Spring 2013

Analysis of Kullback-Leibler Divergence for Masquerade Detection

Geetha Ranjini Viswanathan
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Recommended Citation

Viswanathan, Geetha Ranjini, "Analysis of Kullback-Leibler Divergence for Masquerade Detection" (2013).
Master's Projects. 302.
DOI: <https://doi.org/10.31979/etd.5c9n-a6mc>
https://scholarworks.sjsu.edu/etd_projects/302

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Analysis of Kullback-Leibler Divergence for Masquerade Detection

A Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Geetha Ranjini Viswanathan

May 2013

© 2013

Geetha Ranjini Viswanathan

ALL RIGHTS RESERVED

The Designated Project Committee Approves the Project Titled

Analysis of Kullback-Leibler Divergence for Masquerade Detection

by

Geetha Ranjini Viswanathan

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2013

Dr. Mark Stamp Department of Computer Science

Dr. Chris Pollett Department of Computer Science

Dr. Richard Low Department of Mathematics

ABSTRACT

Analysis of Kullback-Leibler Divergence for Masquerade Detection

by Geetha Ranjini Viswanathan

A masquerader is an attacker who gains access to a legitimate user's credentials and pretends to be that user so as to avoid detection. Several statistical techniques have been applied to the masquerade detection problem, including hidden Markov models (HMM) and one class naïve Bayes (OCNB). In addition, Kullback-Leibler (KL) divergence has been used in an effort to improve detection rates. In this project, we develop and analyze masquerade detection techniques that employ KL divergence, HMMs, and OCNB. Detailed statistical analysis is provided to show that our results outperform previous related research.

ACKNOWLEDGMENTS

I thank Dr. Mark Stamp for his guidance and encouragement in implementing this project. I am much obliged for all his support and patience throughout my master's program.

I thank Dr. Chris Pollett and Dr. Richard Low for serving on my defense committee.

And, I would also like to express my gratitude towards my guru Sai, my family and friends for their love and encouragement.

TABLE OF CONTENTS

CHAPTER

1	Introduction	1
2	Background	3
2.1	Intrusion detection	3
2.1.1	Signature-based intrusion detection	3
2.1.2	Anomaly-based intrusion detection	4
2.2	Schonlau dataset	4
2.3	Performance measures	6
3	Proposed techniques	8
3.1	Hidden Markov model	8
3.1.1	Implementation	10
3.2	One class naïve Bayes	10
3.2.1	Implementation	11
3.3	Kullback-Leibler divergence	12
3.3.1	Probabilistic padding identification	13
3.3.2	Computing scores	17
3.4	Mimicry attack	17
3.4.1	Attack generation	19
4	Experimental results	20
4.1	Analysis of PPI results	21
4.2	Analysis of HMM results	24

4.2.1	Without PPI	24
4.2.2	With padding-first PPI $D1$	25
4.2.3	With attack-first PPI	27
4.2.4	ROC comparison	28
4.2.5	AUC comparison	30
4.3	Analysis of OCNB results	31
4.3.1	Without PPI	31
4.3.2	With padding-first PPI $D1$	31
4.3.3	With attack-first PPI	34
4.3.4	ROC comparison	34
4.3.5	AUC comparison	36
4.4	Performance comparison between HMM and OCNB	38
4.5	Performance comparison using 1 vs 49 tests	38
5	Conclusions and future work	40

APPENDIX

A	Analysis of HMM vs OCNB performance	45
B	Additional results for distance $D1$	46
C	Results for distance $D2$	59

LIST OF TABLES

1	Average number of padding identified during training	22
B.1	AUC comparison for HMM results without PPI (normal face) and padding-first PPI <i>D1</i> (bold face)	51
B.2	AUC comparison for HMM results without PPI (normal face) and attack-first PPI (bold face)	52
B.3	AUC comparison for OCNB results without PPI (normal face) and padding-first PPI <i>D1</i> (bold face)	53
B.4	AUC comparison for OCNB results without PPI (normal face) and attack-first PPI (bold face)	54
B.5	AUCp (FPR 5%) comparison for HMM results without PPI (normal face) and padding-first PPI <i>D1</i> (bold face)	55
B.6	AUCp (FPR 5%) comparison for HMM results without PPI (normal face) and attack-first PPI (bold face)	56
B.7	AUCp (FPR 5%) comparison for OCNB results without PPI (normal face) and padding-first PPI <i>D1</i> (bold face)	57
B.8	AUCp (FPR 5%) comparison for OCNB results without PPI (normal face) and attack-first PPI (bold face)	58
C.1	<i>D2</i> : Average number of padding identified during training	59
C.2	AUC comparison for HMM results without PPI (normal face) and padding-first PPI <i>D2</i> (bold face)	70
C.3	AUC comparison for OCNB results without PPI (normal face) and padding-first PPI <i>D2</i> (bold face)	71
C.4	AUCp (FPR 5%) comparison for HMM results without PPI (normal face) and padding-first PPI <i>D2</i> (bold face)	72
C.5	AUCp (FPR 5%) comparison for OCNB results without PPI (normal face) and padding-first PPI <i>D2</i> (bold face)	73

LIST OF FIGURES

1	Schonlau dataset representation [5]	5
2	Map file for Schonlau dataset [5]	5
3	Confusion matrix [3]	6
4	Example of ROC curves	7
5	Hidden Markov process [21]	9
6	Example of mimicry attack [23]	18
7	Comparison of average accuracy of padding identified	21
8	Average number of attacks and padding identified - intermediate . . .	22
9	Average number of attacks and padding identified - hard	23
10	Average number of attacks and padding identified - easy	23
11	HMM: scores for attack length $ A = 10$ without using PPI	24
12	HMM: scores for attack length $ A = 30$ without using PPI	25
13	HMM: scores for attack length $ A = 10$ using padding-first PPI D1 .	26
14	HMM: scores for attack length $ A = 30$ using padding-first PPI D1 .	26
15	HMM: scores for attack length $ A = 10$ using attack-first PPI	27
16	HMM: scores for attack length $ A = 30$ using attack-first PPI	28
17	HMM: ROC comparison for $ A = 10$	29
18	HMM: ROC comparison for $ A = 30$	29
19	HMM: AUC comparison	30
20	HMM: AUCp comparison (FPR 5%)	31
21	OCNB: scores for attack length $ A = 10$ without using PPI	32

22	OCNB: scores for attack length $ A = 30$ without using PPI	32
23	OCNB: scores for attack length $ A = 10$ using padding-first PPI D1 .	33
24	OCNB: scores for attack length $ A = 30$ using padding-first PPI D1 .	33
25	OCNB: scores for attack length $ A = 10$ using attack-first PPI	34
26	OCNB: scores for attack length $ A = 30$ using attack-first PPI	35
27	OCNB: ROC comparison for $ A = 10$	35
28	OCNB: ROC comparison for $ A = 30$	36
29	OCNB: AUC comparison	37
30	OCNB: AUCp comparison (FPR 5%)	37
31	HMM vs OCNB: without PPI – ROC comparison	38
32	HMM vs OCNB: padding-first PPI D1 – ROC comparison	39
33	HMM vs OCNB: attack-first PPI – ROC comparison	39
A.1	AUCs for 1 vs 49 tests using HMM and OCNB	45
B.1	HMM: scores for attack length $ A = 50$ without using PPI	46
B.2	HMM: scores for attack length $ A = 50$ using padding-first PPI D1 .	47
B.3	HMM: scores for attack length $ A = 50$ using attack-first PPI	47
B.4	HMM: ROC comparison for $ A = 50$	48
B.5	OCNB: scores for attack length $ A = 50$ without using PPI	48
B.6	OCNB: scores for attack length $ A = 50$ using padding-first PPI D1 .	49
B.7	OCNB: scores for attack length $ A = 50$ using attack-first PPI	49
B.8	OCNB: ROC comparison for $ A = 50$	50
C.1	D2: Comparison of average accuracy of padding identified	59
C.2	D2: Average number of attacks and padding identified - intermediate	60

C.3	<i>D2</i> : Average number of attacks and padding identified - hard	60
C.4	<i>D2</i> : Average number of attacks and padding identified - easy	61
C.5	<i>D2</i> : HMM: scores for attack length $ A = 10$ using padding-first PPI .	61
C.6	<i>D2</i> : HMM: scores for attack length $ A = 30$ using padding-first PPI .	62
C.7	<i>D2</i> : HMM: scores for attack length $ A = 50$ using padding-first PPI .	62
C.8	<i>D2</i> : HMM: ROC comparison for $ A = 10$	63
C.9	<i>D2</i> : HMM: ROC comparison for $ A = 30$	63
C.10	<i>D2</i> : HMM: ROC comparison for $ A = 50$	64
C.11	<i>D2</i> : HMM: AUC comparison	64
C.12	<i>D2</i> : HMM: AUCp comparison (FPR 5%)	65
C.13	<i>D2</i> : OCNB: scores for attack length $ A = 10$ using padding-first PPI	65
C.14	<i>D2</i> : OCNB: scores for attack length $ A = 30$ using padding-first PPI	66
C.15	<i>D2</i> : OCNB: scores for attack length $ A = 50$ using padding-first PPI	66
C.16	<i>D2</i> : OCNB: ROC comparison for $ A = 10$	67
C.17	<i>D2</i> : OCNB: ROC comparison for $ A = 30$	67
C.18	<i>D2</i> : OCNB: ROC comparison for $ A = 50$	68
C.19	<i>D2</i> : OCNB: AUC comparison	68
C.20	<i>D2</i> : OCNB: AUCp comparison (FPR 5%)	69
C.21	<i>D2</i> : HMM vs OCNB: padding-first PPI – ROC comparison	69

CHAPTER 1

Introduction

An intruder is an attacker who gains unauthorized access to a system. As the name implies, an intrusion detection system (IDS) is designed to detect such an intruder. Broadly speaking, IDS can be one of two types: signature-based or anomaly-based [22]. A signature-based IDS depends on signatures or attack patterns; whereas, anomaly-based systems focus on anomalous or unusual activity.

The difference between a masquerader and a traitor is that a masquerader gains access to a user's credentials and carries out some malicious activity, whereas a traitor already has the privileges but misuses it for malicious purposes [23]. A masquerader's intent is to masquerade the attacks to avoid detection. A masquerade detection system is designed to detect such masquerades.

The research presented in this project represents an anomaly-based approach to masquerade detection. More specifically, we analyze UNIX commands for anomalous behavior. There is a truly vast amount of prior research on this particular problem; representative examples include [5, 7, 8, 9, 11, 16, 18, 23, 25, 26, 27, 28].

In this project, we have employed hidden Markov models (HMM). HMMs can be viewed as a machine learning technique [28]. In this project, we construct an HMM for each legitimate user's UNIX commands. These models are then used to determine whether a given set of commands is likely from the original user or not. Prior research has shown that HMMs are indeed an effective technique for masquerade detection based on UNIX commands [9].

In this project, we also use of one class naïve Bayes (OCNB). In OCNB, we use

elementary statistical analysis of the legitimate user’s input data as a means to detect masqueraders [9, 23].

Kullback-Leibler (KL) divergence is a statistical method that can be used to separate data from different distributions [10]. This technique has previously been studied in the context of masquerade detection as a means for separating masquerade data (i.e., data intended to mimic the legitimate user) from attack data [23]. If properly separated, we can obtain stronger scores.

In this project, we implement HMM-based and OCNB-based masquerade detection techniques, and we rigorously analyze the effectiveness of KL divergence on detection rates for these two approaches. We show that the results from our research are a significant improvement over the results yielded by prior research.

Chapter 2 introduces different types of intrusion detection, the dataset and performance measures used in this project. Chapter 3 discusses the implementation details for HMM, OCNB, KL divergence and attack generation. Chapter 4 shows the various test cases and their results. Chapter 5 concludes about the effectiveness of KL divergence in masquerade detection and also discusses some future work on the same topic.

CHAPTER 2

Background

2.1 Intrusion detection

An intrusion is said to occur when an attacker gains access to the credentials of the original user and misuses the same to carry out any malicious activity. Intrusion detection is a technique used to identify whether the current activity on the system is from the original user or from an attacker. The rapid increase in the types of attacks that are being introduced everyday, has increased the interest and need for more research in the field of intrusion detection.

To detect intrusion, we make use of intrusion detection system (IDS). IDSs that make use of hidden Markov models [5, 7, 9, 28], one class naïve Bayes [9, 18, 23, 27], support vector machines [8] etc have been analyzed extensively in prior researches.

Intrusion detection can be classified into two categories [22]:

1. Signature-based
2. Anomaly-based

2.1.1 Signature-based intrusion detection

A signature is a byte sequence that is extracted from a file [22]. Signature-based intrusion detection occurs when some known malware or virus is suspected to have infected the computer. For each malware or virus, a signature is extracted and stored in the signature database. To detect whether the software is malicious, the intrusion system compares the signature extracted from the malware with those already present in the signature database.

Signature-based intrusion detection is very simple and efficient. It has minimal false positive rate as long as the malware is already known [22]. But if an unknown malware infects the computer, then signature-based IDS will not be able to detect it unless the signature database is updated with the signature of the new malware [22].

2.1.2 Anomaly-based intrusion detection

An anomaly-based IDS depends on the behavior of the user and the assumption that there will be behavioral difference between attacker and legitimate user [13]. An anomaly-based IDS operates in two phases: training and detection [6]. During the training phase, IDS generates a trained model for the original user's profile. During the detection phase, if an attacker gains access to the computer and pretends to be the original user, then the sequence of activities that the attacker carries out on the system will be compared with the original user's trained model to detect if the person using the computer is the original user or an attacker.

Unlike signature-based systems, anomaly-based system can detect previously unknown attacks or malwares, but the false positive rate can be high and the problem of determining what activities comprise the training model is complex [6].

2.2 Schonlau dataset

The Schonlau dataset [17] consists of truncated, user-issued, UNIX commands collected from 50 users. The data files contain 15000 UNIX commands for each user, and a set of 100 commands represents a block. The first 50 blocks, comprising of 5000 commands, is the training set, and the remaining 100 blocks comprising of 10000 commands is the test set. Figure 1 shows a diagrammatic representation of the Schonlau dataset.

Figure 1. Schonlau dataset representation [5]

The dataset also includes a map file which contains a $100 * 50$ matrix. The 100 rows correspond to the 100 testing blocks and 50 columns correspond to the 50 users. This matrix is filled with 0s and 1s. 0 represents that the corresponding user test block is attack-free and 1 represents that the block contains attacks. Figure 2 depicts the data contained in the map file.

Figure 2. Map file for Schonlau dataset [5]

The blocks that belong to the original user and are attack-free are called *self blocks*. Blocks that contain attacks are called *non-self blocks* [23].

2.3 Performance measures

Receiver operating characteristic (ROC) curves provide a visual representation for measuring the effectiveness of the detection. To identify if the data is an attack or not, a threshold is set and the score for the data is classified as an attack depending on whether it falls above or below the threshold [3].

The detection results can be classified in four ways. If the data is an attack and it is identified as an attack by the IDS, then it is a true positive. If the data is not an attack but it is identified as an attack, then it is a false positive. If an attack data is identified as legitimate data, it is a false negative. Finally, if a legitimate data is identified as legitimate, it is a true negative. These four terms are represented as a confusion matrix in Figure 3.

		<u>True class</u>	
		p	n
<u>Hypothesized class</u>	Y	True Positives	False Positives
	N	False Negatives	True Negatives
Column totals:		P	N

Figure 3. Confusion matrix [3]

Performance measures such as true positive rate (TPR), false positive rate (FPR) and accuracy can be used to determine the effectiveness of any detection system. These are computed as

$$\begin{aligned}\text{True Positive Rate} &= \frac{TP}{P} \\ \text{False Positive Rate} &= \frac{FP}{N} \\ \text{Accuracy} &= \frac{TP + TN}{P + N}\end{aligned}$$

The ROC graph plots FPR on the x-axis and TPR on the y-axis. The values plotted for TRP and FRP by varying the threshold generates the ROC curve [3]. We can then compute the area under the ROC curve (AUC) by using trapezoidal integration. This generates a single number which can be used to measure the performance of the IDS [2]. Figure 4 shows examples of ROC curves for best, intermediate and worst performance. The dashed gray line indicates the limit for FPR of 5%, i.e., 0.05. The partial AUC (AUCp) is computed for the area under the curve within the FPR 0.05. The AUCp thus computed is then normalized to rescale it to 100%.

Figure 4. Example of ROC curves

CHAPTER 3

Proposed techniques

3.1 Hidden Markov model

Hidden Markov model (HMM) is a machine-learning technique which makes use of Markov chains to build models based on the given input data. Extensive research has been done using HMMs for virus detection and it has proved to be very efficient in identifying malwares [15, 20].

HMM is one of the most popular and efficient techniques for building an anomaly-based intrusion detection system. Previous research using HMM for identifying masquerades on Schonlau dataset has shown that attacks of length 30 or more were detected efficiently [9].

The main objective of using HMM in this project is to train a model for each user using the commands available in the training set. Each of these models represents the profile of the original user. Using the trained models we compute the scores for the attack data to determine how close it is to the model.

HMM makes use of three different types of matrices to build a model for the given data:

1. State transition matrix A .
2. Observation matrix B .
3. Initial state distribution π .

All three matrices contain probabilities. And all three matrices need to be row stochastic, i.e., the sum of all elements in each row should be 1. A HMM is represented

as $\lambda = (A, B, \pi)$. The notations [21] used for building a Markov model are given below.

T = length of the observation sequence

N = number of states in the Markov model

M = number of unique observation symbols

$Q = \{q_0, q_1, \dots, q_{N-1}\}$ = the unique states of the Markov process

$V = \{0, 1, \dots, M - 1\}$ = the set of possible observations

$\mathcal{O} = (\mathcal{O}_0, \mathcal{O}_1, \dots, \mathcal{O}_{T-1})$ = observation sequence.

Figure 5 is an example of how a HMM is represented. $X_0, X_1, X_2, \dots, X_{T-1}$ represents the hidden state sequence. The dashed line distinguishes the hidden process from the rest.

Figure 5. Hidden Markov process [21]

For each hidden state there are two probabilities associated with it. First is the probability of transition to another state and the other is the probability of being in the same state. The hidden Markov process is obtained from the current state and state transition matrix A . We can see the observations $\mathcal{O}_i$ which are obtained from the hidden states through the observation matrix B [21].

3.1.1 Implementation

There are three problems that can be solved using HMMs. A brief description of each is given below.

1. Given a set of observations $\mathcal{O}$ and a model $\lambda = (A, B, \pi)$, we need to find $P(\mathcal{O}|\lambda)$ to find out how similar the observations are to the model [21].
2. Given a set of observations $\mathcal{O}$ and a model $\lambda = (A, B, \pi)$, we need to find the hidden state sequence for the Markov process [21].
3. Given a set of observations $\mathcal{O}$, number of hidden states N and number of observation symbols M , we need to find the model $\lambda = (A, B, \pi)$ that increases the observations probability to maximum [21].

For developing a masquerade detection system (MDS), first we need to solve problem 3 and develop a model for the observations in the training data. Then we will solve problem 1 where we will compute the probability $P(\mathcal{O}|\lambda)$ for the observations in the test data to see how close they are to the trained model.

Higher probability scores (i.e., closer to zero) indicate that the observations in the test data are more similar to the training observations. Lower probability scores indicate that the test observations contain attack data.

3.2 One class naïve Bayes

Naïve Bayes classifier is a effective yet simple learning algorithm. Prior research has proved that this method is effective in classification of text and other applications [27]. One class naïve Bayes (OCNB) is a masquerade detection algorithm which is derived from the Bayes rule. The aim of this detection technique is to estimate

the probability that the instance x belongs to class y and to find the class with the highest $P(y|x)$ [23] using

$$P(y|x) = \frac{P(y)}{P(x)} P(y|x) = \frac{P(y)}{P(x)} \prod_{i=1}^m P(x_i|y)$$

where $x = (x_1, x_2, \dots, x_m)$.

In this project, the OCNB algorithm compares the probability of a command occurring in the test data, to the probability of the same command occurring in the training data to classify whether there has been an attack or not. This simple statistical computation has been proved to be efficient in detecting attacks of length 50 or more [9].

3.2.1 Implementation

For the Schonlau dataset, every block B of 100 commands is represented by a vector $[n_1(B), n_2(B), \dots, n_m(B)]$. $n_i(B)$ is the number of times the command c_i appears in the block B and m is the number of distinct commands in the block B [23]. The probability that the block B belongs to user y is computed as

$$P(y|B) = P(y) \prod_{i=1}^m P(c_i|y)^{n_i(B)} \quad (1)$$

where the priors $P(y)$ is ignored because the probability is computed only for one specific user. $P(c_i|y)$ [23] is derived from the training set for user y as

$$P(c_i|y) = \frac{\sum_{B \in T(y)} n_i(B) + \alpha}{|B| \cdot |T(y)| + \alpha \cdot m} \quad (2)$$

where $T(y)$ is the training set for user y which consists of the self data and α is the smoothing parameter. To make sure that all commands appear with a non-zero probability we make use of the smoothing parameter α . The value of α is set to 0.01 in this project.

In equation (1), the value of $n_i(B)$ corresponds to the number of times the command c_i appears in the block B that is to be tested against the training set and m is the number of distinct commands in the test block. Whereas in equation (2), the value of $n_i(B)$ corresponds to the number of times the command c_i appears in the training set and m is the number of distinct commands in the training set.

The result is computed as an anomaly score [23] using

$$\text{score}(B) = -\log P(y|B) = -\sum_{i=1}^m n_i(B) \log P(c_i|y)$$

If the score thus computed is high then the block is considered to be anomalous and contains attacks. But if the score is low (i.e., closer to zero) then it is considered to be a self block.

3.3 Kullback-Leibler divergence

For any two statistical populations, computing the measure of divergence helps to identify whether they differ from each other. This method is called as Kullback-Leibler (KL) divergence [10].

The difference between two probability distributions can be measured using KL divergence. For two discrete distributions P and Q , the KL divergence of Q from P [23] is given by

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \quad (3)$$

Equation (3) can be rewritten as

$$D_{KL}(P||Q) = -\sum_i P(i) \log Q(i) + \sum_i P(i) \log P(i)$$

KL divergence is used extensively for identifying anomalies in wireless signals [1], biomedical data [14], network traffic [4, 12] etc.

3.3.1 Probabilistic padding identification

The probabilistic padding identification (PPI) algorithm [23] makes use of the KL divergence to identify the padding commands from the given block of test commands. A and P represent the attack and padding portions for any given block B and for a given user profile, M represents the trained model. The PPI algorithm is based on two observations [23]:

1. There is an acceptable level of difference between A and M .
2. There is minimal difference between P and M .

In this project, we implemented two versions of PPI algorithm: attack-first and padding-first. The attack-first PPI algorithm attempts to find subsets $\hat{P}, \hat{A} \subseteq B$, with $\hat{P} \cup \hat{A} = B$ and $\hat{P} \cap \hat{A} = \emptyset$, such that $D_{KL}(\hat{P}||M)$ is low (indicates that the padding will be as close as possible to the trained model) and $D_{KL}(\hat{A}||M)$ is high (since the attack will be different from the trained model) [23].

In this research, since the absolute value of the difference of the KL divergence (equation (4)) results provide a more accurate estimate of whether the current command is a padding or not, these two conditions did not have any significance. This is because, $D_{KL}(\hat{P}||M)$ and $D_{KL}(\hat{A}||M)$ are not necessarily low and high for every command during the segregation process. These values keep decreasing and increasing as the commands are being pushed into separate subsets. Another reason is that, if a command does not appear in the trained model then the KL divergence for that command becomes infinity and all the commands that follow are computed to be attacks. To avoid this, the computation of KL divergence for such commands are skipped, which reverses these two conditions. The final padding subset is expected to contain only padding commands and attack subset is expected to contain only attack

commands. Therefore, these two conditions can be verified only after the two subsets are completely formed.

Both algorithms use a boolean vector C of length 100. Each command in a block B correlates to the same location in the C vector. During the segregation process, for each command in the block, we compute the absolute value of the difference between the KL divergence for padding and attack using

$$|D_p - D_a| = \left| \sum_i \hat{P}(i) \log \frac{\hat{P}(i)}{M(i)} - \sum_i \hat{A}(i) \log \frac{\hat{A}(i)}{M(i)} \right| \quad (4)$$

where i represents the number of distinct commands in the putative padding $\hat{P}$ and attack $\hat{A}$ subsets. $M(i)$ represents the relative frequency of command i in the trained model. In this project, we have researched two variations for computing $\hat{P}(i)$ and $\hat{A}(i)$ in equation 4.

Distance $D1$:

$$\begin{aligned} \hat{P}(i) &= \frac{\text{Frequency of command } i \text{ in } \hat{P}}{\text{Total number of commands in } \hat{P}} \\ \hat{A}(i) &= \frac{\text{Frequency of command } i \text{ in } \hat{A}}{\text{Total number of commands in } \hat{A}} \end{aligned}$$

Distance $D2$:

$$\begin{aligned} \hat{P}(i) &= \frac{\text{Frequency of command } i \text{ in } \hat{P}}{\text{Total number of commands in } B} \\ \hat{A}(i) &= \frac{\text{Frequency of command } i \text{ in } \hat{A}}{\text{Total number of commands in } B} \end{aligned}$$

Once the PPI algorithm completes processing the block, the commands in the block will be separated into two sequences P and A . A command is pushed into P if its corresponding position in the C vector is true, but if it is false, then the command is pushed into A .

3.3.1.1 Attack-first PPI

In attack-first PPI, when a block is input, all the values in the C vector are initially set to false, i.e., all the commands in the block are initially considered to be attacks. For each command $B(i)$ in the block, we first compute $|D_p - D_a|$ as $\bar{d}$. Then we set the respective position in the C vector to true and again compute $|D_p - D_a|$ as d . If $d \leq \bar{d}$, $C(i)$ is set to false. The below algorithm was researched in a prior work [23].

Algorithm 1 Attack-first PPI

Input: Block B , Model M

Output: Boolean vector $C(i) = \text{true}$ if $B(i)$ is padding

```

1: Initially  $C(i) \leftarrow \text{false}$  for all  $i$ 
2: for  $i = 1$  to  $|B|$  do
3:    $\bar{d} = \text{DiffKL}(C, B, M)$ 
4:    $C(i) \leftarrow \text{true}$ 
5:    $d = \text{DiffKL}(C, B, M)$ 
6:   if  $(d \leq \bar{d})$  then
7:      $C(i) \leftarrow \text{false}$ 
8:   end if
9: end for
10: return  $P = \text{commands } B(i) \text{ such that } C(i) \text{ is true}$ 

```

Algorithm 2 DiffKL

Input: Boolean vector C , block B , model M

Output: Difference of KL divergences

```

1:  $\hat{A} \leftarrow \text{probability distribution of those } B(i) \text{ such that } C(i) \text{ is false}$ 
2:  $\hat{P} \leftarrow \text{probability distribution of those } B(i) \text{ such that } C(i) \text{ is true}$ 
3:  $D_a \leftarrow D_{KL}(\hat{A}||M)$ 
4:  $D_p \leftarrow D_{KL}(\hat{P}||M)$ 
5: return  $|D_p - D_a|$ 

```

Before computing $|D_p - D_a|$, depending on the values in C , we assign the probability distribution for the corresponding commands in the block to two subsets $\hat{P}$

and $\hat{A}$. If $C(i)$ is true, then the probability distribution of $B(i)$ is pushed into $\hat{P}$. If it is false, it is pushed into $\hat{A}$. Then the KL divergence of trained model M from $\hat{P}$ is computed as D_p and M from $\hat{A}$ is computed as D_a . Attack-first PPI was analyzed for distance $D2$.

3.3.1.2 Padding-first PPI

In padding-first PPI, when a block is input, for each command in B , which is also present in the trained model M , we set the respective position in the C vector to true, i.e., all commands in the test block which are also present in the model are initially marked as padding. We compute $|D_p - D_a|$ as d for the initial vector C . This acts like a threshold. Then for each command in B , we set the corresponding position in C as false and compute $|D_p - D_a|$ as $\bar{d}$. If $d < \bar{d}$, then $C(i)$ is set to true. The algorithm below was developed during this research. DiffKL is computed using Algorithm 2 from Section 3.3.1.1. Padding-first PPI was analyzed for both distance $D1$ and $D2$.

Algorithm 3 Padding-first PPI

Input: Block B , Model M

Output: Boolean vector $C(i) = \text{true}$ if $B(i)$ is padding

```

1: Initially  $C(i) \leftarrow \text{false}$  for all  $i$ 
2: for all  $B(i)$  in  $M$ ,  $C(i) \leftarrow \text{true}$ 
3:  $d = \text{DiffKL}(C, B, M)$ 
4: for  $i = 1$  to  $|B|$  do
5:    $C(i) \leftarrow \text{false}$ 
6:    $\bar{d} = \text{DiffKL}(C, B, M)$ 
7:   if  $(d < \bar{d})$  then
8:      $C(i) \leftarrow \text{true}$ 
9:   end if
10: end for
11: return  $P = \text{commands } B(i) \text{ such that } C(i) \text{ is true}$ 

```

3.3.2 Computing scores

After the two sequences are returned by the PPI algorithm, each is separately scored using the HMM and OCNB-based MDS and the final anomaly score is calculated as a “weighted combination” of these two scores [23].

$$\text{score}(B) = -\text{score}(P) - \beta \cdot \text{score}(A) \quad (5)$$

where $\beta \geq 1$. β is the weight assigned to the sum of anomaly scores for the attack sequences and in this project we have set the value of β as 2. Setting the value of β to be a constant avoids the overhead of finding the optimal β value for different users.

3.4 Mimicry attack

A masquerader is an attacker who impersonates the original user to do something malicious. The masquerader will try to develop attacks that mimic the original user activities, so as to avoid getting detected by any active MDS.

Before generating attacks, it is important to take into account three fundamental assumptions [23].

1. Perfect knowledge: The masquerader has full knowledge about the detection algorithm, all the relevant parameters and the trained model.
2. Non-poisoned detector: The MDS is trained with attack-free data.
3. Attack padding: The attack sequence must be contained within the block. Each attack command from the attack sequence can be placed at any random location in the block.

To generate mimicry attacks [23], the masquerader will generate the attacks first,

which are then placed at random locations in a block and the remaining locations are filled with padding data.

If a block consists of 100 UNIX commands, and the masquerader wants to generate 20 attack commands, these 20 attack commands are placed at random positions and the remaining 80 positions (100 - 20) are filled using padding commands from the original user's profile. The padding commands attempt to make the block look like it belongs to the original user to avoid detection. Figure 6 shows an example of a mimicry attack. The boxed commands represent the attacks and the others are padding.

```
lpdsend grep date cpp lp find expr generic mp sh file post xrdb awk
rm ln getpggrp mkpts LOCK ls env sed FIFO gethost csh download kill
userenv tcpostio UNLOCK rmdir tcpport wait4wm mimencod MediaMai netstat
xhost netscape popper gettxt xsetroot xconfirm endsessi tellwm reaper
xprop xdm cat toolches 4Dwm xterm xwsh sendmail mail gs xdvi.rea xdvi
last dc imgview launchef xv .wrapper uname fmarch .maker_w maker5X.
hostname .java_wr dirname basename egrep java make acroread ps cal xcal
touch nslookup unpack id col ul more man ping finger emacs-20 nawk
PLATFORM Slmhelpe ftp wc mkdir getopt lpdsend tektroni dev.moti Sqpe
```

Figure 6. Example of mimicry attack [23]

For testing the performance of the HMM and OCNB-based MDS developed in this project, mimicry attacks were generated for each user using the commands from other users as attack data. For each user, 50 attack blocks were generated for each of the attack lengths 10, 20, 30, ..., 100. Therefore for each user the performance of the MDS was tested using 500 attack blocks.

3.4.1 Attack generation

If B is a block of 100 commands, then the length of B is represented as $|B| = 100$. Attacks are represented by A and padding by P . If the number of attack commands in a block is 20, then $|A| = 20$, and $|P| = 80$. The algorithm for generating mimicry attacks is given below.

Algorithm 4 Mimicry attack generator

Input: Attack length $|A|$, current user

Output: Attack block B

- 1: Generate $|A|$ random numbers between 1 to 100.
 - 2: Sort the $|A|$ random numbers generated in ascending order and store as positions.
 - 3: Select a random user other than the current user as masquerader.
 - 4: From the training data of the masquerader, retrieve $|A|$ commands from a random starting position.
 - 5: Insert the $|A|$ retrieved commands in block B at the $|A|$ sorted positions.
 - 6: Retrieve $|P| = 100 - |A|$ commands from current user's training data as padding.
 - 7: Fill the $|P|$ empty positions in block B with the retrieved padding data.
 - 8: **return** B
-

Since the padding commands are retrieved directly from the original user's training data, the attack blocks thus generated will try to mimic to be as close as possible to the original user.

CHAPTER 4

Experimental results

For any masquerade detection system (MDS), the performance of the system is measured by how effectively it is able to identify masquerades. After completing the implementation of both versions of the probabilistic padding identification (PPI) algorithm, we rigorously tested the MDS with numerous test blocks containing attacks of various lengths.

The attack blocks were generated using the algorithm from Section 3.4.1, and for each current user, 50 attack blocks were generated for each attack length $|A| = 10, 20, 30, \dots, 100$. For the last section of this chapter, we generated attack blocks of lengths $|A| = 10, 20, 30, 40$ and 50 for each legitimate user, by retrieving attack commands from the training data of the remaining 49 users. Hence, the name “1 vs 49” test.

The performance of the hidden Markov model (HMM) and one class naïve Bayes (OCNB) based MDS were evaluated for each of the following cases:

1. With no PPI
2. Padding-first PPI (distance $D1$ and $D2$)
3. Attack-first PPI

For testing the performance of the original HMM and OCNB-based MDS, the attack blocks were directly scored against the trained model developed by each MDS for the current user. The results obtained without using PPI was used as a scale to determine the performance of PPI-based MDS.

For analyzing the performance of padding-first and attack-first PPI-based HMM and OCNB, the attack blocks were first run through PPI which generated separate sequences of attacks and padding for each block. These sequences were then scored separately and the final score for each block was computed using equation (5).

In this chapter, we analyze and compare the results of padding-first PPI for distance $D1$. The results of padding-first PPI using distance $D2$ are provided in Appendix C.

4.1 Analysis of PPI results

Figure 7 shows the average accuracy of padding identification comparison between padding-first and attack-first PPI for three levels of impersonation difficulty.

Figure 7. Comparison of average accuracy of padding identified

Table 1 shows the average number of commands identified as padding during training phase. During training phase, the input data is attack-free. Therefore the number of padding identified during training should be high. Padding-first PPI for distance $D1$ was able to achieve 15% to 42% identification rates as compared to 11% or lower for attack-first PPI.

Table 1. Average number of padding identified during training

Difficulty of impersonation	Padding-first PPI D1	Attack-first PPI
Hard	15.58	11.34
Intermediate	19.36	7.3
Easy	42.78	5.38

Figures 8, 9 and 10 shows the comparison of average number of commands identified as padding and attacks for three levels of impersonation difficulty.

Figure 8. Average number of attacks and padding identified - intermediate

Figure 9. Average number of attacks and padding identified - hard

Figure 10. Average number of attacks and padding identified - easy

For intermediate and easy impersonation difficulty, compared to padding-first PPI, attack-first PPI was able to identify more padding from blocks with attack lengths 20 or higher and 30 or higher respectively.

4.2 Analysis of HMM results

4.2.1 Without PPI

To begin the analysis of our HMM results, the attacks generated were scored with the original HMM and compared to the scores of the self blocks. Figures 11 and 12 show the score distribution for attack lengths 10 and 30 respectively. From the graphs we can infer that the scores of the self blocks and the blocks containing attacks of length 10 were similar and closer to 0. But for attacks of length 30, the scores of the attack blocks were lower than the scores of self blocks.

Figure 11. HMM: scores for attack length $|A| = 10$ without using PPI

Figure 12. HMM: scores for attack length $|A| = 30$ without using PPI

4.2.2 With padding-first PPI $D1$

Figures 13 and 14 show the padding-first PPI score distribution for attack lengths 10 and 30 respectively. The scores for the self blocks and the blocks containing attacks of length 10 were closer to 0 and similar to the score distribution for tests without using PPI.

For attacks of length 30, the scores of the attack blocks were lower than the self scores. Although it is not obvious from the scatter plots, padding-first PPI for attacks of length 30, when scored using HMM, achieved 15% improved detection rate than HMM without PPI. This can be clearly inferred from Figure 18 - ROC comparison and Figure 19 - AUC comparison. Therefore we can conclude that the objective of this project was achieved here.

Figure 13. HMM: scores for attack length $|A| = 10$ using padding-first PPI D1

Figure 14. HMM: scores for attack length $|A| = 30$ using padding-first PPI D1

4.2.3 With attack-first PPI

Figures 15 and 16 show the score distribution for attack lengths 10 and 30 respectively using attack-first PPI. In both the graphs, we can see that the scores of the attack blocks are similar to the scores of the self blocks.

The reason for this is the low padding identification rate during training. Almost 90% of the commands were identified as attacks during training. While computing the scores for the training blocks, the scores for the attack sequences were doubled, which caused the training scores to be lower than expected. During testing, although the padding identification rates were higher as compared to padding-first PPI, it was also higher than that achieved during training. The lower number of commands in the attack sequences, caused the scores for the attack blocks to be higher than that of the training blocks, thereby preventing them from getting detected.

Figure 15. HMM: scores for attack length $|A| = 10$ using attack-first PPI

Figure 16. HMM: scores for attack length $|A| = 30$ using attack-first PPI

4.2.4 ROC comparison

For attack lengths of 10 and 30, the scores obtained from the previous three sections were used to plot the ROC curves to compare which version of PPI-based HMM performed better than the original HMM.

Figure 17 shows the ROC comparison for attack length 10. Here we can see that both versions of PPI-based HMM range similar to the original HMM. Therefore we can say that neither can detect attacks of length 10 or less.

From the graph in Figure 18 we can infer that padding-first PPI-based HMM performed better than original HMM, thus concluding that it can detect attacks of length 25 or more. This is a 15% improvement over the original HMM detection rate. But attack-first PPI-based HMM failed to improve the detection rate.

Figure 17. HMM: ROC comparison for $|A| = 10$

Figure 18. HMM: ROC comparison for $|A| = 30$

4.2.5 AUC comparison

Figure 19 shows the AUC comparison for original, padding-first PPI and attack-first PPI-based HMM for various attack lengths. Compared to the original HMM, padding-first PPI-based HMM shows improved detection rates for attacks of length 25 or more. On the other hand, attack-first PPI-based HMM detection rates are lower than the rates for the original HMM, for all attack length variations.

Figure 19. HMM: AUC comparison

Figure 20 shows the partial AUC (AUCp) comparison for original, padding-first PPI and attack-first PPI-based HMM for attacks of lengths ranging between 10 to 100. The AUCp values are computed for a FPR of 5%, i.e., 0.05. These values are then normalized to rescale them to 100%.

Figure 20. HMM: AUCp comparison (FPR 5%)

4.3 Analysis of OCNB results

4.3.1 Without PPI

Figures 21 and 22 show the score distribution for original OCNB. In both graphs we can see that the scores of the attack blocks are similar to that of the self blocks. Prior research [9, 23] has shown that OCNB can detect attacks of lengths 50 or more.

4.3.2 With padding-first PPI $D1$

For attacks of length 10, as shown in Figure 23, the scores of attack blocks were similar to those of the self block. Therefore we can say that the detection rate is low. Figure 24 shows that padding-first PPI is very effective in improving detection rate for OCNB. Where original OCNB was able to detect attacks of lengths 50 or more, padding-first PPI-based OCNB was able to detect attacks of lengths 25 or more, thereby improving the detection rate by 50%.

Figure 21. OCNB: scores for attack length $|A| = 10$ without using PPI

Figure 22. OCNB: scores for attack length $|A| = 30$ without using PPI

Figure 23. OCNB: scores for attack length $|A| = 10$ using padding-first PPI D1

Figure 24. OCNB: scores for attack length $|A| = 30$ using padding-first PPI D1

4.3.3 With attack-first PPI

Figures 25 and 26 show that similar to attack-first PPI-based HMM, attack-first PPI-based OCNB MDS failed to improve the detection rate for the same reason stated in Section 4.2.3.

4.3.4 ROC comparison

Figure 27 shows that for attacks of length 10, all three methods had low or zero detection rate. But from Figure 28 we can infer that for attacks of length 30, padding-first PPI-based OCNB achieved 50% improvement over the original OCNB detection rate. Therefore we can conclude that padding-first PPI-based OCNB MDS can detect attacks of length of 25 or more. But, attack-first PPI-based OCNB failed to achieve any improvement.

Figure 25. OCNB: scores for attack length $|A| = 10$ using attack-first PPI

Figure 26. OCNB: scores for attack length $|A| = 30$ using attack-first PPI

Figure 27. OCNB: ROC comparison for $|A| = 10$

Figure 28. OCNB: ROC comparison for $|A| = 30$

4.3.5 AUC comparison

Figure 29 shows the AUCs computed from the ROCs of original, padding-first PPI and attack-first PPI-based OCNB.

For attack lengths of 30 or more, padding-first PPI-based OCNB shows significant improvement in detection rate. Whereas, the detection rates of attack-first PPI-based OCNB for any attack length is much lower than the original OCNB.

Figure 30 shows the AUCp comparison, with FPR 5%, for original, padding-first PPI and attack-first PPI-based OCNB for various attack lengths. We have normalized these results to rescale them to 100%.

Figure 29. OCNB: AUC comparison

Figure 30. OCNB: AUCp comparison (FPR 5%)

4.4 Performance comparison between HMM and OCNB

Figures 31, 32 and 33 show the comparison of ROCs, for various attack lengths, between HMM and OCNB for original version, padding-first PPI and attack-first PPI respectively. From these graphs we can infer that HMM out-performs OCNB in all three cases.

Figure 31. HMM vs OCNB: without PPI – ROC comparison

4.5 Performance comparison using 1 vs 49 tests

The 1 vs 49 tests were effective in analyzing the overall performance of HMM and OCNB-based masquerade detection that employed padding-first and attack-first PPI. Tables B.1 and B.2 contain the AUC results of 1 vs 49 tests for HMM. Tables B.3 and B.4 contain the AUC results of 1 vs 49 tests for OCNB. Tables B.5 and B.6 contain the AUC_p results, for FPR 5%, of 1 vs 49 tests for HMM. Tables B.7 and B.8 contain the AUC_p results, for FPR 5%, of 1 vs 49 tests for OCNB.

Figure 32. HMM vs OCNB: padding-first PPI D1 – ROC comparison

Figure 33. HMM vs OCNB: attack-first PPI – ROC comparison

CHAPTER 5

Conclusions and future work

The two types of probabilistic padding identification (PPI) algorithms, attack-first and padding-first, described in this research, make use of Kullback-Leibler (KL) divergence. These techniques were used to improve the existing hidden Markov model (HMM) and one class naïve Bayes (OCNB) based masquerade detection for UNIX commands. The PPI algorithms separate the padding commands from the attacks and the effective identification of padding, improved the results of masquerade detection.

Results from the PPI algorithms show that the padding-first PPI for distance $D1$, was able to identify 15% to 42% of the commands as padding during the training phase, whereas attack-first PPI identified 11% or less. The padding identification rates for attack-first PPI during training were lower than the rates achieved during testing, which in turn affected the scores of the training blocks, thereby evading detection. Although the padding identification rates for padding-first PPI during testing was a little lower than that of attack-first PPI, the higher padding identification rates during training improved the detection rates of the masquerade detection systems.

For both the PPI algorithms, we measured the detection rates of HMM and OCNB-based masquerade detection systems for attack lengths ranging between 10 to 100, in a block of 100 commands. Results show that, for HMM and OCNB-based masquerade detection that employed padding-first PPI with distance $D1$, the detection rates for attacks of length 25 or more, improved by 15% and 50% respectively. On the other hand, attack-first PPI-based masquerade detection recorded no improvement.

The higher padding identification rate of padding-first PPI for distance $D1$ during

training phase, as compared to the attack-first PPI, proved to improve the overall detection rate of HMM and OCNB-based masquerade detection, based on UNIX commands.

The research in this paper was limited to HMM and OCNB-based masquerade detection. It can be further extended to other techniques such as support vector machines [19] etc. Since this project was based only on UNIX commands, further research can be pursued in using KL divergence for other types of user behavior data.

Although the main objective of achieving improved detection rates was realized, the true efficiency of any masquerade detection system is measured by its ability to detect more sophisticated attacks. Further research needs to be carried out for identifying attacks that could evade the system developed in this project.

Since KL divergence works on the principle of discriminating between statistical sequences, this method can be further analyzed in the context of dead code removal from virus files.

LIST OF REFERENCES

- [1] Afgani, M. (2008). Anomaly detection using the Kullback-Leibler divergence metric. *Applied Sciences on Biomedical and Communication Technologies. ISABEL '08. First International Symposium*, 1–5.
- [2] Bradley, A.P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7), 1145–1159.
- [3] Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8), 861–874.
- [4] Gu, Y., McCallum, A. and Towsley, D. (2005). Detecting anomalies in network traffic using maximum entropy estimation. *IMC '05 Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, 32–37.
- [5] Huang, L. (2010). A study on masquerade detection. *Master's Projects. Paper 9*. Retrieved from:
http://scholarworks.sjsu.edu/etd_projects/9
- [6] Idika, N. and Mathur, A. (2007) A survey of malware detection techniques. *Technical report, Software Engineering Research Center*. Retrieved from:
<http://www.serc.net/system/files/SERC-TR-286.pdf>
- [7] Khanna, R. and Liu, H. (2008). Control theoretic approach to intrusion detection using a distributed hidden Markov model. *IEEE Wireless Communications*, 15(4), 24–33.
- [8] Kim, H. and Cha, S. (2005). Empirical evaluation of svm-based masquerade detection using unix commands. *Computers and Security*, 24(2), 160–168.
- [9] Kothari, A. (2012). Defeating masquerade detection. *Master's Projects. Paper 239*. Retrieved from:
http://scholarworks.sjsu.edu/etd_projects/239
- [10] Kullback, S. and Leibler, R.A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1), 79–86.
- [11] Maxion, R. and Townsend, T. (2002). Masquerade detection using truncated command lines. *Dependable Systems and Networks. DSN 2002. Proceedings. International Conference*, 219–228.

- [12] Mun, G.J., Noh, B.N. and Kim, Y.M. (2009) Enhanced stochastic learning for feature selection in intrusion classification. *International Journal of Innovative Computing, Information and Control*, 5(11A), 3625–3635.
- [13] Murali, A. and Rao, M. (2005). A survey on intrusion detection approaches. *Information and Communication Technologies. ICICT 2005. First International Conference*, 233–240.
- [14] Oh, J.H., Gao, J. and Rosenblatt, K. (2008) Biological data outlier detection based on Kullback-Leibler divergence. *Bioinformatics and Biomedicine. BIBM '08. IEEE International Conference*, 249–254.
- [15] Runwal, N., Low, R. and Stamp, M. (2012). Opcode graph similarity and metamorphic detection. *Journal in Computer Virology*, 8(1–2), 37–52.
- [16] Schonlau, M. and Theus, M. (2000). Detecting masquerades in intrusion detection based on unpopular commands. *Information Processing Letters*, 76(1–2), 33–38.
- [17] Schonlau, M. Masquerading user data. Masquerade data. Retrieved from: <http://www.schonlau.net/intrusion.html>
- [18] Sharma, A. and Paliwal, K. (2007). Detecting masquerades using a combination of naïve Bayes and weighted RBF approach. *Journal in Computer Virology*, 3(3), 237–245.
- [19] Shetty, S., Mukkavilli, S.K. and Keel, L.H. (2011). An integrated machine learning and control theoretic model for mining concept-drifting data streams. *Technologies for Homeland Security (HST), IEEE International Conference*, 75–80.
- [20] Sridhara, S.M. (2012). Metamorphic worm that carries its own morphing engine. *Master's Projects. Paper 240*. Retrieved from: http://scholarworks.sjsu.edu/etd_projects/240
- [21] Stamp, M. (2012). A revealing introduction to hidden Markov model. Retrieved from: <http://www.cs.sjsu.edu/~stamp/RUA/HMM.pdf>
- [22] Stamp, M. (2011). *Information Security: Principles and Practice (2nd ed.)*, Wiley.
- [23] Tapiador, J. and Clark, J. (2011). Masquerade mimicry attack detection: a randomised approach. *Computers and Security*, 30(5), 297–310.
- [24] Tavallaei, M., Stakhanova, N. and Ghorbani, A. (2010). Toward credible evaluation of anomaly-based intrusion-detection methods. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions*, 40(5), 516–524.

- [25] Tejinder, A. (2009). Intrusion detection and prevention system: CGI attacks. *Master's Projects. Paper 41*. Retrieved from:
http://scholarworks.sjsu.edu/etd_projects/41
- [26] Varian, L. (2010). Intrusion detection and prevention system: SQL injection attacks. *Master's Projects. Paper 16*. Retrieved from:
http://scholarworks.sjsu.edu/etd_projects/16
- [27] Wang, K. and Stolfo, S. (2003). One class training for masquerade detection. *3rd IEEE Conference Data Mining Workshop on Data Mining for Computer Security*. Retrieved from:
<http://cs.columbia.edu/~kewang/paper/DMSEC-camera.pdf>
- [28] Yin, Q., Shen, L., Zhang, R., Li, X. and Wang, H. (2003). Intrusion detection based on hidden Markov model. *Machine Learning and Cybernetics, International Conference*, 5, 3115–3118.

APPENDIX A

Analysis of HMM vs OCNB performance

The below graph shows the performance evaluation for the two masquerade detection techniques without using PPI. For each user, the training data of the remaining 49 users was taken as attack data. From this evaluation we can infer that, even without PPI, HMM performs better than OCNB.

Figure A.1. AUCs for 1 vs 49 tests using HMM and OCNB

APPENDIX B

Additional results for distance $D1$

Figure B.1, B.2 and B.3 are the score distributions for attacks of length 50 for original HMM, padding-first and attack-first PPI-based HMM respectively. Figure B.4 is the ROC comparison for these three HMM results.

Figure B.5, B.6 and B.7 are the score distributions for attacks of length 50 for original OCNB, padding-first and attack-first PPI-based OCNB respectively. Figure B.8 is the ROC comparison for these three OCNB results.

Figure B.1. HMM: scores for attack length $|A| = 50$ without using PPI

Figure B.2. HMM: scores for attack length $|A| = 50$ using padding-first PPI D1

Figure B.3. HMM: scores for attack length $|A| = 50$ using attack-first PPI

Figure B.4. HMM: ROC comparison for $|A| = 50$

Figure B.5. OCNB: scores for attack length $|A| = 50$ without using PPI

Figure B.6. OCNB: scores for attack length $|A| = 50$ using padding-first PPI D1

Figure B.7. OCNB: scores for attack length $|A| = 50$ using attack-first PPI

Figure B.8. OCNB: ROC comparison for $|A| = 50$

Table B.1. AUC comparison for HMM results without PPI (normal face) and padding-first PPI $D1$ (bold face)

User	$ A = 10$	$ A = 20$	$ A = 30$	$ A = 40$	$ A = 50$
1	0.47253 / 0.44335	0.54527 / 0.55755	0.61294 / 0.65351	0.67918 / 0.72714	0.74433 / 0.78441
2	0.69906 / 0.75661	0.82637 / 0.83229	0.89029 / 0.88853	0.92082 / 0.92073	0.93539 / 0.93816
3	0.72673 / 0.78743	0.82069 / 0.83918	0.88029 / 0.88331	0.91384 / 0.91106	0.93535 / 0.93331
4	0.59527 / 0.59682	0.67929 / 0.68882	0.76090 / 0.76449	0.80571 / 0.79384	0.84110 / 0.82653
5	0.70722 / 0.78510	0.81300 / 0.85314	0.86008 / 0.87751	0.86718 / 0.88457	0.88033 / 0.89576
6	0.79580 / 0.81518	0.87016 / 0.87992	0.90927 / 0.94208	0.93286 / 0.96869	0.95531 / 0.97457
7	0.48163 / 0.61053	0.56722 / 0.68743	0.64527 / 0.74363	0.71853 / 0.78702	0.77347 / 0.82559
8	0.82498 / 0.81465	0.85939 / 0.86335	0.89384 / 0.90237	0.91673 / 0.92498	0.93559 / 0.94171
9	0.28616 / 0.56878	0.44735 / 0.67624	0.58482 / 0.74416	0.69118 / 0.79200	0.76845 / 0.83702
10	0.70367 / 0.71098	0.73718 / 0.75147	0.78424 / 0.79494	0.84339 / 0.84449	0.89163 / 0.89176
11	0.65645 / 0.71853	0.74359 / 0.80602	0.80827 / 0.86155	0.85265 / 0.89547	0.87737 / 0.92176
12	0.56767 / 0.59698	0.63024 / 0.64592	0.68012 / 0.68767	0.72208 / 0.72710	0.74376 / 0.75073
13	0.60482 / 0.55984	0.68400 / 0.67073	0.74690 / 0.74106	0.78884 / 0.78971	0.81890 / 0.81514
14	0.36710 / 0.55420	0.46314 / 0.66061	0.56147 / 0.75020	0.66780 / 0.81196	0.76935 / 0.85600
15	0.41894 / 0.50882	0.55359 / 0.60771	0.63482 / 0.67947	0.69367 / 0.73037	0.72653 / 0.76682
16	0.51151 / 0.51469	0.57884 / 0.56657	0.63955 / 0.61959	0.69184 / 0.67135	0.73363 / 0.71600
17	0.49592 / 0.51490	0.57388 / 0.60649	0.65163 / 0.66988	0.71494 / 0.72559	0.76718 / 0.77086
18	0.75612 / 0.73016	0.80433 / 0.80176	0.82478 / 0.82980	0.85829 / 0.86180	0.88029 / 0.87890
19	0.55012 / 0.64498	0.68641 / 0.75208	0.78106 / 0.82731	0.82751 / 0.86208	0.85192 / 0.88229
20	0.61143 / 0.51600	0.67563 / 0.62269	0.72147 / 0.68694	0.75947 / 0.73412	0.78429 / 0.76122
21	0.78776 / 0.82837	0.90763 / 0.90155	0.92690 / 0.92604	0.93755 / 0.93467	0.94400 / 0.94376
22	0.67469 / 0.75359	0.77192 / 0.82441	0.82049 / 0.87314	0.85082 / 0.89592	0.86396 / 0.91082
23	0.32506 / 0.41412	0.43990 / 0.48939	0.51653 / 0.55535	0.57608 / 0.61188	0.63384 / 0.66331
24	0.56527 / 0.67057	0.69102 / 0.79057	0.79437 / 0.86322	0.87237 / 0.90241	0.90853 / 0.92661
25	0.51686 / 0.64514	0.62971 / 0.71922	0.71649 / 0.77878	0.79020 / 0.82473	0.84351 / 0.86576
26	0.66208 / 0.62306	0.75816 / 0.76531	0.83261 / 0.83735	0.87245 / 0.88237	0.89457 / 0.91139
27	0.35935 / 0.56629	0.53363 / 0.67457	0.66612 / 0.75527	0.76306 / 0.81065	0.82692 / 0.85139
28	0.73147 / 0.78710	0.86729 / 0.88167	0.92073 / 0.92241	0.93653 / 0.94482	0.94196 / 0.95163
29	0.51629 / 0.56257	0.63029 / 0.68894	0.73302 / 0.76751	0.79792 / 0.81829	0.83673 / 0.84633
30	1.00000 / 1.00000	1.00000 / 1.00000	1.00000 / 1.00000	1.00000 / 1.00000	1.00000 / 1.00000
31	0.56210 / 0.74155	0.70951 / 0.85331	0.79935 / 0.89437	0.85116 / 0.91314	0.88771 / 0.93231
32	0.78694 / 0.78522	0.78931 / 0.78735	0.79441 / 0.79282	0.80078 / 0.79880	0.80861 / 0.80714
33	0.65543 / 0.68914	0.72388 / 0.75865	0.78633 / 0.81020	0.83693 / 0.84849	0.89041 / 0.88792
34	0.67816 / 0.64710	0.78318 / 0.74902	0.84641 / 0.82984	0.87547 / 0.86494	0.89522 / 0.89167
35	0.45704 / 0.62588	0.68669 / 0.75278	0.84710 / 0.86976	0.94424 / 0.94792	0.98151 / 0.98261
36	0.76331 / 0.79559	0.84527 / 0.85661	0.89429 / 0.90706	0.93020 / 0.93914	0.95282 / 0.95506
37	0.54204 / 0.59445	0.63004 / 0.69759	0.71327 / 0.76600	0.78090 / 0.82539	0.82584 / 0.86122
38	0.66180 / 0.74445	0.77886 / 0.82216	0.83286 / 0.86539	0.86898 / 0.89424	0.88906 / 0.91302
39	0.45853 / 0.60057	0.60890 / 0.70257	0.70339 / 0.76265	0.75608 / 0.79967	0.78571 / 0.82612
40	0.60294 / 0.70041	0.71682 / 0.79490	0.79988 / 0.85057	0.83841 / 0.87988	0.86690 / 0.90078
41	0.66233 / 0.78633	0.72718 / 0.84159	0.78943 / 0.86731	0.83041 / 0.88037	0.85518 / 0.90363
42	0.62329 / 0.63971	0.70006 / 0.71151	0.74678 / 0.75498	0.77776 / 0.78131	0.80257 / 0.81441
43	0.72269 / 0.71710	0.75820 / 0.76865	0.79424 / 0.80367	0.82694 / 0.83204	0.85355 / 0.86433
44	0.62727 / 0.64951	0.69016 / 0.71122	0.74812 / 0.75833	0.78829 / 0.79294	0.81869 / 0.81849
45	0.55522 / 0.65822	0.66949 / 0.75371	0.75724 / 0.80147	0.81514 / 0.83596	0.84735 / 0.86400
46	0.66204 / 0.74237	0.77937 / 0.80955	0.85592 / 0.86873	0.90441 / 0.91278	0.93494 / 0.94037
47	0.58045 / 0.62229	0.70592 / 0.76608	0.82576 / 0.85686	0.90049 / 0.92163	0.94396 / 0.94567
48	0.68894 / 0.66204	0.74082 / 0.72722	0.77510 / 0.76727	0.81249 / 0.79751	0.82659 / 0.81665
49	0.32482 / 0.48229	0.46784 / 0.57265	0.57367 / 0.63951	0.63906 / 0.68008	0.68318 / 0.71420
50	0.50657 / 0.62990	0.62502 / 0.72498	0.71031 / 0.78671	0.75245 / 0.81963	0.78502 / 0.84555

Table B.2. AUC comparison for HMM results without PPI (normal face) and attack-first PPI (bold face)

User	$ A = 10$	$ A = 20$	$ A = 30$	$ A = 40$	$ A = 50$
1	0.47253 / 0.45392	0.54527 / 0.52380	0.61294 / 0.59094	0.67918 / 0.65731	0.74433 / 0.72155
2	0.69906 / 0.71322	0.82637 / 0.83049	0.89029 / 0.89065	0.92082 / 0.91690	0.93539 / 0.93073
3	0.72673 / 0.68776	0.82069 / 0.79257	0.88029 / 0.85931	0.91384 / 0.89657	0.93535 / 0.91788
4	0.59527 / 0.55551	0.67929 / 0.63694	0.76090 / 0.71694	0.80571 / 0.76502	0.84110 / 0.80473
5	0.70722 / 0.37237	0.81300 / 0.59873	0.86008 / 0.73890	0.86718 / 0.78094	0.88033 / 0.81543
6	0.79580 / 0.74035	0.87016 / 0.86429	0.90927 / 0.90984	0.93286 / 0.93555	0.95531 / 0.95392
7	0.48163 / 0.43649	0.56722 / 0.51318	0.64527 / 0.59082	0.71853 / 0.66531	0.77347 / 0.72824
8	0.82498 / 0.82204	0.85939 / 0.85527	0.89384 / 0.88959	0.91673 / 0.91245	0.93559 / 0.93114
9	0.28616 / 0.22735	0.44735 / 0.38767	0.58482 / 0.53131	0.69118 / 0.64139	0.76845 / 0.72086
10	0.70367 / 0.71167	0.73718 / 0.74053	0.78424 / 0.78371	0.84339 / 0.83371	0.89163 / 0.88061
11	0.65645 / 0.58535	0.74359 / 0.67143	0.80827 / 0.74694	0.85265 / 0.80335	0.87737 / 0.83780
12	0.56767 / 0.52171	0.63024 / 0.57698	0.68012 / 0.62192	0.72208 / 0.65739	0.74376 / 0.68000
13	0.60482 / 0.55841	0.68400 / 0.63767	0.74690 / 0.70780	0.78884 / 0.75653	0.81890 / 0.79388
14	0.36710 / 0.34694	0.46314 / 0.44351	0.56147 / 0.54596	0.66780 / 0.65388	0.76935 / 0.75457
15	0.41894 / 0.36849	0.55359 / 0.51371	0.63482 / 0.59678	0.69367 / 0.65041	0.72653 / 0.68437
16	0.51151 / 0.49065	0.57884 / 0.55747	0.63955 / 0.61959	0.69184 / 0.67555	0.73363 / 0.72016
17	0.49592 / 0.48727	0.57388 / 0.55718	0.65163 / 0.63016	0.71494 / 0.68914	0.76718 / 0.73955
18	0.75612 / 0.73522	0.80433 / 0.80343	0.82478 / 0.82490	0.85829 / 0.85698	0.88029 / 0.87927
19	0.55012 / 0.42112	0.68641 / 0.58473	0.78106 / 0.69910	0.82751 / 0.76233	0.85192 / 0.80661
20	0.61143 / 0.59163	0.67563 / 0.65461	0.72147 / 0.69710	0.75947 / 0.73755	0.78429 / 0.76424
21	0.78776 / 0.73588	0.90763 / 0.88861	0.92690 / 0.92200	0.93755 / 0.92886	0.94400 / 0.94159
22	0.67469 / 0.50220	0.77192 / 0.62645	0.82049 / 0.69151	0.85082 / 0.73416	0.86396 / 0.76310
23	0.32506 / 0.27800	0.43990 / 0.40331	0.51653 / 0.49212	0.57608 / 0.55469	0.63384 / 0.61029
24	0.56527 / 0.49914	0.69102 / 0.62759	0.79437 / 0.74249	0.87237 / 0.83143	0.90853 / 0.87849
25	0.51686 / 0.49588	0.62971 / 0.60894	0.71649 / 0.69735	0.79020 / 0.77600	0.84351 / 0.83082
26	0.66208 / 0.61820	0.75816 / 0.73110	0.83261 / 0.81233	0.87245 / 0.85588	0.89457 / 0.87955
27	0.35935 / 0.27237	0.53363 / 0.43345	0.66612 / 0.57400	0.76306 / 0.69265	0.82692 / 0.77171
28	0.73147 / 0.54702	0.86729 / 0.72151	0.92073 / 0.83220	0.93653 / 0.88078	0.94196 / 0.89955
29	0.51629 / 0.46159	0.63029 / 0.58098	0.73302 / 0.68204	0.79792 / 0.75633	0.83673 / 0.80673
30	1.00000 / 1.00000	1.00000 / 1.00000	1.00000 / 1.00000	1.00000 / 1.00000	1.00000 / 1.00000
31	0.56210 / 0.45996	0.70951 / 0.62335	0.79935 / 0.74171	0.85116 / 0.81273	0.88771 / 0.85955
32	0.78694 / 0.78514	0.78931 / 0.78735	0.79441 / 0.79163	0.80078 / 0.76718	0.80861 / 0.80449
33	0.65543 / 0.64980	0.72388 / 0.71576	0.78633 / 0.77747	0.83693 / 0.82792	0.89041 / 0.87253
34	0.67816 / 0.57406	0.78318 / 0.69592	0.84641 / 0.78698	0.87547 / 0.83318	0.89522 / 0.86237
35	0.45704 / 0.47988	0.68669 / 0.68220	0.84710 / 0.84420	0.94424 / 0.93714	0.98151 / 0.97788
36	0.76331 / 0.75514	0.84527 / 0.83849	0.89429 / 0.88841	0.93020 / 0.92469	0.95282 / 0.94767
37	0.54204 / 0.49629	0.63004 / 0.57955	0.71327 / 0.66339	0.78090 / 0.73988	0.82584 / 0.79053
38	0.66180 / 0.57335	0.77886 / 0.70180	0.83286 / 0.77176	0.86898 / 0.81902	0.88906 / 0.84710
39	0.45853 / 0.41845	0.60890 / 0.54502	0.70339 / 0.63163	0.75608 / 0.67620	0.78571 / 0.70735
40	0.60294 / 0.53588	0.71682 / 0.67151	0.79988 / 0.76037	0.83841 / 0.80114	0.86690 / 0.83641
41	0.66233 / 0.63412	0.72718 / 0.70416	0.78943 / 0.75743	0.83041 / 0.79592	0.85518 / 0.82184
42	0.62329 / 0.53363	0.70006 / 0.66465	0.74678 / 0.73237	0.77776 / 0.76967	0.80257 / 0.79473
43	0.72269 / 0.70967	0.75820 / 0.74200	0.79424 / 0.77184	0.82694 / 0.80090	0.85355 / 0.82739
44	0.62727 / 0.58580	0.69016 / 0.65706	0.74812 / 0.71559	0.78829 / 0.76016	0.81869 / 0.79404
45	0.55522 / 0.49196	0.66949 / 0.61004	0.75724 / 0.69706	0.81514 / 0.76351	0.84735 / 0.80701
46	0.66204 / 0.63567	0.77937 / 0.77567	0.85592 / 0.85204	0.90441 / 0.89220	0.93494 / 0.91976
47	0.58045 / 0.59873	0.70592 / 0.69392	0.82576 / 0.82384	0.90049 / 0.90159	0.94396 / 0.94155
48	0.68894 / 0.66024	0.74082 / 0.70710	0.77510 / 0.74098	0.81249 / 0.77620	0.82659 / 0.79555
49	0.32482 / 0.24747	0.46784 / 0.40155	0.57367 / 0.52167	0.63906 / 0.59094	0.68318 / 0.63661
50	0.50657 / 0.43763	0.62502 / 0.54914	0.71031 / 0.63506	0.75245 / 0.68396	0.78502 / 0.72135

Table B.3. AUC comparison for OCNB results without PPI (normal face) and padding-first PPI $D1$ (bold face)

User	$ A = 10$	$ A = 20$	$ A = 30$	$ A = 40$	$ A = 50$
1	0.39180 / 0.32543	0.42886 / 0.42935	0.46963 / 0.52278	0.50669 / 0.59139	0.55633 / 0.64196
2	0.58671 / 0.77935	0.72710 / 0.79502	0.82110 / 0.83592	0.87902 / 0.88351	0.91367 / 0.91284
3	0.54414 / 0.74929	0.64453 / 0.78331	0.71804 / 0.81543	0.77414 / 0.83976	0.81747 / 0.86147
4	0.55069 / 0.55486	0.58604 / 0.62514	0.62873 / 0.68233	0.66449 / 0.70700	0.69555 / 0.73090
5	0.31080 / 0.69347	0.40378 / 0.78645	0.49510 / 0.81037	0.53986 / 0.80918	0.59437 / 0.82780
6	0.46018 / 0.75108	0.68688 / 0.83755	0.77114 / 0.89143	0.81380 / 0.93476	0.84998 / 0.94567
7	0.36631 / 0.57082	0.39557 / 0.62882	0.42592 / 0.66637	0.45992 / 0.69535	0.49886 / 0.72310
8	0.77849 / 0.72645	0.79690 / 0.78939	0.81478 / 0.82184	0.82984 / 0.84143	0.84394 / 0.86149
9	0.08755 / 0.53447	0.13090 / 0.60253	0.17931 / 0.63837	0.23635 / 0.66545	0.29398 / 0.69735
10	0.68645 / 0.70086	0.70016 / 0.72273	0.71682 / 0.74220	0.73102 / 0.75598	0.75108 / 0.77086
11	0.46392 / 0.66120	0.48847 / 0.74857	0.51757 / 0.79294	0.56216 / 0.82520	0.58416 / 0.85102
12	0.39263 / 0.53604	0.42394 / 0.56976	0.44537 / 0.58865	0.46580 / 0.60657	0.48218 / 0.61759
13	0.48539 / 0.45496	0.53316 / 0.55914	0.57949 / 0.62573	0.61794 / 0.66906	0.65904 / 0.69310
14	0.27931 / 0.56588	0.31910 / 0.62106	0.37118 / 0.67673	0.43335 / 0.72496	0.50788 / 0.75792
15	0.17222 / 0.46049	0.23145 / 0.54518	0.29731 / 0.59167	0.35973 / 0.62792	0.41404 / 0.65555
16	0.39569 / 0.43229	0.46661 / 0.46004	0.51914 / 0.48696	0.56757 / 0.52049	0.60624 / 0.54951
17	0.41604 / 0.48722	0.47406 / 0.57257	0.53743 / 0.62459	0.58329 / 0.66359	0.62461 / 0.69620
18	0.72098 / 0.51869	0.79082 / 0.66541	0.81931 / 0.78057	0.84604 / 0.83449	0.87392 / 0.86249
19	0.20863 / 0.56412	0.27386 / 0.66576	0.34416 / 0.72894	0.38943 / 0.75869	0.44141 / 0.78657
20	0.54045 / 0.35184	0.59861 / 0.47816	0.64253 / 0.55902	0.68649 / 0.61959	0.71404 / 0.64718
21	0.65176 / 0.78355	0.75841 / 0.81196	0.84488 / 0.85039	0.90347 / 0.87951	0.94639 / 0.91763
22	0.22455 / 0.64108	0.28769 / 0.73710	0.33216 / 0.79249	0.37339 / 0.82927	0.41394 / 0.84682
23	0.13800 / 0.40824	0.23747 / 0.47504	0.34712 / 0.53131	0.44518 / 0.57808	0.51533 / 0.63012
24	0.39204 / 0.60465	0.44365 / 0.72053	0.49927 / 0.78665	0.55841 / 0.82302	0.59663 / 0.85237
25	0.39592 / 0.58353	0.45551 / 0.64845	0.50927 / 0.68498	0.57373 / 0.72657	0.62627 / 0.76012
26	0.43053 / 0.46057	0.48971 / 0.58069	0.54708 / 0.63669	0.59753 / 0.68629	0.62631 / 0.71590
27	0.14571 / 0.50424	0.20302 / 0.55551	0.26857 / 0.59482	0.35384 / 0.64049	0.43024 / 0.67982
28	0.11571 / 0.67784	0.17492 / 0.78661	0.26343 / 0.82216	0.36710 / 0.84290	0.43435 / 0.85053
29	0.32955 / 0.43694	0.39922 / 0.52235	0.46749 / 0.58392	0.54078 / 0.64539	0.60945 / 0.69933
30	0.47543 / 1.00000	0.71273 / 1.00000	0.84384 / 1.00000	0.90571 / 1.00000	0.93367 / 1.00000
31	0.24931 / 0.68131	0.32982 / 0.81069	0.41700 / 0.84173	0.48876 / 0.85735	0.55406 / 0.79939
32	0.84555 / 0.84378	0.85722 / 0.85482	0.86359 / 0.86143	0.86882 / 0.86514	0.87131 / 0.86771
33	0.59376 / 0.62200	0.62424 / 0.67143	0.66082 / 0.70280	0.69463 / 0.72865	0.72616 / 0.74322
34	0.25406 / 0.55269	0.35408 / 0.63651	0.45449 / 0.68763	0.51084 / 0.71571	0.55094 / 0.74094
35	0.31878 / 0.57906	0.46327 / 0.62629	0.61514 / 0.69453	0.74829 / 0.78204	0.83890 / 0.85176
36	0.63878 / 0.77404	0.73155 / 0.81245	0.79622 / 0.84510	0.84004 / 0.87837	0.87310 / 0.90653
37	0.40857 / 0.48192	0.44771 / 0.56649	0.48245 / 0.61629	0.52690 / 0.67780	0.56984 / 0.72035
38	0.36318 / 0.65922	0.44973 / 0.74784	0.51545 / 0.78367	0.57563 / 0.81282	0.61404 / 0.84110
39	0.18147 / 0.56431	0.19978 / 0.64500	0.22086 / 0.68655	0.25976 / 0.70812	0.29057 / 0.72406
40	0.31082 / 0.58802	0.38571 / 0.65492	0.45608 / 0.69469	0.51759 / 0.72710	0.57755 / 0.75255
41	0.46555 / 0.72910	0.51424 / 0.79488	0.55653 / 0.82031	0.59159 / 0.82763	0.61755 / 0.85012
42	0.38588 / 0.58461	0.55343 / 0.66333	0.64139 / 0.71188	0.69173 / 0.74669	0.72171 / 0.78327
43	0.66947 / 0.69882	0.67969 / 0.74196	0.68833 / 0.76127	0.69824 / 0.77563	0.70490 / 0.79482
44	0.50137 / 0.59918	0.54551 / 0.64669	0.57773 / 0.68441	0.60771 / 0.71102	0.63637 / 0.73641
45	0.34120 / 0.57841	0.41663 / 0.67041	0.48159 / 0.70339	0.53429 / 0.72947	0.57341 / 0.75886
46	0.50398 / 0.72298	0.61294 / 0.78049	0.69967 / 0.81118	0.75673 / 0.84151	0.81196 / 0.86180
47	0.49816 / 0.49188	0.57555 / 0.66649	0.65365 / 0.75280	0.74335 / 0.82963	0.81967 / 0.88163
48	0.52190 / 0.58698	0.53647 / 0.62392	0.53935 / 0.64094	0.55410 / 0.65214	0.56320 / 0.65735
49	0.11159 / 0.43196	0.13790 / 0.47565	0.17947 / 0.50861	0.21957 / 0.52216	0.25916 / 0.54063
50	0.20076 / 0.54067	0.23582 / 0.62576	0.27535 / 0.66980	0.31796 / 0.69657	0.33696 / 0.71769

Table B.4. AUC comparison for OCNB results without PPI (normal face) and attack-first PPI (bold face)

User	$ A = 10$	$ A = 20$	$ A = 30$	$ A = 40$	$ A = 50$
1	0.39180 / 0.31298	0.42886 / 0.35392	0.46963 / 0.39292	0.50669 / 0.42880	0.55633 / 0.48171
2	0.58671 / 0.68176	0.72710 / 0.77947	0.82110 / 0.82661	0.87902 / 0.86608	0.91367 / 0.89718
3	0.54414 / 0.26661	0.64453 / 0.27143	0.71804 / 0.32078	0.77414 / 0.45743	0.81747 / 0.59182
4	0.55069 / 0.41882	0.58604 / 0.43786	0.62873 / 0.45488	0.66449 / 0.46173	0.69555 / 0.48478
5	0.31080 / 0.02498	0.40378 / 0.06204	0.49510 / 0.11071	0.53986 / 0.17686	0.59437 / 0.26986
6	0.46018 / 0.20282	0.68688 / 0.36124	0.77114 / 0.52855	0.81380 / 0.70031	0.84998 / 0.80339
7	0.36631 / 0.26649	0.39557 / 0.26849	0.42592 / 0.28233	0.45992 / 0.30612	0.49886 / 0.33535
8	0.77849 / 0.80127	0.79690 / 0.81159	0.81478 / 0.82339	0.82984 / 0.83306	0.84394 / 0.84178
9	0.08755 / 0.02543	0.13090 / 0.03473	0.17931 / 0.05616	0.23635 / 0.09535	0.29398 / 0.12741
10	0.68645 / 0.68210	0.70016 / 0.69633	0.71682 / 0.70955	0.73102 / 0.72233	0.75108 / 0.73494
11	0.46392 / 0.35886	0.48847 / 0.36618	0.51757 / 0.37014	0.56216 / 0.39539	0.58416 / 0.40333
12	0.39263 / 0.26853	0.42394 / 0.25573	0.44537 / 0.26522	0.46580 / 0.26094	0.48218 / 0.26135
13	0.48539 / 0.31329	0.53316 / 0.35892	0.57949 / 0.40343	0.61794 / 0.45088	0.65904 / 0.50984
14	0.27931 / 0.22427	0.31910 / 0.25449	0.37118 / 0.30739	0.43335 / 0.37182	0.50788 / 0.44698
15	0.17222 / 0.18702	0.23145 / 0.20749	0.29731 / 0.24408	0.35973 / 0.27237	0.41404 / 0.30961
16	0.39569 / 0.26700	0.46661 / 0.32729	0.51914 / 0.37959	0.56757 / 0.44314	0.60624 / 0.48649
17	0.41604 / 0.42176	0.47406 / 0.44171	0.53743 / 0.47241	0.58329 / 0.49529	0.62461 / 0.51910
18	0.72098 / 0.60888	0.79082 / 0.73278	0.81931 / 0.78943	0.84604 / 0.80800	0.87392 / 0.80592
19	0.20863 / 0.01906	0.27386 / 0.03906	0.34416 / 0.07467	0.38943 / 0.10231	0.44141 / 0.16012
20	0.54045 / 0.44096	0.59861 / 0.49080	0.64253 / 0.52506	0.68649 / 0.56453	0.71404 / 0.59298
21	0.65176 / 0.39759	0.75841 / 0.42531	0.84488 / 0.50176	0.90347 / 0.58624	0.94639 / 0.73445
22	0.22455 / 0.12057	0.28769 / 0.12988	0.33216 / 0.13208	0.37339 / 0.14433	0.41394 / 0.17592
23	0.13800 / 0.08029	0.23747 / 0.11347	0.34712 / 0.16216	0.44518 / 0.23645	0.51533 / 0.31596
24	0.39204 / 0.24678	0.44365 / 0.26404	0.49927 / 0.31529	0.55841 / 0.35878	0.59663 / 0.40741
25	0.39592 / 0.29218	0.45551 / 0.34904	0.50927 / 0.39437	0.57373 / 0.45335	0.62627 / 0.52878
26	0.43053 / 0.27033	0.48971 / 0.29420	0.54708 / 0.34298	0.59753 / 0.39049	0.62631 / 0.44712
27	0.14571 / 0.07722	0.20302 / 0.09433	0.26857 / 0.11067	0.35384 / 0.15131	0.43024 / 0.20918
28	0.11571 / 0.10208	0.17492 / 0.09986	0.26343 / 0.10437	0.36710 / 0.11204	0.43435 / 0.14498
29	0.32955 / 0.21371	0.39922 / 0.23535	0.46749 / 0.25959	0.54078 / 0.29447	0.60945 / 0.34796
30	0.47543 / 0.99314	0.71273 / 0.99796	0.84384 / 1.00000	0.90571 / 1.00000	0.93367 / 1.00000
31	0.24931 / 0.07353	0.32982 / 0.11004	0.41700 / 0.13951	0.48876 / 0.19310	0.55406 / 0.25798
32	0.84555 / 0.83449	0.85722 / 0.83922	0.86359 / 0.85180	0.86882 / 0.85661	0.87131 / 0.85955
33	0.59376 / 0.48073	0.62424 / 0.53890	0.66082 / 0.58127	0.69463 / 0.61906	0.72616 / 0.65004
34	0.25406 / 0.23218	0.35408 / 0.25286	0.45449 / 0.30041	0.51084 / 0.33320	0.55094 / 0.38857
35	0.31878 / 0.37412	0.46327 / 0.54220	0.61514 / 0.60941	0.74829 / 0.79298	0.83890 / 0.85514
36	0.63878 / 0.57233	0.73155 / 0.63137	0.79622 / 0.67935	0.84004 / 0.72112	0.87310 / 0.76857
37	0.40857 / 0.28418	0.44771 / 0.30567	0.48245 / 0.33204	0.52690 / 0.37135	0.56984 / 0.42024
38	0.36318 / 0.28978	0.44973 / 0.31343	0.51545 / 0.33618	0.57563 / 0.37580	0.61404 / 0.41257
39	0.18147 / 0.15065	0.19978 / 0.13214	0.22086 / 0.12082	0.25976 / 0.12273	0.29057 / 0.13392
40	0.31082 / 0.20922	0.38571 / 0.22459	0.45608 / 0.26531	0.51759 / 0.31092	0.57755 / 0.36424
41	0.46555 / 0.43118	0.51424 / 0.43351	0.55653 / 0.43537	0.59159 / 0.42976	0.61755 / 0.42706
42	0.38588 / 0.22127	0.55343 / 0.27147	0.64139 / 0.35500	0.69173 / 0.46102	0.72171 / 0.56604
43	0.66947 / 0.53741	0.67969 / 0.57229	0.68833 / 0.59865	0.69824 / 0.62980	0.70490 / 0.66094
44	0.50137 / 0.31376	0.54551 / 0.37906	0.57773 / 0.42518	0.60771 / 0.47698	0.63637 / 0.54292
45	0.34120 / 0.16584	0.41663 / 0.20320	0.48159 / 0.25271	0.53429 / 0.29255	0.57341 / 0.36147
46	0.50398 / 0.27871	0.61294 / 0.40759	0.69967 / 0.50067	0.75673 / 0.57810	0.81196 / 0.66306
47	0.49816 / 0.33555	0.57555 / 0.46249	0.65365 / 0.60437	0.74335 / 0.73594	0.81967 / 0.83955
48	0.52190 / 0.44633	0.53647 / 0.44653	0.53935 / 0.43351	0.55410 / 0.43416	0.56320 / 0.44514
49	0.11159 / 0.03220	0.13790 / 0.03641	0.17947 / 0.05906	0.21957 / 0.07822	0.25916 / 0.10731
50	0.20076 / 0.13480	0.23582 / 0.14249	0.27535 / 0.15112	0.31796 / 0.15833	0.33696 / 0.17667

Table B.5. AUCp (FPR 5%) comparison for HMM results without PPI (normal face) and padding-first PPI $D1$ (bold face)

User	$ A = 10$	$ A = 20$	$ A = 30$	$ A = 40$	$ A = 50$
1	0.05714 / 0.02286	0.08490 / 0.06612	0.16327 / 0.09224	0.21959 / 0.12816	0.22531 / 0.24327
2	0.13388 / 0.50612	0.44653 / 0.51184	0.72082 / 0.70204	0.75265 / 0.77714	0.69143 / 0.75184
3	0.36735 / 0.47837	0.50041 / 0.56980	0.59347 / 0.64980	0.73388 / 0.74776	0.81224 / 0.80735
4	0.30041 / 0.21714	0.32327 / 0.24490	0.35265 / 0.27020	0.38612 / 0.27755	0.40000 / 0.30939
5	0.12653 / 0.18286	0.19592 / 0.22367	0.14857 / 0.17796	0.12000 / 0.15592	0.08816 / 0.14122
6	0.59020 / 0.58531	0.66367 / 0.65061	0.74939 / 0.76490	0.71918 / 0.79184	0.76980 / 0.82367
7	0.01714 / 0.03429	0.08000 / 0.07592	0.12408 / 0.21633	0.16245 / 0.18204	0.17224 / 0.25143
8	0.68000 / 0.58776	0.68980 / 0.59673	0.71837 / 0.62776	0.74449 / 0.65551	0.78122 / 0.70612
9	0.00980 / 0.00571	0.04571 / 0.04000	0.06531 / 0.07510	0.10204 / 0.15184	0.10286 / 0.22286
10	0.62612 / 0.52898	0.64816 / 0.61306	0.66286 / 0.64898	0.66612 / 0.66204	0.67265 / 0.69469
11	0.09959 / 0.07510	0.18857 / 0.16408	0.20735 / 0.31347	0.25796 / 0.39673	0.23265 / 0.51020
12	0.17959 / 0.20000	0.21224 / 0.24000	0.23102 / 0.24653	0.27184 / 0.35102	0.20980 / 0.33796
13	0.14857 / 0.09306	0.19755 / 0.19755	0.22449 / 0.22857	0.25714 / 0.26367	0.20735 / 0.24653
14	0.02204 / 0.03347	0.05633 / 0.18041	0.15184 / 0.27265	0.17796 / 0.10204	0.24000 / 0.05224
15	0.02531 / 0.02939	0.04735 / 0.04571	0.08082 / 0.08980	0.09143 / 0.07673	0.05551 / 0.10286
16	0.21469 / 0.06857	0.24408 / 0.04000	0.30531 / 0.03918	0.32327 / 0.04898	0.30449 / 0.08571
17	0.07510 / 0.06531	0.18939 / 0.19265	0.25959 / 0.24816	0.32082 / 0.27755	0.41061 / 0.30857
18	0.50857 / 0.20327	0.66041 / 0.64245	0.68816 / 0.69388	0.70204 / 0.72000	0.71918 / 0.72490
19	0.13959 / 0.15265	0.15918 / 0.17714	0.17959 / 0.19673	0.17388 / 0.17633	0.11184 / 0.15755
20	0.17714 / 0.03592	0.27837 / 0.04735	0.26041 / 0.07184	0.28735 / 0.08735	0.24816 / 0.08327
21	0.47673 / 0.42857	0.61633 / 0.60980	0.82857 / 0.82531	0.86286 / 0.86612	0.87837 / 0.88571
22	0.08490 / 0.11673	0.12490 / 0.14286	0.16816 / 0.20490	0.18122 / 0.20980	0.13061 / 0.21796
23	0.03837 / 0.05796	0.19592 / 0.21061	0.23592 / 0.26449	0.29551 / 0.31837	0.34286 / 0.36408
24	0.12408 / 0.10041	0.16082 / 0.17469	0.20735 / 0.24653	0.26694 / 0.29469	0.31184 / 0.33061
25	0.04898 / 0.12735	0.14367 / 0.13143	0.20408 / 0.20327	0.31429 / 0.27184	0.37551 / 0.34776
26	0.10449 / 0.07184	0.14694 / 0.09959	0.16490 / 0.11837	0.16163 / 0.19918	0.24571 / 0.31020
27	0.01143 / 0.02612	0.01959 / 0.06857	0.06286 / 0.14776	0.15755 / 0.27837	0.25714 / 0.36000
28	0.16735 / 0.16490	0.23347 / 0.28653	0.23510 / 0.32082	0.23918 / 0.33143	0.30531 / 0.36653
29	0.06041 / 0.13061	0.11429 / 0.18531	0.31837 / 0.30286	0.40898 / 0.44490	0.48163 / 0.53490
30	0.97959 / 0.97959	0.97959 / 0.97959	0.97959 / 0.97959	0.97959 / 0.97959	0.97959 / 0.97959
31	0.11510 / 0.12653	0.19673 / 0.22041	0.19592 / 0.22776	0.24000 / 0.29265	0.29061 / 0.36082
32	0.76408 / 0.76408	0.76408 / 0.76408	0.76408 / 0.76408	0.76408 / 0.76408	0.76245 / 0.76408
33	0.50531 / 0.46286	0.52245 / 0.51184	0.56000 / 0.59918	0.59673 / 0.62939	0.62531 / 0.64000
34	0.21143 / 0.16571	0.22857 / 0.19510	0.25143 / 0.24571	0.18694 / 0.18694	0.19592 / 0.23755
35	0.00408 / 0.37878	0.26122 / 0.40163	0.46531 / 0.56163	0.61633 / 0.64082	0.82367 / 0.82939
36	0.34857 / 0.52163	0.54694 / 0.64000	0.71429 / 0.74204	0.77796 / 0.79347	0.79918 / 0.82286
37	0.10776 / 0.13714	0.11837 / 0.13224	0.14204 / 0.11673	0.22122 / 0.16653	0.27837 / 0.22367
38	0.11837 / 0.11265	0.12735 / 0.12735	0.15918 / 0.17633	0.21959 / 0.25306	0.21388 / 0.27510
39	0.00000 / 0.00000	0.00408 / 0.01224	0.01796 / 0.05878	0.05878 / 0.06694	0.08327 / 0.20000
40	0.02204 / 0.04408	0.05061 / 0.05714	0.07347 / 0.09061	0.09551 / 0.14531	0.11184 / 0.25388
41	0.16980 / 0.13388	0.27592 / 0.26122	0.30939 / 0.26776	0.34776 / 0.29061	0.35755 / 0.35592
42	0.28204 / 0.31755	0.44898 / 0.44816	0.52408 / 0.46204	0.55673 / 0.47102	0.58286 / 0.51102
43	0.55102 / 0.45306	0.56163 / 0.46367	0.57959 / 0.48816	0.58776 / 0.50612	0.59918 / 0.57061
44	0.30204 / 0.30041	0.40000 / 0.37878	0.45714 / 0.43755	0.48980 / 0.46367	0.52000 / 0.50612
45	0.15673 / 0.15429	0.23673 / 0.21633	0.30531 / 0.24735	0.36327 / 0.26649	0.34041 / 0.32082
46	0.22694 / 0.35429	0.47837 / 0.62939	0.61633 / 0.68735	0.67265 / 0.73796	0.70122 / 0.72571
47	0.24408 / 0.20980	0.32898 / 0.34612	0.48571 / 0.56490	0.65878 / 0.78694	0.80816 / 0.87592
48	0.10367 / 0.18776	0.17959 / 0.20980	0.19918 / 0.21061	0.21388 / 0.20816	0.18286 / 0.24408
49	0.03592 / 0.02612	0.06286 / 0.05551	0.06367 / 0.04898	0.07837 / 0.06857	0.08571 / 0.06776
50	0.04327 / 0.05143	0.08163 / 0.08735	0.09959 / 0.11673	0.07673 / 0.12000	0.06367 / 0.19592

Table B.6. AUCp (FPR 5%) comparison for HMM results without PPI (normal face) and attack-first PPI (bold face)

User	$ A = 10$	$ A = 20$	$ A = 30$	$ A = 40$	$ A = 50$
1	0.05714 / 0.05714	0.08490 / 0.08490	0.16327 / 0.14449	0.21959 / 0.20327	0.22531 / 0.21714
2	0.13388 / 0.18776	0.44653 / 0.44898	0.72082 / 0.69878	0.75265 / 0.74122	0.69143 / 0.66612
3	0.36735 / 0.32408	0.50041 / 0.46041	0.59347 / 0.56000	0.73388 / 0.70041	0.81224 / 0.77633
4	0.30041 / 0.29388	0.32327 / 0.30612	0.35265 / 0.32490	0.38612 / 0.35347	0.40000 / 0.36082
5	0.12653 / 0.00000	0.19592 / 0.00571	0.14857 / 0.00327	0.12000 / 0.00163	0.08816 / 0.00490
6	0.59020 / 0.10367	0.66367 / 0.56245	0.74939 / 0.67429	0.71918 / 0.67020	0.76980 / 0.67918
7	0.01714 / 0.01224	0.08000 / 0.02776	0.12408 / 0.08327	0.16245 / 0.13306	0.17224 / 0.13388
8	0.68000 / 0.68490	0.68980 / 0.68490	0.71837 / 0.71918	0.74449 / 0.73551	0.78122 / 0.77633
9	0.00980 / 0.00000	0.04571 / 0.02041	0.06531 / 0.05143	0.10204 / 0.07755	0.10286 / 0.07265
10	0.62612 / 0.62694	0.64816 / 0.65143	0.66286 / 0.66612	0.66612 / 0.66694	0.67265 / 0.68980
11	0.09959 / 0.02612	0.18857 / 0.07510	0.20735 / 0.08980	0.25796 / 0.16898	0.23265 / 0.15265
12	0.17959 / 0.17633	0.21224 / 0.17388	0.23102 / 0.16816	0.27184 / 0.18612	0.20980 / 0.16327
13	0.14857 / 0.12245	0.19755 / 0.15755	0.22449 / 0.19102	0.25714 / 0.21714	0.20735 / 0.17796
14	0.02204 / 0.00000	0.05633 / 0.05143	0.15184 / 0.06041	0.17796 / 0.16980	0.24000 / 0.22612
15	0.02531 / 0.01469	0.04735 / 0.02286	0.08082 / 0.02857	0.09143 / 0.03429	0.05551 / 0.02204
16	0.21469 / 0.16408	0.24408 / 0.20816	0.30531 / 0.26367	0.32327 / 0.27510	0.30449 / 0.26204
17	0.07510 / 0.05061	0.18939 / 0.12490	0.25959 / 0.22367	0.32082 / 0.26776	0.41061 / 0.31347
18	0.50857 / 0.50286	0.66041 / 0.65878	0.68816 / 0.68735	0.70204 / 0.70041	0.71918 / 0.71673
19	0.13959 / 0.03265	0.15918 / 0.04082	0.17959 / 0.05796	0.17388 / 0.04163	0.11184 / 0.03429
20	0.17714 / 0.13061	0.27837 / 0.22612	0.26041 / 0.19429	0.28735 / 0.24245	0.24816 / 0.20245
21	0.47673 / 0.36816	0.61633 / 0.51184	0.82857 / 0.75755	0.86286 / 0.75020	0.87837 / 0.86531
22	0.08490 / 0.00898	0.12490 / 0.01224	0.16816 / 0.02694	0.18122 / 0.03265	0.13061 / 0.01143
23	0.03837 / 0.00327	0.19592 / 0.12490	0.23592 / 0.18367	0.29551 / 0.24735	0.34286 / 0.27020
24	0.12408 / 0.11592	0.16082 / 0.14367	0.20735 / 0.18204	0.26694 / 0.23755	0.31184 / 0.25878
25	0.04898 / 0.03347	0.14367 / 0.10612	0.20408 / 0.18122	0.31429 / 0.26857	0.37551 / 0.34857
26	0.10449 / 0.10531	0.14694 / 0.12000	0.16490 / 0.11265	0.16163 / 0.11265	0.24571 / 0.16327
27	0.01143 / 0.00082	0.01959 / 0.01959	0.06286 / 0.01878	0.15755 / 0.08082	0.25714 / 0.13714
28	0.16735 / 0.05143	0.23347 / 0.06531	0.23510 / 0.08082	0.23918 / 0.09469	0.30531 / 0.09306
29	0.06041 / 0.05061	0.11429 / 0.10694	0.31837 / 0.25959	0.40898 / 0.35184	0.48163 / 0.40898
30	0.97959 / 0.97959	0.97959 / 0.97959	0.97959 / 0.97959	0.97959 / 0.97959	0.97959 / 0.97959
31	0.11510 / 0.04000	0.19673 / 0.10041	0.19592 / 0.11592	0.24000 / 0.17224	0.29061 / 0.21222
32	0.76408 / 0.76408	0.76408 / 0.76408	0.76408 / 0.76408	0.76408 / 0.76408	0.76245 / 0.76327
33	0.50531 / 0.51429	0.52245 / 0.52571	0.56000 / 0.54939	0.59673 / 0.57714	0.62531 / 0.60653
34	0.21143 / 0.05551	0.22857 / 0.06286	0.25143 / 0.09224	0.18694 / 0.08571	0.19592 / 0.08653
35	0.00408 / 0.04816	0.26122 / 0.33469	0.46531 / 0.49796	0.61633 / 0.62367	0.82367 / 0.82367
36	0.34857 / 0.30939	0.54694 / 0.50204	0.71429 / 0.69388	0.77796 / 0.76245	0.79918 / 0.78939
37	0.10776 / 0.09714	0.11837 / 0.11020	0.14204 / 0.13959	0.22122 / 0.20816	0.27837 / 0.25714
38	0.11837 / 0.04327	0.12735 / 0.08163	0.15918 / 0.11020	0.21959 / 0.13469	0.21388 / 0.12327
39	0.00000 / 0.00000	0.00408 / 0.00000	0.01796 / 0.00000	0.05878 / 0.00000	0.08327 / 0.00163
40	0.02204 / 0.01959	0.05061 / 0.02939	0.07347 / 0.06531	0.09551 / 0.05878	0.11184 / 0.07020
41	0.16980 / 0.10204	0.27592 / 0.16816	0.30939 / 0.15184	0.34776 / 0.20571	0.35755 / 0.26367
42	0.28204 / 0.03265	0.44898 / 0.21878	0.52408 / 0.41143	0.55673 / 0.50612	0.58286 / 0.53469
43	0.55102 / 0.52327	0.56163 / 0.54531	0.57959 / 0.55347	0.58776 / 0.57388	0.59918 / 0.57959
44	0.30204 / 0.25224	0.40000 / 0.33878	0.45714 / 0.43184	0.48980 / 0.45551	0.52000 / 0.47184
45	0.15673 / 0.03265	0.23673 / 0.10694	0.30531 / 0.15755	0.36327 / 0.20490	0.34041 / 0.23592
46	0.22694 / 0.16490	0.47837 / 0.45959	0.61633 / 0.60653	0.67265 / 0.66204	0.70122 / 0.68571
47	0.24408 / 0.25061	0.32898 / 0.33633	0.48571 / 0.50122	0.65878 / 0.67184	0.80816 / 0.82776
48	0.10367 / 0.09061	0.17959 / 0.14857	0.19918 / 0.15837	0.21388 / 0.16980	0.18286 / 0.15837
49	0.03592 / 0.02939	0.06286 / 0.03755	0.06367 / 0.04408	0.07837 / 0.04245	0.08571 / 0.05633
50	0.04327 / 0.02367	0.08163 / 0.03020	0.09959 / 0.03265	0.07673 / 0.02204	0.06367 / 0.01878

Table B.7. AUCp (FPR 5%) comparison for OCNB results without PPI (normal face) and padding-first PPI $D1$ (bold face)

User	$ A = 10$	$ A = 20$	$ A = 30$	$ A = 40$	$ A = 50$
1	0.00000 / 0.00000	0.00000 / 0.00000	0.00163 / 0.00000	0.00082 / 0.00082	0.00735 / 0.00571
2	0.07429 / 0.07837	0.08327 / 0.08327	0.09959 / 0.09714	0.12939 / 0.13551	0.17959 / 0.19184
3	0.00000 / 0.00000	0.00000 / 0.00000	0.00082 / 0.00000	0.00000 / 0.00000	0.00408 / 0.00245
4	0.07429 / 0.06816	0.07265 / 0.07510	0.07510 / 0.09388	0.10367 / 0.13714	0.13061 / 0.16327
5	0.00000 / 0.00000	0.00000 / 0.00408	0.00082 / 0.01388	0.01224 / 0.03265	0.01959 / 0.04980
6	0.00000 / 0.00163	0.00245 / 0.03429	0.01633 / 0.07429	0.02204 / 0.19496	0.02449 / 0.28408
7	0.00000 / 0.02122	0.00163 / 0.02367	0.00490 / 0.02694	0.01224 / 0.04653	0.02041 / 0.06612
8	0.00245 / 0.01633	0.00327 / 0.01878	0.01306 / 0.02776	0.01388 / 0.03102	0.03429 / 0.05184
9	0.00000 / 0.00000	0.00000 / 0.00000	0.00163 / 0.00000	0.00245 / 0.00163	0.00327 / 0.00245
10	0.00000 / 0.00000	0.00082 / 0.00000	0.00898 / 0.00408	0.01388 / 0.00980	0.02694 / 0.01959
11	0.00000 / 0.01388	0.00163 / 0.03673	0.00816 / 0.06041	0.01959 / 0.08245	0.02612 / 0.10694
12	0.00000 / 0.00000	0.00163 / 0.00163	0.00245 / 0.00327	0.00980 / 0.00327	0.01551 / 0.00980
13	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000
14	0.06612 / 0.04735	0.07020 / 0.08816	0.07510 / 0.10694	0.07918 / 0.12082	0.09796 / 0.16735
15	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00327	0.00490 / 0.01061
16	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00245 / 0.00245	0.00653 / 0.00898
17	0.00000 / 0.00000	0.00000 / 0.00000	0.00163 / 0.00163	0.00408 / 0.00327	0.01306 / 0.00980
18	0.00000 / 0.00000	0.00082 / 0.00000	0.00735 / 0.00000	0.02367 / 0.00653	0.05143 / 0.01959
19	0.00000 / 0.00000	0.00000 / 0.01796	0.00653 / 0.06612	0.00816 / 0.12653	0.01633 / 0.18041
20	0.00000 / 0.00000	0.00000 / 0.00082	0.00000 / 0.00000	0.00163 / 0.00490	0.00735 / 0.01143
21	0.31673 / 0.25061	0.34612 / 0.29714	0.42367 / 0.33551	0.49959 / 0.37551	0.61878 / 0.43102
22	0.00408 / 0.00490	0.01959 / 0.02694	0.04082 / 0.06122	0.07673 / 0.10041	0.10286 / 0.13633
23	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000
24	0.00000 / 0.00245	0.00653 / 0.01224	0.01633 / 0.04245	0.02041 / 0.08082	0.03429 / 0.14612
25	0.00000 / 0.00000	0.00000 / 0.00000	0.00163 / 0.00082	0.00000 / 0.00163	0.00245 / 0.00163
26	0.00000 / 0.00082	0.00245 / 0.00490	0.00653 / 0.00653	0.01633 / 0.01714	0.01959 / 0.01878
27	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00163 / 0.00000
28	0.00000 / 0.00163	0.00000 / 0.00816	0.00980 / 0.02286	0.03673 / 0.05388	0.09959 / 0.09388
29	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000
30	0.13061 / 0.40000	0.24327 / 0.40000	0.31184 / 0.40000	0.35184 / 0.40000	0.36327 / 0.40000
31	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00163	0.00408 / 0.00653	0.00490 / 0.02204
32	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00163 / 0.00000
33	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000
34	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00163	0.02449 / 0.03184	0.05714 / 0.06531
35	0.00000 / 0.00000	0.00163 / 0.00408	0.03020 / 0.02857	0.11184 / 0.11918	0.23673 / 0.22612
36	0.00163 / 0.00000	0.01143 / 0.00735	0.01959 / 0.01633	0.02204 / 0.01714	0.03429 / 0.03102
37	0.00000 / 0.00245	0.00735 / 0.02612	0.01143 / 0.02857	0.01714 / 0.04571	0.02694 / 0.07102
38	0.00000 / 0.00082	0.00000 / 0.00490	0.00000 / 0.01551	0.00163 / 0.03592	0.01143 / 0.06041
39	0.00000 / 0.02041	0.00000 / 0.01061	0.00000 / 0.01388	0.00245 / 0.01633	0.00408 / 0.01878
40	0.00000 / 0.01061	0.00245 / 0.04653	0.00571 / 0.06367	0.01469 / 0.10122	0.04245 / 0.13061
41	0.07429 / 0.06041	0.07102 / 0.06857	0.07510 / 0.06857	0.08816 / 0.07265	0.11020 / 0.08327
42	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00082
43	0.00000 / 0.00000	0.00000 / 0.00000	0.00735 / 0.00653	0.00980 / 0.00980	0.01306 / 0.01388
44	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00163
45	0.00082 / 0.00000	0.00980 / 0.02204	0.03510 / 0.05061	0.04898 / 0.06531	0.07837 / 0.10286
46	0.00000 / 0.02857	0.01959 / 0.05551	0.03755 / 0.05796	0.04980 / 0.07102	0.07673 / 0.09061
47	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00082	0.00327 / 0.00327	0.01224 / 0.01306
48	0.00000 / 0.00000	0.00327 / 0.00327	0.00327 / 0.00408	0.01224 / 0.01143	0.02286 / 0.01714
49	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00163	0.00327 / 0.00245
50	0.00653 / 0.02041	0.02531 / 0.01796	0.04898 / 0.03673	0.07184 / 0.06041	0.09714 / 0.06531

Table B.8. AUCp (FPR 5%) comparison for OCNB results without PPI (normal face) and attack-first PPI (bold face)

User	$ A = 10$	$ A = 20$	$ A = 30$	$ A = 40$	$ A = 50$
1	0.00000 / 0.00000	0.00000 / 0.00000	0.00163 / 0.00000	0.00082 / 0.00245	0.00735 / 0.00408
2	0.07429 / 0.02776	0.08327 / 0.04816	0.09959 / 0.06122	0.12939 / 0.08327	0.17959 / 0.13388
3	0.00000 / 0.00000	0.00000 / 0.00000	0.00082 / 0.00000	0.00000 / 0.00245	0.00408 / 0.00245
4	0.07429 / 0.00245	0.07265 / 0.00571	0.07510 / 0.00490	0.10367 / 0.01633	0.13061 / 0.02449
5	0.00000 / 0.00000	0.00000 / 0.00000	0.00082 / 0.00000	0.01224 / 0.00082	0.01959 / 0.00163
6	0.00000 / 0.00082	0.00245 / 0.01633	0.01633 / 0.03429	0.02204 / 0.05878	0.02449 / 0.10367
7	0.00000 / 0.00000	0.00163 / 0.00082	0.00490 / 0.00163	0.01224 / 0.00735	0.02041 / 0.00816
8	0.00245 / 0.00163	0.00327 / 0.00327	0.01306 / 0.01143	0.01388 / 0.02041	0.03429 / 0.03592
9	0.00000 / 0.00000	0.00000 / 0.00000	0.00163 / 0.00000	0.00245 / 0.00163	0.00327 / 0.00163
10	0.00000 / 0.00000	0.00082 / 0.00082	0.00898 / 0.00163	0.01388 / 0.00490	0.02694 / 0.00898
11	0.00000 / 0.00000	0.00163 / 0.00327	0.00816 / 0.00490	0.01959 / 0.00816	0.02612 / 0.01959
12	0.00000 / 0.00000	0.00163 / 0.00000	0.00245 / 0.00000	0.00980 / 0.00163	0.01551 / 0.00490
13	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000
14	0.06612 / 0.05143	0.07020 / 0.05796	0.07510 / 0.06449	0.07918 / 0.07102	0.09796 / 0.07510
15	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00490 / 0.00163
16	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00245 / 0.00163	0.00653 / 0.00571
17	0.00000 / 0.00000	0.00000 / 0.00000	0.00163 / 0.00163	0.00408 / 0.00163	0.01306 / 0.00653
18	0.00000 / 0.00000	0.00082 / 0.00000	0.00735 / 0.00000	0.02367 / 0.00163	0.05143 / 0.01714
19	0.00000 / 0.00000	0.00000 / 0.00000	0.00653 / 0.00653	0.00816 / 0.01224	0.01633 / 0.02367
20	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00163 / 0.00082	0.00735 / 0.00408
21	0.31673 / 0.24000	0.34612 / 0.24735	0.42367 / 0.26939	0.49959 / 0.31510	0.61878 / 0.36163
22	0.00408 / 0.00000	0.01959 / 0.00082	0.04082 / 0.00490	0.07673 / 0.01224	0.10286 / 0.01878
23	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000
24	0.00000 / 0.00000	0.00653 / 0.00000	0.01633 / 0.00898	0.02041 / 0.01469	0.03429 / 0.02857
25	0.00000 / 0.00000	0.00000 / 0.00000	0.00163 / 0.00000	0.00000 / 0.00163	0.00245 / 0.00163
26	0.00000 / 0.00245	0.00245 / 0.00163	0.00653 / 0.00480	0.01633 / 0.00980	0.01959 / 0.00980
27	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00163 / 0.00000
28	0.00000 / 0.00000	0.00000 / 0.00000	0.00980 / 0.00000	0.03673 / 0.00490	0.09959 / 0.01878
29	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000
30	0.13061 / 0.39673	0.24327 / 0.39918	0.31184 / 0.40000	0.35184 / 0.40000	0.36327 / 0.40000
31	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00408 / 0.00327	0.00490 / 0.00245
32	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00163	0.00163 / 0.00082
33	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000
34	0.00000 / 0.00000	0.00000 / 0.00082	0.00000 / 0.00571	0.02449 / 0.01388	0.05714 / 0.03265
35	0.00000 / 0.01633	0.00163 / 0.00327	0.03020 / 0.02776	0.11184 / 0.11918	0.23673 / 0.20122
36	0.00163 / 0.00000	0.01143 / 0.02286	0.01959 / 0.02449	0.02204 / 0.02531	0.03429 / 0.02776
37	0.00000 / 0.00000	0.00735 / 0.00000	0.01143 / 0.00408	0.01714 / 0.00408	0.02694 / 0.01143
38	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00245	0.00163 / 0.00653	0.01143 / 0.01714
39	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00082	0.00245 / 0.00163	0.00408 / 0.00245
40	0.00000 / 0.06857	0.00245 / 0.00000	0.00571 / 0.00082	0.01469 / 0.00735	0.04245 / 0.01714
41	0.07429 / 0.00000	0.07102 / 0.07020	0.07510 / 0.06612	0.08816 / 0.06204	0.11020 / 0.07020
42	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00082
43	0.00000 / 0.00000	0.00000 / 0.00000	0.00735 / 0.00163	0.00980 / 0.00816	0.01306 / 0.00980
44	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000
45	0.00082 / 0.00000	0.00980 / 0.00000	0.03510 / 0.00000	0.04898 / 0.00735	0.07837 / 0.02122
46	0.00000 / 0.00000	0.01959 / 0.00000	0.03755 / 0.00531	0.04980 / 0.01469	0.07673 / 0.02612
47	0.00000 / 0.00000	0.00000 / 0.00163	0.00000 / 0.00653	0.00327 / 0.03592	0.01224 / 0.07102
48	0.00000 / 0.00000	0.00327 / 0.00163	0.00327 / 0.00327	0.01224 / 0.00980	0.02286 / 0.00653
49	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00163	0.00327 / 0.00082
50	0.00653 / 0.00000	0.02531 / 0.00327	0.04898 / 0.00898	0.07184 / 0.00816	0.09714 / 0.02286

APPENDIX C

Results for distance $D2$

The results for padding-first PPI using distance $D2$ are given below.

Difficulty of impersonation	Intermediate	Hard	Easy
	Padding-first PPI D2 ———	Padding-first PPI D2 - - -	Padding-first PPI D2
	Attack-first PPI ———	Attack-first PPI - - -	Attack-first PPI

Figure C.1. $D2$: Comparison of average accuracy of padding identified

Table C.1. $D2$: Average number of padding identified during training

Difficulty of impersonation	Padding-first PPI $D2$	Attack-first PPI
Hard	34.12	11.34
Intermediate	4.58	7.3
Easy	1	5.38

Figure C.2. *D2*: Average number of attacks and padding identified - intermediate

Figure C.3. *D2*: Average number of attacks and padding identified - hard

Figure C.4. *D2*: Average number of attacks and padding identified - easy

Figure C.5. *D2*: HMM: scores for attack length $|A| = 10$ using padding-first PPI

Figure C.6. *D2*: HMM: scores for attack length $|A| = 30$ using padding-first PPI

Figure C.7. *D2*: HMM: scores for attack length $|A| = 50$ using padding-first PPI

Figure C.8. *D2*: HMM: ROC comparison for $|A| = 10$

Figure C.9. *D2*: HMM: ROC comparison for $|A| = 30$

Figure C.10. *D2*: HMM: ROC comparison for $|A| = 50$

Figure C.11. *D2*: HMM: AUC comparison

Figure C.12. *D2*: HMM: AUCp comparison (FPR 5%)

Figure C.13. *D2*: OCNB: scores for attack length $|A| = 10$ using padding-first PPI

Figure C.14. $D2$: OCNB: scores for attack length $|A| = 30$ using padding-first PPI

Figure C.15. $D2$: OCNB: scores for attack length $|A| = 50$ using padding-first PPI

Figure C.16. *D2*: OCNB: ROC comparison for $|A| = 10$

Figure C.17. *D2*: OCNB: ROC comparison for $|A| = 30$

Figure C.18. D_2 : OCNB: ROC comparison for $|A| = 50$

Figure C.19. D_2 : OCNB: AUC comparison

Figure C.20. *D2*: OCNB: AUCp comparison (FPR 5%)

Figure C.21. *D2*: HMM vs OCNB: padding-first PPI – ROC comparison

Table C.2. AUC comparison for HMM results without PPI (normal face) and padding-first PPI *D2* (bold face)

User	$ A = 10$	$ A = 20$	$ A = 30$	$ A = 40$	$ A = 50$
1	0.47253 / 0.46339	0.54527 / 0.53580	0.61294 / 0.60449	0.67918 / 0.67233	0.74433 / 0.74049
2	0.69906 / 0.70167	0.82637 / 0.83282	0.89029 / 0.89339	0.92082 / 0.91812	0.93539 / 0.93302
3	0.72673 / 0.67445	0.82069 / 0.79518	0.88029 / 0.87143	0.91384 / 0.90914	0.93535 / 0.93057
4	0.59527 / 0.58882	0.67929 / 0.67198	0.76090 / 0.75437	0.80571 / 0.79918	0.84110 / 0.83671
5	0.70722 / 0.60820	0.81300 / 0.76980	0.86008 / 0.84024	0.86718 / 0.84620	0.88033 / 0.86469
6	0.79580 / 0.77869	0.87016 / 0.88776	0.90927 / 0.91502	0.93286 / 0.93122	0.95531 / 0.95347
7	0.48163 / 0.46384	0.56722 / 0.54563	0.64527 / 0.62420	0.71853 / 0.69359	0.77347 / 0.75729
8	0.82498 / 0.82547	0.85939 / 0.85873	0.89384 / 0.89082	0.91673 / 0.91465	0.93559 / 0.93355
9	0.28616 / 0.26245	0.44735 / 0.42384	0.58482 / 0.56682	0.69118 / 0.67559	0.76845 / 0.75731
10	0.70367 / 0.71596	0.73718 / 0.74682	0.78424 / 0.78931	0.84339 / 0.84408	0.89163 / 0.89057
11	0.65645 / 0.63510	0.74359 / 0.72576	0.80827 / 0.79016	0.85265 / 0.84200	0.87737 / 0.86931
12	0.56767 / 0.56616	0.63024 / 0.62678	0.68012 / 0.67690	0.72208 / 0.71657	0.74376 / 0.73600
13	0.60482 / 0.58865	0.68400 / 0.67327	0.74690 / 0.73563	0.78884 / 0.77971	0.81890 / 0.81465
14	0.36710 / 0.33547	0.46314 / 0.43167	0.56147 / 0.53380	0.66780 / 0.64678	0.76935 / 0.75069
15	0.41894 / 0.39694	0.55359 / 0.54194	0.63482 / 0.62384	0.69367 / 0.68322	0.72653 / 0.72008
16	0.51151 / 0.50135	0.57884 / 0.57137	0.63955 / 0.63318	0.69184 / 0.68702	0.73363 / 0.73069
17	0.49592 / 0.49106	0.57388 / 0.56969	0.65163 / 0.64441	0.71494 / 0.70731	0.76718 / 0.76184
18	0.75612 / 0.75196	0.80433 / 0.80269	0.82478 / 0.82657	0.85829 / 0.86020	0.88029 / 0.88118
19	0.55012 / 0.55486	0.68641 / 0.69184	0.78106 / 0.78453	0.82751 / 0.82429	0.85192 / 0.85012
20	0.61143 / 0.60404	0.67563 / 0.67061	0.72147 / 0.71776	0.75947 / 0.75659	0.78429 / 0.78090
21	0.78776 / 0.73694	0.90763 / 0.87576	0.92690 / 0.92273	0.93755 / 0.93649	0.94400 / 0.94371
22	0.67469 / 0.62188	0.77192 / 0.73429	0.82049 / 0.79706	0.85082 / 0.82927	0.86396 / 0.84596
23	0.32506 / 0.29624	0.43990 / 0.42310	0.51653 / 0.50441	0.57608 / 0.57069	0.63384 / 0.63149
24	0.56527 / 0.55208	0.69102 / 0.67771	0.79437 / 0.78339	0.87237 / 0.86559	0.90853 / 0.90298
25	0.51686 / 0.51106	0.62971 / 0.62102	0.71649 / 0.70976	0.79020 / 0.78531	0.84351 / 0.84012
26	0.66208 / 0.63310	0.75816 / 0.74041	0.83261 / 0.82245	0.87245 / 0.86678	0.89457 / 0.89039
27	0.35935 / 0.34608	0.53363 / 0.50898	0.66612 / 0.64573	0.76306 / 0.74865	0.82692 / 0.81790
28	0.73147 / 0.69139	0.86729 / 0.83984	0.92073 / 0.90922	0.93653 / 0.93002	0.94196 / 0.93661
29	0.51629 / 0.50253	0.63029 / 0.62143	0.73302 / 0.72455	0.79792 / 0.79433	0.83673 / 0.83539
30	1.00000 / 1.00000	1.00000 / 1.00000	1.00000 / 1.00000	1.00000 / 1.00000	1.00000 / 1.00000
31	0.56210 / 0.47288	0.70951 / 0.64653	0.79935 / 0.76449	0.85116 / 0.82837	0.88771 / 0.87049
32	0.78694 / 0.78665	0.78931 / 0.78898	0.79441 / 0.79351	0.80078 / 0.80086	0.80861 / 0.80849
33	0.65543 / 0.65265	0.72388 / 0.72016	0.78633 / 0.78220	0.83693 / 0.83469	0.89041 / 0.88437
34	0.67816 / 0.66200	0.78318 / 0.77065	0.84641 / 0.83731	0.87547 / 0.87237	0.89522 / 0.89363
35	0.45704 / 0.45151	0.68669 / 0.68906	0.84710 / 0.84102	0.94424 / 0.93918	0.98151 / 0.98106
36	0.76331 / 0.75053	0.84527 / 0.83800	0.89429 / 0.89151	0.93020 / 0.92686	0.95282 / 0.95171
37	0.54204 / 0.52939	0.63004 / 0.61253	0.71327 / 0.69641	0.78090 / 0.76731	0.82584 / 0.81261
38	0.66180 / 0.61502	0.77886 / 0.74539	0.83286 / 0.80796	0.86898 / 0.85110	0.88906 / 0.87388
39	0.45853 / 0.45473	0.60890 / 0.60069	0.70339 / 0.69433	0.75608 / 0.74278	0.78571 / 0.77282
40	0.60294 / 0.53955	0.71682 / 0.69237	0.79988 / 0.78269	0.83841 / 0.82780	0.86690 / 0.85837
41	0.66233 / 0.65188	0.72718 / 0.72404	0.78943 / 0.78118	0.83041 / 0.82265	0.85518 / 0.84902
42	0.62329 / 0.57273	0.70006 / 0.68147	0.74678 / 0.74008	0.77776 / 0.77910	0.80257 / 0.81073
43	0.72269 / 0.71967	0.75820 / 0.75645	0.79424 / 0.79049	0.82694 / 0.82257	0.85355 / 0.85102
44	0.62727 / 0.60208	0.69016 / 0.67616	0.74812 / 0.74090	0.78829 / 0.78698	0.81869 / 0.82155
45	0.55522 / 0.54731	0.66949 / 0.66224	0.75724 / 0.75041	0.81514 / 0.80959	0.84735 / 0.84437
46	0.66204 / 0.62502	0.77937 / 0.77718	0.85592 / 0.86653	0.90441 / 0.90996	0.93494 / 0.93351
47	0.58045 / 0.59714	0.70592 / 0.69265	0.82576 / 0.82208	0.90049 / 0.89886	0.94396 / 0.94192
48	0.68894 / 0.67416	0.74082 / 0.72710	0.77510 / 0.76555	0.81249 / 0.80229	0.82659 / 0.82114
49	0.32482 / 0.29943	0.46784 / 0.44682	0.57367 / 0.59371	0.63906 / 0.62637	0.68318 / 0.67351
50	0.50657 / 0.48727	0.62502 / 0.60537	0.71031 / 0.69347	0.75245 / 0.74212	0.78502 / 0.76984

Table C.3. AUC comparison for OCNB results without PPI (normal face) and padding-first PPI $D2$ (bold face)

User	$ A = 10$	$ A = 20$	$ A = 30$	$ A = 40$	$ A = 50$
1	0.39180 / 0.37196	0.42886 / 0.41935	0.46963 / 0.46241	0.50669 / 0.50576	0.55633 / 0.55180
2	0.58671 / 0.60133	0.72710 / 0.74204	0.82110 / 0.80410	0.87902 / 0.85514	0.91367 / 0.89416
3	0.54414 / 0.40143	0.64453 / 0.52635	0.71804 / 0.65147	0.77414 / 0.72845	0.81747 / 0.79037
4	0.55069 / 0.56314	0.58604 / 0.58278	0.62873 / 0.61876	0.66449 / 0.64106	0.69555 / 0.67324
5	0.31080 / 0.17253	0.40378 / 0.24855	0.49510 / 0.34727	0.53986 / 0.40618	0.59437 / 0.46804
6	0.46018 / 0.54992	0.68688 / 0.68118	0.77114 / 0.77067	0.81380 / 0.82898	0.84998 / 0.86243
7	0.36631 / 0.33927	0.39557 / 0.36020	0.42592 / 0.38690	0.45992 / 0.40384	0.49886 / 0.45202
8	0.77849 / 0.78465	0.79690 / 0.80135	0.81478 / 0.81298	0.82984 / 0.82759	0.84394 / 0.84049
9	0.08755 / 0.09412	0.13090 / 0.13208	0.17931 / 0.16171	0.23635 / 0.21820	0.29398 / 0.25922
10	0.68645 / 0.67790	0.70016 / 0.70033	0.71682 / 0.72229	0.73102 / 0.73478	0.75108 / 0.75747
11	0.46392 / 0.44267	0.48847 / 0.46073	0.51757 / 0.47598	0.56216 / 0.52449	0.58416 / 0.53496
12	0.39263 / 0.40614	0.42394 / 0.42690	0.44537 / 0.44386	0.46580 / 0.45396	0.48218 / 0.45590
13	0.48539 / 0.46680	0.53316 / 0.51388	0.57949 / 0.55457	0.61794 / 0.60122	0.65904 / 0.64310
14	0.27931 / 0.24735	0.31910 / 0.27514	0.37118 / 0.31971	0.43335 / 0.38996	0.50788 / 0.46506
15	0.17222 / 0.18731	0.23145 / 0.23422	0.29731 / 0.28784	0.35973 / 0.33980	0.41404 / 0.39257
16	0.39569 / 0.35110	0.46661 / 0.41016	0.51914 / 0.46278	0.56757 / 0.52159	0.60624 / 0.57141
17	0.41604 / 0.43682	0.47406 / 0.48380	0.53743 / 0.53537	0.58329 / 0.57273	0.62461 / 0.61845
18	0.72098 / 0.73980	0.79082 / 0.78951	0.81931 / 0.82106	0.84604 / 0.85188	0.87392 / 0.86812
19	0.20863 / 0.23827	0.27386 / 0.29845	0.34416 / 0.34255	0.38943 / 0.37484	0.44141 / 0.42718
20	0.54045 / 0.52924	0.59861 / 0.57690	0.64253 / 0.61839	0.68649 / 0.66624	0.71404 / 0.69731
21	0.65176 / 0.53963	0.75841 / 0.60939	0.84488 / 0.69608	0.90347 / 0.78204	0.94639 / 0.84853
22	0.22455 / 0.15967	0.28769 / 0.20880	0.33216 / 0.26157	0.37339 / 0.28939	0.41394 / 0.33104
23	0.13800 / 0.12433	0.23747 / 0.20616	0.34712 / 0.29882	0.44518 / 0.39947	0.51533 / 0.48978
24	0.39204 / 0.41308	0.44365 / 0.46718	0.49927 / 0.51400	0.55841 / 0.57931	0.59663 / 0.62149
25	0.39592 / 0.40731	0.45551 / 0.46224	0.50927 / 0.51208	0.57373 / 0.57241	0.62627 / 0.62029
26	0.43053 / 0.32727	0.48971 / 0.38951	0.54708 / 0.45947	0.59753 / 0.52322	0.62631 / 0.55122
27	0.14571 / 0.15686	0.20302 / 0.19865	0.26857 / 0.25751	0.35384 / 0.32745	0.43024 / 0.39245
28	0.11571 / 0.16408	0.17492 / 0.20853	0.26343 / 0.23731	0.36710 / 0.23869	0.43435 / 0.35271
29	0.32955 / 0.31231	0.39922 / 0.36220	0.46749 / 0.42561	0.54078 / 0.48433	0.60945 / 0.55992
30	0.47543 / 0.99657	0.71273 / 1.00000	0.84384 / 1.00000	0.90571 / 1.00000	0.93367 / 1.00000
31	0.24931 / 0.20878	0.32982 / 0.26390	0.41700 / 0.30576	0.48876 / 0.37200	0.55406 / 0.44371
32	0.84555 / 0.84780	0.85722 / 0.85849	0.86359 / 0.86404	0.86882 / 0.86788	0.87131 / 0.86976
33	0.59376 / 0.56947	0.62424 / 0.60694	0.66082 / 0.64257	0.69463 / 0.66863	0.72616 / 0.70120
34	0.25406 / 0.27269	0.35408 / 0.33780	0.45449 / 0.41394	0.51084 / 0.43908	0.55094 / 0.48008
35	0.31878 / 0.31747	0.46327 / 0.46304	0.61514 / 0.56739	0.74829 / 0.68916	0.83890 / 0.79220
36	0.63878 / 0.65094	0.73155 / 0.71192	0.79622 / 0.77608	0.84004 / 0.81355	0.87310 / 0.85506
37	0.40857 / 0.40098	0.44771 / 0.43433	0.48245 / 0.47037	0.52690 / 0.51004	0.56984 / 0.53843
38	0.36318 / 0.32500	0.44973 / 0.37624	0.51545 / 0.42696	0.57563 / 0.49686	0.61404 / 0.53198
39	0.18147 / 0.15355	0.19978 / 0.17594	0.22086 / 0.19804	0.25976 / 0.22800	0.29057 / 0.25008
40	0.31082 / 0.23971	0.38571 / 0.29424	0.45608 / 0.34976	0.51759 / 0.41710	0.57755 / 0.49327
41	0.46555 / 0.45176	0.51424 / 0.48159	0.55653 / 0.51367	0.59159 / 0.54610	0.61755 / 0.56608
42	0.38588 / 0.28182	0.55343 / 0.44627	0.64139 / 0.59373	0.69173 / 0.67073	0.72171 / 0.73273
43	0.66947 / 0.64522	0.67969 / 0.67465	0.68833 / 0.68359	0.69824 / 0.70355	0.70490 / 0.72078
44	0.50137 / 0.44735	0.54551 / 0.50196	0.57773 / 0.54824	0.60771 / 0.59739	0.63637 / 0.65461
45	0.34120 / 0.36455	0.41663 / 0.42157	0.48159 / 0.48320	0.53429 / 0.52792	0.57341 / 0.56498
46	0.50398 / 0.42604	0.61294 / 0.56596	0.69967 / 0.67851	0.75673 / 0.75727	0.81196 / 0.82543
47	0.49816 / 0.46506	0.57555 / 0.53506	0.65365 / 0.62020	0.74335 / 0.72061	0.81967 / 0.79176
48	0.52190 / 0.48861	0.53647 / 0.50876	0.53935 / 0.52178	0.55410 / 0.53388	0.56320 / 0.54888
49	0.11159 / 0.08396	0.13790 / 0.09994	0.17947 / 0.14433	0.21957 / 0.17898	0.25916 / 0.21163
50	0.20076 / 0.17894	0.23582 / 0.20555	0.27535 / 0.24204	0.31796 / 0.27814	0.33696 / 0.29027

Table C.4. AUCp (FPR 5%) comparison for HMM results without PPI (normal face) and padding-first PPI *D2* (bold face)

User	$ A = 10$	$ A = 20$	$ A = 30$	$ A = 40$	$ A = 50$
1	0.05714 / 0.05551	0.08490 / 0.08816	0.16327 / 0.16490	0.21959 / 0.21469	0.22531 / 0.21959
2	0.13388 / 0.11755	0.44653 / 0.44571	0.72082 / 0.71347	0.75265 / 0.75020	0.69143 / 0.67510
3	0.36735 / 0.17143	0.50041 / 0.46204	0.59347 / 0.58857	0.73388 / 0.72490	0.81224 / 0.78939
4	0.30041 / 0.34449	0.32327 / 0.35673	0.35265 / 0.36327	0.38612 / 0.38122	0.40000 / 0.38449
5	0.12653 / 0.02286	0.19592 / 0.06041	0.14857 / 0.08327	0.12000 / 0.03592	0.08816 / 0.02939
6	0.59020 / 0.32980	0.66367 / 0.63347	0.74939 / 0.70776	0.71918 / 0.65551	0.76980 / 0.71020
7	0.01714 / 0.01061	0.08000 / 0.05061	0.12408 / 0.10776	0.16245 / 0.15429	0.17224 / 0.14776
8	0.68000 / 0.67347	0.68980 / 0.69224	0.71837 / 0.72082	0.74449 / 0.74449	0.78122 / 0.77633
9	0.00980 / 0.00735	0.04571 / 0.04000	0.06531 / 0.06449	0.10204 / 0.09143	0.10286 / 0.09469
10	0.62612 / 0.62367	0.64816 / 0.64653	0.66286 / 0.66531	0.66612 / 0.68735	0.67265 / 0.69796
11	0.09959 / 0.08816	0.18857 / 0.17714	0.20735 / 0.18531	0.25796 / 0.23347	0.23265 / 0.21224
12	0.17959 / 0.18286	0.21224 / 0.20816	0.23102 / 0.23265	0.27184 / 0.29551	0.20980 / 0.20735
13	0.14857 / 0.14449	0.19755 / 0.19592	0.22449 / 0.22449	0.25714 / 0.25143	0.20735 / 0.20816
14	0.02204 / 0.00000	0.05633 / 0.00327	0.15184 / 0.06041	0.17796 / 0.16653	0.24000 / 0.21469
15	0.02531 / 0.02204	0.04735 / 0.03837	0.08082 / 0.07020	0.09143 / 0.07592	0.05551 / 0.03918
16	0.21469 / 0.18204	0.24408 / 0.24000	0.30531 / 0.28735	0.32327 / 0.30122	0.30449 / 0.30204
17	0.07510 / 0.03020	0.18939 / 0.11918	0.25959 / 0.23592	0.32082 / 0.30612	0.41061 / 0.39592
18	0.50857 / 0.51429	0.66041 / 0.65306	0.68816 / 0.67918	0.70204 / 0.69469	0.71918 / 0.72000
19	0.13959 / 0.18122	0.15918 / 0.16816	0.17959 / 0.22694	0.17388 / 0.18204	0.11184 / 0.13796
20	0.17714 / 0.13633	0.27837 / 0.23592	0.26041 / 0.24408	0.28735 / 0.27265	0.24816 / 0.22776
21	0.47673 / 0.40327	0.61633 / 0.52980	0.82857 / 0.77633	0.86286 / 0.85388	0.87837 / 0.88000
22	0.08490 / 0.04408	0.12490 / 0.07020	0.16816 / 0.08082	0.18122 / 0.11673	0.13061 / 0.09959
23	0.03837 / 0.00082	0.19592 / 0.15102	0.23592 / 0.21469	0.29551 / 0.28816	0.34286 / 0.32898
24	0.12408 / 0.11265	0.16082 / 0.14367	0.20735 / 0.18857	0.26694 / 0.25796	0.31184 / 0.30286
25	0.04898 / 0.05306	0.14367 / 0.13388	0.20408 / 0.19592	0.31429 / 0.28082	0.37551 / 0.36653
26	0.10449 / 0.11184	0.14694 / 0.12490	0.16490 / 0.13224	0.16163 / 0.14612	0.24571 / 0.24163
27	0.01143 / 0.00571	0.01959 / 0.01959	0.06286 / 0.02286	0.15755 / 0.13388	0.25714 / 0.22857
28	0.16735 / 0.19184	0.23347 / 0.22857	0.23510 / 0.25143	0.23918 / 0.25469	0.30531 / 0.27837
29	0.06041 / 0.07184	0.11429 / 0.12816	0.31837 / 0.31388	0.40898 / 0.40571	0.48163 / 0.47755
30	0.97959 / 0.97959	0.97959 / 0.97959	0.97959 / 0.97959	0.97959 / 0.97959	0.97959 / 0.97959
31	0.11510 / 0.01469	0.19673 / 0.08082	0.19592 / 0.14449	0.24000 / 0.17551	0.29061 / 0.22122
32	0.76408 / 0.76408	0.76408 / 0.76408	0.76408 / 0.76408	0.76408 / 0.76408	0.76245 / 0.76327
33	0.50531 / 0.50286	0.52245 / 0.52163	0.56000 / 0.55837	0.59673 / 0.57878	0.62531 / 0.62041
34	0.21143 / 0.18776	0.22857 / 0.19918	0.25143 / 0.20735	0.18694 / 0.19102	0.19592 / 0.19265
35	0.00408 / 0.00000	0.26122 / 0.25061	0.46531 / 0.42041	0.61633 / 0.56082	0.82367 / 0.79592
36	0.34857 / 0.24163	0.54694 / 0.46694	0.71429 / 0.69714	0.77796 / 0.77714	0.79918 / 0.80571
37	0.10776 / 0.06449	0.11837 / 0.10939	0.14204 / 0.14694	0.22122 / 0.22204	0.27837 / 0.27837
38	0.11837 / 0.06694	0.12735 / 0.09551	0.15918 / 0.13469	0.21959 / 0.16082	0.21388 / 0.15429
39	0.00000 / 0.01633	0.00408 / 0.01878	0.01796 / 0.03184	0.05878 / 0.03020	0.08327 / 0.05959
40	0.02204 / 0.02122	0.05061 / 0.03755	0.07347 / 0.06286	0.09551 / 0.08980	0.11184 / 0.09224
41	0.16980 / 0.10776	0.27592 / 0.21878	0.30939 / 0.26612	0.34776 / 0.30857	0.35755 / 0.35265
42	0.28204 / 0.08980	0.44898 / 0.26041	0.52408 / 0.43020	0.55673 / 0.51510	0.58286 / 0.55510
43	0.55102 / 0.46041	0.56163 / 0.53959	0.57959 / 0.54857	0.58776 / 0.59020	0.59918 / 0.61878
44	0.30204 / 0.15918	0.40000 / 0.32163	0.45714 / 0.42122	0.48980 / 0.49061	0.52000 / 0.52816
45	0.15673 / 0.10694	0.23673 / 0.20571	0.30531 / 0.24898	0.36327 / 0.33714	0.34041 / 0.34286
46	0.22694 / 0.04490	0.47837 / 0.38449	0.61633 / 0.58776	0.67265 / 0.69551	0.70122 / 0.72245
47	0.24408 / 0.24735	0.32898 / 0.33224	0.48571 / 0.49796	0.65878 / 0.65714	0.80816 / 0.81388
48	0.10367 / 0.07755	0.17959 / 0.16245	0.19918 / 0.18204	0.21388 / 0.18367	0.18286 / 0.18449
49	0.03592 / 0.03020	0.06286 / 0.04571	0.06367 / 0.04980	0.07837 / 0.05633	0.08571 / 0.07429
50	0.04327 / 0.03510	0.08163 / 0.06694	0.09959 / 0.09143	0.07673 / 0.06367	0.06367 / 0.05306

Table C.5. AUCp (FPR 5%) comparison for OCNB results without PPI (normal face) and padding-first PPI *D2* (bold face)

User	$ A = 10$	$ A = 20$	$ A = 30$	$ A = 40$	$ A = 50$
1	0.00000 / 0.00000	0.00000 / 0.00000	0.00163 / 0.00163	0.00082 / 0.00327	0.00735 / 0.00980
2	0.07429 / 0.04327	0.08327 / 0.06857	0.09959 / 0.08163	0.12939 / 0.10857	0.17959 / 0.15918
3	0.00000 / 0.00000	0.00000 / 0.00000	0.00082 / 0.00082	0.00000 / 0.00000	0.00408 / 0.00327
4	0.07429 / 0.08000	0.07265 / 0.07265	0.07510 / 0.06612	0.10367 / 0.08816	0.13061 / 0.10857
5	0.00000 / 0.00000	0.00000 / 0.00000	0.00082 / 0.00000	0.01224 / 0.01143	0.01959 / 0.01714
6	0.00000 / 0.00653	0.00245 / 0.02531	0.01633 / 0.03184	0.02204 / 0.03673	0.02449 / 0.05306
7	0.00000 / 0.00000	0.00163 / 0.00163	0.00490 / 0.00408	0.01224 / 0.00653	0.02041 / 0.01143
8	0.00245 / 0.00163	0.00327 / 0.00327	0.01306 / 0.01224	0.01388 / 0.01388	0.03429 / 0.02694
9	0.00000 / 0.00000	0.00000 / 0.00000	0.00163 / 0.00000	0.00245 / 0.00408	0.00327 / 0.00327
10	0.00000 / 0.00000	0.00082 / 0.00163	0.00898 / 0.00898	0.01388 / 0.01633	0.02694 / 0.02939
11	0.00000 / 0.00000	0.00163 / 0.00163	0.00816 / 0.00816	0.01959 / 0.01551	0.02612 / 0.02367
12	0.00000 / 0.00000	0.00163 / 0.00082	0.00245 / 0.00163	0.00980 / 0.00816	0.01551 / 0.01469
13	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000
14	0.06612 / 0.03755	0.07020 / 0.05551	0.07510 / 0.06286	0.07918 / 0.07020	0.09796 / 0.08980
15	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00245	0.00490 / 0.00735
16	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00245 / 0.00245	0.00653 / 0.00816
17	0.00000 / 0.00000	0.00000 / 0.00000	0.00163 / 0.00163	0.00408 / 0.00245	0.01306 / 0.01224
18	0.00000 / 0.00000	0.00082 / 0.00000	0.00735 / 0.00163	0.02367 / 0.01143	0.05143 / 0.03347
19	0.00000 / 0.00000	0.00000 / 0.00000	0.00653 / 0.00490	0.00816 / 0.00816	0.01633 / 0.02694
20	0.00000 / 0.00000	0.00000 / 0.00082	0.00000 / 0.00082	0.00163 / 0.00408	0.00735 / 0.01061
21	0.31673 / 0.24245	0.34612 / 0.25714	0.42367 / 0.28408	0.49959 / 0.34449	0.61878 / 0.41878
22	0.00408 / 0.00490	0.01959 / 0.02286	0.04082 / 0.03102	0.07673 / 0.05224	0.10286 / 0.07102
23	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000
24	0.00000 / 0.00000	0.00653 / 0.00490	0.01633 / 0.01143	0.02041 / 0.02204	0.03429 / 0.04163
25	0.00000 / 0.00000	0.00000 / 0.00000	0.00163 / 0.00000	0.00000 / 0.00163	0.00245 / 0.00571
26	0.00000 / 0.00000	0.00245 / 0.00163	0.00653 / 0.00571	0.01633 / 0.01143	0.01959 / 0.00898
27	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00082	0.00163 / 0.00245
28	0.00000 / 0.00000	0.00000 / 0.00000	0.00980 / 0.00653	0.03673 / 0.01878	0.09959 / 0.04490
29	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000
30	0.13061 / 0.37143	0.24327 / 0.40000	0.31184 / 0.40000	0.35184 / 0.40000	0.36327 / 0.40000
31	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00408 / 0.00490	0.00490 / 0.00816
32	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00163 / 0.00163
33	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000
34	0.00000 / 0.00000	0.00000 / 0.00327	0.00000 / 0.01469	0.02449 / 0.04816	0.05714 / 0.08408
35	0.00000 / 0.00000	0.00163 / 0.00490	0.03020 / 0.03102	0.11184 / 0.08571	0.23673 / 0.18612
36	0.00163 / 0.03837	0.01143 / 0.03592	0.01959 / 0.03429	0.02204 / 0.03918	0.03429 / 0.04980
37	0.00000 / 0.00000	0.00735 / 0.00735	0.01143 / 0.01143	0.01714 / 0.01551	0.02694 / 0.02204
38	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00163 / 0.00163	0.01143 / 0.01143
39	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00245 / 0.00082	0.00408 / 0.00408
40	0.00000 / 0.00000	0.00245 / 0.00163	0.00571 / 0.00816	0.01469 / 0.02122	0.04245 / 0.05714
41	0.07429 / 0.07347	0.07102 / 0.07102	0.07510 / 0.07020	0.08816 / 0.07673	0.11020 / 0.09224
42	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00082
43	0.00000 / 0.00000	0.00000 / 0.00000	0.00735 / 0.00653	0.00980 / 0.00816	0.01306 / 0.01143
44	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00163
45	0.00082 / 0.00000	0.00980 / 0.00163	0.03510 / 0.02449	0.04898 / 0.04163	0.07837 / 0.05061
46	0.00000 / 0.00163	0.01959 / 0.01306	0.03755 / 0.02449	0.04980 / 0.03592	0.07673 / 0.06612
47	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00327 / 0.00327	0.01224 / 0.01224
48	0.00000 / 0.00000	0.00327 / 0.00327	0.00327 / 0.00327	0.01224 / 0.00898	0.02286 / 0.01469
49	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	0.00327 / 0.00245
50	0.00653 / 0.01143	0.02531 / 0.02531	0.04898 / 0.03918	0.07184 / 0.05959	0.09714 / 0.08082