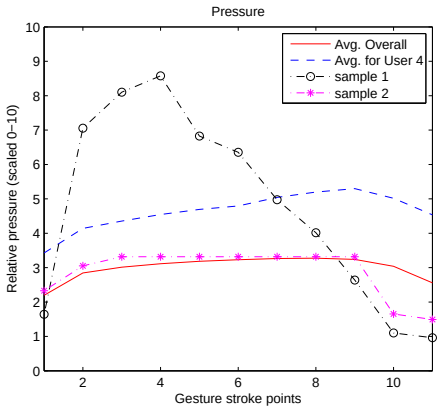


Figure 3.4: Touch pressure

3.5.2 The Randomization Approach

Numerical algorithms have to be customized according to the features available in the compromised system. Hence, the previous algorithm was specific to one compromised system. In this section, we present an algorithm that can work with almost any compromised system without the need for customizing it by the adversary according to the available features. This stems from utilizing a small subset of features (e.g. starting and ending gesture points, or gesture duration) that we have found to be used in almost all systems due to having high importance when doing feature selection.

This approach aims to produce a feature vector that is as close to the original feature vector as possible. The main idea of the randomization approach is adding Gaussian noise to certain values in the raw data matrix (e.g. time vector T); extracting the feature vector from the randomized raw data; and checking if the new randomized values have made the feature vector closer to the original feature vector. If it did become closer, we would update the raw data with the randomized values and repeat the process until an upper limit of number of iterations is reached. This upper limit is found experimentally in section 3.6.1, and it represents a number after which no significant improvement might happen.

Since changing one raw data value affects most of the extracted features, and might delay converging to the original feature vector, we opted to handle the problem with a divide and conquer

approach. Therefore, we define three categories of features: (1) those affected only by X and Y coordinates; (2) those affected either only by time, or time and X and Y coordinates; and (3) the rest that has only one feature per raw data type like pressure and area. For the latter, we use the same set of functions that were described in section 3.5.1.4 because randomization will not help with them. As for the first two categories, we process each one in a different stage. Features related to X and Y coordinates have to be optimized first before moving on to the time-related features. While velocity depends on X, Y, and timestamps, it is still possible to work on X and Y first to optimize the location-related features. Then one can move on to change the timestamps to optimize the velocity-related features, while having the X and Y coordinates fixed. In other compromised systems where feature vectors have rate of change features for pressure and area, the same approach can be used to optimize these features.

To evaluate how close two feature vectors are, we rely on the mean square error (MSE) between them. Of course, only the part of the feature vector related to the randomized values is tested. MSE is used to compare the normalized (or standardized) feature vectors to make sure that all features have similar value ranges and thus have equal importance. For this comparison, the function $testRawData()$ (please refer to algorithm 3) is used. It takes the new randomized values, the feature vector F and the indices of the relevant feature vector subset as inputs, and returns the MSE.

3.5.2.1 Reconstructing X and Y coordinates

Since the deviation from straight line defines the shape of the gesture (section 3.5.1.2), we apply the Gaussian noise to these values (lines 6-9 in algorithm 3). This has to be preceded by shifting and rotating the gesture to be horizontal, and followed by rotating and shifting back. After that, the new randomized gesture is tested to see if it decreases the MSE (lines 10-14 in algorithm 3), and if it does, the Y vector is updated accordingly. In particular, the function $testRawData()$ takes the current randomized raw data and the target feature vector and computes the MSE between the target feature vector and the one generated from the randomized raw data. That $tmpScore$

would be tested against the previous best MSE (*score*) to see if we need to update the best raw data estimation.

Algorithm 3 Reconstruct X and Y vectors randomly

```
1: function reconstructRandXY(F, startP, endP, LD, length, iterLimit, indFeatXY, sRatio)
2:     [X, Y, score] = initalizeXY(F, length, indFeatXY);
3:     theta = getAngle(firstP, lastP);
4:     for i = 1 → iterLimit do
5:         [tmpX, tmpY] = rotateShiftXY(X, Y, −theta);
6:         for j = 2 → length − 1 do
7:             σ = |tmpY_j| ;
8:             tmpY_j = tmpY_j + normRand(0, sRatio ∗ σ);
9:         [tmpX, tmpY] = rotateShiftXY(tmpX, tmpY, theta);
10:        tmpScore = testRawData(tmpX, tmpY, F, indFeatXY);
11:        if tmpScore < score then
12:            score = tmpScore;
13:            Y = tmpY;
14:            sRatio = 0.999 ∗ sRatio;
15:    return [X, Y];
```

Algorithm 3 starts by initializing X and Y. X is initialized in the same way as in section 3.5.1.2, and it does not change throughout the execution of this algorithm. whereas Y is initialized to one of eight gesture shapes depending on which one yields lower MSE (*score*). This step could minimize the amount of time needed to optimize the features. The eight gesture shapes are all scaled to the largest deviation (LD) with two only having positive deviations, two only having negative deviations, and four having both positive and negative deviations. This way, we can start with the gesture pattern closest to the original gesture.

The remaining part of the algorithm contains the randomization steps. Similar to algorithm 1, we use additive Gaussian noise with mean zero, and a standard deviation σ scaled by *sRatio* which is determined experimentally. However, in this algorithm, each point's additive noise has its own standard deviation, which equals that point's deviation from the straight line. Hence, points on the gesture edges that normally have less deviation would have less noise applied to them; thus maintaining its relatively less deviation compared to other points. This helps maintain an almost realistic gesture shape even after randomization, especially since the algorithm starts by initializing the gesture to one of eight realistic gesture shapes. Preserving such realistic gesture shapes would

be desirable in case future AA systems flag unrealistic gesture shapes as being forgeries. Finally, similar to algorithm 1, *sRatio* is reduced after each successful update (line 14 in algorithm 3).

3.5.2.2 Reconstructing time vector T

The algorithm starts by initializing the time vector T in a similar way to initializing Y in the previous section. Based on the total gesture duration, from the feature vector F, we set five variations of the time vector T that produce five different velocity curves, and then select the one that produces the least MSE. After that, this algorithm is similar to algorithm 3 described in the previous section except for the randomization part. Hence, algorithm 4 only describes the randomization of the time vector T.

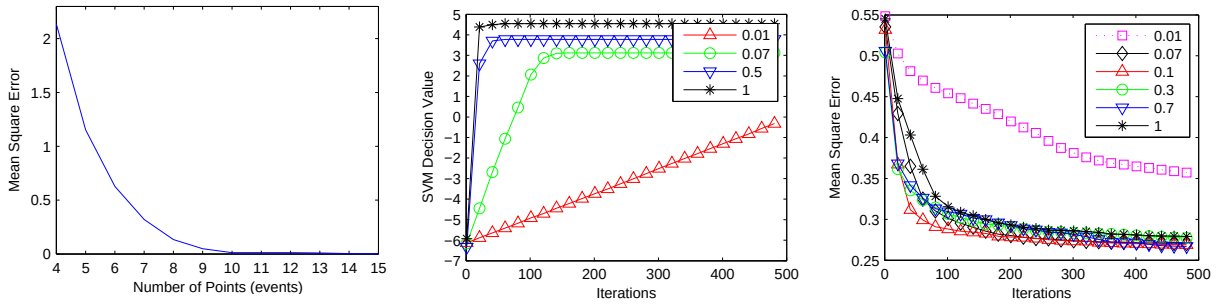
In algorithm 4, all points, except for the first and last (that are previously set to zero and the gesture duration respectively), are randomized successively. To ensure that the new randomized values ($rndT$) for point j won't fall before the preceding timestamp ($tmpT_{j-1}$) or after the following timestamp ($tmpT_{j+1}$), and that enough room is left for randomization in subsequent iterations, we do not set the newly generated random timestamp unless its difference from the preceding or the following timestamps exceeds a required time buffer ($bufT = 0.1 * (duration/length)$).

Algorithm 4 Randomize time vector T

1: **function randomizeTime**($tmpT$, $length$, $sRatio$, $bufT$)
2: **for** $j = 2 \rightarrow length - 1$ **do**
3: $\sigma = min(tmpT_{j+1} - tmpT_j, tmpT_j - tmpT_{j-1})$;
4: $rndT = tmpT_j + normRand(0, sRatio * \sigma)$;
5: $tmpT_j = conditionalUpdate(tmpT_j, rndT, bufT)$;
6: return $tmpT$;

To limit the chance of having any new randomized value very close to either the preceding or the following timestamps, we generate the additive noise for every point using a different standard deviation value σ scaled by *sRatio* (similar to algorithm 3). σ is set to be the smallest difference between the current timestamp and the preceding or following timestamps. By having this value, we limit the possibility of generating additive noise that would result in a value $rndT$ so close to

the preceding or following timestamps. Thus, we ensure successful updates of the randomized time vector and possibly achieve better use of each randomization iteration.



(a) *length* parameter (b) *sRatio* for Dec. Value Attack (c) *sRatio* for time randomization

Figure 3.5: Parameter Selection

3.6 Experimental Results and Discussion

3.6.1 Experimental Setup and Remarks

We used MATLAB and LIBSVM [67] to conduct the experiments. LIBSVM was chosen because it is more efficient than the MATLAB SVM functions. We used SVM with RBF kernel, and we tuned the parameters C and γ using five-fold cross validation. For prediction, we did not use the sign function where the threshold would be 0 to determine different classes. We change the threshold to find the equal error rate (EER) which is the value where both FAR and FRR are equal. This threshold is used throughout the experiments.

For full profile attacks, PSVs and NSVs are used to attack one of the four attacked systems. For each user's sample used (whether it was PSV or NSV), that sample is excluded from the data set before training user's models for the attacked systems. This is to avoid having biased results if that sample contributes to building the user model on the attacked system.

Because the reconstruction attacks given in the related work are in a different biometric research area (namely fingerprints), they are not directly comparable to our attacks and algorithms. Hence, our experiments only compare the attacks and algorithms we proposed, because our work is the first to address reconstruction attacks for mobile gestures.

Classifiers Selection: We used the data set provided by Frank et al [35] in our experiments. To test the attacks and our algorithms, we chose two classifiers: left-to-right (LTR) swipe, and right-to-left (RTL) swipe. We do not consider them to be one horizontal classifier since we believe that they each have distinct characteristics, and thus we classify them separately as was done in [32]. In the public data set we have, and for many users, one of the two vertical classifiers did not have enough samples to train an SVM classifier without overfitting. For this reason, and the fact that our algorithms are not dependent on gestures' directional angles (RTL and LTR are 180part, and our algorithms work successfully on both of them), the results are limited to LTR and RTL swipes. Finally, in table 3.3, we report the EER values of the four attacked systems when using RTL or LTR classifiers.

Table 3.3: EERs of different attacked systems

	RTL	LTR
$SysA1$ [32]	7%	7.3%
$SysA2$ [33]	13%	10.2%
$SysA3$ [34]	14.6%	12.8%
$SysA4$ [38]	11.1%	8.9%

Parameter Selection: We use data from a single user, chosen randomly, to perform parameter selection since the attacker will not have access to the original data of many users.

The parameter *length* (number of reconstructed points) is used in all our reconstruction algorithms in section 3.5. We needed to find the smallest number of events that would preserve the characteristics of gestures. When testing $length = k$, (1) we took a gesture from the data set (say of $length = 25$); (2) we interpolated this gesture to only have a *length* of k; then (3) interpolated the resulting reduced gesture to find the original 25 points; and (4) we finally measure the MSE between the original gesture, and the interpolated one. We did the previous steps for a number of gestures and aggregated the error. As shown in fig. 3.5a, the error could not be decreased after $k = 11$, thus we chose *length* to equal that number.

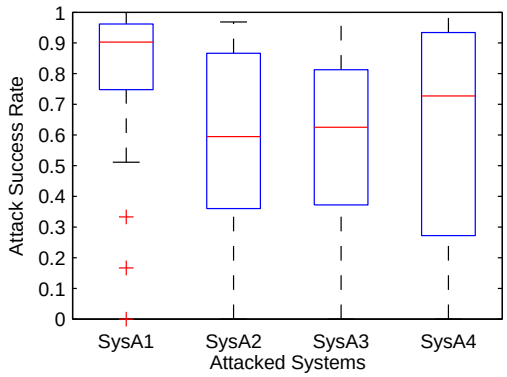
Figure 3.5b shows the effect of choosing different values of *sRatio* on the performance of the decision value attack algorithm (algorithm 1). In this case, we chose the target SVM decision

value, V_{th}, to be 3 in order to show that we can obtain samples with much higher SVM decision values than PSVs. It can be noticed that setting $sRatio$ to be 1 would ensure the least number of iterations for this algorithm 1 (about 20).

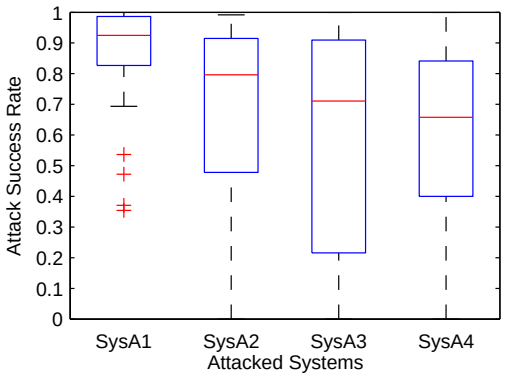
For the randomized time reconstruction algorithm (algorithm 4), we also show the effect of using different $sRatio$ values on the performance of this algorithm (fig. 3.5c). We selected $sRatio$ to be 0.1 since it required the least number of iterations to reach the almost plateau stage of MSE. Since the plot is based on average MSE values, it is not possible to determine the plateau stage MSE value for each reconstructed gesture. Hence, we selected an upper limit of iterations as the stopping condition for this algorithm (which was 300 as can be seen from fig. 3.5c). As for the randomized coordinates reconstruction algorithm (algorithm 3), the plots were very similar, so we also selected $sRatio$ to be 0.1, and $iterLimit$ to be 300.

Negative Support Vectors' Results: It is worth noting that NSVs were not found for all users in the latter attack; however, they were only three users for RTL swipes and two for LTR swipes (out of 41 users). The results in the following section are reported excluding these users since we wanted to test how the algorithms would perform using NSVs regardless of the percentage of users that didn't have corresponding NSVs, which might vary from one compromised system to another.

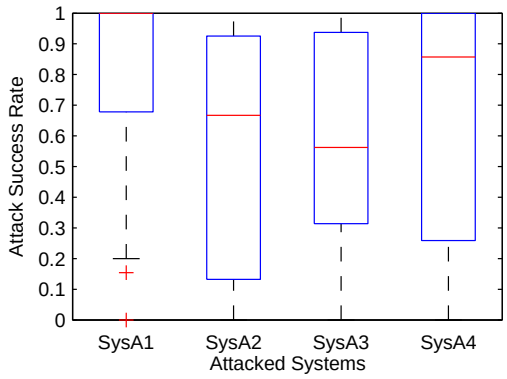
Attack Evaluation Metrics and Results Representation: To evaluate the attacks, we use the successfully reconstructed samples from the compromised system to attack four other systems. The choice of these systems was explained in section 3.4.6. For user u with N_c^u reconstructed samples from system $SysC$, if N_{ai}^u samples were successful at attacking system $SysAi$, then the success rate would be equal to N_{ai}^u/N_c^u. Hence, the success rate value is always between 0 and 1. For the randomization algorithms, we ran each test ten times, and reported the average. To show the results of the different attacks with our algorithms, box plots of these success rates are used (figures 3.6 and 3.7). The box plots show the median success rate (center red line) and the 25^{th} and 75^{th} percentiles which are the edges of the box. The whiskers extend to the most extreme



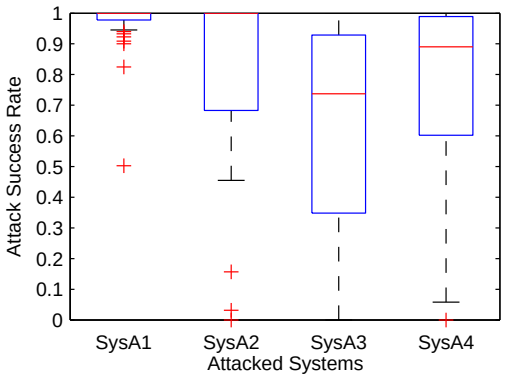
(a) Numerical Alg. - Positive SVs Attack



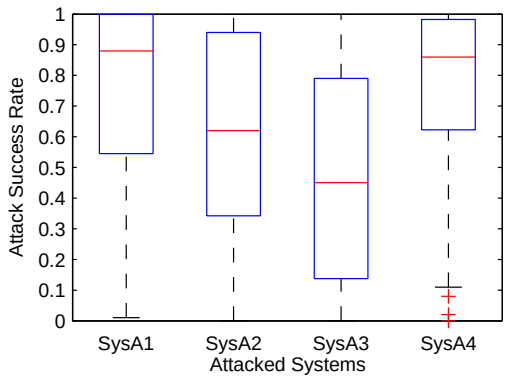
(b) Randomized Alg. - Positive SVs Attack



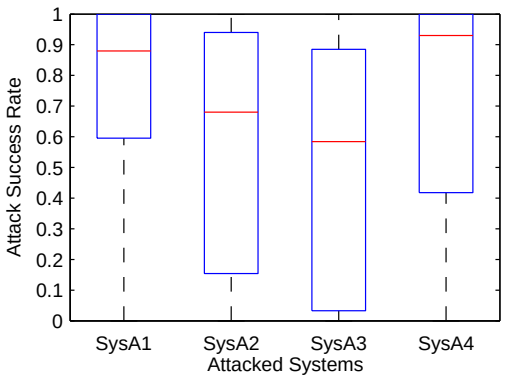
(c) Numerical Alg. - Negative SVs Attack



(d) Randomized Alg. - Negative SVs Attack



(e) Numerical Alg. - Decision Value Attack



(f) Randomized Alg. - Decision Value Attack

Figure 3.6: Results for left-to-right swipe classifier (LTR)

data points that are not considered outliers. The default MATLAB whisker spans 1.5 times the inter-percentile distances. Outlying values are represented by red plus signs.

3.6.2 Results and Analysis

Gesture stroke profiles (and our synthetic sample generation algorithm) provide a pool of feature vectors that can be used with the reconstruction algorithms. If the attempts to reconstruct any of these samples were successful, then an attack could be launched against the compromised system successfully. In our experiments, the reconstructed raw data is first tested against the compromised system, and it is only used for attacks if it succeeds at impersonating the legitimate user. Because both of our algorithms were successful at reconstructing a subset of gestures' raw data that would always pass the compromised system tests, results for success against the same system are not shown.

Results Overview: Figures 3.6 and 3.7 show attack results for LTR and RTL classifiers, respectively. Each of these figures contains six sub-figures that represent the combination of three types of attacks (positive SVs, negative SVs and decision value attacks) and two algorithms (numerical and randomization algorithms). Each sub-figure includes four box plots representing the four attacked systems: $SysA1 - SysA4$.

Impact of EER on Attack Success: In all types of attacks, it was clear that both our algorithms performed better against $SysA1$ compared to the other three. This was not initially expected as we thought that our algorithms would perform best against $SysA2$, since it uses a subset of the compromised system features. From observing the EER of both systems (table 3.3), we noted that $SysA1$'s EER was lower than $SysA2$'s EER by 3% to 6% for both classifiers. Higher EER indicates higher FRR which means that more legitimate user's samples would be falsely rejected by the authentication system. This might lead to more reconstructed raw data being rejected by the system with the higher ERR, and it is possibly the reason behind having a lower attack success rate against $SysA2$ compared to $SysA1$. Moreover, the EER for $SysA1$ was also lower than both $SysA3$ and $SysA4$, and the attack success rate was better against $SysA1$. This coincides with the

previous observation regarding $SysA1$ and $SysA2$. That being said, EER is probably not the only reason why our attacks did not perform so well against $SysA3$ and $SysA4$.

Impact of Features' Differences on Attack Success: In the following discussion, we analyze how the attack success rates are affected by the proportion of similar features and the relation of features in compromised and attacked systems.

It can be seen from figures 3.6 and 3.7 that attacks against $SysA3$ [34] did not reach "conditional success" in 7 out of the 12 attack combinations. In comparison, attacks against other systems were all at least conditionally successful except for one case each for $SysA2$ and $SysA4$ (figs. 3.7e and 3.7a respectively). From tables 3.1 and 4.1, it can be noticed that $SysC$ (the compromised system) only has one feature each for both pressure and area. Examining the 10 features used by $SysA3$, it can be noted that at least four of them cannot be deduced from $SysC$ features. These are the three area features and the average moving curvature. Such difference in the features, combined with $SysA3$ having the lowest EER, could have caused these low attack success rates. Intuitively, the relation between the features in the compromised and attacked systems would have an effect on the attack success rates.

Examining $SysA1$, which has less than 40% of its features similar to $SysC$ (as opposed to 50% for $SysA3$), it can be noted that there were 3 "successful" attacks against $SysA1$. The remaining 9 attacks were all on the high end of "conditional success". Hence, by comparing $SysA1$ to $SysA3$, we can deduce that it is not just the proportion of similar features that matters, but the possibility of deducing the attacked system features from the compromised system features. In the case of $SysA1$, most of its features could be deduced from $SysC$ using the reconstruction algorithms. The high success rates for $SysA1$ could also be attributed to its low EER.

On the other hand, $SysA4$ [38] had 2 "successful" attacks against it, 9 "conditionally successful" attacks, and one failed attack. The proportion of similar features with $SysC$ was only 19%, and this could have caused the lower attack success rates compared to $SysA1$. While $SysA4$ had features that cannot be deduced from $SysC$ features (e.g. the straight LDP-to-stop length), these features had lower importance in their respective feature vector (see [38] for features ranking). For this

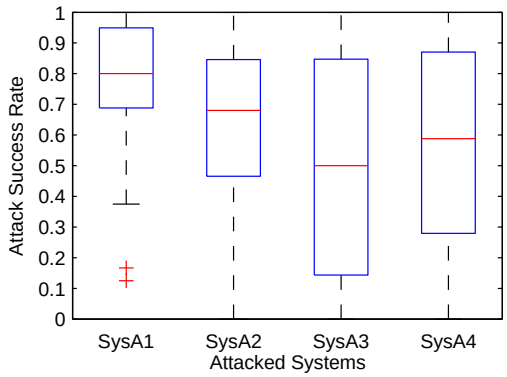
reason, and the lower EER of $SysA4$, the attacks against it were more successful than against $SysA3$ mentioned previously. Finally, the results of $SysA2$ were comparable to $SysA4$. However, these results are mostly tied to $SysA2$'s low EER rather than its features, since its features are a subset of those used in $SysC$.

Hence, we conclude that the attack success rates decrease if: (1) there are fewer similar (common) features between the compromised and attacked systems, (2) the similar features have lower importance (ranking) than differing features in their respective feature vector, and (3) fewer of the attacked system features can be deduced from the compromised system features.

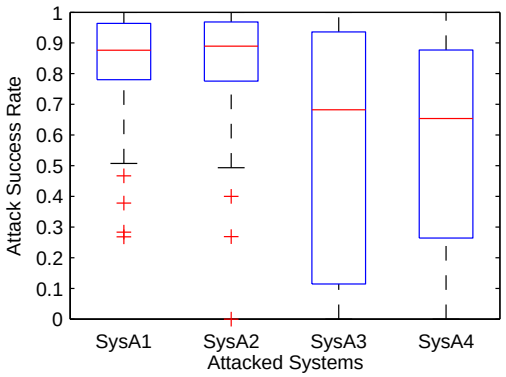
Different Attacks Performance: Table 3.4 shows that the NSV attack performed better than both the PSV and the decision value attacks. Furthermore, in comparing the results shown in figures 3.6 and 3.7, it is clear that using NSVs yielded noticeably better attack success rates than using PSVs to the point that the median attack success rates against $SysA1$ and $SysA2$ reached 100% for both LTR and RTL swipes using the randomization algorithm. This was expected since the NSVs found for a user (from other users' profiles) would normally have a higher decision value than the PSVs in her own profile.

Finally, the results of the decision value attacks indicate that even without actual feature vector samples, it is feasible to launch successful attacks against many systems. In fact, the attacks achieved at least "conditional success" against three systems out of four using both reconstruction algorithms. The results against $SysA1$ were also in the lead with medians exceeding 88% for both algorithms and classifiers, while the results against $SysA4$ showed even better median attack success rates than the PSVs attack results against the same system.

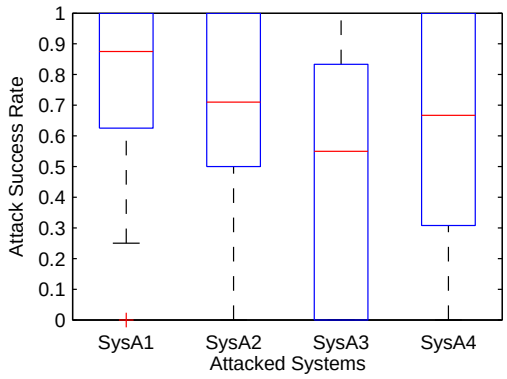
Numerical vs. Randomization Algorithms: Considering reconstruction algorithms, it can be seen from table 3.4 that the randomization approach performed better than the numerical. While the randomization algorithm was mainly designed to make it easier to utilize any compromised system without having to make changes to our reconstruction algorithm, it also had the added benefit of reconstructing raw data which can generate a feature vector that is very close to the target feature vector. Therefore, it can reach closer SVM decision values to those achieved using original



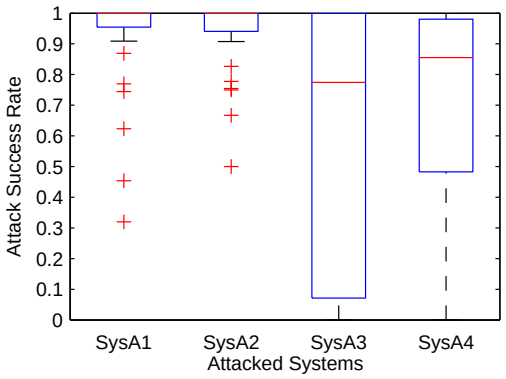
(a) Numerical Alg. - Positive SVs Attack



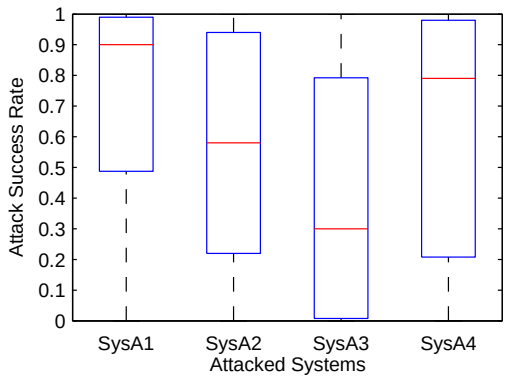
(b) Randomized Alg. - Positive SVs Attack



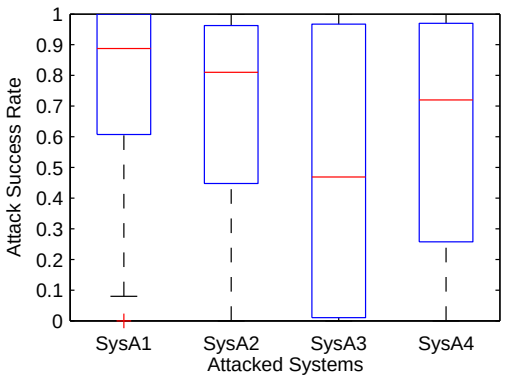
(c) Numerical Alg. - Negative SVs Attack



(d) Randomized Alg. - Negative SVs Attack



(e) Numerical Alg. - Decision Value Attack



(f) Randomized Alg. - Decision Value Attack

Figure 3.7: Results for right-to-left swipe classifier (RTL)

Table 3.4: Pairwise comparisons of attacks and algorithms using paired t-test. The alternative hypothesis was that the first of the compared pair has greater mean than the second. In all comparisons, the difference in mean was significant as evident by the p-values (much smaller than 0.05).

Compared Pair	Mean Difference	p-value
Randomized vs. Numerical	0.076	$3.1 * 10^{-15}$
Negative SV vs. Positive SV	0.066	$1.6 * 10^{-13}$
Negative SV vs. Decision Value	0.112	$2.6 * 10^{-16}$
Positive SV vs. Decision Value	0.046	$8.5 * 10^{-5}$

feature vectors. On the other hand, the numerical algorithm does not use an iterative approach; nor does it refine the raw data. The numerical estimation is done by one-way functions, with no feedback on how well they performed. Hence, it is not always guaranteed that the reconstructed raw data would yield a sample that is the closest possible to the original feature vector.

Table 3.5: Average reconstruction attack times per user

Attack Type	Time
Numerical - Positive SV	14.7 sec
Numerical - Negative SV	4.7 sec
Numerical - Decision Value	79.4 sec
Randomized - Positive SV	154.4 sec
Randomized - Negative SV	79.4 sec
Randomized - Decision Value	483.8 sec

Attacks Efficiency: The experiments were performed on a desktop computer with i5-3470 processor and 8GB of RAM. The average times required for the attacker to perform reconstruction attacks against one user are shown in table 3.5 (run using single-threaded MATLAB script). The PSVs attacks took longer than the NSVs attacks because the number of PSVs was greater than the number of NSVs for each user on average (38 and 11 respectively). All attacks had reasonable efficiency except for the randomized reconstruction algorithm with the decision value attack which took almost 8 minutes per user to generate ten user samples. This performance could be easily improved since the generation of these user samples can be done in parallel. Moreover, the

randomized reconstruction algorithm can be modified to have its optimization target be the SVM decision value, thus eliminating the need for running two consecutive randomized algorithms. This improvement is not our main target, and can be left for future work.

3.6.3 Discussion and Recommendations

From the analysis in the previous section, our observations can be summarized as follows:

First. Full profile attacks can yield high attack success rates, especially NSVs attacks. Hence, it is vital to avoid exposing user samples when building users' authentication profiles either by utilizing a machine learning algorithm that does not use them, or by using a privacy-preserving algorithm.

Second. Even in the absence of user samples, it is feasible to synthesize user samples using decision value attacks. This indicates that machine learning algorithms (like logistic regression) that do not store user samples, but return a probability, can also be susceptible to reconstruction attacks. Hence, it is recommended to only return binary classification results, and to rely on other methods to provide a trade-off between security and usability (other than using a threshold with the SVM decision value or logistic regression probability).

Third. It is not necessary for the attacker to spend a long time tailoring a reconstruction algorithm to a compromised system since a general randomization algorithm can yield high success rates. This shows that the AA system designer must not rely on choosing features that seem harder to reverse using numerical methods. It also further signifies the importance of using privacy-preserving (PP) techniques.

Fourth. Success rates are lower when the attacked systems share fewer features with the compromised system, especially if these features cannot be deduced from the compromised feature vectors. However, this should not deter the AA system designers from using PP techniques just because they think their feature vectors are so different from other systems, since this includes high uncertainty and great risk.

Fifth. Systems with lower EERs can be more susceptible to reconstruction attacks. This indicates that just because a system has a lower EER, it does not mean it is ready for deployment. In fact, it is more dangerous to deploy such a system before taking preventative measures against reconstruction attacks, than with a higher EER system.

3.6.3.1 Recommendations

Based on these observations, we provide our recommendations for building AA systems starting with modifications to the system model that was described in section 3.4.2.1 (fig. 3.1). The improved system architecture is shown in fig. 3.8, where the modifications are two-fold:

AA Server: To prevent full-profile attacks, it is imperative to use privacy-preserving machine learning techniques. This includes having "transformed" profiles (P_T in fig. 3.8) and a privacy-preserving matching algorithm (PP Matcher in fig. 3.8). Such transformation can be achieved using encryption [39] or reduced SVM [68], and the PP Matcher would be tied to the specific transformation performed. Unfortunately, such approaches come at the cost of decreasing the accuracy of the system. For example, the lowest reported EER in [39] was 18% when using the features from [35]. Nevertheless, it is still recommended to use such approaches, both to circumvent full-profile attacks, and to report the realistic readiness of such AA systems for deployment.

Client App and Gesture aggregation: To thwart decision value attacks, the decision value of testing a sample must not be returned. In fact, returning a binary result for each tested sample is not recommended as well. Instead, the client app should collect multiple gestures (represented by their raw data $\{R_1, \cdots, R_S\}$ in fig. 3.8, where S is the number of gestures), extract their individual feature vectors, and send them to the AA server ($\{F_1, \cdots, F_S\}$ in fig. 3.8). The *PP Matcher* can classify each sample F_i, and take a majority vote before returning a single binary result *binRes* (which equals 1 for a legitimate user, and 0 for an intruder).

This way, no individual sample can be tested by the adversary. Moreover, the possibility of finding a useful sample would decrease since the adversary has to come up with more than one successful synthetic sample through randomization (increased entropy). To avoid having the

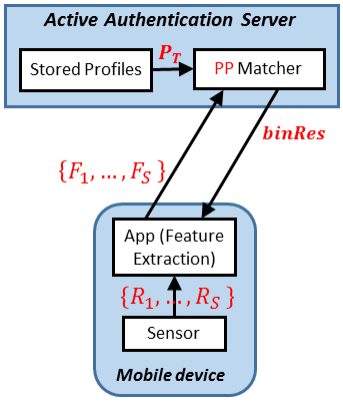


Figure 3.8: System Architecture

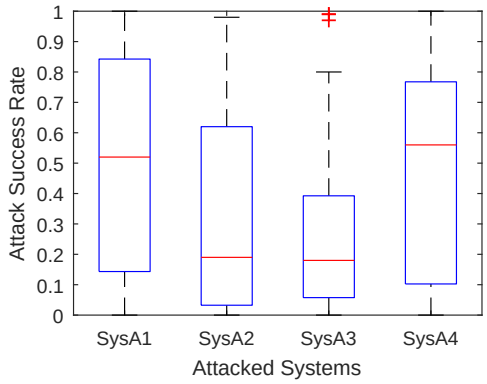
adversary send the same group of samples each time with one different sample to test that one specifically, the AA system can maintain an adequate history of submitted feature vectors to avoid repetition. It is worth noting that previous research (e.g., [35, 38]) has shown that aggregating multiple users' test samples for authentication proved to lower the EER of many systems. Hence, it comes with an additional advantage in this case.

To demonstrate the effectiveness of our recommendations, we conduct further experiments in the following section. Since the security and privacy of privacy-preserving machine learning algorithms were discussed in their respective work [39], we concentrate on the gesture aggregation advantages.

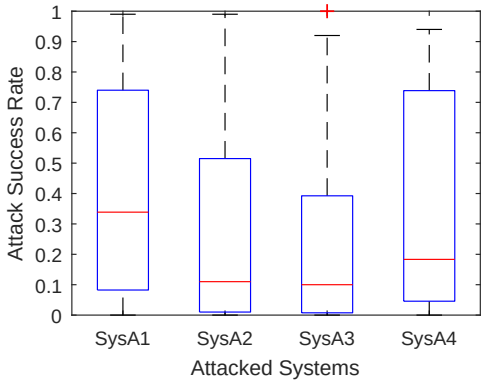
3.6.3.2 Binary and Aggregated Classification Results

We assume a ***binary result adversary (BR)*** that is similar to the decision value adversary described in section 3.4.4. However, for BR adversary, only binary results are returned whether it was for a single sample or an aggregation of samples. To obtain a feature vector, the BR adversary uses algorithm 5 which is similar in concept to algorithm 1. However, since no decision value is returned, it is reduced to random guessing in each iteration.

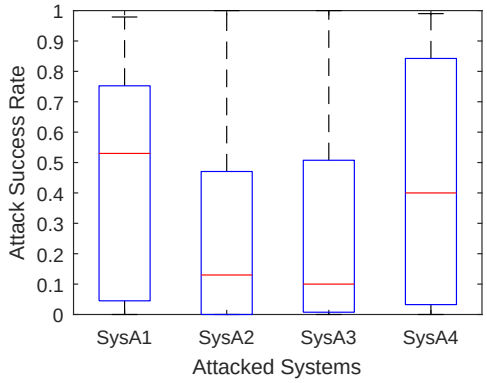
Figure 3.9 shows the results for using algorithm 5. By comparing the binary result attack (figures 3.9a, 3.9c, 3.9b and 3.9d) and the decision value attack (figures 3.6e, 3.6f, 3.7e and 3.7f), we can see that returning a binary result lowers the attack success rates. In fact, the binary result attack mean was less than the decision value attack mean by 0.27 (was found significant using



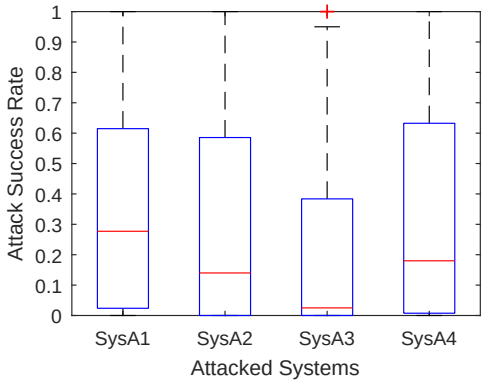
(a) LTR: Numerical Algorithm



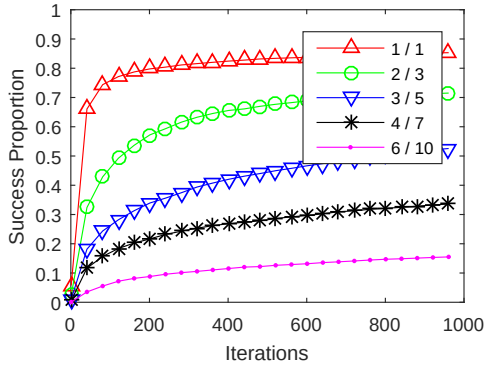
(b) RTL: Numerical Algorithm



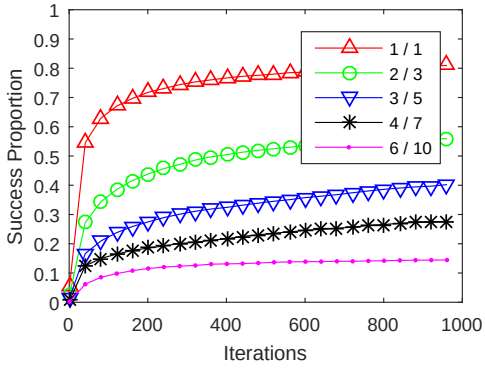
(c) LTR: Randomized Algorithm



(d) RTL: Randomized Algorithm



(e) LTR: Success proportion vs. iterations for multiple majority vote settings



(f) RTL: Success proportion vs. iterations for multiple majority vote settings

Figure 3.9: Results of the binary result attack for left-to-right (LTR) and right-to-left (RTL) swipes

paired t-test with $p - value = 2.2 * 10^{-16}$). It is also worth noting that none of the binary result attacks achieved "conditional success" against any system.

We show the benefit of aggregating gestures in figures 3.9e and 3.9f. These figures show the number of iterations required to generate synthetic feature vectors (using algorithm 5) that can pass as user samples. The y-axis in figures 3.9e and 3.9f represent the proportion of working synthetic samples to the total required samples at a specific iteration (we required 100 synthetic samples for each user to account for randomness). In figures 3.9e and 3.9f, we show different majority vote schemes and how they compare to each other (these curves do not include the iterations required for the reconstruction algorithms to represent the worst case scenario if the adversary had a perfect algorithm). From figures 3.9e and 3.9f, it can be seen that aggregating more gestures lowers the proportion of successfully synthesized feature vectors. For example, when the majority scheme is 6 out of 10 gestures (6 / 10), an adversary can only obtain 15% of the required samples even after 1000 iterations. In conclusion, more aggregated samples make it harder for the adversary to obtain synthetic samples, and even these generated ones cannot achieve attacks that are at least "conditionally successful".

Algorithm 5 Generate synthetic feature vectors for user u using binary results

```
1: function binGenSyntheticFV(model_u, initF, initR)
2:     binRes = testFV(model_u, initF);
3:     while binRes ≠ 1 do
4:         rndF = initF + normRand(0, initR);
5:         binRes = testFV(model_u, rndF);
6:     return rndF;
```

3.7 Conclusions

The aim of this work was to highlight the importance of carefully designed privacy-preserving AA systems. Otherwise, AA systems would fail at fulfilling their premise of providing security and privacy for mobile devices' users, especially while accessing cloud services. To show this, we proposed a numerical-based and a randomization-based reconstruction algorithms. These algorithms were utilized to reconstruct raw gesture data from users' authentication profiles (the full-profile

attack), and from the decision value returned by SVM sample testing (the decision value attack). We further used such reconstructed raw data for attacking users' accounts on other AA systems. To test our algorithms, the literature was surveyed to select one compromised system and four attacked systems. The experimental results showed that reconstruction attacks against gesture-based AA systems are feasible, and that the most effective attack was using negative SVs. The most effective reconstruction algorithm was the randomization-based one. The results also showed that even when the adversary had no access to the users' samples, reconstruction attacks were still feasible against most systems. In our future work, we will pursue the problem of designing privacy-preserving systems that are resilient to reconstruction attacks.

CHAPTER 4. PRIVACY-AWARE GESTURE-BASED CONTINUOUS AUTHENTICATION BY REDUCING DIMENSIONS

Mohammad Al-Rubaie [1] and J. Morris Chang

4.1 Abstract

Behavioral biometrics can be utilized to continuously and transparently authenticate users while interacting with their mobile devices. Handling behavioral biometrics users' profiles raises security and privacy challenges especially when relying on a 3rd party to perform mobile identity management. To protect physical biometrics profiles, cancelable biometrics techniques were proposed to transform the biometrics profile (profile) in a revocable and non-invertible way. Traditionally, cancelable biometrics requirements included security (non-invertibility), performance, reovacability and diversity. Recognizing the possibility of performing undesired inference on behavioral biometrics profiles, we introduce a new cancelable biometrics requirement: Privacy (inference restriction). We propose a gesture-based continuous authentication framework that utilizes supervised dimensionality reduction (S-DR) techniques to protect against undesired inference attacks. These S-DR methods are Discriminant Component Analysis (DCA), and Multiclass Discriminant Ratio (MDR). Using experiments on a public data set, our results show that DCA and MDR provide better privacy/utility performance than random projection, which was extensively utilized in cancelable biometrics. Finally, we outline how these techniques can be used to enhance previous cancelable biometrics approaches in terms of privacy (inference restriction).

[1] primary researcher and author. Mohammad is the primary author of this paper that conducted the research, performed the experiments, and wrote the paper under the supervision and advice of Prof. J. Morris Chang

4.2 Introduction

Mobile devices are highly portable, a property that makes them more susceptible to theft (or loss) than desktop computers. In order to augment initial authentication on mobile devices (i.e. passwords, lock patterns, fingerprints or face IDs), continuous authentication (CA) was explored in the recent years [32, 35, 34, 52, 69, 70, 40, 71]. Continuous authentication (also called implicit or active authentication) systems continue to authenticate the user transparently after the initial authentication. CA mostly utilizes behavioral biometrics which rely on how users interact with their devices through touchscreen gestures, taps, and keystrokes. In fact, CA technology has matured enough to be deployed in real environments by companies such as BehavioSec and Appdome. These companies partnered to augment Appdome's Mobile Identity solution (a mobile Identity as a Service - IDaaS) with BehavioSec's Behavioral-based CA system (CAaaS) [72].

In a study conducted by Balebako *et al.* [73], it was found that most app developers rely on 3rd party initial authentication IDaaS systems, e.g. facebook connect. Developers preferred such systems because they believed that 3rd party IDaaS systems were more secure than any solution they could develop as these IDaaS systems were built by security experts. Unfortunately, such 3rd party IDaaS systems introduce privacy challenges since these IDaaS systems leak information about installed apps [73]. Furthermore, login attempts reveal users' app usage patterns to the 3rd party IDaaS. If mobile IDaaS were complemented with CA capabilities (CAaaS), tracking users' activities would be even more fine-grained. A 3rd party CAaaS server could learn what virtual keyboard keys are being pressed via tap locations, or discover the type of app being used by utilizing gesture direction information (e.g. maps or a browser) [74]. Thus, biometrics security and privacy should be thoroughly considered when designing CAaaS systems.

Earlier work on privacy and security for physical biometrics recommended transforming the users' biometrics data using user-specific one-way functions to achieve ***Cancelable Biometrics*** properties [75]. Such transformations would enable ***Revocability*** where a biometric profile could be revoked if compromised, and ***Security (Non-Invertibility)*** since such one-way transformations are computationally difficult to reverse; hence they prevent recovering the original biometric

from the transformed one. In addition, two other requirements for biometric profile protection are: ***Performance (Utility)*** which indicates that the protection method should not degrade the accuracy of distinguishing a legitimate user from an intruder, and ***Diversity*** for preventing cross-matching across profile databases. While these four requirements were sufficient for physical biometrics, behavioral biometrics introduce inference-related challenges.

Behavioral biometrics are susceptible to inferring context and personal information using the same feature vectors used for CA. For example, the touch gesture feature vectors used for CA could also be utilized for inferring gender or experience level [33], or recognizing the operating hand [76]. Furthermore, Zhu *et al.* [37] was able to predict gender, occupation, and age group with good accuracy using the CA feature vectors extracted from mobile sensors (accelerometer, gyroscope, magnetometer and orientation). Hence, we propose adding a fifth requirement for biometric profile protection: ***Privacy (Inference Restriction)***. Undesired and unauthorized inference from the feature vectors should be restricted, while still allowing the intended use of the data, i.e. continuous authentication. More specifically, inference attempts of any non-user-authentication targets from the behavioral biometrics feature vectors should perform similar to random guessing.

In this paper, we address the privacy issues related to using Continuous Authentication as a Service (CAaaS) for mobile devices. Continuous authentication is expected to be outsourced to 3rd party CAaaS since CA requires specialized expertise and infrastructure [77] (similar to the initial authentication case noted by Balebako *et al.* [73]). Such outsourcing necessitates privacy and security measures when handling behavioral-based CAaaS systems because of the sensitivity and amount of the data they utilize. With the exception of papers like [74] and [78], almost all behavioral-based continuous authentication research on mobile devices did not pay much attention to privacy issues. However, the cryptographic techniques in [74] relied on one-class classification that decreased the accuracy of CA, while the random projection techniques in [78] can not address the unauthorized inference privacy leaks, as will be demonstrated in this paper.

Since touch gestures received the most attention in the recent behavioral-based CA literature ([51, 34, 45, 70, 79, 80, 81, 32, 35, 46, 82, 40, 53, 38, 52, 47, 83, 69, 71]), it was chosen to be studied

in this paper. We further selected the gesture direction to be the privacy target that should be protected from leakage to the 3rd party CAaaS server. We chose this case study for the following reasons: (1) Gesture direction is sensitive information [74]. If the CAaaS server receives the app name with the login attempt, then providing gesture direction in the uploaded feature vector would indeed provide the CAaaS with very fine-grained tracking of the users activity within the app. Even if the app name was masked from the CAaaS, the gesture direction still leaks information about the task being performed by the user, as noted by earlier research [74, 70, 80], (2) Since almost all feature vectors proposed in the recent literature include direction-related features, gesture direction provides a good opportunity to study reconstruction attacks (security requirement), and (3) we can also study inference-based attacks since it is possible to infer the gesture direction from the feature vector (privacy requirement).

The main contributions of this paper are as follows:

First. Based on our analysis of the recent behavioral biometrics literature, we found that ***Privacy (Inference Restriction)*** should be added as a fifth requirement for securing behavioral biometrics data. We propose utilizing the following supervised dimensionality reduction techniques to address this requirement: (1) Disrimenant Component Analysis (DCA) [84], and (2) Multiclass Discriminant Ratio (MDR) [85]. We compare these supervised techniques to the non-supervised dimensionality reduction methods: Principal Component Analysis (PCA) and Random Projection (RND). We also evaluate the proposed techniques in terms of *Performance (Utility)* and *Security (Non-Invertibility)*. Furthermore, we show how these techniques can be combined with well-known cancellable biometrics techniques that provide the *Revocability* and *Diversity* properties.

Second. We assembled a gesture-based feature vector that ensures balanced coverage of the different categories of touch features: location, shape, velocity, acceleration, pressure, area, time and orientation. We perform experiments using a public data set [35] to demonstrate the ability of supervised dimensionality reduction to resist inference and reconstruction-based attacks (which correspond to the *Security* and *Privacy* properties, respectively). Our experiments show that DCA and MDR provide better privacy/utility tradeoff than random projection which was extensively

utilized in previous cancelable biometrics approaches. DCA and MDR also performed better than PCA in terms of security and privacy.

4.3 Related Work

Gesture-based Continuous Authentication

Almost all behavioral-based continuous authentication (CA) systems proposed in the recent literature did not consider privacy, and only concentrated on increasing the accuracy of such systems. Many of the enhancements that targeted accuracy were only suitable for on-device CA solutions, and cannot be directly ported to a CAaaS environment for privacy-related issues. An example of such enhancements is context-aware CA, where for each user, different context-based-classifiers are built.

For gesture-based CA, contexts included music [86], apps or tasks [87, 50, 88], screen size [88], physical activity of user (walking or stationary) and device (In-Left-Hand, In-Right-Hand, In-Both-Hands, On-Table) [88], or a combination of app-based and UI-based modalities [89]. Utilizing context in a CAaaS environment would result in leaking privacy-sensitive information to a 3rd party CAaaS server. Thus, it is recommend to restrict context-aware CA to device-specific CA systems.

Even in the absence of explicit context information, it is inevitable to send the touch gesture feature vectors to the CAaaS server for authentication. These feature vectors contain features that might leak sensitive information to the CAaaS entity. For example, the gesture directional angle leaks information about the task being performed by the user [74, 70, 80]. However, similar to context-based classifiers, gestures direction was used to build classifiers as follows: (1) classifiers for each of the four gesture directions (Left-to-Right, Right-to-Left, Upward, Downward) were utilized in [51, 34, 45, 70, 79, 80, 81], (2) classifiers for horizontal and vertical gestures were used in [32, 35, 46, 82, 40], and (3) while other work utilized one classifier for all gesture directions [53, 38, 52, 47, 83, 69, 71], gesture angle was still used as a feature in almost all recent research. Hence, the feature vector itself could leak sensitive information to the 3rd party CAaaS server.

In our work, we build one classifier for all gesture directions, and do not use any context-related classifiers. Furthermore, our proposed enhancements to current CA systems ensure that the original feature vector is not revealed to the CAaaS server. We transform the feature vector in a non-invertible way, which prevents the CAaaS from learning the features that might leak sensitive information. Moreover, our proposed transformation not only prevents reconstruction of the original features, but also prevents the adversary from inferring these features from the transformed feature vectors.

Secure Biometrics' profiles

Cancelable Biometrics techniques were proposed to ensure that a physical biometric profile could be revoked if compromised [75]. This was typically achieved by providing two lines of defense: (1) secret-key based transformation, and (2) a one-way transformation that should be irreversible (in case the secret key was leaked). Many cancelable biometrics' approaches relied on reducing dimesionality using random projection to perform non-invertible transformation [90, 78]. In such systems, the features dimensions reduction is done by projecting the features using a random matrix generated from a user-specific secret-key. Biohashing [91] added another layer of security by passing the reduced feature vector to a discretizing mechanism where the output features would be binary based on a specific threshold. It can be noted that these approaches prepare for the possibility of losing the first line of defense (secret-key) by relying on a non-invertible transformation (such as dimensionality reduction).

Securing the profiles of behavioral-based CA did not receive as much attention as its physical biometrics authentication counterpart. The few exceptions such as [74, 92] relied on cryptographic approaches, but did not consider adding non-invertible transformation as a second line of defense. In [92], PCA with dimensionality reduction (DR) was used to partially circumvent the accuracy decrease due to using one-class classification. However, dimensionality reduction using PCA cannot be considered a second line of defense as it does not provide adequate protection against reconstruction and inference attacks (sections 4.5.4 and 4.6).

As we will demonstrate later in this paper, dimensionality reduction using random projection or PCA do not protect against undesired inference attacks; thus, these approaches cannot meet the privacy requirement. In contrast, our proposed framework which utilizes supervised dimensionality reduction techniques (DCA and MDR) provides the required protection against undesired inference.

4.4 Compressive Privacy

Dimensionality reduction is an important machine learning tool that is traditionally used to overcome issues like: (a) over-fitting, (b) performance degradation due to suboptimal search, and (c) higher computational cost and power consumption resulting from using high dimensional feature vectors. Dimensionality reduction can be demonstrated visually by Fig. 5.2 where the red dots $\widetilde{\mathbf{x}}_\mathbf{i}$ are the projections of the blue dots $\mathbf{x_i}$ on the principal component $\mathbf{u^1}$.

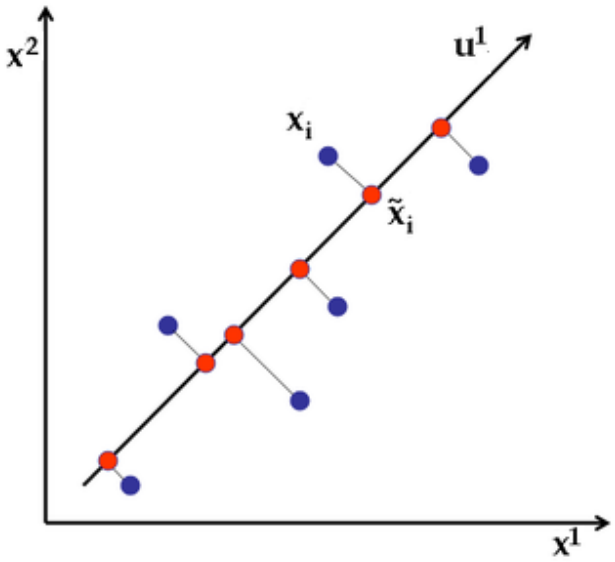


Figure 4.1: Data Projection

A dimensionality-reducing transformation is more resilient to reconstruction attacks than a dimensionality preserving one [1]. Since the mapping from the original feature vectors to a low dimensional subspace is a many-to-one mapping, it is not possible to determine the privately held features from the reduced feature vectors, as there are infinite possible feature vectors which lead

to identical reduced feature vector. For example, a vector $\mathbf{x} \in \mathbb{R}^M$ can be projected using a matrix $\mathbf{U}_m \in \mathbb{R}^{M \times m}$ where $m < M$, to obtain $\widetilde{x}$:

$$\widetilde{x} = \mathbf{U}_m^T \mathbf{x} \tag{4.1}$$

It would not be possible to find the exact values of the vector $\mathbf{x}$ by solving an underdetermined linear equation system (more unknowns than equations). Thus, previous work utilized dimensionality reduction for privacy-preservation in the area of biometrics security; but they mostly considered random projection techniques [91, 90, 78].

4.5 Proposed Framework

4.5.1 Problem Description

Most mobile app developers choose to rely on 3rd party IDaaS systems developed by groups of experts, e.g. facebook connect, rather than implementing their own authentication mechanisms [73]. Similarly, continuous authentication is expected to be outsourced to 3rd party CAaaS systems since CA requires even more specialized expertise and infrastructure than initial authentication [77].

In addition to the risk of releasing app usage information to the 3rd party IDaaS server, CAaaS systems could reveal more fine-grained behavioral data. For example, tap locations could enable the 3rd party CAaaS server to learn virtual keyboard keys that are being pressed, and the gesture direction could give clues about the task being performed, e.g. reading or viewing images [74, 70, 80]. Furthermore, undesired inference could be done using the same feature vectors used for CAaaS systems, such as predicting gender, occupation, or age group [37].

We note that preventing undesired inference was never addressed for CA systems. In this work, we propose methods that could specifically ensure the privacy through preventing undesired inference. Such methods maintain the accuracy of CA systems, while enhancing both security and privacy.

4.5.2 Design Goals and Threat Model

We mainly consider the case of outsourcing continuous authentication to a 3rd party CAaaS server; hence, our proposed system will be evaluated based on these design goals:

- ***Security (Non-Invertibility)***: The CAaaS should be prevented from learning the original feature values submitted to the cloud for CA. Hence, non-invertible transformation should be used that is easy to compute, but hard to invert.
- ***Privacy (Inference Restriction)***: Undesired inference from the transformed feature vector should be prevented, which is especially important for behavioral biometrics.
- ***Performance (Utility)***: The accuracy of the CA classifiers should not degrade as a result of applying the biometric profile protection mechanism.

In addition to these design goals, we will also outline how our techniques can be combined with well-known cancellable biometrics techniques that provide the ***Revocability*** and ***Diversity*** properties. Furthermore, it should be noted that our proposed techniques are ***energy efficient*** which is an important design requirement especially for mobile devices [93].

Similar to earlier cancelable biometrics research, we assume a powerful adversary that managed to compromise the first line of defense which is the secret key used to protect the biometric profile. Hence, the only thing preventing the adversary from obtaining the original feature vector is the non-invertible transformation applied to the original data. We further assume that this adversary obtained the projection matrix used to transform the data. For privacy evaluation, we assume that the adversary has access to an ML model that was built for performing a certain undesired inference (unauthorized by the user who submitted its transformed data to the CAaaS server for the sole purpose of performing CA).

4.5.3 Gesture-based Continuous Authentication

Gesture touch raw data cannot be used directly as input to classification algorithms. A feature extraction step has to be done to convert raw touch gesture data into feature vectors. Raw data of

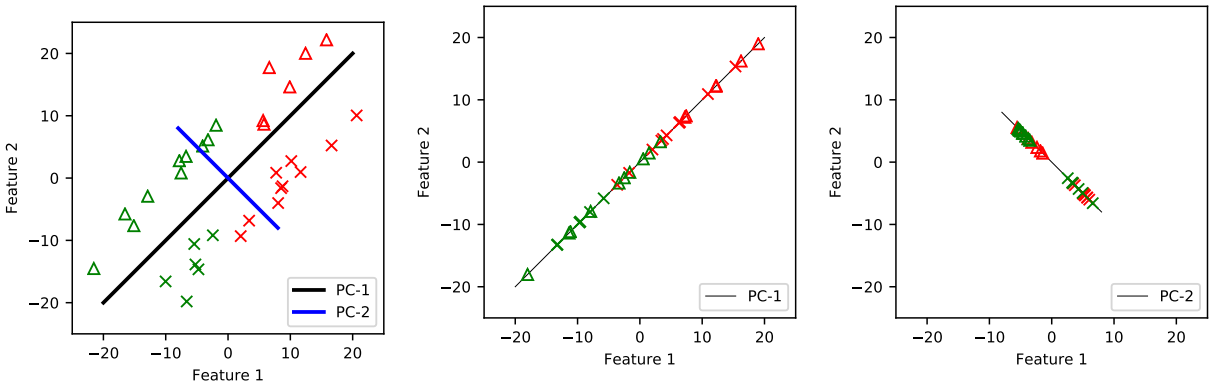
each touch gesture stroke can be represented by a series of vectors $(S_1, S_2, ..., S_N)$ with N being the number of touch events. Each touch event (denoted by a vector, S_i) corresponds to the following data: location coordinates, time stamp, pressure, area, phone and finger orientation [35]. Out of this touch sequence, a feature vector is extracted, which is formed of features like average velocity, directional angle, and trajectory length.

We reviewed the recent literature of gesture-based CA to choose a set of features to be used in our framework. We selected feature sets from the following four papers: Frank *et al.* [35], Serwadda *et al.* [32], Li *et al.* [34] and Xu *et al.* [38]. These papers were selected as they achieved low error rates, and each paper described its feature vector clearly. Furthermore, as can be seen in table 4.1, the features in these papers complement each other in terms of their coverage of the 8 different categories of touch features.

The union of the features from these four papers [35, 32, 34, 38] resulted in a feature vector composed of 73 unique touch features. To prevent over-fitting caused by the high feature dimensions, previous gesture-based CA work performed feature selection using a variety of methods. In this work, we rely on supervised dimensionality reduction (S-DR) techniques to reduce the feature vector length. While feature selection eliminates some lower weighted features, S-DR makes sure that all features can contribute to the reduced feature vector, and that each feature's contribution is relative to its importance. In short, S-DR has the advantage (over feature selection) that no feature discriminate power is lost.

4.5.4 Privacy Enhancement

In this section, we discuss our privacy enhancements that achieve the design goals presented in section 4.5.2. The main idea stems from observing that by projecting the data in the direction that maximizes the users' approved utility target, the possibility of utilizing the data for any other (undesired inference) target would be minimized. Such optimization cannot be achieved by random projection, or PCA, since they do not utilize the data labels. This intuition will be domonstrated in a visual example followed by outlining the necessary techniques to prevent undesired inference.



(a) Original Data

(b) Projection on principal component 1

(c) Projection on principal component 2

Figure 4.2: Dimensionality Reduction: the data has two utility labels ($\times$ and $\triangle$) and two privacy labels (red and green)

Table 4.1: Features categories of different papers

	Frank [35]	Serwadda [32]	Li [34]	Xu [38]
Location	4	4	2	6
Shape	9	3	4	13
Velocity	5	5	-	4
Acceleration	4	5	-	-
Pressure	1	5	-	5
Area	1	5	3	5
Time	2	1	1	3
Orientation	3	-	-	1

Consider a data set with N training samples $\mathbf{x}_1, \cdots, \mathbf{x}_N$, with each sample having M features ($\mathbf{x}_i \in \mathbb{R}^M$). Each sample $\mathbf{x}_i$ has an associated utility label, y_i, and a privacy-sensitive label, p_i. Predicting the utility labels is approved by the data owner, while predicting any privacy-sensitive label should be prevented. An example is given in Fig. 4.2a where the data has two types of labels: (1) utility labels: representing two utility classes denoted by '$\times$' and '$\triangle$' symbols, and (2) privacy labels: representing two privacy-sensitive classes denoted by red and green colors.

In Fig. 4.2a, PCA would select principal component 1 (PC-1) for projecting the data since PC-1 is the direction of maximum variance. This results in mixing the samples from both utility classes (Fig. 4.2b); which leads to lower utility classification accuracy. However, projecting the data on

PC-2 would have been a better choice as can be seen in Fig. 4.2c since the '×' and '△' classes can be clearly separated. PC-2 would be the choice of Disrimenant Component Analysis (DCA) which will be described shortly. It can also be seen from Fig. 4.2c that projecting the data to maximize the utility target has resulted in mixing the privacy labels; potentially preventing prediction of the privacy-sensitive labels. Finally, random projection which was extensively utilized in cancelable biometrics would choose a projection hyperplane at random; thus it is not guaranteed that PC-2 would be selected.

Supervised-DR methods are crucial for ensuring protection against undesired inference. In this work, we will rely on, and compare, two supervised DR methods:

- ***Disrimenant Component Analysis (DCA)*** is a supervised method for dimensionality reduction [84]. The key intuition behind DCA is selecting a projection hyperplane that can effectively discriminate between different classes. In Fig. 4.2a, DCA would choose PC-2 as the principal component on which the data has to be projected because it provides the highest discriminant power.

  Computing the DCA projection matrix $\mathbf{W}_{DCA}$ involves the following optimization criterion [84]:

$$\mathbf{W}_{DCA} = \underset{W:W^T[\overline{\mathbf{S}}+\rho\mathbf{I}]W=\mathbf{I}}{\arg\max} \; tr(W^T\mathbf{S}_B W) \tag{4.2}$$

  where $\mathbf{S}_B$ is the between-class scatter matrix, and $\overline{\mathbf{S}}$ is the center-adjusted total scatter matrix. $\mathbf{W}_{DCA}$ can also be obtained by solving the generalized eigenvalue decomposition problem: $eig(\mathbf{S}_B, \overline{\mathbf{S}}+\rho\mathbf{I})$. This would ensure selecting a subspace that would maximize the separability between the utility classes (Fig. 4.2c).

- ***Multiclass Discriminant Ratio (MDR)*** provides joint utility-privacy optimization [85]. In Fig. 4.2c, DCA does not mix the privacy labels perfectly as red and green triangles are still partially separate. MDR can account for such issues by slightly tilting PC-2 clockwise to provide perfect mixing of the privacy labels.

MDR aims to maximize the separability of the utility classification problem (intended task), while minimizing the separability of the privacy-sensitive classification problem (undesirable or sensitive task). It assumes that each data sample has two labels: a utility label (for the intended purpose of the data), and a privacy label (sensitive information). Hence, we can obtain two between-class scatter matrices: one that is based on the utility labels $\mathbf{S}_{B_U}$, and the second computed using the privacy labels $\mathbf{S}_{B_P}$. To obtain a projection matrix $\mathbf{W}_{MDR}$ that maximizes the accuracy of the utility classification problem, and minimizes the accuracy of the privacy classification problem, MDR uses the following optimization criterion [85]:

$$J(W) = \frac{det(W^T \mathbf{S}_{B_U} W)}{det(W^T \mathbf{S}_{B_P} W)} \tag{4.3}$$

Similar to DCA, the MDR projection matrix $\mathbf{W}_{MDR}$ can be obtained by solving the the generalized eigenvalue decomposition problem: $eig(\mathbf{S}_{B_U}, \mathbf{S}_{B_P})$

As shown above, DCA (or MDR) has the potential to ensure the privacy and performance design goals. In addition, reducing the dimensionality of the data achieves the security (non-invertibility) requirement as was discussed in sections 5.3 and 4.4. In section 4.6, we will evaluate DCA and MDR experimentally in terms of privacy, security and performance.

4.6 Experimental Evaluation

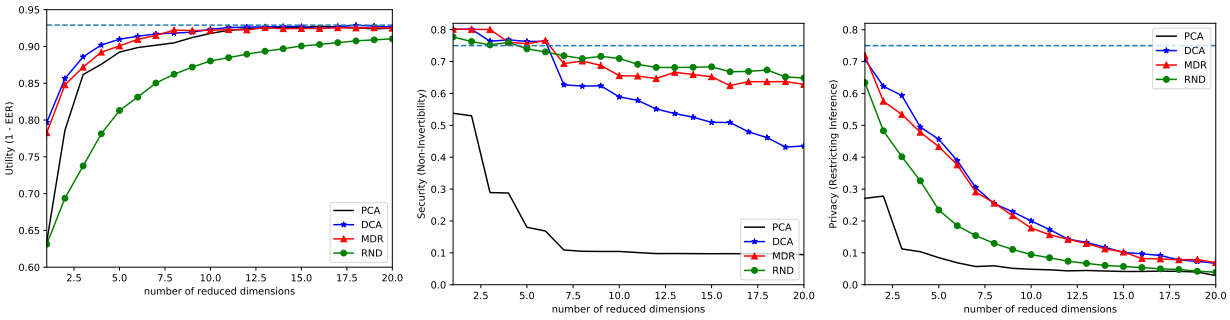
In this section, we evaluate our proposed framework in terms of privacy, security and performance using the public dataset published by Frank *et al.* [35]. This dataset contains touch gesture data collected from 41 users.

4.6.1 Case Study: Gesture-based Continuous Authentication

We consider a case study of gesture-based CA on mobile devices to compare DCA, MDR, PCA and Random Projection in terms of their ability to resist reconstruction and inference-based attacks. For this evaluation, we aim to enable continuous authentication using gesture data while

preventing the 3rd party CAaaS server from learning gesture directions. The security/privacy target is protecting the gesture direction.

One might think of a simple solution to this problem, which is excluding any gesture direction related features from the feature vector. After performing initial experiments, we noticed that removing the direction features increased the error rate from 7.1% to 8.1%. While this utility loss is small, we also found that this feature removal did not solve the problem. It was still possible to predict the gesture direction from "no-direction-information feature vectors" with an accuracy of 83.35%. It seems that features related to velocity, pressure, timing and others can also help infer the gesture direction. Besides, removing groups of features would not provide an effective solution especially if we considered many privacy targets, or when there is no clear relationship between the privacy target and feature values.



(a) Utility (Continuous Authentication)

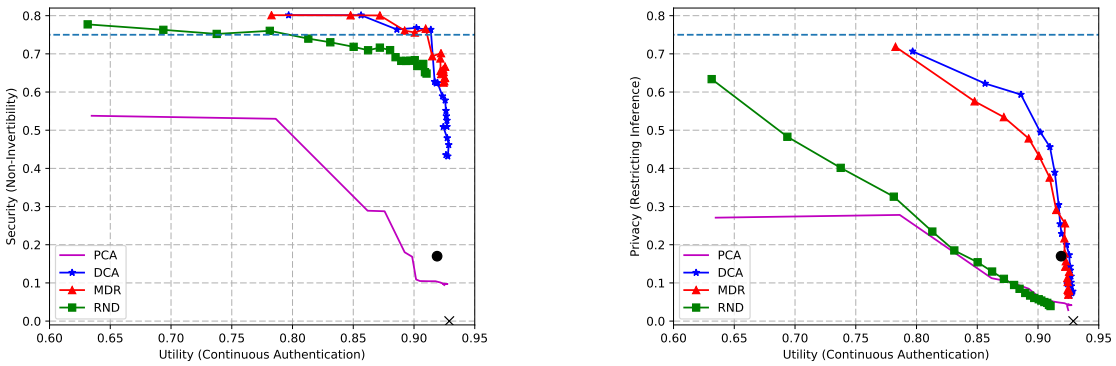
(b) Security (Non-Invertibility)

(c) Privacy (Inference Restriction)

Figure 4.3: Effect of number of reduced dimensions on utility, privacy and security. The dashed line in (a) denotes the performance when using the original feature vector. In (b) and (c), the dashed lines represent the security and privacy, respectively, of random guessing.

4.6.2 Evaluation Criteria

The gesture-based CA literature mainly addresses four gesture directions: Left-to-Right, Right-to-Left, Upward, and Downward [51, 34, 45, 70, 79, 80, 81]. The same four gesture directions were used for designing touch interactions with mobile App GUI interfaces, and these predefined interactions led privacy researchers to determine that certain gesture directions can leak the type of tasks performed by the users [74, 70, 80]. Hence, we evaluate the DR techniques in terms of



(a) Utility vs. Security ROC Curves

(b) Utility vs. Privacy ROC Curves

Figure 4.4: Security/Privacy vs. Utility ROC Curves. the dashed lines represent the security and privacy, respectively, of random guessing. The black symbol × denotes using the original feature vector (no security/privacy), while the black symbol • denotes using a "no-direction-information feature vectors".

their ability to prevent the adversary from predicting gesture directions with accuracy higher than 25% (which corresponds to random guessing of the four gesture directions).

Since gesture direction is given as a feature in gesture-based CA, and can also be inferred from the feature vector (section 4.6.1), it provides a good opportunity to study both reconstruction and inference-based attacks. These two types of attacks correspond to the ***Security (Non-Invertibility)*** and ***Privacy (Inference Restriction)*** requirements, respectively, in our design goals (section 4.5.2):

- ***Security (Non-Invertibility)***: Assuming the adversary has access to both the transformed feature vector, and the projection matrix, an approximation of the original feature vector could be computed by the adversary (reconstruction attack). Since the gesture angle is included in the feature vector, we measure of the accuracy of determining the gesture direction from the reconstructed gesture angle. By subtracting this accuracy from 1, we can report the security (non-Invertibility) score in our experiments.

- ***Privacy (Inference Restriction)***: We build a machine learning model to predict the gesture directions from the transformed feature vectors. In our experiments, privacy (inference restriction) score is found by subtracting the gesture prediction accuracy from 1.

It should be noted that the security (privacy) of random guessing is 75% since random guessing from four directions equals 0.25%. We also study the trade-off between security/privacy and utility (performance), which is our third design goal (section 4.5.2). The utility (performance) is defined here as $1-EER$ where EER is the equal error rate. EER was the main measurement for comparing different continuous authentication systems in the literature. It is the value where False Rejection Rate and False Acceptance Rate are equal.

4.6.3 Experimental Results

In figures 4.3 and 4.4, we compare the different dimensionality reduction techniques: DCA, MDR, PCA and random projection (RND) in terms of performance, security and privacy. In Fig. 4.3, the effect of the number of reduced features dimension is shown, while the security/privacy vs. performance trade-off is demonstrated in Fig. 4.4.

In terms of performance (utility), it can be seen in Fig. 4.3a that DCA, PCA and MDR perform better than RND. This is expected as RND does not utilize the data in determining the projection matrix. On the other hand, RND performs the best in terms of security (non-invertibility) with MDR being a very close second (Fig. 4.3b). This result seems to confirm the choice made by earlier cancelable biometrics approaches that chose RND as their main technique to achieve security (non-invertibility). On the other hand, it can be seen in Fig. 4.3c that neither RND nor PCA performed as well as the supervised methods: DCA and MDR in terms of privacy (as anticipated by our analysis in section 4.5.4). PCA's formulation aims to minimize the reconstruction error, which made it perform the worst in terms of security (non-invertibility). Furthermore, PCA performed the worst in terms of privacy, which could be attributed to it being unsupervised DR technique.

In order to get a better understanding of how different DR techniques compare to each other, we generated the security/privacy vs. performance ROC curves in Fig. 4.4. Each point on the curves in Fig. 4.4 represents one of the reduced dimensions in Fig. 4.3. The best technique would have a curve that is closer to the upper right corner since both security/privacy and performance (utility) would be at their maximum in that corner. It is clear from Fig. 4.4 that both DCA and MDR

provide the best security vs. utility tradeoff. It is due to the fact that they both perform much better than RND in terms of utility (Fig. 4.3a). Furthermore, as anticipated in 4.5.4, both DCA and MDR show clear advantage over RND and PCA in terms of privacy as can be seen in Fig. 4.4b. For DCA, the privacy score reaches 60% (only 15% short of random guessing) when the utility is 88% (i.e., EER is 12%). This utility is still reasonable considering that only one gesture stroke was used, and that many past studies recommended aggregating the result of multiple gesture strokes for increased utility [35], and for resisting model inversion attacks [8]. For example, Frank *et al.* [35] aggregated the testing results of 11 gesture strokes to lower the EER to an average of 2-3% down from 13% for one gesture only.

We conclude that utilizing DCA or MDR is preferred in terms of privacy, security and performance over RND or PCA. These supervised DR methods provide a better second line of defense than RND which was extensively utilized in cancelable biometrics.

4.7 Compatibility with Cancelable Biometrics Approaches

One of the compelling reasons for using random projection in cancelable biometrics is the ability to generate the projection matrix randomly from a seed specific to the data owner (i.e., U_{RND} in Fig. 4.5). This user-specific projection enables the approaches shown in Fig. 4.5 to meet the revocability and diversity requirements of cancelable biometrics. By default, the DCA (or MDR) projection matrix would be the same for all data owners; however, it is still possible to replace U_{RND} in Fig. 4.5 with a user-specific DCA (or MDR) projection matrix.

Similar to the case in Fig. 4.5, assume that the original feature dimension was M, and that k was selected as the reduced feature dimension. After computing the DCA projection matrix $\mathbf{W}_{DCA} \in \mathbb{R}^{M \times M}$ according to eq. 5.6, the first k columns are kept to yield the projection matrix $\mathbf{W}^k_{DCA} \in \mathbb{R}^{M \times k}$. This matrix would be the same for all users. However, each user can still generate their own random matrix $\mathbf{W}_{RND} \in \mathbb{R}^{k \times k}$ from a user-specific seed. By multiplying the two matrices, each user could obtain their own projection matrix $U_{DCA} = \mathbf{W}^k_{DCA} \times \mathbf{W}_{RND}$. The resulting matrix

$U_{DCA} \in \mathbb{R}^{M \times k}$ can replace U_{RND} in Fig. 4.5 to enable our approach to be compatible with other cancelable biometrics approaches. The same discussion applies to MDR's projection matrix.

Applying U_{DCA} to the data directly would be equivalent to first (a) projecting the data using $\mathbf{W}^k_{DCA}$, an operation that ensures privacy, security and performance requirements, and then (b) rotating the projected data using a user-specific projection matrix to ensure the revocability and diversity properties of cancelable biometrics.

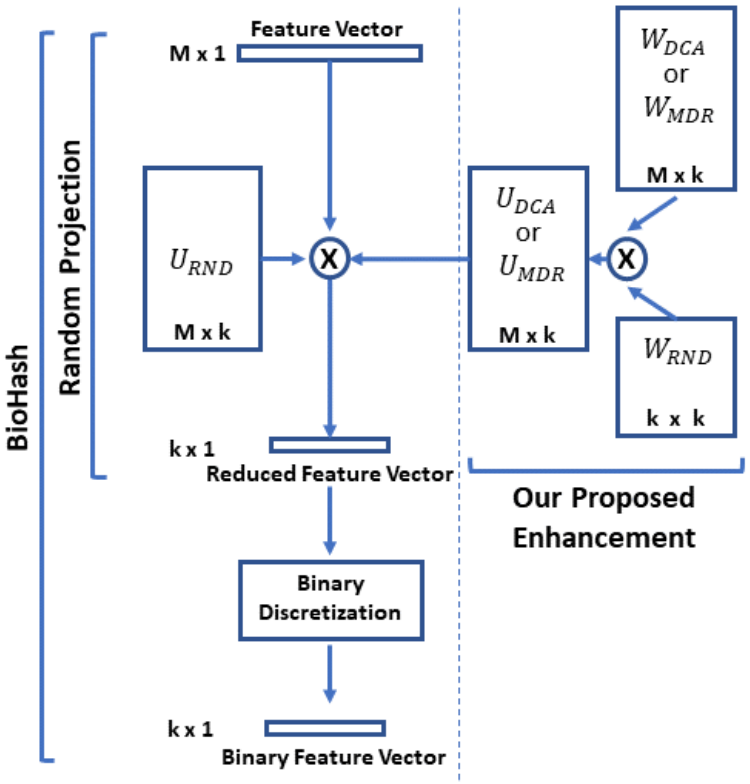


Figure 4.5: Enhancement to Cancelable Biometrics

4.8 Conclusion

We investigated the potential use of supervised dimensionality reduction (S-DR) techniques for securing behavioral-biometrics-based continuous authentication systems. We observed that the original four properties of cancelable biometrics: security, performance, revocability and diversity can not fully address the challenges associated with behavioral biometrics CA. We proposed a fifth

requirement which is privacy for protecting against undesired inference on the transformed data sent to a 3rd party CAaaS entity. We proposed a framework that relies on the S-DR methods: DCA and MDR to meet the privacy, security and performance design goals. Using experiments on a public dataset [35], our results show that the supervised DR methods (DCA and MDR) provide a better second line of defense than random projection, which was extensively utilized in cancelable biometrics. We further demonstrate how to replace random projection in previous cancelable biometrics solutions by DCA, or MDR, to make such solutions ready for behavioral biometrics applications.

CHAPTER 5. PRIVACY-PRESERVING PCA/DCA ON HORIZONTALLY-PARTITIONED DATA

Extended from a paper published in the IEEE Conference on Dependable and Secure Computing, Aug. 2017.

Mohammad Al-Rubaie [1], Pei-yuan Wu, Emre Yilmaz, J. Morris Chang, and Sun-Yuan Kung

5.1 Abstract

Private data is used on daily basis by a variety of applications, where machine learning algorithms predict our shopping patterns and movie preferences. Reducing the features dimension is an essential machine learning tool for preventing overfitting, and data visualization. Two popular dimensionality reduction (DR) techniques are *principal component analysis (PCA)* and *discriminant component analysis (DCA)*. PCA is a popular method for unsupervised learning, while DCA is a highly effective method for the prevailing supervised learning. Finding the projection matrices for these two DR methods include computing the scatter matrices followed by performing eigenvalue decomposition. In this paper, we propose protocols that enable the PCA/DCA computation to be performed, on horizontally-partitioned data distributed among multiple data owners, without exposing the private data of any of the participating data owners. To preserve the data owners' privacy, we propose hybrid protocols that utilize additive homomorphic encryption to compute the scatter matrices on encrypted data, and then perform the Eigen decomposition using Garbled circuits. Our protocols are practical as they do not require the data owners to interact with each other, or remain online throughout the execution of the protocol. We implemented our protocols using

[1] Primary researcher and author. The main idea of this paper was based on discussions between the authors especially Mohammad and Dr. Wu. The rest of the work was done by Mohammad under the supervision and advice of Prof. J. Morris Chang

Java and Obliv-C, and conducted experiments on public datasets. These experiments demonstrated that our protocols are efficient, and preserve the privacy while maintaining the accuracy.

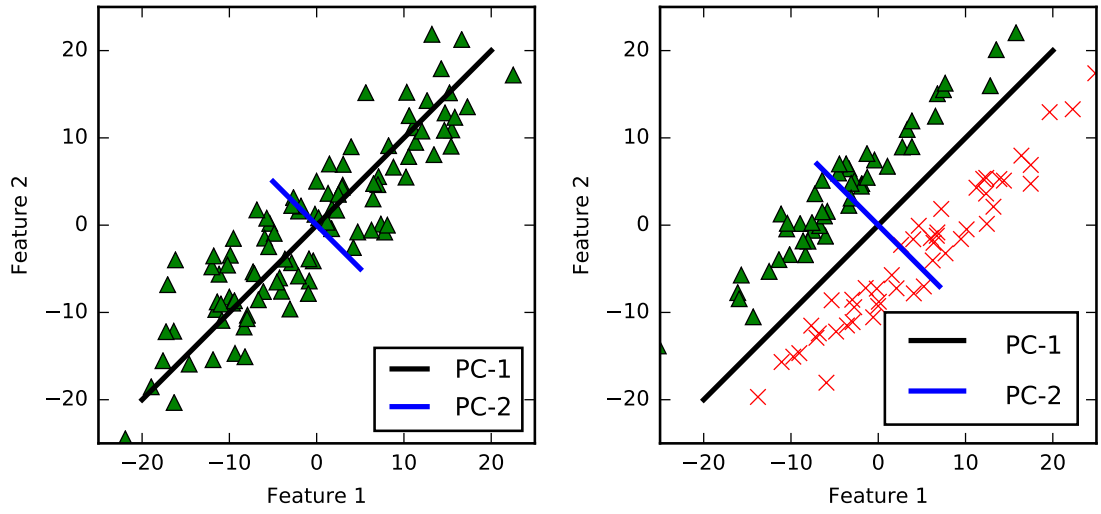
5.2 Introduction

Machine learning tasks include supervised tasks like regression and classification, and unsupervised tasks like clustering. These techniques are widely used in modern applications: identity and access management [94, 35], detecting email spam [95], identifying fraudulent credit card transactions [96], or building clinical decision support systems [97]. Very often, these applications use personal data (e.g. biometrics, health care records, or financial datasets) during the training (enrollment) and testing (prediction) phases of machine learning.

Dimensionality reduction is an important machine learning tool that is used to overcome issues like: (a) over-fitting when the features' dimensions far exceed the number of training samples, (b) performance degradation due to suboptimal search, and (c) higher computational cost and power consumption resulting from high dimensionality in the feature space. Principal Component Analysis (PCA) is a widely used method for dimensionlity reduction. PCA aims to project the data on the principal components with the highest variance; thus preserving most of the information in the data while reducing the data dimensions. From looking at Fig. 5.1a, it can be noticed that most of the variability happens along the first principal component (denoted PC-1). Hence, projecting all the points on that new axis could reduce the dimensions without sacrificing much of the data variability.

On the other hand, Discriminant Component Analysis (DCA) [98] is tailor designed for (the prevailing) supervised learning applications. The objective of DCA is not recoverability (reducing the reconstruction error like PCA), but rather improving the discriminant power of the learned classifiers, so that they can effectively discriminate between different classes. Fig. 5.1b shows an example of a supervised learning problem where PCA would choose PC-1 as the principal component on which the data would be projected because PC-1 is the direction of most variability.

However, DCA would choose PC-2 since it provides the highest discriminant power (as can be seen from Fig. 5.1b).



(a) Unlabeled Data (b) Two utility labels (× and △)

Figure 5.1: Dimensionality Reduction Techniques

PCA/DCA were traditionally performed by gathering the data in a centralized location, however, in many applications (e.g. continuous authentication [39, 35]), the data is distributed across multiple data owners. In such cases, collaborative learning would be done on a joint dataset formed of samples held by different data owners, where each sample contains the same attributes (features). Such data is described as horizontally-partitioned since the data is represented as rows of similar features (columns), and each data owner holds a different set of rows in the joint data matrix. In this work, we consider the case where a central entity (the data user) would like to compute the PCA (or DCA) projection matrix using the data distributed across multiple data owners in a privacy-preserving way. The projection matrix would then be used by the data owners to reduce the dimensions of their data. Such reduced dimensions data could later be used as an input to certain privacy-preserving ML algorithms that perform classification [39] or regression [11], for example.

In this paper, we address the problem of computing the PCA (and DCA) projection matrices by a certain entity (the data user) without compromising the privacy of the data owners. Early distributed PCA approaches did not consider privacy at all [99, 100]. Later on, privacy-preserving

approaches were proposed for PCA [101] and Fisher Discrimenant Analysis (FDA) [102] but they required all the data owners to remain online throughout the execution of their protocols, which is not very practical. Furthermore, the approaches proposed for Linear Discrimenant Analysis (LDA) either did not protect the scatter matrices [103] (thus revealing intermediate results which compromises security), or were limited to LDA, and would not extend to DCA (or similar linear dimensionality reduction approaches) [104]. In our work, we provide a solution to such issues by proposing and implementing a practical method to perform dimensionality reduction using PCA/DCA in a privacy-preserving way. In contrast to earlier approaches, our protocols do not reveal any intermediate results (e.g., the scatter matrices), and do not require data owners to interact with each other, or remain online after submitting their individual encrypted shares.

Furthermore, our protocols could be utilized as a privacy-preserving data preprocessing stage that comes before applying other privacy-preserving machine learning algorithms for the purpose of classification [39] or regression [11]. Thus, ensuring privacy and utility throughout the process of applying machine learning algorithms on private data.

Our contributions are as follows:

First. Computing the projection matrices for PCA (or DCA) requires computing the scatter matrices in a distributed way. Thus, we developed protocols that use additive homomorphic encryption to compute the scatter matrices. In addition to the data owners (who hold the data), and the data user (who aims to compute the projection matrix), we assume the existence of a third entity, a crypto service provider (CSP), that would not collude with the data user (similar assumption to [10, 11]). Participating data owners are required to compute their individual shares, encrypt them using homomorphic encryption with the CSP's public key, and send these shares to the data user that would aggregate the shares. The CSP builds a garbled circuit that performs Eigen Decomposition (or generalized Eigen decomposition) on the scatter matrices computed from the aggregated shares (only revealed in clear text form inside the garbled circuit). Neither the data user nor the CSP can see the aggregated shares in their clear text form; thus, our protocols do not reveal any intermediate values (such as the user shares or scatter matrices). Furthermore, our

approach does not burden the data owners with having to interact with other data owners, and do not require the data owners to remain online after sending their encrypted shares; a property that makes our protocols practical.

Second. We implement our privacy-preserving protocols using Java and Obliv-C. Through the experiments we performed using public datasets, we show that our protocols are efficient. We further demonstrate that our approach do not degrade the accuracy of machine learning tasks.

This paper extends an earlier version [105] which only considered PCA. The improvement over [105] is twofold: (1) this extended paper considers DCA projection matrix computation as well, and proposes protocols for computing the signal and noise scatter matrices, and a new garbled circuit implementation for the generalized eigenvalue problem, and (2) improved PCA computation (over [105]) by further optimizing the garbled circuit implementation.

5.3 Related Work

Cryptographic and Distributed PCA/DCA Computation: Earlier approaches that considered distributed PCA computation ([99, 100]) did not consider privacy in computing the projection matrices, and their methods included sharing private information such as the data mean with other parties. Other protocols revealed the users' shares as well as the scatter matrix [106]. In contrast, our protocols do not reveal any intermediate results (e.g., user shares or scatter matrices).

Pathak *et al.* [101] had the privacy of the individual users' shares in mind (when performing eigenvalue decomposition), however, their protocol required multiple stages of computation, and all the data owners had to remain online during these stages. In our approach, the data owners do not have to remain online after submitting their individual encrypted shares; which makes our protocols more practical. More recently, Mirhoseini *et al.* [107] proposed a framework for delegating matrix-based machine learning applications to untrusted cloud service providers, and they demonstrated their methods using PCA. However, their framework does not consider the multi-party situation, and only considers a single data owner delegating their PCA computation to the cloud. Another approach only supported two-party eigenvalue decomposition [108], while our protocols do not have

a limit on the number of participating data owners, and do not require any interaction between the data owners.

While computing the PCA projection matrix was not their main topic, Han *et al.* [109] and Chen *et al.* [110] proposed approaches to compute the singular value decomposition (SVD). Han *et al.* [109] proposed a protocol to perform SVD using QR decomposition. Their protocol is limited to the two-party situation (who have to remain online for the execution of the protocol), while our protocols address the multiple data owners case. Chen *et al.* [110] proposed a multi-party protocol for SVD, however, the protocol involves revealing the scatter matrix, and performing eigenvalue decomposition on the clear text scatter matrix. In contrast, our protocols do not reveal any intermediate values.

DCA computation was not previously addressed in the literature, however, we list similar formulations such as Linear Discrimenant Analysis (LDA) and Fisher Discrimenant Analysis (FDA). Han *et al.* [102] proposed a two-party protocol to perform Fisher Discrimenant Analysis (FDA). Their work required the two parties to remain online for the execution of the protocol. In contrast, our protocols support multiple data owners, and they do not need to remain online after submitting their individual shares. Sedenka *et al.* [103] proposed a privacy-preserving LDA protocol where the data owners' shares are encrypted and aggregated into the scatter matrices. However, these matrices are decrypted before computing the LDA projection matrix. Our approach does not reveal the scatter matrices to any party. Tian *et al.* [104] proposed a privacy-preserving sparse LDA protocol, however, their protocol relies on computing local approximations, and averaging these approximations using MPC. They do not implement generalized eigenvalue decomposition (GEVD) which prevents their methods from being extended to other DR methods that utilize GED such as DCA, Multiclass Discrimenant Ratio (MDR) [85] and Differential Utility-Cost Analysis (DUCA) [26].

PCA Perturbation Methods

Different methods for differentially-private PCA were proposed in the recent literature. All of these approaches aim to provide a differentially-private PCA projection matrix; however, none of

them considered the multiple data owner case. They can be categorized according to where, and how, the randomness is applied:

(1) Input Perturbation where the noise is added to the covariance matrix, and after the desired non-private-computation (Eigen decomposition) is performed on the noisy input, the output would be differentially-private. Dwork *et al.* [19] adds symmetric Gaussian noise matrix to the covariance matrix before performing Eigen-decomposition. The output would be a (ϵ, δ)-DP projection matrix (the target here is not releasing the projected data, but the DP projection matrix). Later on, other works proposed adding wishart noise to the covariance matrix [111, 112], which produced a $(\epsilon, 0)$-DP projection matrix (pure differential privacy). Their approach also maintains the positive semidefinite property of the perturbed covariance matrix.

(2) Another approach is perturbing intermediate values in iterative algorithms. Hardt and Price [20] proposed adding Gaussian noise in each iteration of the power method algorithm which operates on the non-perturbed covariance matrix, leading to DP-PCA.

(3) Output Perturbation involves running the non-private-learning algorithm, and then adding noise to the generated model. For PCA, this involves adding noise to the projection matrix [113]. An alternative approach would be using the exponential mechanism to sample a random k-dimensional subspace that approximates the top-k PCA subspace [114].

For LDA, Chakrabarti *et al.* [115] proposed privacy-preserving LDA using input perturbation methods by adding noise to the scatter matrices (no differential privacy guarantees were provided).

All of the approaches mentioned above work on data held by a single data owner, or collected by a trusted server. They do not provide a method for distributed computation of differentially private PCA (or LDA). Our work complements these approaches by providing a mechanism to collect data from multiple data owners. Furthermore, it would be trivial to modify our protocols to accommodate any of the above approaches. Our garbled circuit implementation could be slightly modified to perturb the covariance matrix [19, 111, 112], the power method [20], or the resulting projection matrix [113, 114] such that the output would be differentially private. However, such modification is out of the scope of this paper, and will be left for future work.

General Cryptographic Methods: Multiple secure multiparty computation (MPC) techniques have been utilized to solve a variety of problems including creating privacy-preserving versions of many machine learning algorithms. These approaches mainly used additive homomorphic encryption [116, 117, 10], commutative keyed hash function [118] or hybrid approaches using additive homomorphic encryption and garbled circuits [11]. Many solutions that only rely on additive homomorphic encryption (e.g. [101]) require all data owners to remain online during the computation stage which would not be practical for many applications. Some exceptions like [10] relied on two computation parties, each with a different role, to avoid having data owners remain online. It is notable that Garbled Circuit solutions have two main parties: a garbler and an evaluator. Nikolaenko *et al.* [11] utilized this fact to allow performing a complex computation by using a hybrid approach of additive homomorphic encryption and garbled circuits, without requiring data owners to remain online. In both approaches [10, 11], the two computation parties are trusted not to collude as they perform different roles, and could essentially be represented by different companies.

5.4 Preliminaries

5.4.1 Dimensionality Reduction

5.4.1.1 Principal Component Analysis (PCA)

PCA is an unsupervised dimensionality reduction technique (i.e. it does not utilize data labels). Consider a dataset with N training samples $\mathbf{x}_1, \cdots, \mathbf{x}_N$, where each sample has M features ($\mathbf{x}_i \in \mathbb{R}^M$). PCA performs spectral decomposition of the center-adjusted scatter matrix $\overline{\mathbf{S}} \in \mathbb{R}^{M \times M}$:

$$\overline{\mathbf{S}} = \sum_{i=1}^{N} (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \tag{5.1}$$

where μ is the mean vector $\mu = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i$, and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \cdots, \lambda_M)$ is a diagonal matrix of eigenvalues, with the eigenvalues arranged in a monotonically decreasing order (i.e. $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M$). The matrix $\mathbf{U} = [\mathbf{u}_1\ \mathbf{u}_2\ \cdots \mathbf{u}_M]$ is an $M \times M$ unitary matrix where $\mathbf{u}_j$ denotes the j^{th} eigenvector of the scatter matrix. For PCA, we retain the m principle components that correspond

to the m highest eigenvalues in order to obtain the projection matrix $\mathbf{U}_m \in \mathbb{R}^{M \times m}$. We obtain the reduced-dimensions' feature vector by:

$$\widetilde{\mathbf{x}}_\mathbf{i} = \mathbf{U}_m^T \mathbf{x}_\mathbf{i} \tag{5.2}$$

The parameter m determines to which extent the signal power is retained after dimensionality reduction. More precisely, while the original feature vectors have signal power $\sum_{j=1}^{M} \lambda_j$, the reduced-dimensions' feature vectors have power $\sum_{j=1}^{m} \lambda_j$. In Fig. 5.2, $m = 1$ while $M = 2$, and the red dots $\widetilde{\mathbf{x}}_\mathbf{i}$ are the projections of the blue dots $\mathbf{x}_\mathbf{i}$ on the principal component $\mathbf{u}^\mathbf{1}$.

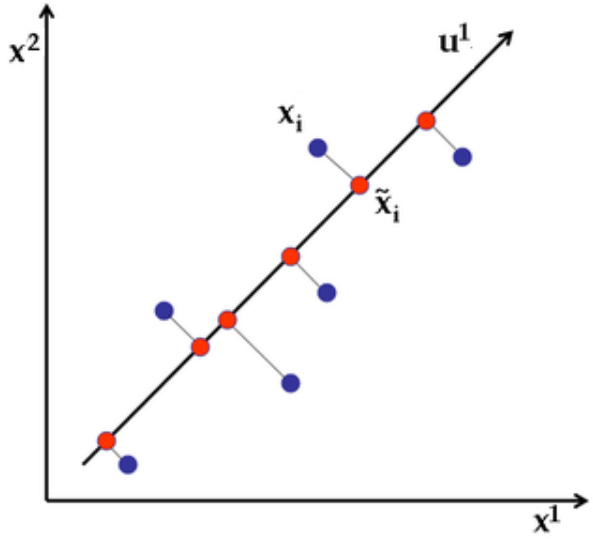


Figure 5.2: Data Projection

5.4.1.2 Discriminant Componenet Analysis (DCA)

DCA is a supervised method for dimensionality reduction [84, 98]. The target of DCA is selecting a projection hyperplane that can effectively discriminate between different classes.

In supervised learning, each sample $\mathbf{x}_i$ has a label y_i indicating its belonging to one of the K classes $C_1, C_2, \cdots, C_K$. Namely $y_i \in \{C_1, C_2, \cdots, C_K\}$. Let μ denotes the mean vector (of all the training samples), μ_k denotes the mean vector of the training samples in class C_k, and N_k denotes the number of samples in class C_k. The signal matrix is represented by the between-class scatter matrix:

$$\mathbf{S}_B = \sum_{k=1}^{K} N_k(\mu_k - \mu)(\mu_k - \mu)^T \tag{5.3}$$

The noise matrix is characterized by the following within-class scatter matrix:

$$\mathbf{S}_W = \sum_{k=1}^{K} \sum_{i:y_i=C_k} (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^T \tag{5.4}$$

The total scatter matrix $\overline{\mathbf{S}}$ can be written as follows:

$$\overline{\mathbf{S}} = \mathbf{S}_W + \mathbf{S}_B \tag{5.5}$$

Computing the DCA projection matrix $\mathbf{W}_{DCA}$ involves the following optimization criterion [84]:

$$\mathbf{W}_{DCA} = \underset{W:W^T[\overline{\mathbf{S}}+\rho\mathbf{I}]W=\mathbf{I}}{\arg\max} \; tr(W^T\mathbf{S}_B W) \tag{5.6}$$

where ρ is a ridge parameter added to the potentially ill-conditioned scatter matrix. $\mathbf{W}_{DCA}$ can also be obtained by solving the ***generalized eigenvalue decomposition (GEVD) problem***: $eig(\mathbf{S}_B, \overline{\mathbf{S}} + \rho\mathbf{I})$. This would ensure selecting a subspace that would maximize the separability between the utility classes (Fig. 5.1b).

To project the data, and obtain $\widetilde{x}_i$, we can also use Eq. 5.2; however, we set $m = K - 1$ where K is the number of classes since the maximum discriminant power is attained with that value of m [98].

5.4.1.3 Other Dimensionality Reduction (DR) Methods

Similar to PCA and DCA, many other DR approaches rely on these two steps: (1) computing the scatter matrices, (2) solving the (generalized) eigenvalue problem. These DR methods can

be distinguished in terms of which scatter matrices are needed, and which eigenvalue problem need to be solved: the regular eigenvalue problem $eig(A) : A\lambda = \lambda u$, or the generalized version $eig(A, B) : A\lambda = \lambda B u$.

Linear Discrimenant Analysis (LDA) is a method that is similar to DCA (utility-driven), however, rather than solving $eig(\mathbf{S}_B, \overline{\mathbf{S}} + \rho\mathbf{I})$ like DCA, LDA solves $eig(\mathbf{S}_B, \mathbf{S}_W)$, where $\mathbf{S}_W$ is the within-class scatter matrix (Eq. 5.4).

While DCA and LDA are utility-driven DR techniques, another class of DR approaches concentrates on utility-privacy optimization problems such as ***Multiclass Discriminant Ratio (MDR)*** [85]. MDR aims to maximize the separability of the utility classification problem (intended task), while minimizing the separability of the privacy-sensitive classification problem (undesirable or sensitive task). It assumes that each data sample has two labels: a utility label (for the intended purpose of the data), and a privacy label (sensitive information). Hence, we can obtain two between-class scatter matrices: one that is based on the utility labels $\mathbf{S}_{B_U}$, and the second computed using the privacy labels $\mathbf{S}_{B_P}$. To obtain a projection matrix $\mathbf{W}_{MDR}$ that maximizes the accuracy of the utility classification problem, and minimizes the accuracy of the privacy classification problem, MDR uses the following optimization criterion [85]:

$$J(W) = \frac{det(W^T \mathbf{S}_{B_U} W)}{det(W^T \mathbf{S}_{B_P} W)} \tag{5.7}$$

Similar to DCA, the MDR projection matrix $\mathbf{W}_{MDR}$ can be obtained by solving the the generalized eigenvalue decomposition problem: $eig(\mathbf{S}_{B_U}, \mathbf{S}_{B_P})$.

Other utility-privacy optimization DR methods can be found in [26] such as Differential Utility-Cost Analysis (DUCA). They also can be solved by using generalized eigenvalue decomposition (GEVD).

The above discussion highlights the importance of proposing protocols to compute the scatter matrices, and to perform the (generalized) eigenvalue problem. These two steps are the topics of sections 5.6 and 5.7.

5.4.2 Cryptographic Background

An important building block of our protocols is additive homomorphic encryption. There are multiple semantically-secure additive homomorphic encryption schemes, and without loss of generality, Paillier [119] will be used in this work as an example of such encryption schemes. Let function $\mathcal{E}_{pk}[\cdot]$ be an encryption operation indexed by a public key pk, and let $\mathcal{D}_{sk}[\cdot]$ be a decryption operation indexed by a secret key sk. The following rule holds for additive homomorphic encryption:

$$\mathcal{E}_{pk}[a+b] = \mathcal{E}_{pk}[a] \otimes \mathcal{E}_{pk}[b]$$

where $\otimes$ denotes the modulo multiplication operator in the encrypted domain. In addition, scalar multiplication can be achieved by: $\mathcal{E}_{pk}[a]^b = \mathcal{E}_{pk}[a \cdot b]$. Such encryption schemes only accept integers as plain text while machine learning data is expected to have real values. Hence, the data (feature values) should be discretized to obtain integer values [74].

We also utilize garbled circuits and oblivious transfer in our protocols. The main idea is having one party (the garbler) create an encrypted circuit that computes a function f, while the second party (the evaluator) executes this circuit on garbled input, and obtains the function output without learning any intermediate values. The garbled circuit is basically a collection of garbled gates, where each wire of this encrypted circuit would have two random cryptographic keys associated with it (for one and zero). The garbled input for the evaluator party is obtained from the garbling party by using oblivious transfer that does not allow the garbler to learn anything about the evaluator's input. The garbler would also provide the mapping from garbled output keys to bits to enable the evaluator to obtain said output in plain-text.

5.5 Problem Statement

5.5.1 Overview

We consider the case of horizontally-partitioned data across multiple data owners (Fig. 5.3). Suppose there are L data owners. Each data owner ℓ holds a set of feature vectors $x_i^\ell \in \mathbb{R}^M$ where

M is the number of features (dimensions), and $i = 1, ..., N^\ell$ (N^ℓ is the number of feature vectors held by data owner ℓ). Hence, each data owner ℓ would have a data matrix $X^\ell \in \mathbb{R}^{N^\ell \times M}$. Moreover, in supervised learning, each sample x_i^ℓ has a label y_i^ℓ indicating its belonging to one of K classes. Namely $y_i^\ell \in \{C_1, C_2, \cdots, C_K\}$.

In our system, a central entity (the data user) would like to compute the PCA (or DCA) projection matrix from the data horizontally-partitioned among multiple data owners. Many practical applications fall into this data sharing model. For example, with continuous authentication (CA) applications, a CA server needs to build authentication profiles for a group of registered users using machine learning algorithms, including dimensionality reduction techniques like PCA [39]. Hence, the CA server (the data user) would need to compute the PCA/DCA projection matrix using data distributed (horizontally) across all registered users (the data owners).

Traditionally, PCA/DCA was only used on the joint dataset (formed of data contributed by multiple data owners) in a centralized location. Computing the projection matrix $\mathbf{U}$ required the data owners to reveal their data before applying PCA/DCA. Hence, it is necessary to modify the computation of the projection matrix to make it distributed and privacy-preserving. This will be presented in sections 5.6.1 and 5.7, where we enable the data user to compute the projection matrices without compromising the privacy of data owners. To facilitate this privacy-preserving computation, we utilize a third-party (the crypto service provider) that will engage in a relatively short one round step with the data user to compute the projection matrix.

The projection matrix would then be used by the data owners to reduce the dimensions of their data. Such reduced dimensions data could later be used as an input to a certain privacy-preserving ML algorithm that performs classification [39] or regression [11].

5.5.2 Threat Model

The main privacy requirement of our protocols is enabling the data owners to preserve the privacy of their data. We consider the adversaries to be the computation parties: the data user and the crypto service provider (CSP). Neither of these parties should have access to any of the

data owner's input data or any intermediate values (e.g. data owner shares or scatter matrices) in clear text format. The data user should only learn the output which is the PCA/DCA projection matrix and the Eigen values.

The CSP's main role is facilitating the privacy-preserving computation of the scatter matrices (section 5.6.1). The CSP is assumed not to collude with the data user (similar to the assumptions for the privacy service provider in [10] and the CSP in [11]). The data user and the CSP can be different corporations that would not collude, at least in the interest of maintaining their reputation and their customer base.

We assume all participants to be honest-but-curious, i.e., we consider the semi-honest adversarial model. This means that all parties would correctly follow the protocol specification, but try to use the protocol transcripts to extract new information. Hence, both the data user and the CSP are considered semi-honest, non-colluding but otherwise untrusted servers. We also discuss extensions that could account for the possibility of collusion between the data user and a subset of the data owners in order to glean private information pertaining to a single data owner (section 5.6.3).

5.6 Privacy-Preserving PCA/DCA

In this section, we describe the privacy preserving computation of the scatter matrices $(\overline{\mathbf{S}}, \mathbf{S}_W, \mathbf{S}_B)$ and the PCA/DCA projection matrix. We assume that there are L data owners that are willing to cooperate with a certain data user to compute the scatter matrices. We also assume the existence of a crypto service provider (CSP) who is trusted not to collude with the data user. Figure 5.3 shows the overall system architecture and the interaction between different parties.

This section covers steps 1-4 in Fig. 5.3, while steps 5 and 6 show the usage of the PCA/DCA projection matrix by the data owner to transform its private data to lower dimensional data. This could be followed by using a privacy-preserving ML algorithm on the transformed data (such as classification [39] or regression [11]).

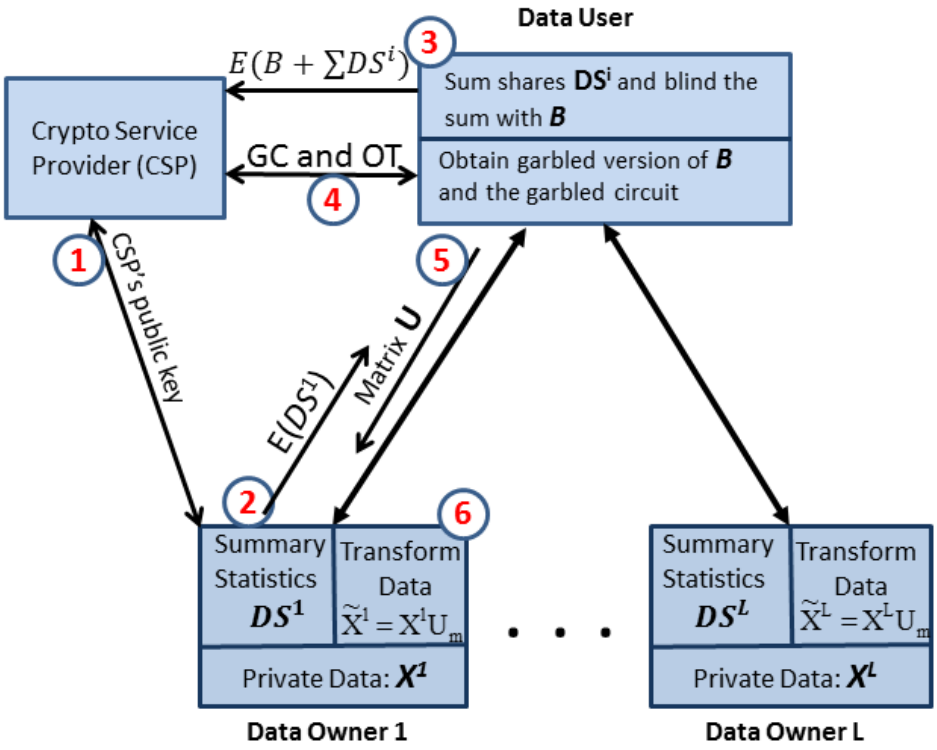


Figure 5.3: System architecture and protocols

In the following, all communication between the data owners, the data user and the CSP is assumed to be carried on secure channels using well-known methods like SSL/TLS, digital certificates, and signatures. Hence, we only concentrate on our protocols for brevity.

As mentioned in section 5.4.2, the individual shares have to be discretized before applying homomorphic encryption. Since such discretization involves scaling by an integer, such as $2^b - 1$, the resulting scatter matrix would take larger values than if it was directly computed from the original data. Therefore, the data user should divide all the scatter matrix elements by $(2^b - 1)^2$ after the protocol concludes.

We first present our protocols for the semi-honest model (sections 5.6.1 and 5.6.2), and follow that by an analysis of these protocol in section 5.6.3.

5.6.1 Privacy-Preserving PCA

We first describe the necessary equations to compute the total scatter matrix $\overline{\mathbf{S}}$ in a distributed way, and we follow that by presenting the protocol that performs the PCA computation in a privacy-preserving way.

The total Scatter Matrix $\overline{\mathbf{S}}$:

$\overline{\mathbf{S}}$ can be computed in an iterative fashion. Suppose there are L data owners, and denote P^ℓ as the set of training samples held by data owner ℓ. Each data owner ℓ can locally compute:

$$\mathbf{R}^\ell = \sum_{i \in P^\ell} \mathbf{x}_i \mathbf{x}_i^T \quad , \quad \mathbf{v}^\ell = \sum_{i \in P^\ell} \mathbf{x}_i \quad \text{and} \quad N^\ell = \left| P^\ell \right| \tag{5.8}$$

It can be shown that the total scatter matrix (eq. 5.1) is can be found by summing the partial contributions from each party:

$$\begin{aligned} \overline{\mathbf{S}} &= \sum_{i=1}^{N} (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T = \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T - N\mu\mu^T \\ &= \sum_{\ell=1}^{L} \mathbf{R}^\ell - \frac{1}{N}\mathbf{v}\mathbf{v}^T = \mathbf{R} - \frac{1}{N}\mathbf{v}\mathbf{v}^T \end{aligned} \tag{5.9}$$

where

$$\mathbf{R} = \sum_{\ell=1}^{L} \mathbf{R}^\ell \quad , \quad \mathbf{v} = \sum_{\ell=1}^{L} \mathbf{v}^\ell \quad \text{and} \quad N = \sum_{\ell=1}^{L} N^\ell \tag{5.10}$$

The data owners must not send the local shares ($[\mathbf{R}^\ell, \mathbf{v}^\ell, N^\ell]$ for $\overline{\mathbf{S}}$) to the data user in cleartext since they include statistical summaries of their data. Alternatively, the local shares can be encrypted using an additive homomorphic encryption scheme (such as Paillier's cryptosystem [119]) where the public key is provided by the CSP. After receiving these encrypted shares, the data user can aggregate them to compute the encrypted intermediate values (namely $\mathbf{R}$, $\mathbf{v}$ and N), send them to the CSP for decryption (after blinding the values), and use these aggregate values to compute the scatter matrix and the PCA projection matrix by using garbled circuits and oblivious transfer. Blinding refers to adding random numbers to these encrypted values to prevent the CSP from learning anything about the data even in its aggregated form (*"Blinding"* the values using an equivalent of one-time pad).

Privacy-Preserving PCA Protocol:

For computing the PCA projection matrix, the protocol is as follows (Fig. 5.3):

1. **Setup**: The CSP sends its public key pk for Paillier's cryptosystem to the data owners and the data user based on their request. This step could also include official registration of the data owners with the CSP.

2. **The data owners**: Each data owner ℓ would compute its own share $\mathbf{DS}^\ell = \{\mathbf{R}^\ell, \mathbf{v}^\ell, N^\ell\}$ using eq. 5.8. After discretizing all the values to obtain integer values, the data owners would then encrypt $\mathbf{R}^\ell$, $\mathbf{v}^\ell$ and N^ℓ using the CSP's public key to obtain $\mathcal{E}_{pk}[\mathbf{DS}^\ell] = \{\mathcal{E}_{pk}[\mathbf{R}^\ell], \mathcal{E}_{pk}[\mathbf{v}^\ell], \mathcal{E}_{pk}[N^\ell]\}$. Finally, each data owner ℓ sends $\mathcal{E}_{pk}[\mathbf{DS}^\ell]$ to the data user.

3. **The data user**: The data user receives $\mathcal{E}_{pk}[\mathbf{DS}^\ell] = \{\mathcal{E}_{pk}[\mathbf{R}^\ell], \mathcal{E}_{pk}[\mathbf{v}^\ell], \mathcal{E}_{pk}[N^\ell]\}$ from each data owner ℓ, and proceeds to compute the encryption of $\mathbf{R}$, $\mathbf{v}$ and N given by eq. 5.10. More explicitly, the data user is capable of computing $\mathcal{E}_{pk}[\mathbf{R}]$, $\mathcal{E}_{pk}[\mathbf{v}]$ and $\mathcal{E}_{pk}[N]$ from the encrypted data owner shares as follows:

$$\begin{aligned} \mathcal{E}_{pk}[\mathbf{R}] &= \otimes_{\ell=1}^{L} \mathcal{E}_{pk}[\mathbf{R}^\ell] \\ \mathcal{E}_{pk}[\mathbf{v}] &= \otimes_{\ell=1}^{L} \mathcal{E}_{pk}[\mathbf{v}^\ell] \\ \mathcal{E}_{pk}[N] &= \otimes_{\ell=1}^{L} \mathcal{E}_{pk}[N^\ell] \end{aligned} \tag{5.11}$$

   The data user adds some random integers to these aggregated values to mask them from the CSP, thus obtaining the blinded aggregated shares $\mathcal{E}_{pk}[\mathbf{R}']$, $\mathcal{E}_{pk}[\mathbf{v}']$ and $\mathcal{E}_{pk}[N']$ that can be sent to the CSP for decryption.

4. **The CSP and Data User**: Eigenvalue Decomposition (EVD) using Garbled Circuits

   (a) The CSP would use its private key to decrypt the blinded aggregated shares $\mathcal{E}_{pk}[\mathbf{R}']$, $\mathcal{E}_{pk}[\mathbf{v}']$ and $\mathcal{E}_{pk}[N']$ received from the data user. Without knowing the random values added by the data user, the CSP can not learn the aggregated values.

   (b) The CSP would then proceed to construct a garbled circuit to perform EVD on the scatter matrix computed from the aggregated shares. The input to this garbled circuit

is the garbled version of: (i) the blinded aggregate shares which were decrypted by the CSP (step 4a), and (ii) the blinding values which are generated and held by the data user (step 3). Since the CSP constructs the garbled circuit, it can obtain the garbled version of its input by itself. However, the data user needs to interact with the CSP using oblivious transfer to obtain the garbled version of its input: the blinding values. Using oblivious transfer guarantees that the CSP would not learn the blinding values held by the data user.

The garbled circuit constructed by the CSP takes the two garbled inputs and does the following: (1) computes the scatter matrix from the shares $\mathbf{R}'$, $\mathbf{v}'$ and N' after subtracting the data user blinding values added in the previous steps, and (2) follows that by performing Eigen decomposition on the scatter matrix to obtain the PCA projection matrix.

(c) The data user would receive the garbled circuit as its evaluator. This garbled circuit already has the CSP's garbled input which is the decrypted and blinded aggregate shares, and obtains the garbled version of the blinding values using oblivious transfer.

(d) Finally, the data user executes the garbled circuit and obtains the projection matrix and Eigen values as output.

The details of implementing the Eigenvalue Decomposition using garbled circuits is deferred to section 5.7.

5.6.2 Privacy-Preserving DCA

DCA computation requires the computation of the total scatter matrix $\overline{\mathbf{S}}$ and the signal matrix $\mathbf{S}_B$ (Eq. 5.6). Furthermore, some other DCA formulations [84] (in addition to LDA and some other DR techniques [26]) also used the noise matrix $\mathbf{S}_W$. Hence, we will outline the distributed computation of $\mathbf{S}_B$ and $\mathbf{S}_W$ in this section.

Computation of $\mathbf{S}_W$ and $\mathbf{S}_B$ could be different depending on whether the data owner has data that belongs to a single class or multiple classes. An example of multiple-classes data owner

(MCDO) would be spam email detection application where each data owner has spam and non-spam emails, which represent two classes. On the other hand, in active authentication systems, each data owner represents one class; hence, all their data would have the same label. The latter case is called single-class data owners (SCDO).

We will first describe the necessary equations to compute the scatter matrices $\mathbf{S}_B$ and $\mathbf{S}_W$ in a distributed way, and follow that by presenting the protocol that performs the DCA computation in a privacy-preserving way.

The Noise Matrix $\mathbf{S}_W$:

Note that the noise matrix $\mathbf{S}_W$ can be written as (please recall the notation in section 5.4.1.2):

$$\begin{aligned}\mathbf{S}_W &= \sum_{k=1}^{K} \sum_{i:y_i=C_k} (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^T \\ &= \sum_{k=1}^{K} \left(\sum_{i:y_i=C_k} \mathbf{x}_i \mathbf{x}_i^T - N_k \mu_k \mu_k^T \right) = \sum_{k=1}^{K} Q_k\end{aligned} \tag{5.12}$$

with the matrix $\mathbf{Q}_k = \sum_{i:y_i=C_k} \mathbf{x}_i \mathbf{x}_i^T - N_k \mu_k \mu_k^T$ representing a share that pertains to class k. Since each data owner ℓ maps to a single class k (in the SCDO case), we can write $\mathbf{Q}_k = \mathbf{Q}^\ell$ to have: $\mathbf{S}_W = \sum_{k=1}^{K} \mathbf{Q}_k = \sum_{\ell=1}^{L} \mathbf{Q}^\ell$.

As for the case of multiple-classes data owners (MCDO), each data owner ℓ would hold data belonging to different classes. Denote P_k^ℓ as the set of training samples held by data owner ℓ that belong to class k. For each class k, a data owner ℓ can locally compute:

$$\nu_k^\ell = \sum_{i \in P_k^\ell} \mathbf{x}_i \quad \text{and} \quad N_k^\ell = \left| P_k^\ell \right| \tag{5.13}$$

and would also compute $\mathbf{R}^\ell = \sum_{i \in P^\ell} \mathbf{x}_i \mathbf{x}_i^T$ which is not restricted to a certain class (similar to eq. 5.8).

It can be shown that eq. 5.12 can be written in terms of the above partial contributions from data owners as follows:

$$\mathbf{S}_W = \sum_{\ell=1}^{L} \mathbf{R}^{\ell} - \sum_{k=1}^{K} \frac{1}{N_k} \nu_k \nu_k^T = \mathbf{R} - \sum_{k=1}^{K} \frac{1}{N_k} \nu_k \nu_k^T \tag{5.14}$$

where

$$\mathbf{R} = \sum_{\ell=1}^{L} \mathbf{R}^{\ell} \quad , \quad \nu_k = \sum_{\ell=1}^{L} \nu_k^{\ell} \quad \text{and} \quad N_k = \sum_{\ell=1}^{L} N_k^{\ell} \tag{5.15}$$

The Signal Matrix $\mathbf{S}_B$:

For the signal matrix $\mathbf{S}_B$, both the single class (SCDO) and multiple classes (MCDO) can be computed similarly. The signal matrix $\mathbf{S}_B$ can be computed directly from the aggregate data $(\nu_k, N_k, \mathbf{v}, N)$ described in equations 5.10 and 5.15.

$$\begin{aligned} \mathbf{S}_B &= \sum_{k=1}^{K} N_k (\mu_k - \mu)(\mu_k - \mu)^T \\ &= \sum_{k=1}^{K} N_k (\frac{\nu_k}{N_k} - \frac{\mathbf{v}}{N})(\frac{\nu_k}{N_k} - \frac{\mathbf{v}}{N})^T \end{aligned} \tag{5.16}$$

Privacy-Preserving DCA Protocol:

Unlike the scatter matrix, both the noise and signal matrices use class mean values in their computation. This would require the data owners to send per-class shares to the data user. If a data owner wants to send only the shares related to the classes s/he has, this would leak knowledge of which classes this data owner has. Hence, it is recommended that all data owners send send shares representing all classes, and when a data owner does not own a certain class data, s/he can set that class shares to all zeros.

For the DCA formulation given in Eq. 5.6, we only need to compute the total scatter matrix $\overline{\mathbf{S}}$ and the signal matrix $\mathbf{S}_B$; hence, the following protocol will only concentrate on these two matrices.

For computing the DCA projection matrix, the protocol is as follows (Fig. 5.3):

1. **Setup**: The CSP sends its public key pk for Paillier's cryptosystem to the data owners and the data user based on their request. This step could also include official registration of the data owners with the CSP.

2. **The data owners**: Each data owner ℓ computes $\mathbf{R}^\ell$ from all its data, and for each class labeled k, the data owner computes ν_k^ℓ and N_k^ℓ. As noted earlier, if a certain data owner does not have samples pertaining to class k, s/he still generates ν_k^ℓ and N_k^ℓ but sets them to zeros. After discretizatiion and obtaining integer values, the data owner would encrypt $\mathbf{R}^\ell$ using the CSP's public key to obtain $\mathcal{E}_{pk}[\mathbf{R}^\ell]$, and would also encrypt the class-based shares (ν_k^ℓ, N_k^ℓ) to obtain $\left\{k, \mathcal{E}_{pk}[\nu_k^\ell], \mathcal{E}_{pk}[N_k^\ell]\right\}$ for each class k. Finally, the data owner would send its own share $\mathbf{DS}^\ell$, which includes the encrypted data mentioned above, to the data user.

3. **The data user**: From each data owner ℓ, the data user receives $\mathbf{DS}^\ell$ which includes $\mathcal{E}_{pk}[\mathbf{R}^\ell]$, and for each class k, it also includes $\left\{k, \mathcal{E}_{pk}[\nu_k^\ell], \mathcal{E}_{pk}[N_k^\ell]\right\}$. The data user then proceeds to reconstruct the encryption of $\mathbf{R}$, ν_k's and N_k's as follows:

$$\mathcal{E}_{pk}[\mathbf{R}] = \otimes_{\ell=1}^{L} \mathcal{E}_{pk}[\mathbf{R}^\ell] \tag{5.17}$$

and for each class $k \in 1, \cdots, K$, the data user computes

$$\begin{aligned} \mathcal{E}_{pk}[\nu_k] &= \otimes_{\ell=1}^{L} \mathcal{E}_{pk}[\nu_k^\ell] \\ \mathcal{E}_{pk}[N_k] &= \otimes_{\ell=1}^{L} \mathcal{E}_{pk}[N_k^\ell] \end{aligned} \tag{5.18}$$

The data user adds some random integers to these aggregated values to mask them from the CSP, thus obtaining the blinded shares $\mathcal{E}_{pk}[\mathbf{R}']$ in addition to $\mathcal{E}_{pk}[\nu_k']$'s and $\mathcal{E}_{pk}[N_k']$. These blinded values would then be sent to the CSP.

4. **The CSP and Data User**:

   Generalized Eigenvalue Decomposition (GEVD) using Garbled Circuits

   (a) The CSP would use its private key to decrypt the blinded shares $\mathcal{E}_{pk}[\mathbf{R}']$, $\mathcal{E}_{pk}[\nu_k']$'s and $\mathcal{E}_{pk}[N_k']$'s received from the data user. Without knowing the random values added by the data user, the CSP can not learn the aggregated values.

   (b) The CSP would then proceed to construct a garbled circuit to perform GEVD on the signal and the total scatter matrices computed from the aggregated shares. The input

to this garbled circuit is the garbled version of: (i) the blinded aggregate shares which were decrypted by the CSP (step 4a), and (ii) the blinding values which are generated and held by the data user (step 3). Since the CSP constructs the garbled circuit, it can obtain the garbled version of its input by itself. However, the data user needs to interact with the CSP using oblivious transfer to obtain the garbled version of its input: the blinding values. Using oblivious transfer guarantees that the CSP would not learn the blinding values held by the data user.

The garbled circuit constructed by the CSP takes the two garbled inputs and does the following:

(1) removes the blinding values from the aggregated shares, uses the shares ν_k's and N_k's to compute $\mathbf{v}$ and N using:

$$\mathbf{v} = \sum_{k=1}^{K} \nu_k \quad \text{and} \quad N = \sum_{k=1}^{K} N_k \tag{5.19}$$

(2) compute the total scatter matrix $\overline{\mathbf{S}}$ using Eq. 5.9, and the signal matrix $\mathbf{S}_B$ using Eq. 5.16, and (3) follows that by performing Generalized Eigen decomposition $eig(\mathbf{S}_B, \overline{\mathbf{S}}+\rho\mathbf{I})$ to compute the DCA projection matrix.

(c) The data user would receive the garbled circuit as its evaluator. This garbled circuit already has the CSP's garbled input which is the decrypted and blinded aggregate shares, and obtains the garbled version of the blinding values using oblivious transfer.

(d) Finally, the data user executes the garbled circuit and obtains the projection matrix and Eigen values as output.

The details of implementing the GEVD using garbled circuits will be deferred to section 5.7.

5.6.3 Security Analysis

To prove the security of the proposed protocols, we use the simulation paradigm given in [120]. Our proposed protocols in sections 5.6.1 and 5.6.2 are secure multi-party protocols in which the

participants are the data owners, the data user and the CSP. We assume that all the parties participating in our protocols are semi-honest. In the semi-honest adversarial model, corrupted parties would still correctly follow the protocol specification, however, the adversary obtains the internal state of the corrupted parties including the exchanged messages, and attempts to use this to extrapolate new information. We further assume that the data user and the CSP do not collude, which can be ensured by having two different organizations control each of these two parties.

In general, in a secure m-party computation, m participants $P = \{P_1, ..., P_m\}$ having input values $\bar{x} = \{x_1, ..., x_m\}$ aim to compute a function $f(\bar{x}) = (f_1(\bar{x}), ..., f_m(\bar{x}))$ without revealing their inputs to each other. At the end, each party P_i obtains its output $f_i(\bar{x})$. A protocol securely computes the function f if the messages received by parties during the protocol do not leak information. To prove it formally, simulation technique [120] can be used to show that each party's view can be simulated using its input and output. View of a party is defined as its input, internal random tape, and the messages received throughout the protocol.

Let π be a m-party protocol to compute function f. The view of the party P_i is denoted as $\mathbf{view}_i^{\pi}(\bar{x})$ and $\overset{c}{\equiv}$ denotes the computational indistinguishability. The security of a deterministic multi-party protocol under semi-honest model is formally defined as follows:

Definition 1 *The protocol π securely computes function f under semi-honest threat model if there exist probabilistic polynomial-time algorithms $\mathcal{S}_1$,..., $\mathcal{S}_m$ such that*

$$\{\mathcal{S}_i(x_i, f_i(\bar{x}))\} \overset{c}{\equiv} \{\mathbf{view}_i^{\pi}(\bar{x})\}$$

for all $i \in \{1, ..., m\}$

Hence, a multi-party protocol is secure if each party's view can be simulated by a simulator. Since there are three types of parties in our protocols such as the data owners, the data user and the CSP, we define the simulator for each of them to prove the security of our protocols. The output of the proposed protocols is the projection matrix which must be obtained by the data user only (i.e. $f_i(\bar{x}) = \mathbf{U}$ for the data user). Hence, data owners and the CSP receive no output at the end of the protocols (i.e. $f_i(\bar{x}) = \perp$ for all data owners and the CSP).

Data Owners: None of the data owners would have any interaction with other data owners in our protocols. They only send their encrypted shares to the data user (step 2 in sections 5.6.1 and 5.6.2). Each data owner has its own data and the CSP's public key as her input. The data owners do not get any output. Since the data owners do not receive any message during the protocol execution, the view of each data owner consists of her input only. It is clear that the view of each data owner is simulatable given her input and output. Therefore, none of the data owners can learn extra information during the protocol.

Data User: The data user's input is the CSP's public key and the blinding values, and the output is projection matrix $\mathbf{U}$. The view of the data user includes its input, the data owner shares $\mathcal{E}_{pk}[\mathbf{DS}^{\ell}]$, the garbled circuit and the input for the garbled circuit. The data user receives $\mathcal{E}_{pk}[\mathbf{DS}^{\ell}]$ from each one of the L data owners (step 3 in sections 5.6.1 and 5.6.2). For privacy-preserving PCA (section 5.6.1), the data user receives $(M^2 + M + 1) \cdot L$ Paillier ciphertexts (where M is the number of features). The simulator generates $(M^2 + M + 1) \cdot L$ random numbers in the range of a Paillier ciphertext, and these random numbers are computationally indistinguishable from the $(M^2+M+1)\cdot L$ Paillier ciphertexts (data owner shares) because of the semantic security of Paillier cryptosystem. In addition, the data user receives the garbled circuit and an input for the garbled circuit from the CSP. When the simulator produces a random garbled circuit and a random number as input, these are computationally indistinguishable from the ones provided by CSP. Therefore, none of the messages received by the data user during the protocol provide any information except the output of the protocol.

For DCA, a similar discussion follows, with the exception that the data user receives an additional $(K \cdot M + 1) \cdot L$ Paillier ciphertexts where K is the number of class labels.

CSP: The CSP's input is its public and secret keys and the output is nothing. During the protocol the CSP does not receive any value from the data owners. The CSP only receives $\mathcal{E}_{pk}[\mathbf{R}']$, $\mathcal{E}_{pk}[\mathbf{v}']$ and $\mathcal{E}_{pk}[N']$ from the data user to decrypt (step 4 in sections 5.6.1 and 5.6.2). Since the CSP has secret key, it can decrypt these values and obtain $\mathbf{R}'$, $\mathbf{v}'$, and N'. These values are masked by the data user by adding random numbers to $\mathbf{R}$, $\mathbf{v}$, and N. The simulator generates random

values equivalent in size to $\mathbf{R}'$, $\mathbf{v}'$, and N', and they will be computationally indistinguishable from $\mathbf{R}'$, $\mathbf{v}'$, and N'. As a result, the messages received by the CSP during the protocol do not leak information.

Since we provide a simulator for each party in the protocols, it can be concluded that the proposed protocols are secure under semi-honest threat model. However, a malicious data user may use a single data owner input to compute the projection matrix in an attempt to reveal some private information about that data owner. Certain measures from the literature could be utilized to enable the CSP to ensure that the aggregates received were based on input from all the intended data owners. Such method was outlined by [11] where zero knowledge proofs could be used by the data user to prove to the CSP that the ciphertexts it received were the product of all data owners' ciphertexts (section IV.G in [11]). Obviously, such precautions for the malicious adversary increase the computation cost.

5.7 Eigenvalue Decomposition Garbled Circuit

5.7.1 Eigenvalue Decomposition (EVD)

EVD aims to find the eigenvalues λ's and eigenvectors u's of a matrix A that satisfy the following equation:

$$Au = \lambda u \tag{5.20}$$

EVD could be performed in a variety of ways. Some methods such as the QR algorithm find all the Eigen vectors/values at once. However, PCA is used to reduce the dimensions; hence, not all the Eigen vectors are needed. For this reason, we will rely on methods that only find a subset of the Eigenvalues/vectors to avoid the extra computation associated with finding unneeded Eigenvectors.

One of the most notable algorithms for EVD is the power iteration method. This method finds the dominant Eigen value (largest value) with its associated Eigen vector. A matrix deflation method could be used afterwards to remove the effect of the already found dominant Eigen value

while leaving the remaining Eigen values unchanged. By repeatedly applying the power iteration method and matrix deflation, we can find the required number of Eigen vectors.

The power iteration method starts with a non-zero vector x_0, and attempts to reach a good approximation of the dominant Eigenvector in a number of iterations J. In each iteration j, a new approximation is computed as $x_j = Ax_{j-1}$ where A is the matrix for which we want to find the Eigen values/vectors. The algorithm stops when the difference between x_j and x_{j-1} is negligible. It can be seen that if J is known, this iterative computation is equivalent to $x_J = A^J x_0$; however, computing A^J would most likely lead to an overflow. Hence, scaling x_j in each iteration is important (see algorithm 6).

Algorithm 6 Eigen Decomposition of matrix A

Input: the matrix A of dimensions M x M
Input: the number of required Eigen values/vectors m
Input: the number of iterations J
Output: Eigenvectors U
Output: Eigenvalues Λ

Set x_0 to be a unit vector
for `i = 1 to` m **do**
    $x = x_0$ ▷ initialize the Eigenvector
    for `j = 1 to` J **do**
        $x = Ax$
        $\lambda = ||x||_\infty$
        $x = \frac{x}{\lambda}$
    $x = \frac{x}{||x||_2}$ ▷ Normalize vector x
    $A = A - xx^T Axx^T$ ▷ Deflation
    Set the i^{th} Eigenvalue $\Lambda_i = \lambda$
    Set the i^{th} Eigenvector $U_i = x$

As input, algorithm 6 takes a matrix A, the number of required Eigenvalues/vectors m, and the number of iterations for the power method J. The inputs m and J are provided by the data user. Algorithm 6 outputs the Eigenvalues and vectors. The algorithm has two loops: the outer loop runs m times which is the number of required Eigenvectors (principal components), while the inner loop performs the power method in J iterations. For each Eigenvector, the power method is

run followed by the deflation method. In line 6, the infinity norm of the vector x is computed, and used in step 7 to scale the vector x down (to prevent overflow).

The deflation method is shown in step 10 and is preceded by normalizing the Eigenvector x in step 9. It should be noted that the order of performing the deflation term xx^TAxx^T is important for computational reasons. A less efficient way would involve the following order (starting from the inside brackets): $((xx^T)A)xx^T$ which would cost one outer product and two full matrix multiplications bringing the cost up to $\mathcal{O}(M^2 + 2M^3)$. Alternatively, recognizing that the term x^TAx is a number would motivate the following order of operations: $x(x^T(Ax))x^T$ bringing the cost down to $\mathcal{O}(M + 3M^2)$. Thus, dropping the expensive $\mathcal{O}(M^3)$ from the cost of deflation.

As can be seen from algorithm 6, setting the number of required Eigenvectors m to a high number would increase the computation time. It would be possible for m to be set to a low number initially, and based on the Eigenvalues obtained as output, the data user can choose to generate additional Eigenvectors. The data user can determine if enough number of Eigenvectors were obtained either using (1) the Kaiser method: if any Eigenvalues had values less than one, then there is no need to generate more Eigenvectors, or (2) the scree test: where the Eigenvalues can be plotted, and if the curve is still steep after having m Eigenvalues, more Eigenvectors could be generated.

5.7.2 Generalized Eigenvalue Decomposition (GEVD)

Given the matrices A and B, the GEVD aims to find the eigenvalues λ's and eigenvectors u's that satisfy the following equation:

$$Au = \lambda Bu \tag{5.21}$$

One might think of reducing this problem to the regular EVD by computing the inverse of B, and attempting to solve $B^{-1}Au = \lambda u$ instead (using algorithm 6). However, $B^{-1}A$ is not guaranteed to be a symmetric matrix. An important property of the scatter matrices is that they are symmetric. Eigenvalues of a symmetric matrix are always real; thus enabling us to have simpler

implementation of the power method that does not involve complex values (section 5.7.3). Hence, an alternative formulation would be needed.

Assuming that the matrix B is positive definite, we can decompose B using Cholesky decomposition to obtain a lower triangular matrix L such that $B = LL^T$.

$$\begin{aligned} Au &= \lambda Bu \\ Au &= \lambda LL^T u \\ L^{-1}AL^{-T}L^T u &= \lambda L^T u \\ Cx &= \lambda x \end{aligned} \tag{5.22}$$

Hence, we can solve $Cx = \lambda x$ where $x = L^T u$ and $C = L^{-1}AL^{-T}$. It should be noted that the matrix C is symmetric.

$$(L^{-1}AL^{-T})^T = (L^{-T})^T A^T (L^{-1})^T = L^{-1}AL^{-T} \tag{5.23}$$

Thus, by solving $Cx = \lambda x$, we can ensure that all eigenvalues are real, which would reduce the computational complexity (by avoiding complex numbers).

Algorithm 7 outlines the GEVD procedure. For the specific case of DCA, the matrix $A = \mathbf{S}_B$ while $B = \overline{\mathbf{S}} + \rho I$ (section 5.4.1.2). We know that the total scatter matrix is positive semi-definite, and by adding ρI, we can ensure that matrix B would be positive definite. Thus, it would be safe to use Cholesky decomposition on B (step 1 of algorithm 7). Steps 2 and 3 are required to compute the matrix C which is real symmetric matrix; thus, we can use algorithm 6 to obtain real Eigenvalues and Eigenvectors. The final step would be obtaining the Eigenvectors u's associated with the original problem $Au = \lambda Bu$ by performing back substitution (step 5 of algorithm 7).

5.7.3 Implementation

We implemented our garbled circuit using Obliv-C [121]. Obliv-C includes an implementation of Yao's garbled circuit protocol and oblivious transfer, and it further incorporates recent optimizations. Obliv-C provides an extensible secure computation programming tool with the ability

Algorithm 7 Generalized Eigen Decomposition of (A, B)

Input: the symmetric matrix A of dimensions M x M
Input: the symmetric matrix B of dimensions M x M
Input: the number of required Eigen values/vectors m
Input: the number of iterations J
Output: Eigenvectors U
Output: Eigenvalues Λ

1: L = Cholesky(B)
2: Solve for Y: $LY = A$ ▷ Forward Substitution
3: Solve for C: $LC^T = Y^T$ ▷ Forward Substitution
4: Use algorithm 6 on C to find Eigenvalues Λ and Eigenvectors X
5: Solve for U: $L^T U = X$ ▷ Back Substitution

to be integrated with standard C code. It basically provides garbled integer and binary data types similar to the standard C data types, and the basic arithmetic and logic operations associated with them.

In order to implement algorithms 6 and 7, we implemented fixed-point arithmetic functions, and a linear algebra library. We used fixed-point arithmetic for its speed in comparison to floating point arithmetic. We utilized 32-bits integer and used the format Q15.16. Functions for multiplication and division were created to work with Q15.16 fixed-point numbers.

Because of the need to normalize the Eigenvectors (step 9 in algorithm 6), we needed to compute the square root using an iterative algorithm: $y_{i+1} = 0.5(y_i + \frac{x}{y_i})$ where x is the value for which the square root is desired, and y_i and y_{i+1} are the previous and current estimations. It can be seen that each iteration includes a division. Furthermore, to normalize a certain vector, each of its values has to be divided by the resulting square root. Knowing that multiplication is generally more efficient than division, we should find the inverse square root (instead of the square root); thus, performing multiplication by the vector rather than division. Moreover, the iterations required to compute the inverse square root do not include divisions: $y_{i+1} = y_i(1.5 - 0.5xy_i^2)$. Hence, avoiding divisions for all of the vector normalization steps.

In addition, we implemented a basic linear algebra library that performs matrix multiplication and addition, dot products, scaling or matrices and vectors, computing norms, and some other minor

Table 5.1: Distributed PCA Efficiency

Dataset	Features	Classes	Avg. Data Owner time	Avg. Data User Coll / Add time	CSP Decryption time	Eigenvalue Decomposition using Garbled Circuits
Diabetes	8	2	0.63 sec	10 ms	0.67 sec	13.8 sec (8)
Breast Cancer	10	2	0.93 sec	11 ms	1 sec	20.7 sec (8)
Australian	14	2	1.7 sec	12 ms	1.8 sec	37.4 sec (8)
German	24	2	5 sec	17 ms	5 sec	3.28 min (15)
Ionosphere	34	2	9.8 sec	24 ms	9.9 sec	6.44 min (15)
SensIT Acoustic	50	3	22.5 sec	40 ms	22.7 sec	13.8 min (15)

Table 5.2: Distributed DCA Efficiency

Dataset	Features	Classes	Avg. Data Owner time	Avg. Data User Coll / Add time	CSP Decryption time	Generalized Eigenvalue Decomposition using Garbled Circuits
Diabetes	8	2	0.7 sec	12 ms	0.8 sec	4.0 sec (1)
Breast Cancer	10	2	1.2 sec	13 ms	1.3 sec	5.8 sec (1)
Australian	14	2	2.1 sec	15 ms	2.2 sec	12.1 sec (1)
German	24	2	5.67 sec	22 ms	5.87 sec	46.6 sec (1)
Ionosphere	34	2	11.2 sec	29 ms	11.6 sec	1.96 min (1)
SensIT Acoustic	50	3	26.2 sec	48 ms	26.9 sec	6.72 min (2)

operations. Since most of the matrices are symmetric, even after deflation, we also implemented symmetric matrices specific data structures and math operations. Furthermore, for algorithm 7, we implemented Cholesky decomposition, and forward and back substitution.

5.8 Experiments

In this section, we evaluate our privacy-preserving PCA/DCA protocols in terms of efficiency and accuracy (performance of ML algorithms). We implemented our protocols using Java and Obliv-C [121]. For homomorphic encryption using Paillier's cryptosystem, we used the Java library: THEP [122]. For some of our experiments, we also utilized python including libraries like Scikit-Learn and NumPy.

Table 5.3: Privacy-Preserving DCA Accuracy

Dataset	F1 Score (Our Implementation)	F1 Score (Numpy)
Diabetes	76.5%	76.4%
Breast Cancer	96.9%	96.8%
Australian	86.1%	85.5%
German	72.7%	73.8%
Ionosphere	84.3%	83.4%
SensIT Acoustic	67.8%	68.4%

The datasets used were from the UCI machine learning repository [123]. While PCA and DCA work for any type of machine learning algorithm, we chose classification since the number of datasets available for classification, which was 255, far outnumbered those available for clustering or regression (around 55 each). Since the efficiency of our protocols depend on the data dimensions (number of features), we chose datasets with varying numbers of features: 8-50. Both the number of features and classes for each dataset can be seen in tables 5.1 and 5.2.

All the experiment were performed on a commodity computer with i5-6600K CPU @ 3.5GHz and 8GB of RAM. In all experiments, we used SVM for classification, and performed cross validation and grid search to find the SVM parameters C and γ. The number of data owners was set to 10 in all experiments. The initial stage of aggregating encrypted shares was implemented using Java, while the second stage where these encrypted shares were utilized to compute the projection matrix (using a garbled circuit) was implemented using Obliv-C.

5.8.1 Efficiency

Tables 5.1 and 5.2 show the timing data for performing privacy-preserving PCA/DCA on different datasets when using Paillier's key length of 1024 bits. In these tables, the *"Avg. data owner time"* refers to the total time it took the data owner to compute the individual shares and encrypt them. The *"Avg. data user Coll/Add time"* represents the time needed to *collect* each individual share from each data owner, and *add* it to the current sum of these shares (in the encrypted domain). We also show the time it took the CSP to decrypt the blinded aggregated values received from the data user. Finally, we show the time needed to run the Eigenvalue Decomposition using

garbled circuits in order to compute the PCA/DCA projection matrix. The value between brackets refers to the number of principal components generated using the garbled circuit. Naturally, for a given dataset, reducing this number would decrease the computation time. As will be seen in the next section, even 15 PCs for the SensIT Acoustic dataset is enough to achieve adequate accuracy. Finally, it can be noticed that increasing the dimension of data would increase the computation time for all stages of our protocols especially the Eigenvalue Decomposition. However, even at 50 dimensions, the computation time was still reasonable especially that no specialized servers were used in our experiments.

5.8.2 Accuracy

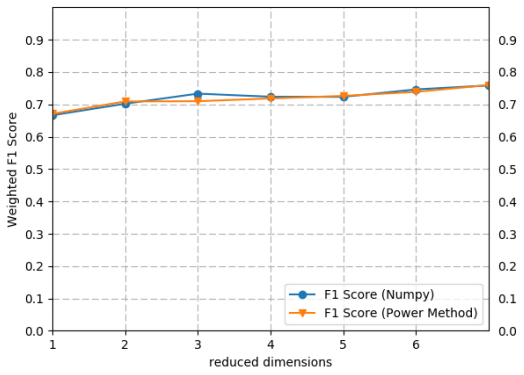
Fig. 5.4 and Table 5.3 show the results of performing experiments to test the accuracy of classification tasks after using our privacy-preserving protocols (in order to compare such results to those obtained using the python library Numpy).

We use the weighted F1 score as a measure for testing the accuracy of classifiers (it is basically an F1 score that considers the labels imbalance). The F1 score can be thought of as a weighted average of the precision and recall scores, and a classifier is at its best when its F1 score is 1, and worst at 0.

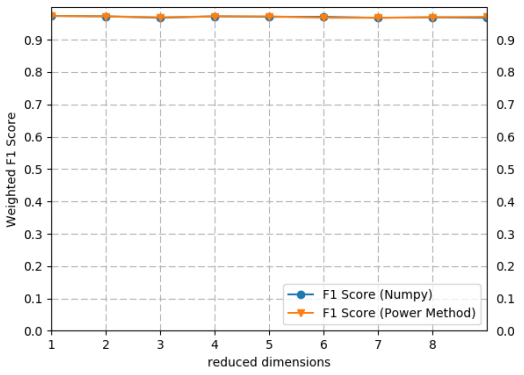
Fig. 5.4 and Table 5.3 show the accuracy results for our privacy-preserving PCA and DCA protocols, respectively. The DCA results are shown in a table with one value each since DCA projects the data to $K-1$ dimensions (where K is the number of class labels), while PCA projects the data to a variable number of dimensions. These results show that our protocols are correct, and their results are equivalent to the ones obtained using Numpy. It should be noted that the small fluctuation in the weighted F1 score is mostly due to the SVM parameter selection, however, it can still be seen that the accuracy of both methods are almost the same.

5.9 Conclusion

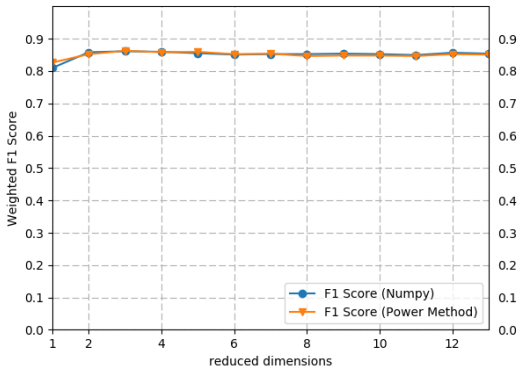
We introduced efficient privacy-preserving protocols for computing PCA/DCA on horizontally-partitioned data (distributed across multiple data owners). These protocol are based on additive homomorphic encryption and garbled circuits, and they were implemented using Java and Obliv-C. We performed experiments which have shown that our protocols are efficient, and that they maintain the utility for data users. In our future work, we intend to extend our protocols to the Kernel version of PCA/DCA. This would mostly require utilizing approximate kernels such as random Fourier features. They would also require extending our protocols to support higher dimensions which can be achieved by Lanczos methods, multi-threading or a combination of both.



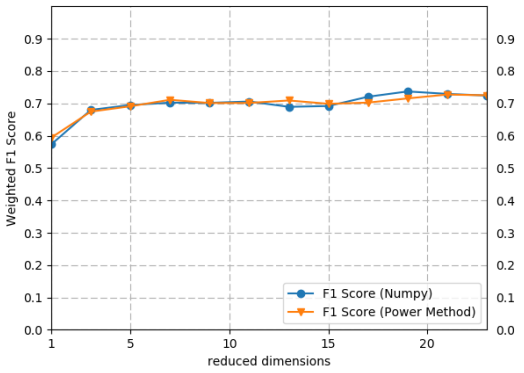
(a) Diabetes dataset



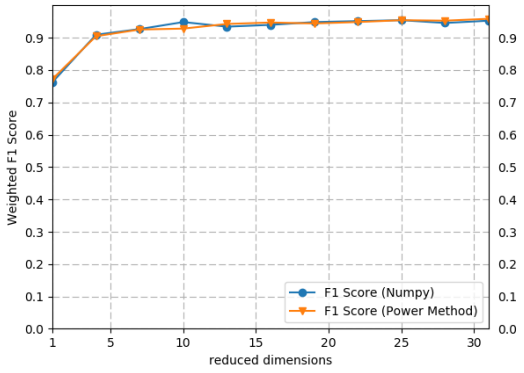
(b) Breast cancer dataset



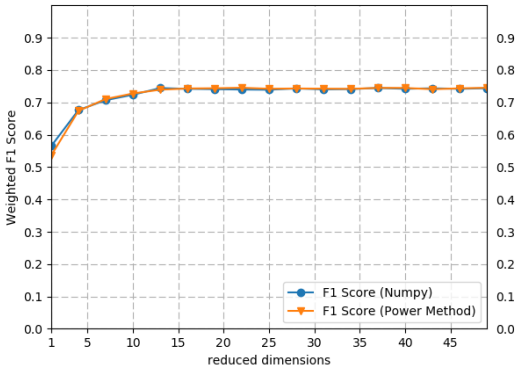
(c) Australian credit dataset



(d) German credit dataset



(e) Ionosphere dataset



(f) SensIT Acoustic dataset

Figure 5.4: Privacy-Preserving PCA Accuracy

CHAPTER 6. GENERAL CONCLUSIONS AND FUTURE WORK

This research is focused on building continuous authentication systems that are more resilient to attacks related to the use of machine learning algorithms. We first surveyed the recent literature, in chapter 2, to explore the privacy-related threats to machine learning applications, and the techniques utilized for privacy-preservation: cryptographic, and perturbation approaches. In chapter 3, we demonstrated the feasibility of reconstruction attacks against gesture-based continuous authentication systems. We concluded that such vulnerabilities can be thwarted using two steps: (1) utilizing privacy-preserving machine learning algorithms in order to prevent full-profile attacks, and (2) avoiding returning the decision value (or probability) of testing a single sample. For the second step, we proposed aggregating the results of testing multiple samples, and only returning a binary result of this testing. We showed the effectiveness of this approach in chapter 3. As for the first step, we explored the potential of using supervised dimensionality reduction (DR) techniques for privacy-preservation in chapter 4. We performed experiments to demonstrate that supervised DR techniques (DCA and MDR) are effective at achieving two goals: (1) Security: non-invertibility of the dimensionality reduced feature vector, and (2) privacy: preventing undesired inference.

Since many DR techniques (PCA/DCA/MDR) require users' private data to compute the best projection matrix, we proposed privacy-preserving PCA/DCA protocols that can work on horizontally-partitioned data (in chapter 5). For this purpose, we designed and implemented hybrid protocols that utilize additive homomorphic encryption and garbled circuits. These protocols are needed as many systems, including continuous authentication, would have multiple data owners that might be reluctant to share their data in plain-text format. The resulting projection matrices are used by data owners to transform their data, and only send the transformed feature vectors to the data user. This system is suitable for continuous authentication as it enables training and testing using transformed data from multiple data owners.

For future research, we intend on focus on two directions: (1) extending our privacy-preserving protocols for PCA/DCA to the non-linear domain (specifically Kernel methods), and (2) working on Locally-differentially private (LDP) machine learning algorithms.

It is known that Kernel PCA could improve the utility as there are cases where different classes are not linearly separable. However, the common technique for performing Kernel PCA relies on computing the kernel matrix which requires dot products between all samples from all data owners. Such operations might require interaction between data owner which is not practical. Furthermore, projecting new samples would require storing original data samples. Hence, Adapting the PCA protocols to the commonly used Kernel PCA is not straightforward. Thus, we intend to use approximate Kernel functions to transform each sample explicitly, which can be used for computing the projection matrices using the same protocols we proposed in chapter 5. Since the approximate kernels produce high-dimensional explicit non-linear features, we will also work on distributed DR-technique-specific feature selection methods, to control the high number of dimensions. As further research, we intend to extend our garbled circuits implementation in chapter 5 to enable Eigenvalue decomposition on higher dimensions by utilizing Lanczos algorithm.

Finally, we also intend to work on Locally-differentially private (LDP) machine learning algorithms. Unlike traditional differential privacy, LDP does not require a trusted data curator, and each data owner can send a perturbed version of their data to an untrusted server (with LDP guarantees). LDP is based on randomized response techniques, and is used mostly for frequency estimation. We intend to use LDP techniques to enable building ML classifiers. We found that Naive Bayes would be the most suitable candidate, since estimating the probabilities of classes, and features, would be similar to frequency estimation. Hence, we intend to build an LDP Naive Bayes algorithm that would utilize the recent encoding/perturbation advances in LDP [124].

BIBLIOGRAPHY

[1] Kun Liu, Hillol Kargupta, and Jessica Ryan. "Random Projection-Based Multiplicative Data Perturbation for Privacy Preserving Distributed Data Mining". In: *IEEE Trans. on Knowl. and Data Eng.* 18.1 (Jan. 2006), pp. 92–106. ISSN: 1041-4347. DOI: 10.1109/TKDE.2006.14. URL: http://dx.doi.org/10.1109/TKDE.2006.14.

[2] Arvind Narayanan and Vitaly Shmatikov. "Robust de-anonymization of large sparse datasets". In: *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE. 2008, pp. 111–125.

[3] Matt Fredrikson, Somesh Jha, and Thomas 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.

[4] M Bishop Christopher. *PATTERN RECOGNITION AND MACHINE LEARNING*. Springer-Verlag New York, 2006.

[5] Martin Abadi et al. "Deep learning with differential privacy". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2016, pp. 308–318.

[6] Dan Bogdanov et al. "Privacy-preserving statistical data analysis on federated databases". In: *Annual Privacy Forum*. Springer. 2014, pp. 30–55.

[7] Jianjiang Feng and Anil K Jain. "Fingerprint reconstruction: from minutiae to phase". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 33.2 (2011), pp. 209–223.

[8] Mohammad Al-Rubaie and J Morris Chang. "Reconstruction Attacks Against Mobile-Based Continuous Authentication Systems in the Cloud". In: *IEEE Transactions on Information Forensics and Security* 11.12 (2016), pp. 2648–2663.

[9] Reza Shokri et al. "Membership inference attacks against machine learning models". In: *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE. 2017, pp. 3–18.

[10] Zekeriya Erkin et al. "Generating private recommendations efficiently using homomorphic encryption and data packing". In: *Information Forensics and Security, IEEE Transactions on* 7.3 (2012), pp. 1053–1066.

[11] Valeria Nikolaenko et al. "Privacy-preserving ridge regression on hundreds of millions of records". In: *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE. 2013, pp. 334–348.

[12] Raphael Bost et al. "Machine learning classification over encrypted data." In: *NDSS*. Vol. 4324. 2015, p. 4325.

[13] Dan Bogdanov et al. “Implementation and Evaluation of an Algorithm for Cryptographically Private Principal Component Analysis on Genomic Data”. In: *Proceedings of the 3rd International Workshop on Genome Privacy and Security (GenoPri'16)*. GenoPri '16. Chicago, USA, 2016, pp. 1–8. URL: http://2016.genopri.org/uploads/3/9/9/9/39999711/genopri16_paper_10.pdf.

[14] Keith Bonawitz et al. “Practical Secure Aggregation for Privacy-Preserving Machine Learning”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2017, pp. 1175–1191.

[15] Olga Ohrimenko et al. “Oblivious Multi-Party Machine Learning on Trusted Processors.” In: *USENIX Security Symposium*. 2016, pp. 619–636.

[16] Cynthia Dwork, Aaron Roth, et al. “The algorithmic foundations of differential privacy”. In: *Foundations and Trends® in Theoretical Computer Science* 9.3–4 (2014), pp. 211–407.

[17] Nicolas Papernot et al. “Semi-supervised knowledge transfer for deep learning from private training data”. In: *arXiv preprint arXiv:1610.05755* (2016).

[18] Ilya Mironov. “Renyi differential privacy”. In: *Computer Security Foundations Symposium (CSF), 2017 IEEE 30th*. IEEE. 2017, pp. 263–275.

[19] Cynthia Dwork et al. “Analyze gauss: optimal bounds for privacy-preserving principal component analysis”. In: *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*. ACM. 2014, pp. 11–20.

[20] Moritz Hardt and Eric Price. “The noisy power method: A meta algorithm with applications”. In: *Advances in Neural Information Processing Systems*. 2014, pp. 2861–2869.

[21] Kamalika Chaudhuri, Anand D Sarwate, and Kaushik Sinha. “A near-optimal algorithm for differentially-private principal components”. In: *The Journal of Machine Learning Research* 14.1 (2013), pp. 2905–2943.

[22] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. “Differentially private empirical risk minimization”. In: *Journal of Machine Learning Research* 12.Mar (2011), pp. 1069–1109.

[23] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. “Rappor: Randomized aggregatable privacy-preserving ordinal response”. In: *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. ACM. 2014, pp. 1054–1067.

[24] Bennett Cyphers and Kalyan Veeramachaneni. “AnonML: Locally private machine learning over a network of peers”. In: ().

[25] Xiaoqian Jiang et al. “Differential-private data publishing through component analysis”. In: *Transactions on data privacy* 6.1 (2013), p. 19.

[26] Sun-Yuan Kung. “Compressive privacy: From information\/estimation theory to machine learning [lecture notes]”. In: *IEEE Signal Processing Magazine* 34.1 (2017), pp. 94–112.

[27] Richard Mortier et al. “Human-data interaction: the human face of the data-driven society”. In: (2014).

[28] Yinzhi Cao and Junfeng Yang. “Towards making systems forget with machine unlearning”. In: *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE. 2015, pp. 463–480.

[29] *The U.S. Mobile App Report*. White Paper. Accessed 17-August-2015. comScore, 2014. URL: http://www.comscore.com/Insights/Presentations-and-Whitepapers/2014/The-US-Mobile-App-Report.

[30] *Update to Celebrity Photo Investigation*. Apple Media Advisory. Accessed 17-August-2015. Apple, 2014. URL: http://www.apple.com/pr/library/2014/09/02Apple-Media-Advisory.html.

[31] Michael Rose. *Think iCloud's two-factor authentication protects your privacy? It doesn't*. Report. Accessed 17-August-2015. Engadget, 2014. URL: http://www.engadget.com/2014/09/02/think-iclouds-two-factor-authentication-protects-your-privacy/.

[32] Abdul Serwadda, Vir V Phoha, and Zhen Wang. “Which verifiers work?: A benchmark evaluation of touch-based authentication algorithms”. In: *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*. IEEE. 2013, pp. 1–8.

[33] Margit Antal, Zsolt Bokor, and László Zsolt Szabó. “Information revealed from scrolling interactions on mobile devices”. In: *Pattern Recognition Letters* 56 (2015), pp. 7–13.

[34] Lingjun Li, Xinxin Zhao, and Guoliang Xue. “Unobservable Re-authentication for Smartphones.” In: *NDSS*. 2013.

[35] Mario Frank et al. “Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication”. In: *IEEE transactions on information forensics and security* 8.1 (2013), pp. 136–148.

[36] Tao Feng et al. “Continuous mobile authentication using virtual key typing biometrics”. In: *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on*. IEEE. 2013, pp. 1547–1552.

[37] Jiang Zhu et al. “Sensec: Mobile security through passive sensing”. In: *Computing, Networking and Communications (ICNC), 2013 International Conference on*. IEEE. 2013, pp. 1128–1133.

[38] Hui Xu, Yangfan Zhou, and Michael R Lyu. “Towards continuous and passive authentication via touch biometrics: An experimental study on smartphones”. In: *Symposium On Usable Privacy and Security, SOUPS*. Vol. 14. 2014, pp. 187–198.

[39] Jaroslav Sedenka et al. “Secure Outsourced Biometric Authentication With Performance Evaluation on Smartphones”. In: *Information Forensics and Security, IEEE Transactions on* 10.2 (2015), pp. 384–396.

[40] Neil Zhenqiang Gong et al. "Forgery-Resistant Touch-based Authentication on Mobile Devices". In: *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. ACM. 2016, pp. 499–510.

[41] Sheng Li and Alex C Kot. "An improved scheme for full fingerprint reconstruction". In: *Information Forensics and Security, IEEE Transactions on* 7.6 (2012), pp. 1906–1912.

[42] Arun Ross, Jidnya Shah, and Anil K Jain. "From template to image: Reconstructing fingerprints from minutiae points". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 29.4 (2007), pp. 544–560.

[43] Raffaele Cappelli et al. "Fingerprint image reconstruction from standard templates". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 29.9 (2007), pp. 1489–1503.

[44] Xi Zhao, Tao Feng, and Weidong Shi. "Continuous mobile authentication using a novel graphic touch gesture feature". In: *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*. IEEE. 2013, pp. 1–6.

[45] Xi Zhao et al. "Mobile user authentication using statistical touch dynamics images". In: *Information Forensics and Security, IEEE Transactions on* 9.11 (2014), pp. 1780–1789.

[46] Anirban Roy, Tzipora Halevi, and Nasir Memon. "An HMM-based behavior modeling approach for continuous mobile authentication". In: *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE. 2014, pp. 3789–3793.

[47] Michael Velten et al. "User Identity Verification Based on Touchscreen Interaction Analysis in Web Contexts". In: *Information Security Practice and Experience*. Springer, 2015, pp. 268–282.

[48] Yuxin Meng, Duncan S Wong, Roman Schlegel, et al. "Touch gestures based biometric authentication scheme for touchscreen mobile phones". In: *Information Security and Cryptology*. Springer. 2013, pp. 331–350.

[49] Heng Zhang et al. "Touch Gesture-Based Active User Authentication Using Dictionaries". In: *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*. IEEE. 2015, pp. 207–214.

[50] Hassan Khan and Urs Hengartner. "Towards application-centric implicit authentication on smartphones". In: *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*. ACM. 2014, p. 10.

[51] Tao Feng et al. "Continuous mobile authentication using touchscreen gestures". In: *Homeland Security (HST), 2012 IEEE Conference on Technologies for*. IEEE. 2012, pp. 451–456.

[52] Premkumar Saravanan et al. "LatentGesture: active user authentication through background touch analysis". In: *Proceedings of the Second International Symposium of Chinese CHI*. ACM. 2014, pp. 110–113.

[53] Hojin Seo, Eunjin Kim, and Huy Kang Kim. "A novel biometric identification based on a users input pattern analysis for intelligent mobile devices". In: *International Journal of Advanced Robotic Systems* (2012).

[54] Kim Zetter. *Sony Got Hacked Hard: What We Know and Dont Know So Far*. Article. Accessed 17-August-2015. Wired, 2014. URL: http://www.wired.com/2014/12/sony-hack-what-we-know/.

[55] Joseph Bonneau et al. "Privacy concerns of implicit secondary factors for web authentication". In: *SOUPS Workshop on Who are you*. 2014.

[56] Abdul Serwadda and Vir 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.

[57] Margit Antal, LÁSZLÓ ZSOLT SZABÓ, and Zsolt Bokor. "IDENTITY INFORMATION REVEALED FROM MOBILE TOUCH GESTURES". In: *Studia Universitatis Babes-Bolyai, Informatica* 59 (2014).

[58] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*. Vol. 1. Wiley New York, 1998.

[59] Koichiro Niinuma and Anil K Jain. "Continuous user authentication using temporal information". In: *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics. 2010, pp. 76670L–76670L.

[60] Kasper Bonne Rasmussen et al. "Authentication Using Pulse-Response Biometrics." In: *NDSS*. 2014.

[61] Nik Cubrilovic. *RockYou Hack: From Bad To Worse*. Report. Accessed 30-April-2016. 2009. URL: http://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook-passwords/.

[62] Mohit Kumar. *China Software Developer Network (CSDN) 5 million user data leaked*. Report. Accessed 30-April-2016. 2011. URL: http://thehackernews.com/2011/12/china-software-developer-network-csdn-6.html.

[63] Nalini K Ratha, Jonathan H Connell, and Ruud M Bolle. "An analysis of minutiae matching strength". In: *Audio-and Video-Based Biometric Person Authentication*. Springer. 2001, pp. 223–228.

[64] Lorenzo Gomez et al. "Reran: Timing-and touch-sensitive record and replay for android". In: *Software Engineering (ICSE), 2013 35th International Conference on*. IEEE. 2013, pp. 72–81.

[65] *Decompile Android Apps*. Apple Media Advisory. Accessed 30-April-2016. 2016. URL: http://www.decompileandroid.com/.

[66] C.B. Moler. *Numerical Computing with MATLAB*. Society for Industrial and Applied Mathematics, 2004. ISBN: 9780898715606. URL: https://books.google.com/books?id=-vPtcriflH0C.

[67] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: A library for support vector machines". In: *ACM Transactions on Intelligent Systems and Technology* 2 (3 2011). Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm, 27:1–27:27.

[68] Olvi L Mangasarian and Edward W Wild. "Privacy-Preserving Classification of Horizontally Partitioned Data via Random Kernels." In: *DMIN*. 2008, pp. 473–479.

[69] Zahid Syed et al. "Effect of user posture and device size on the performance of touch-based authentication systems". In: *High Assurance Systems Engineering (HASE), 2015 IEEE 16th International Symposium on*. IEEE. 2015, pp. 10–17.

[70] Chao Shen et al. "Touch-interaction behavior for continuous user authentication on smartphones". In: *Biometrics (ICB), 2015 International Conference on*. IEEE. 2015, pp. 157–162.

[71] Xiao Wang et al. "Towards continuous and passive authentication across mobile devices: an empirical study". In: *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM. 2017, pp. 35–45.

[72] *BehavioSec announces partnership with Appdome*. Website. Accessed 5-October-2017. 2017. URL: https://www.behaviosec.com/behaviosec-today-announces-new-partnership-appdome/.

[73] Rebecca Balebako et al. "The privacy and security behaviors of smartphone app developers". In: (2014).

[74] Sathya Govindarajan, Paolo Gasti, and Kiran S Balagani. "Secure privacy-preserving protocols for outsourcing continuous authentication of smartphone users with touch data". In: *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*. IEEE. 2013, pp. 1–8.

[75] Vishal M Patel, Nalini K Ratha, and Rama Chellappa. "Cancelable biometrics: A review". In: *IEEE Signal Processing Magazine* 32.5 (2015), pp. 54–65.

[76] Hansong Guo et al. "Recognizing the Operating Hand and the Hand-Changing Process for User Interface Adjustment on Smartphones". In: *Sensors* 16.8 (2016), p. 1314.

[77] Vishal M Patel et al. "Continuous user authentication on mobile devices: Recent progress and remaining challenges". In: *IEEE Signal Processing Magazine* 33.4 (2016), pp. 49–61.

[78] Somayeh Taheri, Md Morshedul Islam, and Reihaneh Safavi-Naini. "Privacy-Enhanced Profile-Based Authentication Using Sparse Random Projection". In: *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer. 2017, pp. 474–490.

[79] Ankita Jain and Vivek Kanhangad. "Exploring orientation and accelerometer sensor data for personal authentication in smartphones using touchscreen gestures". In: *Pattern Recognition Letters* 68 (2015), pp. 351–360.

[80] Chao Shen et al. "Performance analysis of touch-interaction behavior for active smartphone authentication". In: *IEEE Transactions on Information Forensics and Security* 11.3 (2016), pp. 498–513.

[81] Zaire Ali, Jamie Payton, and Vincent Sritapan. "At Your Fingertips: Considering Finger Distinctness in Continuous Touch-Based Authentication for Mobile Devices". In: *Security and Privacy Workshops (SPW), 2016 IEEE*. IEEE. 2016, pp. 272–275.

[82] Soumik Mondal and Patrick Bours. "Swipe gesture based continuous authentication for mobile devices". In: *Biometrics (ICB), 2015 International Conference on*. IEEE. 2015, pp. 458–465.

[83] Li Lu and Yongshuai Liu. "Safeguard: User reauthentication on smartphones via behavioral biometrics". In: *IEEE Transactions on Computational Social Systems* 2.3 (2015), pp. 53–64.

[84] Sun Yuan Kung. *Kernel methods and machine learning*. Cambridge University Press, 2014.

[85] Konstantinos Diamantaras and Sun-Yuan Kung. "Data privacy protection by kernel subspace projection and generalized eigenvalue decomposition". In: *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*. IEEE. 2016, pp. 1–6.

[86] Abena Primo and Vir V Phoha. "Music and images as contexts in a context-aware touch-based authentication system". In: *Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE 7th International Conference on*. IEEE. 2015, pp. 1–7.

[87] Cheng Bo et al. "Silentsense: silent user identification via touch and movement behavioral biometrics". In: *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM. 2013, pp. 187–190.

[88] Tao Feng et al. "An investigation on touch biometrics: Behavioral factors on screen size, physical context and application context". In: *Technologies for Homeland Security (HST), 2015 IEEE International Symposium on*. IEEE. 2015, pp. 1–6.

[89] Vaibhav Sharma and Richard Enbody. "User authentication and identification from user interface interactions on touch-enabled devices". In: *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM. 2017, pp. 1–11.

[90] Jaishanker K Pillai et al. "Secure and robust iris recognition using random projections and sparse representations". In: *IEEE transactions on pattern analysis and machine intelligence* 33.9 (2011), pp. 1877–1893.

[91] Andrew BJ Teoh, Yip Wai Kuan, and Sangyoun Lee. "Cancellable biometrics and annotations on biohash". In: *Pattern recognition* 41.6 (2008), pp. 2034–2044.

[92] Jaroslav Šeděnka et al. “Secure outsourced biometric authentication with performance evaluation on smartphones”. In: *IEEE Transactions on Information Forensics and Security* 10.2 (2015), pp. 384–396.

[93] Tapalina Bhattasali et al. “A survey of security and privacy issues for biometrics based remote authentication in cloud”. In: *IFIP International Conference on Computer Information Systems and Industrial Management*. Springer. 2014, pp. 112–121.

[94] Anil K Jain, Arun Ross, and Sharath Pankanti. “Biometrics: a tool for information security”. In: *Information Forensics and Security, IEEE Transactions on* 1.2 (2006), pp. 125–143.

[95] Gordon V Cormack. “Email spam filtering: A systematic review”. In: *Foundations and Trends in Information Retrieval* 1.4 (2007), pp. 335–455.

[96] Linda Delamaire, HAH Abdou, and John Pointon. “Credit card fraud and detection techniques: a review”. In: *Banks and Bank systems* 4.2 (2009), pp. 57–68.

[97] Mark A Musen, Blackford Middleton, and Robert A Greenes. “Clinical decision-support systems”. In: *Biomedical informatics*. Springer, 2014, pp. 643–674.

[98] Sun-Yuan Kung. “Discriminant component analysis for privacy protection and visualization of big data”. In: *Multimedia Tools and Applications* (2015), pp. 1–36.

[99] Yongming Qu et al. “Principal component analysis for dimension reduction in massive distributed data sets”. In: *Proceedings of IEEE International Conference on Data Mining (ICDM)*. 2002.

[100] Zheng-Jian Bai, Raymond H Chan, and Franklin T Luk. “Principal component analysis for distributed data sets with updating”. In: *International Workshop on Advanced Parallel Processing Technologies*. Springer. 2005, pp. 471–483.

[101] Manas A Pathak and Bhiksha Raj. “Efficient Protocols for Principal Eigenvector Computation over Private Data.” In: *Transactions on Data Privacy* 4.3 (2011), pp. 129–146.

[102] Shuguo Han and Wee Keong Ng. “Privacy-preserving linear fisher discriminant analysis”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2008, pp. 136–147.

[103] Jaroslav Šeděnka et al. “Privacy-preserving population-enhanced biometric key generation from free-text keystroke dynamics”. In: *Biometrics (IJCB), 2014 IEEE International Joint Conference on*. IEEE. 2014, pp. 1–8.

[104] Lu Tian et al. “Aggregating Private Sparse Learning Models Using Multi-Party Computation”. In: *NIPS Workshop on Private Multi-Party Machine Learning, Barcelona, Spain*. 2016.

[105] Mohammad Al-Rubaie et al. “Privacy-preserving PCA on horizontally-partitioned data”. In: *Dependable and Secure Computing, 2017 IEEE Conference on*. IEEE. 2017, pp. 280–287.

[106] Hee-Sun Won et al. “Secure principal component analysis in multiple distributed nodes”. In: *Security and Communication Networks* 9.14 (2016), pp. 2348–2358.

[107] Azalia Mirhoseini, Ahmad-Reza Sadeghi, and Farinaz Koushanfar. “CryptoML: Secure outsourcing of big data machine learning applications”. In: *Hardware Oriented Security and Trust (HOST), 2016 IEEE International Symposium on*. IEEE. 2016, pp. 149–154.

[108] Bernardo David et al. “Unconditionally Secure, Universally Composable Privacy Preserving Linear Algebra”. In: *IEEE Transactions on Information Forensics and Security* 11.1 (2016), pp. 59–73.

[109] Shuguo Han, Wee Keong Ng, and S Yu Philip. “Privacy-preserving singular value decomposition”. In: *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*. IEEE. 2009, pp. 1267–1270.

[110] Shuo Chen, Rongxing Lu, and Jie Zhang. “A Flexible Privacy-Preserving Framework for Singular Value Decomposition Under Internet of Things Environment”. In: *IFIP International Conference on Trust Management*. Springer. 2017, pp. 21–37.

[111] Wuxuan Jiang, Cong Xie, and Zhihua Zhang. “Wishart Mechanism for Differentially Private Principal Components Analysis.” In: *AAAI*. 2016, pp. 1730–1736.

[112] Hafiz Imtiaz and Anand D Sarwate. “Symmetric matrix perturbation for differentially-private principal component analysis”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE. 2016, pp. 2339–2343.

[113] Alon Gonen and Gilad Ran-Bachrach. “Smooth Sensitivity Based Approach for Differentially Private Principal Component Analysis”. In: *arXiv preprint arXiv:1710.10556* (2017).

[114] Kamalika Chaudhuri, Anand Sarwate, and Kaushik Sinha. “Near-optimal differentially private principal components”. In: *Advances in Neural Information Processing Systems*. 2012, pp. 989–997.

[115] Somnath Chakrabarti et al. “Privacy preserving linear discriminant analysis from perturbed data”. In: *Proceedings of the 2010 ACM Symposium on Applied Computing*. ACM. 2010, pp. 610–615.

[116] Justin Zhan, L Chang, and Stan Matwin. “Privacy-preserving support vector machines learning”. In: *Proceedings of the 2005 International Conference on Electronic Business (ICEB05)*. 2005.

[117] Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. “Cryptographically private support vector machines”. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2006, pp. 618–624.

[118] Hwanjo Yu, Xiaoqian Jiang, and Jaideep Vaidya. “Privacy-preserving SVM using nonlinear kernels on horizontally partitioned data”. In: *Proceedings of the 2006 ACM symposium on Applied computing*. ACM. 2006, pp. 603–610.

[119] Pascal Paillier. "Public-key cryptosystems based on composite degree residuosity classes". In: *Advances in cryptologyEUROCRYPT99*. Springer. 1999, pp. 223–238.

[120] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[121] Samee Zahur and David Evans. "Obliv-C: A Language for Extensible Data-Oblivious Computation." In: *IACR Cryptology ePrint Archive* 2015 (2015), p. 1153.

[122] THEP. *The Homomorphic Encryption Project*. Java Library. Accessed 3-April-2016. 2016. URL: https://github.com/diegode/thep.

[123] M. Lichman. *UCI Machine Learning Repository*. 2013. URL: http://archive.ics.uci.edu/ml.

[124] Tianhao Wang et al. "Locally differentially private protocols for frequency estimation". In: *Proc. of the 26th USENIX Security Symposium*. 2017, pp. 729–745.