5-2012

Recognizing Patterns in Transmitted Signals for Identification Purposes

Baha' A. Alsaify
University of Arkansas, Fayetteville

RECOGNIZING PATTERNS IN TRANSMITTED SIGNALS
FOR IDENTIFICATION PURPOSES

RECOGNIZING PATTERNS IN TRANSMITTED SIGNALS FOR IDENTIFICATION PURPOSES

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Computer Engineering

By

Baha' A. Alsaify
Jordan University of Science and Technology
Bachelor of Science in Computer Engineering, 2007
University of Arkansas
Master of Science in Computer Engineering, 2009

May 2012
University of Arkansas

Abstract

The ability to identify and authenticate entities in cyberspace such as users, computers, cell phones, smart cards, and radio frequency identification (RFID) tags is usually accomplished by having the entity demonstrate knowledge of a secret key. When the entity is portable and physically accessible, like an RFID tag, it can be difficult to secure given the memory, processing, and economic constraints. This work proposes to use unique patterns in the transmitted signals caused by manufacturing differences to identify and authenticate a wireless device such as an RFID tag. Both manufacturer identification and tag identification are performed on a population of 300 tags from three different manufacturers. A methodology to select features for identifying signals with high accuracy is developed and applied to passive RFID tags. The classifier algorithms K-Nearest Neighbors, Parzen Windows, and Support Vector Machines are investigated. The tag's manufacturer can be identified with 99.93% true positive rate. An individual tag is identified with 99.8% accuracy, which is better than previously published work. Using a Hidden Markov Model with framed timing and power data, the tag manufacturer can be identified with 97.37% accuracy and has a compact representation. An authentication system based on unique features of the signals is proposed assuming that the readers that interrogate the tags may be compromised by a malicious adversary. For RFID tags, a set of timing-only features can provide an accuracy of 97.22%, which is better than previously published work, is easier to measure, and appears to be more stable than power features.

This dissertation is approved for
recommendation to the
Graduate Council

Dissertation Director:

Dale R. Thompson, Ph.D.

Dissertation Committee:

Gordon Beavers, Ph.D.

Jia Di, Ph.D.

Roy McCann, Ph.D.

Dissertation Duplication Release

I hereby authorize the University of Arkansas Libraries to duplicate this dissertation when needed for research and/or scholarship.

Agreed ———————————————————-
Baha' A. Alsaify

Refused ———————————————————

Acknowledgments

This dissertation would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study. First and foremost, I would like to express my utmost gratitude to my advisor, Dr. Dale R. Thompson, for his guidance, caring, patience, and for providing me with the confidence to explore and research that I find interesting. Dr. Thompson, despite your numerous academic and professional commitments, you helped me to finish my master's thesis and continued your support throughout my doctoral studies.

Dr. Jia Di, your suggestions, feedbacks, and positive attitude made it possible for this work to be complete.

Dr. Gordon Beavers, your always open door policy, your patience, and your valuable inputs made all the difference for the quality of this work.

Dr. Roy McCann, thank you very much for agreeing to be part of this work. Your comments and suggestions made it possible to complete this work.

Dr. Senthilkumar, you helped me at the beginning of my doctoral studies. You were always there whenever I asked for your assistance and inputs. Thank you very much for your assistance throughout the years.

Last, but not least, I would like to thank my family for always being there for me and for believing in me. Their patience and sacrifice will never be forgotten.

Baha' A. Alsaify

Dedication

Dedicated to my late father.

Table of Contents

List of Figures

List of Tables

List of Equations

Chapter 1

Introduction

Cyber security refers to security as it applies to computers and networks. It refers to the security of any entity that communicates. Two main processes are key to cyber security: authentication and identification. Authentication is the process of confirming or denying that a claimed identity is correct by comparing the credentials of user/object with those previously proven, stored, and associated with the identity being claimed. On the other hand, identification is the process of discovering the true identity of an item from the entire collection of similar items, which requires a one-to-many matching.

Classic authentication and identification techniques depend on using a key or a password to identify or authenticate a certain entity, which can be a user, a computer, a smart card, etc. The entity will transmit its key whenever it is required to verify its identity. This approach has many assumptions. First, the entity is assumed to be secure enough so that it can hold the secret key without disclosing it to an unauthorized party. Second, the entity is assumed to be capable of performing complex encryption and decryption operations based on an available key. These encryption and decryption operations require the entity to have a large memory, a power supply capable of supporting these complex operations, and a computational unit capable of performing the encryption and decryption operations under predefined timing constraints.

Some entities lack the previous capabilities including RFID, smart cards, wireless sensor networks, etc. Therefore, researchers are looking for ways to authenticate and identify objects other than relying on their capabilities to hold secrets. Authentication and identification techniques are moving away from what an object "has" toward what an object "is".

1.1 Motivation

We can identify a specific object based on what it "has", which means that the object must maintain a secret so that it can be accurately identified among a collection of similar objects. The secret key can be duplicated which will allow an unauthorized access to physical or logical resources. To prevent this threat, many systems are moving away from secret key authentication. Instead they authenticate and identify objects based on what they "are". The same transition has already happened with human authentication, which relies on the anatomical or behavioral characteristics [51] that makes us different from each other such as fingerprints [62][15], Iris scans [20], voice recognition [44], face recognition [11], and palm prints [32].

The motivation of this work is to build a fingerprint for objects based on their physical characteristics so that they can be identified by what they "are". As a representative of the objects to identify and authenticate, we use passive ultra-high frequency (UHF) radio-frequency identification (RFID) tags. RFID tags communicate with the world that surrounds it using radio signals, which are easy to capture and inexpensive to analyze. RFID tags are challenging to use because of their constrained resources. The proposed identification technique for RFID described in this work can be applied to other objects with wireless communication capabilities.

There are several challenges of identifying RFID tag signals. Tags need to be as inexpensive as possible so that they can be used in mass scale [33] [60]. Retail stores need to have an inexpensive tag so that the added tag's overhead will not affect the price of the goods they are selling. Passive UHF RFID tags have no power supply; it gathers its required energy by harvesting the energy of the carrier waves sent by the RFID reader. RFID tags have little memory and few processing capabilities, which makes adding any encryption technique to the tag a hard, if not an impossible, task given the current power and cost limitations.

1.2 Objectives

The objectives of this work are listed below:

1. Create a methodology for identifying signals with high probability that is scalable and can be deployed efficiently;
2. Investigate and evaluate a set of performance metrics;
3. Investigate classification and pattern recognition techniques for the purpose of identifying both the tag manufacturer and individual tags;
4. Investigate hidden Markov model and compare the performance with the results obtained using classical classification techniques; and
5. Create a system that identifies and authenticates RFID tags even with the presence of malicious or hacked readers.

1.3 Background

In this section, background information for this work is presented. The background includes information about RFID technology, its usage, and operations. In addition, the theory and usage of a set of pattern recognition techniques are provided.

1.3.1 Radio Frequency Identification

Radio frequency identification (RFID) is a wireless technology that is used to identify objects. There are three main parts of any RFID system, a reader, a server, and a tag. The reader (or interrogator) sends commands to tags. The reader is also responsible for receiving the

responses from the tags, demodulating, and decoding the tag's transmission, and presenting the tags response to the user in a way that will be easy to be understood. A low-power RFID reader is shown in Figure 1.1. The second part of any RFID system is the server. The server is responsible of collecting tag readings from the field via the reader(s), organizing the collected reads, eliminating redundant reads if necessary, and presenting the data to the system user in a concise and easy way to understand.

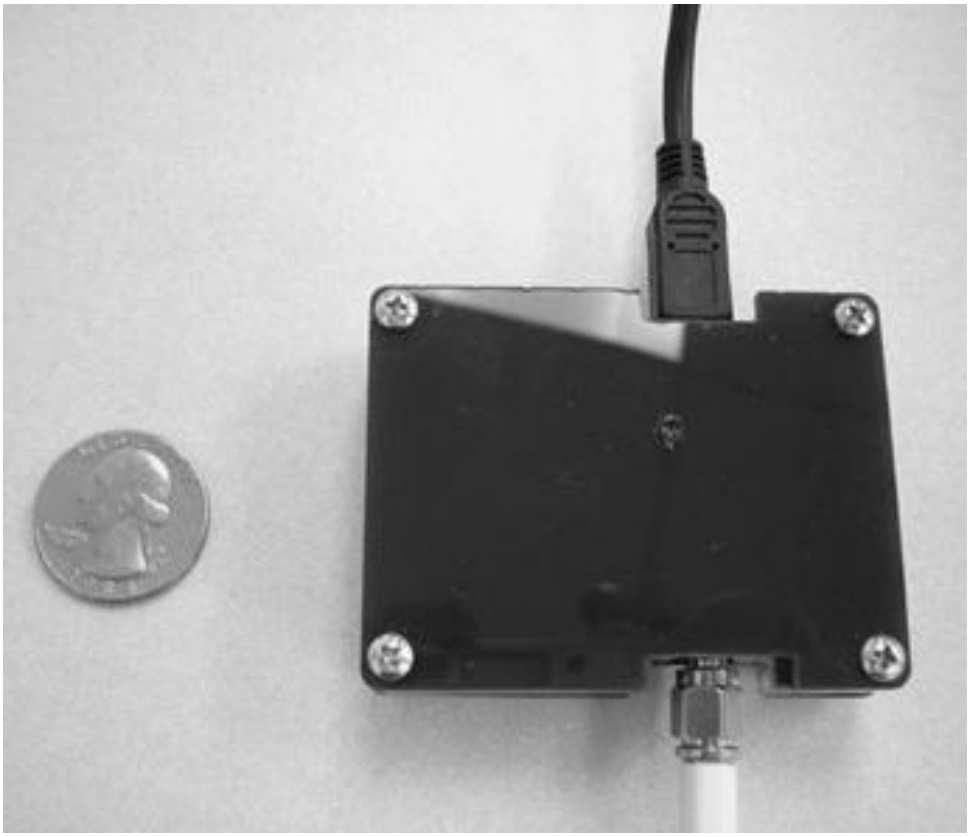

Figure 1.1: RFID Reader

RFID tags are responsible for receiving the commands from the reader, decoding them, and responding with the appropriate data. An example of an RFID tag is shown in Figure 1.2. Each RFID tag is composed of several components such as an antenna, a power supply, a memory chip, and a microprocessor chip [43]. An example of an RFID system is provided in Figure 1.3.

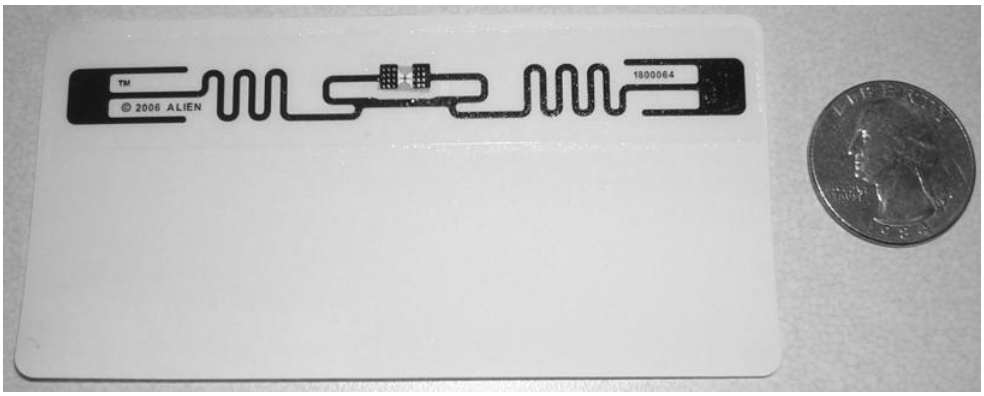

Figure 1.2: RFID Tag

RFID tags are classified based on their power source into three categories: active, semi-active, and passive tags. Active RFID tags have a battery onboard, which means that they can communicate over longer distances than other types and they are capable of performing more complex operations. On the other hand, having a battery means that the tags will consume more space and that they will be more expensive. Semi-passive tags have a battery onboard but it is only used for running the onboard logical chip and sensors, not for communication purposes, which means that the battery lasts longer than active tags [30][16]. The last category of RFID tags based on their power supply is called passive tags. Passive RFID tags do not have a power supply onboard but they rely on the energy contained in the reader's carrier-wave. The process in which the passive tag gets its energy is called energy harvesting [29]. Because passive tags do not have a dedicated power supply, they are small in size and inexpensive. Therefore, they are widely used in many applications such as animal tracking, inventory control, and access control.

RFID tags can be also categorized based on the frequency bands they are using. Several frequency bands can be used in RFID communication including low-frequency (LF), high-frequency (HF), and ultra-high-frequency (UHF). The focus of this work is on passive UHF RFID tags that operate between 860 MHz and 960 MHz.

In order to control the communication data stream between the reader and the tag, a

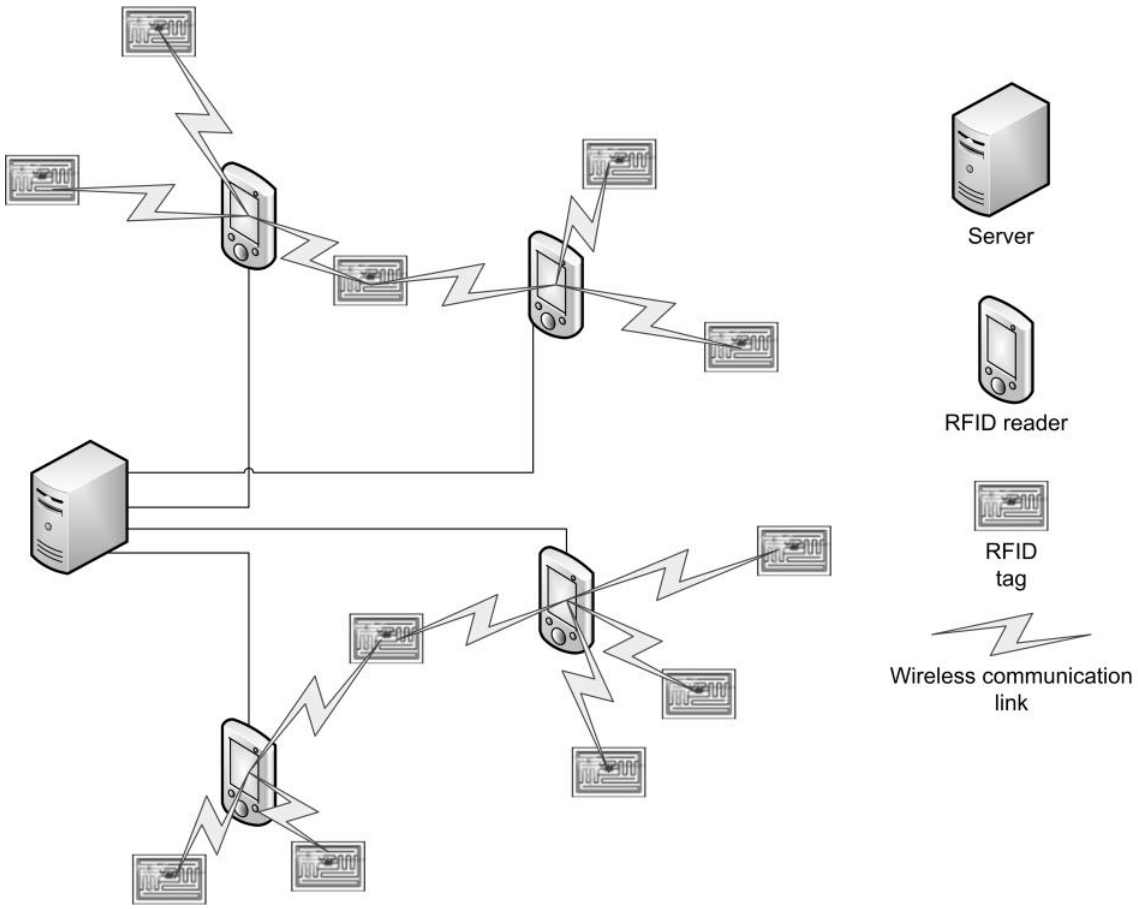

Figure 1.3: RFID System

standard was introduced. EPCglobal is a non-profit organization that developed a standard called EPCglobal Class-1 Gen-2 (C1G2) [1], which "defines the physical and logical requirements for a passive-backscatter, interrogator-talks-first (ITF), radio-frequency identification (RFID) system operating in the 860 MHz - 960 MHz frequency range". In EPCglobal C1G2, the reader and the tag take turns in communicating with each other as shown in Figure 1.4. The reader will select an individual tag to query based on an anti-collision protocol [14][67] that is similar to the slotted Aloha wireless protocol [3][5]. Once a tag is selected, it sends its handle (a 16-bit random number (RN16)) to the reader. This handle is used by the reader for any future communication with the tag.

Because RFID technology does not need line-of-sight to operate alongside its inexpensive

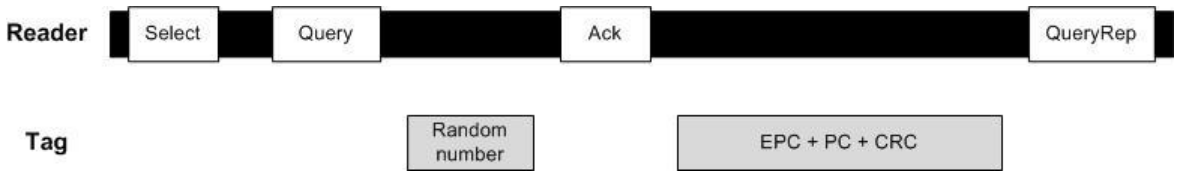

Figure 1.4: EPC Class-1 Gen-2

price, made it a perfect candidate for many inventory and supply chain applications. RFID technology is currently being used by large companies such as Walmart [12] to keep track of inventory and is included in some goods in its retail stores. RFID is also being used in many applications other than inventory control, such as:

- Contactless payment systems;
- Access control;
- Tracking assets in hospitals;
- Animal tracking and monitoring;
- Building management;
- Library services;
- Municipal solid waste management; and
- Enterprize feedback control.

1.3.2 Classification

Classification refers to a set of methods from which an object is mapped to a cluster of similar objects. Classification is used to solve problems such as robotic vision, fingerprint matching, voice recognition, and medical decision making. Classification techniques work by creating a rule based on previously known objects. This rule is often called classifier.

Classification techniques can be divided into two main groups, rule-based and learning-based [69]. Rule-based classifiers determine their final decision by comparing an overall similarity score and a threshold value. Rule-based classifiers are good for capturing domain knowledge and are applicable to many scenarios. Decision tree classifiers such as [52] [64] [57] are categorized under rule-based classifiers. On the other hand, learning-based classifiers depend on statistically creating the decision rule(s) based on the sample data. Users of the learning-based approach need to explicitly provide classifier examples rather than just rules. Support vector machines [18], k nearest neighbors (k-NN) [21], and Parzen windows [45] are techniques that use the learning-based approach.

1.3.2.1 K Nearest Neighbors

K nearest neighbors (k-NN) is a non-probabilistic, non-parametric classification method that classifies an unlabeled test sample based on its similarity with samples in the training set [61][21]. k-NN is among the simplest of all classification and machine learning algorithms.

There are three main components that are needed so that k-NN can work correctly. First, k-NN requires an integer k that represents how many close neighbors the algorithm should use to formulate its final decision. Second, k-NN requires a distance metric that measures the difference in a test sample and training sample. Finally, k-NN requires a set of training samples. Several distance metrics can be used in k-NN such as the Euclidian distance, Manhattan distance, and Hamming distance.

k-NN makes several assumptions. It assumes that the training and testing data is already in the feature space. It assumes that each of the training samples consists of a feature vector and a class label associated with it. In the simplest case, this class label can be either positive or negative (binary classifier), but k-NN also works with multiple class labels (multi-class classifier).

The most common k value in k-NN is 1. When 1 is used as k, the classifier is often called $1-NN$. What $1-NN$ means is that we need to find the closest training instance to the test sample. Other common values are 3 and 5. If the value of k was even, then a draw between two or more class labels may occur. In order to prevent that possible draw, weights are added to the final decision of k-NN as shown in Equation 1.1, where d_1 is the distance to the closest training sample, d_k is the distance to the k^{th} closest training sample, and d_j is the distance to the j^{th} closest training sample. The closest training sample will have the highest weight and the k^{th} training sample will have the lowest weight.

$$w_j = \begin{cases} \frac{d_k - d_j}{d_k - d_1}, & d_k \neq d_1 \\ 1, & d_k = d_1 \end{cases} \tag{1.1}$$

k-NN is a simple approach that is easy to understand and use. The main drawback of using k-NN is that it needs to compute the distance between the test sample and all of the training samples and then select the closest k training samples to formulate its decision. Having all training samples is the main challenge. If the number of training samples is moderate, then k-NN will have a fast computation time and it will not require a large memory, but it will have a larger error rate. On the other hand, if we use a large training data set, the decision error will be minimal but the time and memory requirement will grow large, which makes k-NN unusable for a large number of online and offline applications.

1.3.2.2 Parzen Windows

Parzen windows is a technique that was invented by Emanuel Parzen [45] [6]. The goal of Parzen windows is to estimate the probability density function (PDF) from the training data-set. The reason behind using the term window in the process name is due to the argument

that "if we want to estimate the value of PDF at observation x, we can place a window function at x and determine how many training observations fall within the window, or rather, what the contribution of each training observation to the window" [6]. The final score of the Parzen classifier is the sum total of the contributions from the training observations to the window. The Parzen window estimation formula is shown in Equations 1.2, where n is the number of training observations, h is the smoothing parameter, X_i is the i^{th} training observations, x is the data point to be classified, and K is the kernel function.

$$f(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right) \tag{1.2}$$

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{(h\sqrt{2\pi})^d} exp\left(-\frac{1}{2}\left(\frac{x - x_i}{h}\right)^2\right) \tag{1.3}$$

There are several choices for kernel functions. Gaussian, uniform, triangular, biweight, and Epanechnikov are among the popular kernel functions. Equation 1.3 shows an example of a Parzen windows estimator when Gaussian is being used as a kernel function. Trial-and-error are the best way to determine the best kernel function to use since it differs based on the problem. Other than the choice of the kernel function, the choice of the smoothing parameter h (window width) is very important. Parzen windows evolved from histograms, in which the smoothing parameter is used to determine the width of the bins. If h is selected to be large, it will over-smooth the result, which means that some of the data will be hidden. On the other hand, using a small value for h will under-smooth the results, which means that any noise in the data will have an effect on the end classification result. The previous argument demonstrates the importance of selecting the appropriate value of h since it directly affects

the end result.

1.3.2.3 Support Vector Machines

Support vector machines (SVM) [59][18][13] is a classification technique introduced by Vapnik in 1995. The main idea behind SVM is that it maps the input vectors into some high dimensional feature space Z through some non-linear mapping chosen a priori. In this space, a linear decision surface (hyperplane) is constructed with special properties that ensure high generalization ability of the network. Given a training set of N data points $\{y_k, x_k\}_{k=1}^{N}$, where $x_k \in \mathbb{R}^n$ is the k^{th} data training sample, $y_k \in \mathbb{R}$ is the k_{th} output class label, the SVM formula has a form similar to that which is shown in Equation 1.4, where α_k is a positive real constant, b is a real constant, and $\Psi(x, x_k)$ depends on the type of SVM we are using (linear $\left(x_k^T x\right)$, polynomial of degree d $\left(x_k^T x + 1\right)^d$, RBF SVM $\left(exp\left\{-\left\|x - x_k\right\|_2^2 / \sigma^2\right\}\right)$, or two layer neural svm $\left(tanh\left[k x_k^T x + \theta\right]\right)$).

$$f(x) = sign\left[\sum_{k=1}^{N} \alpha_k y_k \Psi(x, x_k) + b\right] \tag{1.4}$$

Using Equation 1.4, SVM generates a set of hyperplanes that separates the training set based on its labels. The main problem that SVM tries to solve is finding an optimal distance between the hyperplane and every other class. This distance is called margins and the problem SVM tries to solve is called margin maximization.

1.3.3 Hidden Markov Models

Hidden Markov model (HMM) [54] is a technique that is used for pattern recognition. In HMM if there exist i number of classes to classify the test sample into, then i models must

be built. Figure 1.5 shows conceptually what each of the models should look like.

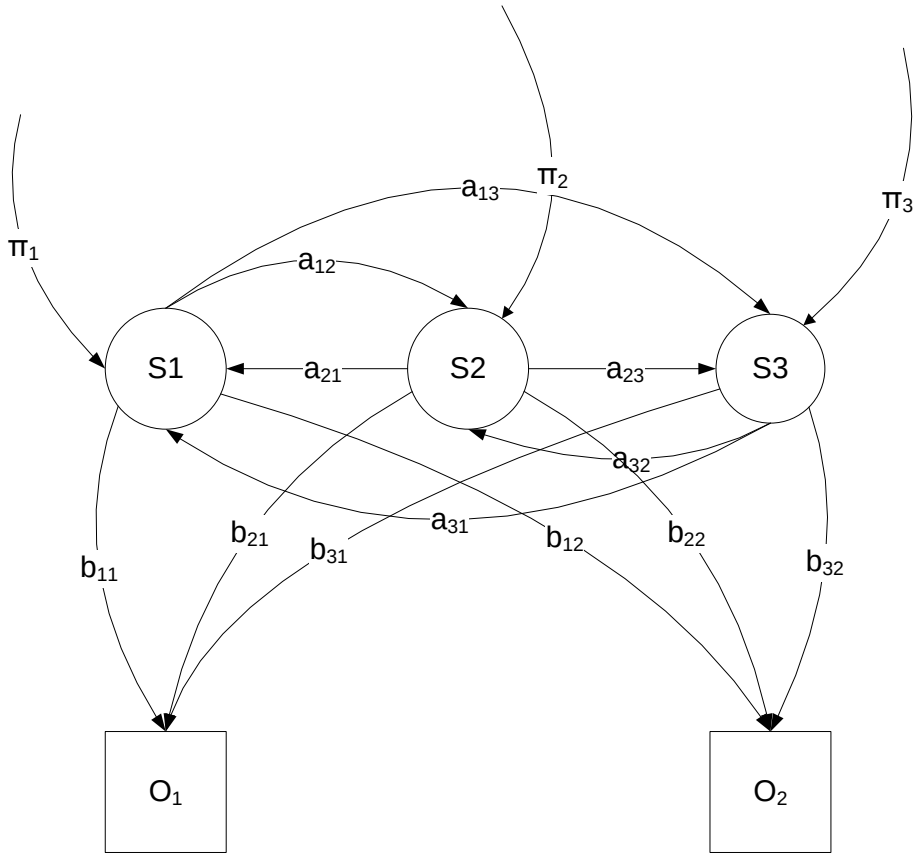

Figure 1.5: Hidden Markov Model Parameters

In order to use HMM efficiently, the system to be modeled must have its output quantized into an observation stream. Observations can be anything that results from the system when it runs. Observations can be the result of a series of coin tosses (head or tails); it can be the result of throwing a rolling dice for several times (1-6); it can be a voltage-time waveform that results from processing some speech patterns, etc.

As Figure 1.5 shows, any HMM is built using a set of probabilities, a set of states, and a set of observations. The states are the engine that runs the model. Each state will generate a possible observation. The model keeps running until the appropriate number of observations are generated. Once the observations are generated, the resultant observation stream is compared with the test sample stream to determine if they are close. There are 5

elements for any HMM:

1. N, the number of states in the model. In general, the states are interconnected in a way that allows any state to reach any other state;
2. M, the number of distinct observation symbols per state;
3. The state transition probability distribution $A = \left\{a_i^j\right\}$, where a_i^j is the probability that the current state is i and the next state for the system to be in is j;
4. The observation symbol probability distribution in state j, $B = \{b_j(k)\}$, where $b_j(k)$ is the probability that the k^{th} observation will appear at state j; and
5. The initial state distribution $\pi = \{\pi_i\}$, where π_i is the probability that the model will initially start at state i.

To apply HMM, three main problems are solved:

1. Given the observation sequence $O = O_1O_2O_3...O_T$, and a model $\lambda = P(O|\lambda)$, what is the probability that the observation sequence was generated using the model?;
2. Given the observation sequence $O = O_1O_2O_3...O_T$, and a model λ, how to select a corresponding state sequence $Q = q_1q_2q_3...q_T$, which best explains the observation sequence?; and
3. How to adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$?

For more information regarding the mathematical models that are responsible for solving the previous problems please refer to [54][7][63].

One of the issues that arises when using HMM is how to initialize all of the model parameters. To solve this problem, researchers often initialize the model parameters using

a uniform distribution. All the probabilities will be equivalent to each other while some rules hold in both the initialization and optimization steps. The first rule that needs to hold true is that the sum of probabilities for the model moving to any state j from state i is equal to 1 $\left(\sum_{j=1}^{N} a_i^j = 1\right)$. The second rule that must hold true is that the sum of initial probability π must be equal to 1 $\left(\sum_{i=1}^{N} \pi_i = 1\right)$. Finally, the summation of the all the possible observations at state i must be also equal to 1 $\left(\sum_{j=1}^{T} b_i(j)\right)$, where T is the number of distinct observations that can be generated at state i.

To classify a new observation sequence, a HMM for each of the possible classes is generated $\lambda_1\lambda_2...\lambda_l$, where l is the number of classes (patterns). Each of the available models describes only one of the classes. When we evaluate the new observation sequences O, $P(O|\lambda_1)P(O|\lambda_2)P(O|\lambda_3)...P(O|\lambda_l)$ must be computed. The model with the maximum probability i assumed to be the one that can generate the observation sequence O and thus O belongs to that model.

1.4 Dissertation Organization

The rest of this dissertation is organized as follows. First, related work in the area of identifying and authenticating RFID tags is presented in Chapter 2. In Chapter 3, the feature extraction, feature selection, and the classification experiments are presented. The work on hidden Markov models is provided in Chapter 4. Signals authentication using the fingerprints is presented in Chapter 5. Finally, the conclusions and the planned future work are presented in Chapter 6.

Chapter 2

Related Work

Fingerprinting is the process of creating a unique identifier that can be used to accurately identify and/or authenticate an object. The fingerprinting process can be applied to any object as long as the result can uniquely identify that object. For example, researchers introduced the idea of fingerprinting an official paper (or any paper) by measuring the three dimensional paper's surface using commodity scanners, and without modifying the document in any way [17]. The authors of [17] demonstrated that it is possible to generate a fingerprint of a document using a flatbed scanner coupled with the appropriate software. The proposed system can be used like a digital signature to authenticate official and important documents.

Bluetooth networks can also have a fingerprint. In [26], an intrusion detection system that uses radio frequency fingerprints (RFF) for profiling, Hotelling's T^2 for classification, and a decision filter is introduced. The transient portion of the exchanged signal is used to identify a bluetooth transceiver. The reason behind using the transient response is because it depends on the hardware of the transceiver, which makes it hard to replicate. Features such as frequency, amplitude, and phase are extracted. For each transceiver, multiple fingerprints are created and the outliers are removed. K-means clustering technique of [39] can be used to select a representative set of the enrolled fingerprints. One of the issues this work had to deal with is the presence of noise and interference, which may lead to errors in the final decision making process. To work around the errors, a set of fingerprints are independently classified and a decision filter is applied to determine the final decision based on multiple observed fingerprints.

Additional work that aims to enhance bluetooth security through fingerprints is pre-

sented in [46]. The purpose is to allow bluetooth devices manufacturers to implement a user friendly, and efficient intrusion detection and prevention systems. The radio frequency (RF) fingerprint depends on the transient phase of the bluetooth transceiver signal, which lasts an interval between 2 - 10 ms. The proposed system is a local system that is designed to work within a single company in a single building that includes a bluetooth enabled server at a fixed location. The server has the role of saving all the fingerprints of the legitimize bluetooth devices. Bluetooth devices have the option of not using the system; but if they want to use the system, all communication must go through the server. The system is efficient since it requires small physical size (the server), all intensive operations are taking place at the server, and no modifications are required on the bluetooth devices.

Another form of fingerprinting physical devices is presented in [34]. In this work, ethernet cards are fingerprinted by exploiting small deviations in the card's hardware that appear as differences in the clock skews. The technique does not require any modification of the card's hardware, and can be employed without the knowledge and/or cooperation of the card itself. TCP timestamps, periodic activities, or ICMP timestamps are used to measure the clock skew. Fourier transform is used to extract the time skew, which is independent of the distance, access technology, and topology. The system was demonstrated to work well over international networks.

Cellular networks use RF fingerprinting to prevent fraud. According to [55], Corsair communications provides an RF fingerprinting system for cellular networks. RF characteristics are extracted and used as fingerprints. Also, TRW Inc. had developed a similar system that is used by the military to track enemies radio signals by recognizing these fingerprints.

In the rest of this chapter, related work on identifying and/or authenticating RFID tags is presented. RFID technology is different than the previously discussed technologies (Bluetooth, cellular networks, etc.). From here on we will focus on passive RFID tags that are

characterized by having very constrained resources such as power, computation, and memory. Having constrained resources makes the identification and authentication problems harder than systems with high-end resources. For example, having limited resources such as memory and power makes it hard to implement security measures to assist with the identification and authentication process.

2.1 Identifying RFID tags using minimum power response

In [49] and [48], the authors present a way to fingerprint passive UHF RFID tags based on their minimum power response measured at multiple frequencies. The minimum power response was measured using a Voyantic Tagformance Lite system [38] in an anechoic chamber, which was designed for best tag placement on objects. Using an anechoic chamber means that all the experiments are conducted in a controlled RF environment. Before each experiment, the system is calibrated using a calibration tag.

To measure the minimum power response, they use a bottom-up algorithm that sends signals to the tag. The power of the sent signals starts at -20 dBm and keeps incrementing by 0.01 dBm until the tag responds. When the tag response is detected, the power of the signal at that instant is recorded and is considered to be the minimum power response. Frequencies that range from 860 MHz to 960 MHz are tested on 100 passive RFID tags. Each of the tags is measured 6 times. Two-way analysis of variance (ANOVA) [24] is applied on the gathered data. It shows that both the frequency and the particular tag significantly affect the minimum power response. The authors use the k-NN classifier based on the minimum power response. They are able to classify the tags with an average true positive rate (TPR) of 90.5% for two manufacturers.

2.2 Identifying RFID tags using certificate of authority

Fingerprinting RFID tags by adding a random scattering structure to the tag is introduced in [35]. These structures consist of a difficult to replicate, random arrangement of a conductive material, such as copper wire, mixed with a firm dielectric material, such as plastic PET mold, that produces a unique and repeatable response. The additional structure renders RFID tags physically unique and is hard to replicate. The authors measure the near-field response of the added structure when creating the fingerprint. The distance between the reader's antenna and the tag has to be very close, between 1mm and 8mm. A misalignment noise in the readings exists because of mounting issues. In order to verify that the fingerprint acquired works, a binary classification problem is assumed.

Kernel Density Estimation (KDE) [53] is used to estimate the underlying distribution of the fingerprints. The results of the classification show that the probability to confuse a fake tag with a real one is less than 10^{-200} if the same copper-based structure is used and that probability will be reduced to 10^{-300} considering all fingerprints. The disadvantage of this work is that the distance between the reader antenna and tag has to be very close when measuring unlike in our work.

2.3 Identifying RFID tags using power and timing features

In [68], timing and power features extracted from UHF RFID tag for identification purposes is discussed. A population of 70 tags from three different manufacturers is used. A special-built RFID reader is used to challenge the tags by initiating an inventory process. RF features, such as power and timing, are extracted by measuring the tag's (RN16) preamble sent by the reader as part of the tag's reply to the inventory command.

The main goal of the work in [68] is to study the feasibility and the accuracy of physical-layer identification and classification of passive UHF RFID tags. The experimental setup consists of a special purpose tag reader for signal acquisition, and a feature extracting and matching module. Between the performed experiments, the tag is powered down and is assumed to have lost all of its stored energy before being activated again for the next experiment.

The first experiment that is conducted is the extraction of the time interval error (TIE) feature. The TIE measures how far each active edge of the clock varies from its ideal position. As time goes by, a constant increase in the TIE (δ_{TIE}) was observed. This (δ_{TIE}) value has been considered as the timing feature to be used for identification and classification. Power features are also extracted. The average baseband power $\left(\overline{P}_B\right)$ of the acquired RN16 is calculated. $\overline{P}_B$ relates the backscatter power transferred from the tag to the reader during data modulation phase. The results of the classification experiments demonstrate that the TIE has an accuracy of 71.4% while the average baseband power has an accuracy of 43.2%. The experiments also demonstrate that combining the time and power features results in a 98.7% accuracy. In addition, it shows that the timing results are stable but the power results are not stable across different configurations. Finally, the spectral feature method from [19] is applied to UHF tags in [68] resulting in an accuracy of 99.6% but the results are not stable at varying distances.

2.4 RFID tag counterfeit detection using timing features from transient responses

Work on fingerprinting HF RFID tags is presented in [56]. Since HF RFID tags are used, inductive coupling, rather than radiation or backscattering, is the primary electromagnetic transmission mechanism. A closed loop test system is used to test the RFID tags. In the test

setup, a computer and a commercial RFID reader provide the digital commands necessary to advance a reader-tag transaction. The computer initiates the transaction via the reader, and it keeps on recording the electromagnetic field measurement of that transaction.

The authors extract timing features from the transient response of the tag as well as frequency and phase components. The techniques are tested on a set of 20 RFID smart cards. There are zero classification errors in their small sample size. The features in [56] are measured in the HF frequency band and in the near field unlike our measurements that are from the UHF frequency band and in the far field. Therefore, the distance between the reader antenna and tag is necessarily smaller than the distances in our measurement.

2.5 Physical unclonable functions

Physical unclonable functions (PUFs) [58] are innovative circuit primitives that extract secrets from the physical characteristics of integrated circuits rather than storing them in a digital memory. PUF is a way to fingerprint IC chips by generating volatile secret keys for cryptographic operations. These volatile secret keys can only be attacked when the chip is powered on, which makes attacking the volatile memory to discover the secret key a much harder problem than attacking a non-volatile memory that stores secret keys.

PUFs are functions that are embedded within the chips to be authenticated. PUF maps a set of challenges to a set of responses based on an intractably complex physical system. The function can only be evaluated with the physical system, and is unique for each physical instance.

To authenticate an RFID tag using PUFs, the tag must go through and enrollment phase in which a trusted party applies a set of challenges to the PUF and records the response, which can be either stored at a database (off-line authentication) or directly on the tag (on-

line authentication) as seen in Figure 2.1. During verification, a random challenge (from the set of challenges that the tag was subjected to at enrollment phase) is sent to the tag. The tag's response is recorded and compared with the one that was previously stored (from the enrollment phase). If the tag's response matches the one that was previously recorded, then the tag is assumed to be authentic.

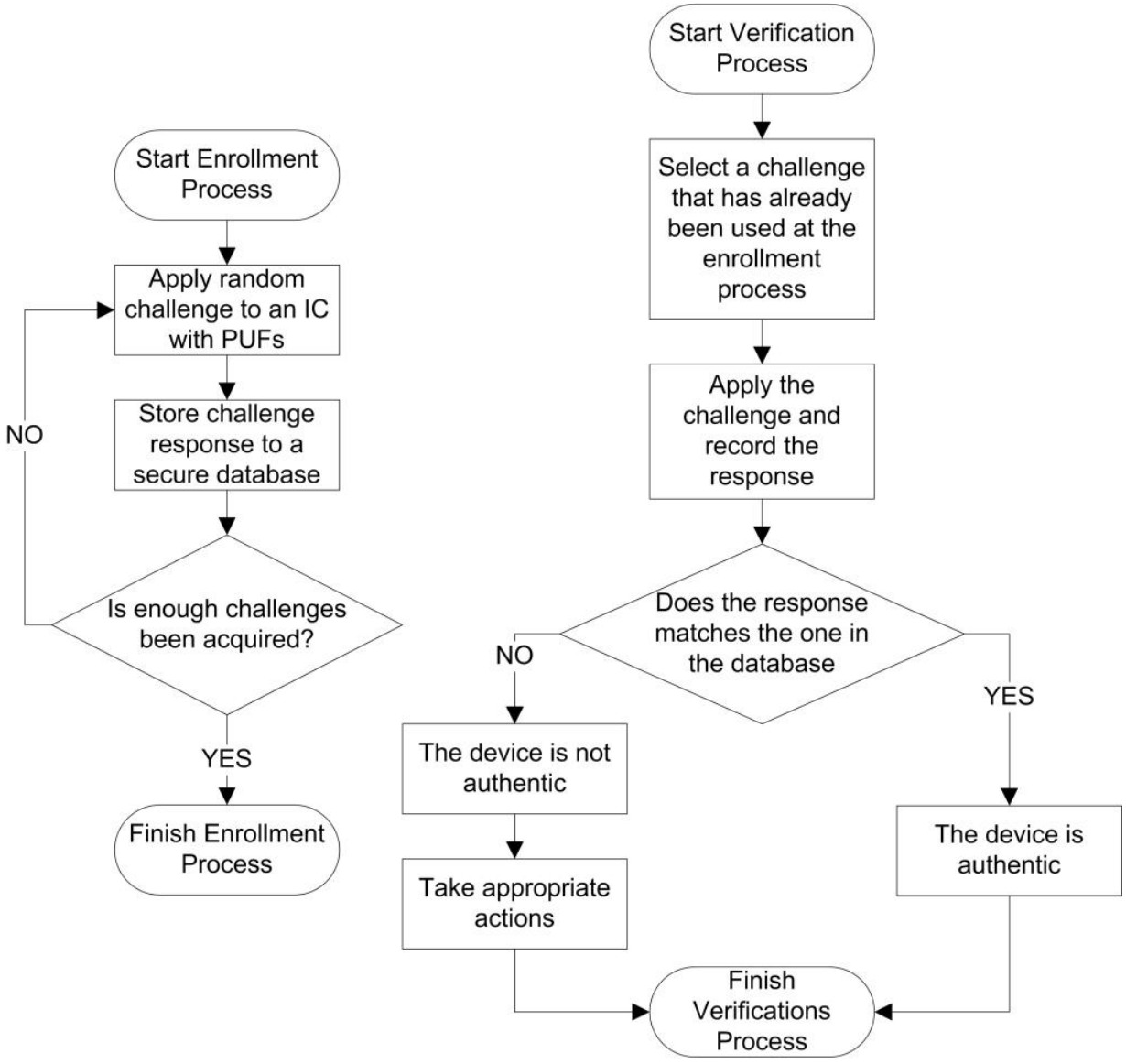

Figure 2.1: PUF Authentication System. The flowchart on the left presents the enrollment process, while the one on the right presents the authentication process.

Chapter 3

Classification

In this section, the set of experiments that are conducted for RFID tag identification is discussed. We start by describing the setup used for the experiments. An overview of the testing environment and the methodology is provided. The extracted features and the steps that have been taken to finalize the feature set are presented. We conclude this section by providing the final classification results for identification of the tag manufacturer and for individual tag identification.

3.1 Testing Environment

All the measurements are performed using a specialized RFID conformance test system, which is shown in Figure 3.1 [2]. This programmable conformance test system is capable of sending valid and invalid commands to test how will the tags respond. The conformance test system is capable of capturing the response from the tags, processing these responses, and extracting important features for fingerprinting. An example of the captured transmission is shown in Figure 3.2. The conformance test system includes a 2.7 GHz RF up-converter, a 2.7 GHz RF down-converter, and an FPGA based IF transceiver.

In order to mimic real world environment, no special precautions are taken when measuring the tags. In other words, the measurements are taken in a noisy environment, where thermal fluctuations occur, cell-phone noise is available, WiFi noise is present, and RF noise occurs in a random fashion. The only restriction we apply while measuring the tags is the distance between the reader antenna and the tag to be tested. In all the experiments, the

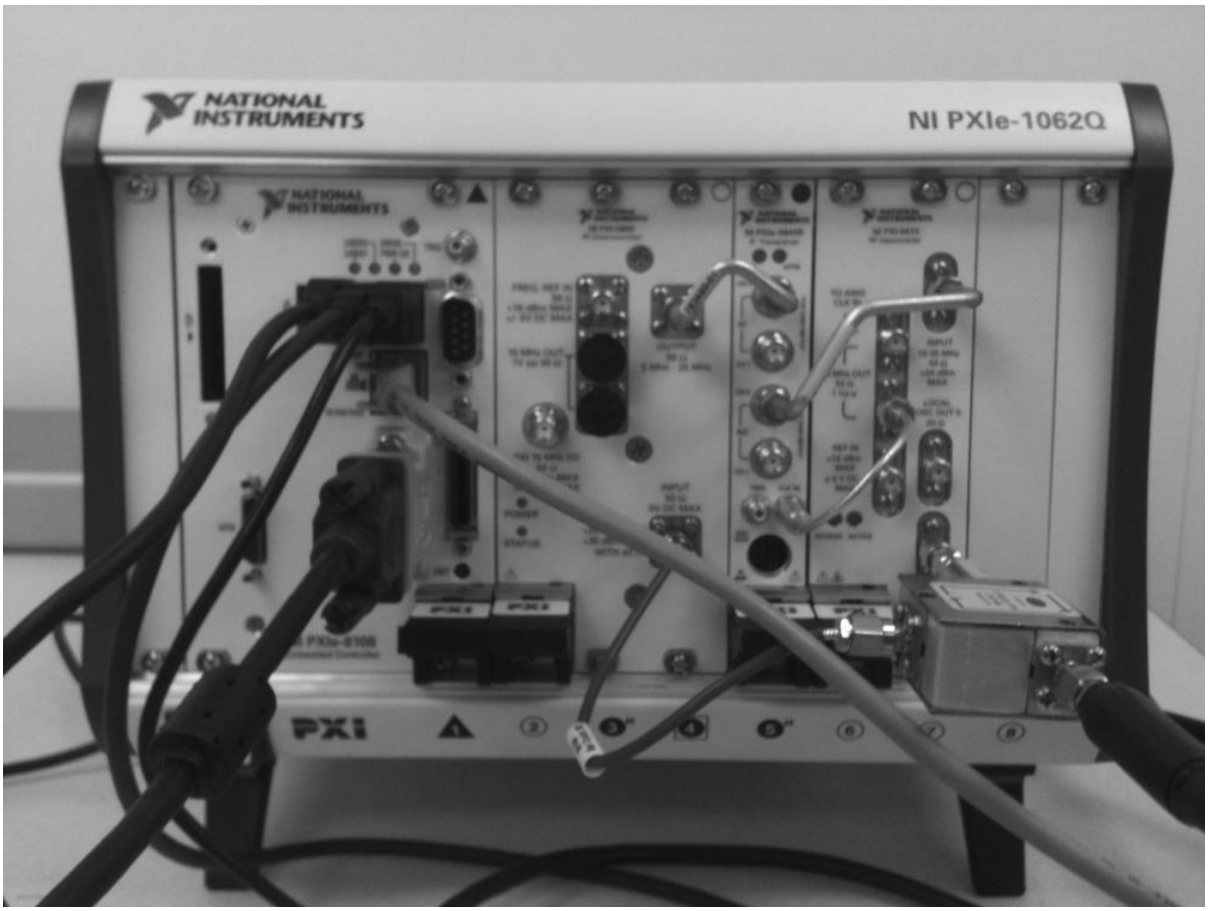

Figure 3.1: Conformance Test System

distance between the reader antenna and the tag is fixed. Tags from three different manufacturers are being tested. For each manufacturer, 100 tags are measured. Each tag was measured for 5 times. A total of 1500 overall measurements are gathered. After each of the measurements, the tags are assumed to have lost all of the energy it stored in the previous measurement.

3.2 Feature Descriptors

Each RFID tag is characterized by a set of features. These features are used by the classifier's algorithm to determine if the presented tag features match the enrolled tag features.

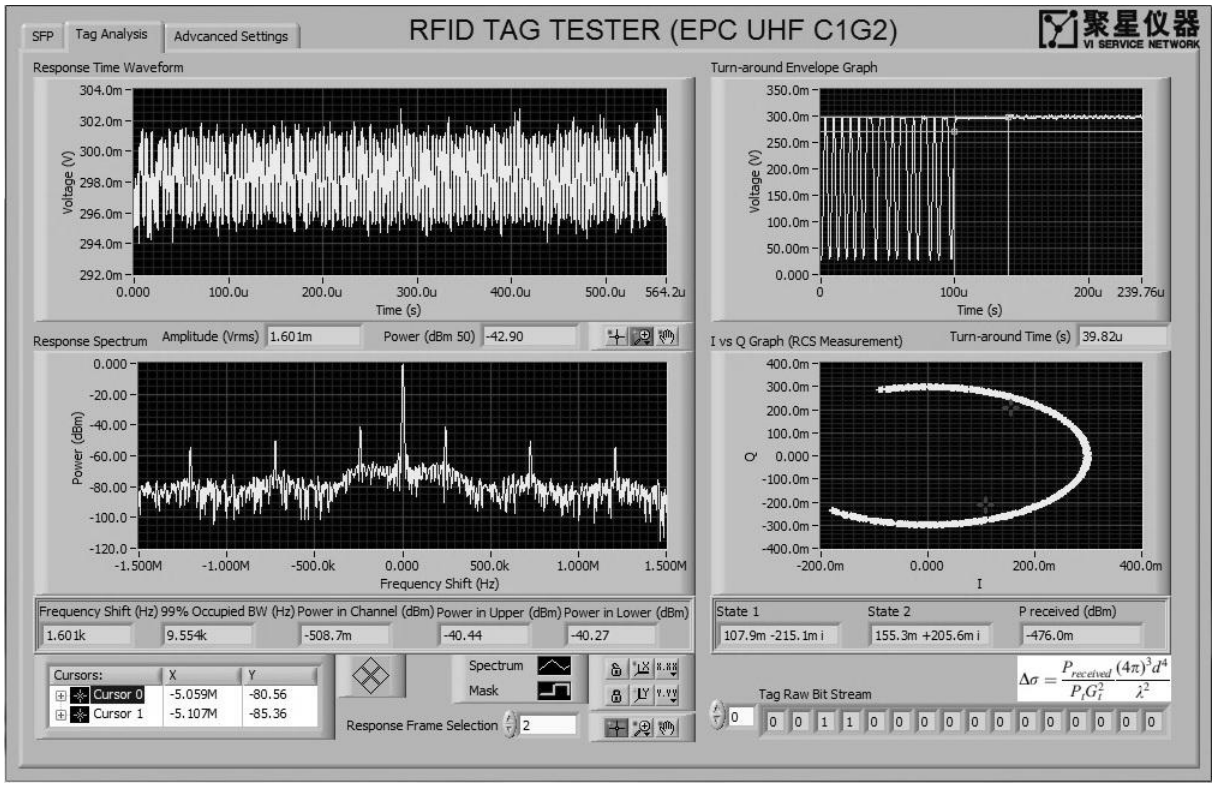

Figure 3.2: RFID tag sample transmission

Therefore, the chosen features will be the main factor in determining the performance of the classifier. Adding more features does not necessarily improve the performance of the classifier. In fact, adding more features can increase the noise in the data. Therefore, choosing features is usually a two-step process, that is, feature extraction and feature selection.

In the proposed methodology, regular commands are sent to the RFID tags by the conformance test system, which acts as a specialized reader, and the response from the tags is measured and recorded. The reader demodulates the received transmission and extracts the features, which include timing, voltage, and power measurements. Several high-level observations have been discovered:

1. For each of the tested tag manufacturers, some tags have large variance in the time needed to send data to the reader. For example, the tags from Manufacturers A and

B alternate between two different timing delays;

2. Tags from the same manufacturer tend to have the same transmission time range; and

3. There are differences in features among tags, especially in timing and power measurements.

Given the previous observations, we hypothesized that we could distinguish tags by their measured features if we determined good combinations of these features. In other words, if we are given enough features and enough training data we will be able to identify the manufacturer and/or the individual tag identity of the tag that is being tested.

3.3 Feature Extraction

Using the custom built reader's software, we can focus our attention on any part of the tag's response. We choose to focus on the tag's transmission of the PC+EPC+CRC as Figure 3.2 shows a sample for that transmission. The reason for selecting the PC+EPC+CRC is because it can be repeated as many times as possible and it is a long transmission. This gives the extracted features the highest possibility to be distinct among different tags. Long transmissions have higher probability of exhibiting distinct patterns than short transmissions.

The data analysis software we use generates most of the features we needed such as, power, and timing features. Other features, such as actual time-voltage waveform, can not be generated from the software as is. In order to record actual time-voltage waveform, we modified the software. The performed modification are shown in Figure 3.3 where we added recording hooks to record when the PC+EPC+CRC starts and stops. The second modification we did is shown in Figure 3.4 where we added a recording hook to record the entire reader-tag communication session.

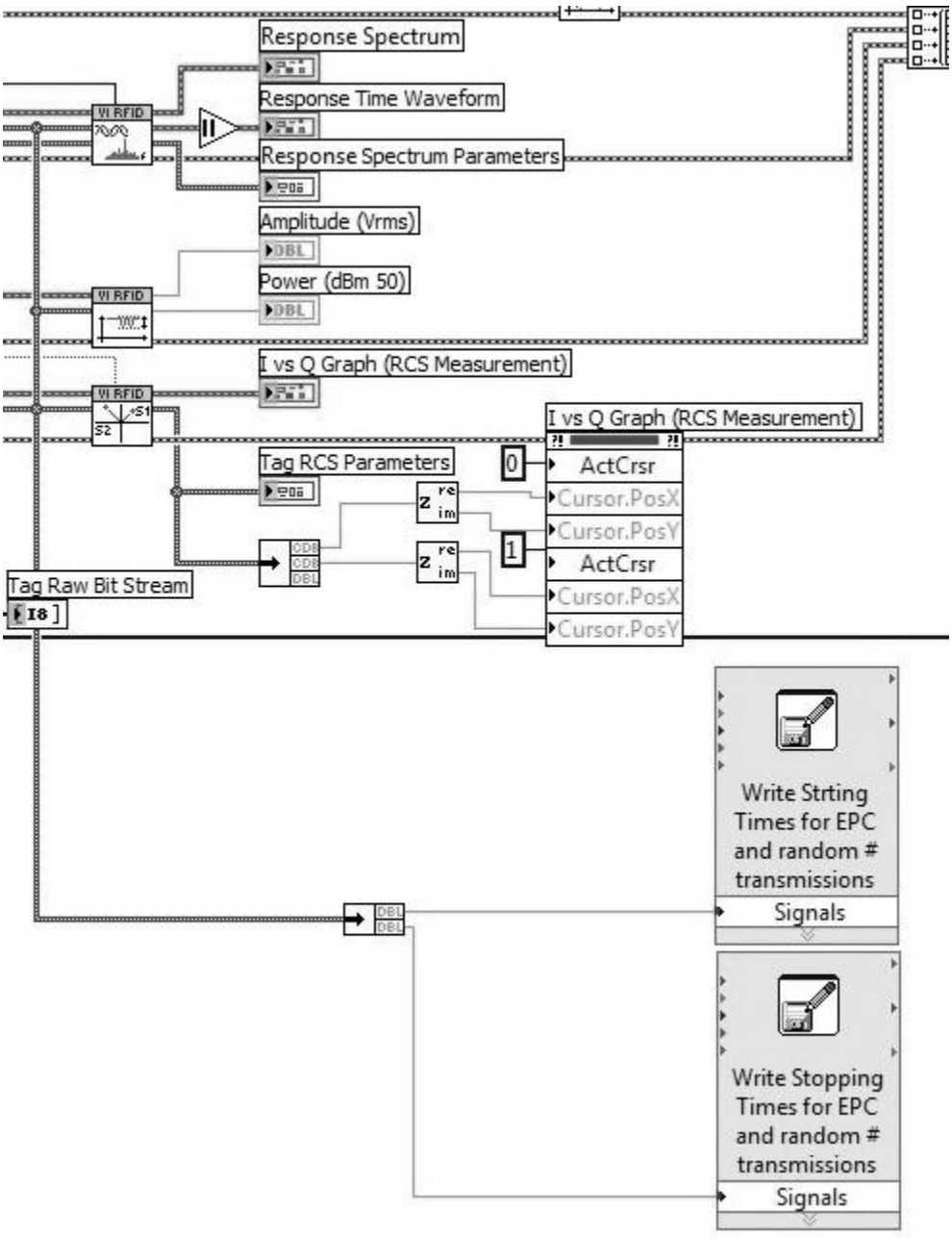

Figure 3.3: LabView modification to record timing

The extracted features can be categorized into three groups.

- **Power features**: These features represent the power measurement that are acquired from the PC+EPC+CRC transmission;

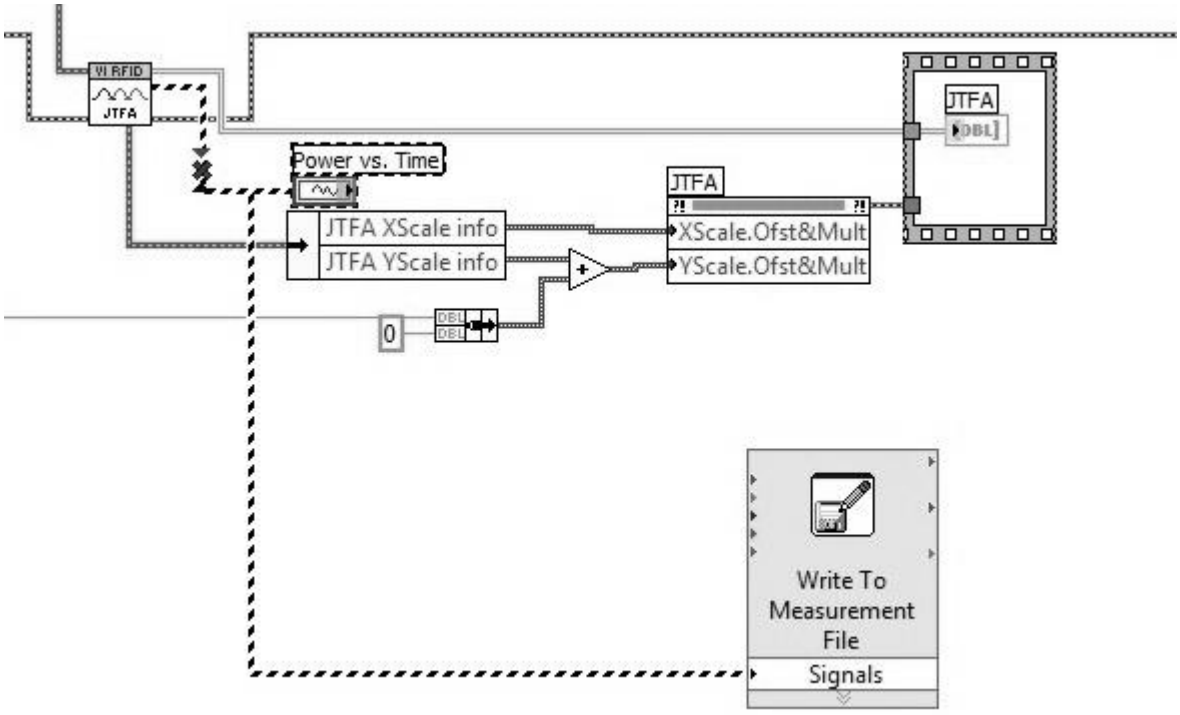

Figure 3.4: LabView modification to record time-voltage waveform

- **Timing Features**: These features represent how long will it take the tag to perform a certain operation; and
- **Voltage Features**: These features represent information that is gathered from the tag's response time-voltage waveform of the demodulated signal.

A total of 14 features are acquired. The following is a list of these features with a short description of each. In the rest of this dissertation, we will refer to the features by their number in the list.

1. **PC+EPC+CRC Transmission time:** It is measured from the demodulated waveform. This is the time it takes for the tag to transmit its protocol-control information (PC), EPC, and cyclic-redundancy check (CRC), after it is queried by a reader and is measured in microseconds. The PC+EPC+CRC sequence is the most common se-

quence that a tag responds with when identifying itself. Typical measurements are 500-575 microseconds;

2. **PC+EPC+CRC Amplitude:** It is measured from the demodulated waveform. This is the equivalent DC component of the PC+EPC+CRC transmission and is measured in V_{rms}. Typical measurements were 3.278 millivolts RMS ($mVrms$);

3. **PC+EPC+CRC Power:** It is measured from the demodulated waveform. This is the received power of PC+EPC+CRC that is transmitted by the tag and is measured in dBm 50, which means dBm at 50 ohm impedance. It is derived from the PC+EPC+CRC Amplitude. Typical measurements are -43 dBm 50;

4. **First Random Number T1:** It is measured from the demodulated waveform. This is the time delay from the end of the Query command by the reader to the beginning of the 16-bit random number transmitted by the tag and is measured in microseconds. Typical measurements are 35-40 microseconds. T1 refers to the time the tag needs to recharge after it responds to the reader;

5. **PC+EPC+CRC T1:** It is measured from the demodulated waveform. This is the time delay from the end of the ACK by the reader to the beginning of the PC+EPC+CRC transmitted by the tag and is measured in microseconds. Typical measurements are 35-40 microseconds;

6. **Second Random Number T1:** It is measured from the demodulated waveform. This is the time delay from the end of the request from the reader for a new handle after it receives the PC+EPC+CRC to the beginning of the new 16-bit random number transmitted by the tag and is measured in microseconds. Note that the conformance test system is programmed to ask for another handle from the tag after it receives the PC+EPC+CRC. Typical measurements are 35-40 microseconds;

7. **Power in Channel:** It is measured from the frequency spectrum. This is the power received by the reader from the tag in an RF channel while the tag is transmitting the PC+EPC+CRC and is measured in dBm. It is the power in the 500 KHz channel at which we take the measurements. We fix the frequency to be 915 MHz. Therefore, the channel is 915 MHz +/- 250 KHz, or 500 KHz wide. A typical measurement is $-508.7m$ dBm or -0.5087 dBm;

8. **Power in Upper:** It is measured from the frequency spectrum. This is the power received by the reader from the tag in the channel above the measured RF channel while the tag is transmitting the PC+EPC+CRC and is measured in dBm. We take measurements in the 500 KHz channel that has a center frequency of 915 MHz. Therefore, the Power in the Upper is the power in the channel with a center frequency of 915 MHz + 0.5 MHz. A typical measurement is -40 dBm;

9. **Power in Lower:** It is measured from the frequency spectrum. This is the power received by the reader from the tag in the channel below the measured RF channel while the tag is transmitting the PC+EPC+CRC and is measured in dBm. We take measurements in the 500 KHz channel that has a center frequency of 915 MHz. Therefore, the Power in the Lower is the power in the channel with a center frequency of 915 MHz - 0.5 MHz. A typical measurement is -40 dBm;

10. **Power Received:** This is the power received by the reader from the tag at the base frequency of 915 MHz +/- 1.5 MHz, a 3 MHz channel, and is measured in dBm. In other words, this is the total power received in a 3 MHz channel with the reader frequency centered in the channel. For example, we measure -0.476 dBm from 915-1.5 MHz to 915+1.5 MHz, which is larger than the Power in Channel feature;

11. **Voltage Maximum in PC+EPC+CRC Transmitted Data:** This is the maximum voltage received by the reader from the tag and is measured in volts;

12. **Voltage Minimum in PC+EPC+CRC Transmitted Data:** This is the minimum voltage received by the reader from the tag and is measured in volts;

13. **Voltage Variance in PC+EPC+CRC Transmitted Data:** This is the variance of the voltage received by the reader from the tag and is measured in volts squared; and

14. **Voltage Standard Deviation in PC+EPC+CRC Transmitted Data:** This is the standard deviation of the voltage received by the reader from the tag and is measured in volts.

3.4 Feature Selection

Features selection refers to the process in which a subset of the available features will be used in classification experiments. One might wonder why the need for feature selection? If we used all the available features, will it not be better than using a subset of features?

Many issues might arise from using all of the features. First, using all the features will require using large memory to store each of the object's feature descriptor array. Processing all the features requires substantial computation and processing capabilities, which in turn requires longer time to compute. Finally and most importantly, having all features available does not necessarily result in the best performance. The previous phenomenon is known as the "curse of dimensionality" [28].

For the previous reasons, we can reduce the number of features [36] using many techniques such as principal component analysis (PCA) [66], and linear discriminant analysis (LDA) [23][40]. In our work, we started by testing all the feature as shown in Section 3.4.1 to determine which of the features contributes greatly in discriminating between tags and which features do not. After that, we tested the correlation between the different features to

eliminate the ones that are closely related as shown in Section 3.4.3.

3.4.1 Preliminary Classification

In order to have an initial idea on which of the features affects the classification results, some preliminary classification is performed. With preliminary classification we train and test the k-NN (k = 1, 3, and 5), Parzen windows, and SVM classifiers on the features. We train the classifiers using 66% of the available data and we test the classifiers performance using the remaining 34% of the data. The previous balance between the training and testing data is referred to as 66/34.

We apply the classifiers to all possible features combinations $C_m^d = \binom{d}{m}$ for both the manufacturer and tag identification problems. Therefore, there are four sets of experiments. In the manufacturer identification problem, when presented with a tag, we classify it as belonging to one of the three enrolled manufacturers, A, B, or C. In addition, we have three sets of experiments for the tag identification problem, one for each of the three manufacturers. In the tag identification problem, when presented with a tag from a particular manufacturer we classify it as belonging to a particular enrolled tag. We decided that performing the tag identification problem within a set of tags from one manufacturer is a more difficult, or worst-case, test of tag identification.

For each classifier in each of the four sets of experiments, we calculate the true positive rate (TPR) and we identify the top-ten set of features based on this rate. The TPR is defined in terms of the number of true positives (TP) and the number of false negatives (FN), $TPR = \frac{TP}{TP+FN}$. All of the performance evaluation terms will be described in depth in Section 3.6.1.

For both manufacturer and tag identification problems, 1-NN, 3-NN, 5-NN, and Parzen

Window have reasonable $TPRs$. However, SVM does not perform as well for either manufacturer or tag identification problems. In the tag identification problem, SVM performs poorly with less than 50% TPR and is computationally intensive. The manufacturer identification problem appears to be easier for all classifiers with many of the classifiers identifying the correct manufacturer without making any classification error. The tag identification problem for the three manufacturers is more difficult but for certain sets of features the TPR is above 77% and for some classifiers above 95%. Tables 3.1 and 3.2 show the top-ten set of features for tag identification of manufacturer A, when classifying with the 1-NN and Parzen Window classifiers, respectively. Note that adding more features does not necessarily result in better performance (curse of dimensionality), and none of the sets have all 14 features. This gives us evidence that we can reduce the number of features without compromising the classifier's performance. The 1-NN classifier performs best in all four sets of experiments so it is chosen for determining the final feature set.

Table 3.1: 1-NN Top-Ten Results for Tag Identification of Manufacturer A

True positive rate	**Features responsible**
99%	1 2 7 8 11 12 13 14
98%	1 2 7 9 10 12 14
97%	1 3 6 7 8 9 13 14
97%	1 3 6 7 8 9 12
97%	1 3 6 7 8 9 10 14
97%	1 3 4 6 7 8 10 12 13
97%	1 2 7 9 10 11 12
97%	1 2 7 8 10 11 12 14
97%	1 2 7 8 9 10 12 13
97%	1 2 5 7 8 10 12 14

3.4.2 Features' Histogram

After gathering the top-ten results for each of the classifiers (k-NN, Parzen windows, and SVM) we noticed that 1-NN had the best performance. Since 1-NN had the best performance, we used its results for tag identification problems to determine how many times each of the

Table 3.2: Parzen Window Top-Ten Results for Tag Identification of Manufacturer A

True positive rate	**Features responsible**
81%	2 7 9 10 11 12
80%	3 7 8 9 10 14
80%	2 7 9 12
79%	2 7 8 12 13 14
79%	2 7 8 11 14
79%	2 7 8 9 13
79%	2 7 8 9 10 12
78%	3 7 8 9 10 12 14
78%	2 7 8 10 11
77%	3 4 6 7 8 9 10 12

features appear in a set that had high TPR. To determine the number of times a certain feature appear in the top-ten results, a histogram is used.

Histograms are a graphical representation showing visual impression of the data distribution [47]. Histograms are used to plot the density of the data and often used for density estimation, which is another name for Parzen windows classifier. The mathematical foundation of any histogram can be summarized as shown in Equation 3.1 where, n is the total number of observations, k is the total number of bins, and m_i is the function of the histogram.

$$n = \sum_{i=1}^{k} m_i \tag{3.1}$$

Figure 3.5 shows the histogram of the number of times that a feature appears in the top-ten set of results for manufacturer identification, tag identification for manufacturer A, tag identification for manufacturer B, and tag identification for manufacturer C. The histogram shows a strong overlap of features that are good for identifying both the manufacturer and the individual tag. Even though there are slight differences, we decided that having the ability to identify both a manufacturer and tag is desirable.

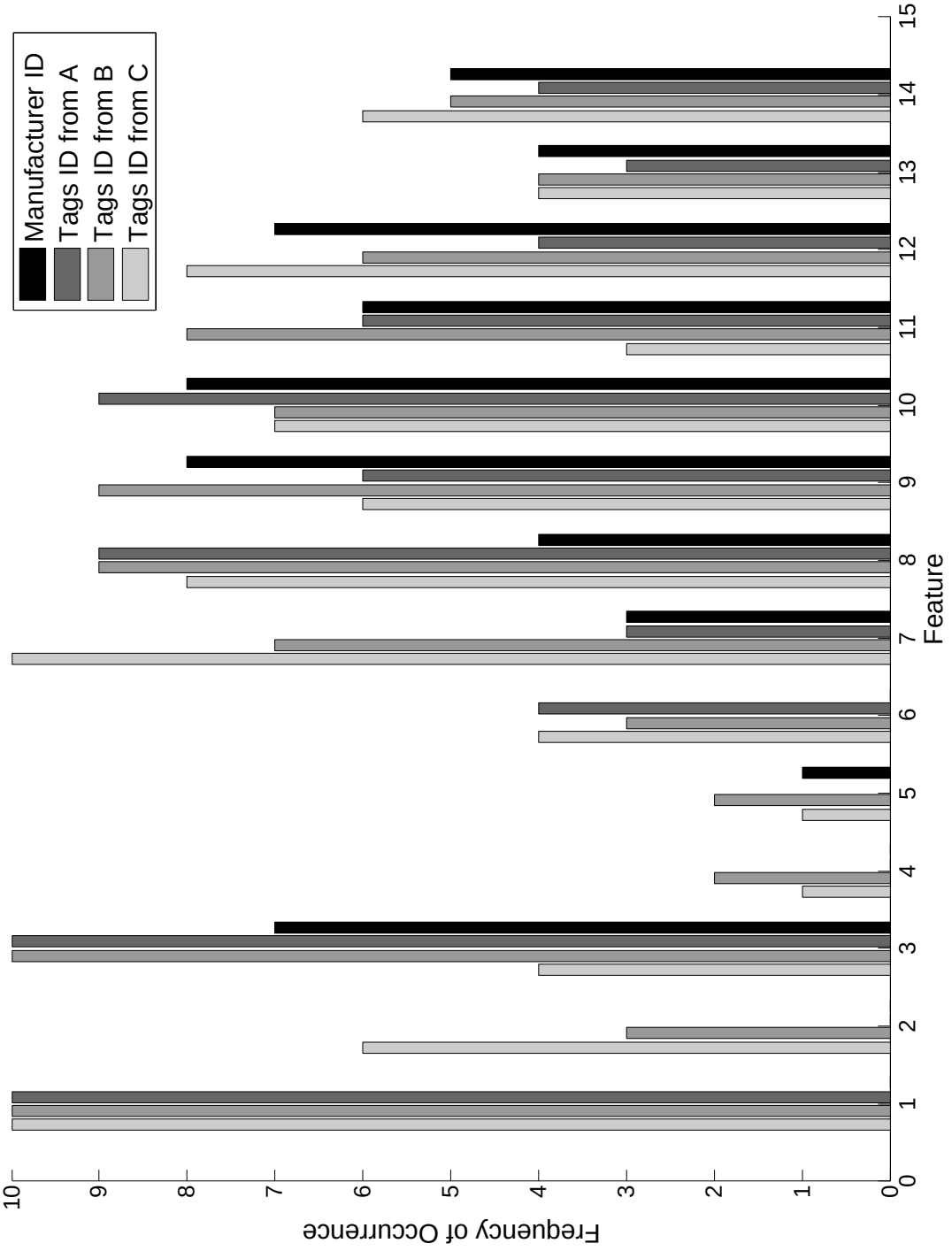

Figure 3.5: Histogram of the Number of Times a Feature Appears in Good Sets Based on the Preliminary Classification Experiments Using the 1-NN Classifier Applied to Manufacturer and Tag Identification

3.4.3 Features' Correlation

The main goal behind applying correlation on the available features set is to determine the dependencies among the features, thus, eliminating any redundant feature. Correlation is a measure of how much two variable are related. After applying the correlation analysis on a set of variable (features) n, a correlation matrix $n \times n$ is generated. The values in the correlation matrix range between +1 and -1. A value of +1 in the correlation matrix means that there is a perfect positive correlation (if one variable increases, the other one will increase by the same ratio as well) between the two variables, which are represented by the row and column numbers. A value of -1 in the correlation matrix means that there is a perfect negative correlation (if one variable increases, the other variable will decrease by the same ration as well) between the two variables, which are represented by the row and column numbers.

$$C(i,j) = E\left[(X_i - \mu_i)\,(X_j - \mu_j)\right] \tag{3.2}$$

$$\rho(i,j) = \frac{C(i,j)}{\sqrt{C(i,i)C(j,j)}} \tag{3.3}$$

To build a correlation matrix, we first calculate the covariance matrix and use its elements for the correlation calculations. Equation 3.2 is usually used to calculate the covariance of two elements i and j. In Equation 3.2, X_i refers to element i and μ_i refers to the mean of element i over all readings. The correlation matrix is then built using Equation 3.3. After applying the correlation analysis on the feature set we have, several dependencies appeared

Table 3.3: Correlation Analysis

Features	Correlation Score
Feature #2 and Feature #3	0.9946
Feature #4, Feature #5, and Feature #6	0.859 - 0.875
Feature #7 and Feature #10	0.9944
Feature #11 and Feature #12	0.9978
Feature #13 and Feature #14	0.9955
Feature #8 and Feature #9	0.9696

as shown in Table 3.3.

By looking at Table 3.3, we notice the following

1. The correlations between the First Random Number T1 (feature #4), the PC+EPC+CRC T1 (feature #5), and the Second Random Number T1 (feature #6) are between 0.859 to 0.875 because it is dependant on the power supply unit of the tag;

2. There is a high correlation between the PC+EPC+CRC power (feature #3), and the amplitude of PC+EPC+CRC (feature #2). The reason behind such high correlation is because the power is actually derived from the amplitude; and

3. There is a high correlation between the voltage variance in PC+EPC+CRC transmitted data (feature #13), and voltage standard deviation in PC+EPC+CRC transmitted data (feature #14). Since the standard deviation is derived from the variance, the correlation between them is high.

Based on the results of the histogram and the correlation analysis, we decided to reduce the feature set. Instead of using all 14 features, only 7 features will be used in the classification experiments in Section 3.6. The following list shows the final 7 features we are using.

- **PC+EPC+CRC Transmission Time** (original feature #1);

- **PC+EPC+CRC Power** (original feature #3);
- **Second Random Number T1** (original feature #6);
- **Power in Channel** (original feature #7);
- **Power in Lower** (original feature #9);
- **Voltage Minimum in PC+EPC+CRC Transmitted Data** (original feature #12); and
- **Voltage Variance in PC+EPC+CRC Transmitted Data** (original feature #13)

3.5 Features' Ranking

After determining which features to use, we need to determine the significance of each of the features as it applies to identification. Therefore, we ranked the features using a feature scoring technique. Feature scoring is a feature selection technique that associates a score with each of the features. For example, to select d features out of m original features, we can use feature scoring to select the features with the highest (or lowest) d scores. In this work, we are using a well-known approach that is based on Fisher criterion [25][27].

The key idea behind Fisher Scoring is to find a subset of features such that the data points in different classes are as far as possible from each other, and the data points in the same class are as close as possible. The Fisher Score of the j^{th} feature is computed as:

$$F(x^j) = \frac{\sum_{k=1}^{c} \eta_k (\mu_k^j - \mu^j)^2}{\sum_{k=1}^{c} \eta_k \sigma_k^2} \tag{3.4}$$

Table 3.4: Feature Scoring of Feature when Identifying Manufacturer Models and Individual Tags

	f_1	f_2	f_3	f_4	f_5	f_6	f_7
Manufacturer Identification	9	6	13	3	7	11	1
Tag Identification from A	13	11	7	3	9	1	6
Tag Identification from B	7	11	3	13	9	1	6
Tag Identification from C	7	11	3	13	9	1	6

Note: In this table, the features that are shown under f_1 have the highest score while the ones under f_7 have the lowest score.

where c represents the class, η_k represents the number of data points in class k, μ_k is the mean of class k and σ_k is the variance of class k.

We apply Fisher Scoring on the four sets of data. The first set represents all the readings we have with three manufacturer models, which corresponds to three classes. The second, third, and fourth data sets are associated with 100 different individual tags. Each set contains the same manufacturer model tag. In other words, there are 100 classes in each of the last three data sets. The rankings from Fisher Scoring are presented in Table 3.4.

The results in Table 3.4 show the significance of each feature in discriminating between classes. As the results show, both manufacturer and tag identification require a combination of power and timing features. Power features dominate, but the two timing features (#1 and #6) do help during individual tag identification. Without the timing features, the TPRs using 1-NN classifier decrease to 69%, 80%, and 62% for tag identification, which hints to the importance of using multiple features for improving the performance of the classifiers.

3.6 Classification Experiments and Results

In this section, the classification experiments and the obtained results are presented. Before presenting the results, a quick introduction of the performance metrics is provided in

Section 3.6.1. After introducing the used metrics, the three main experiments and their results are provided in Section 3.6.2, Section 3.6.3, and Section 3.6.4. All the performed classification results are done using a pattern recognition toolbox (PRTools) for Matlab [22].

3.6.1 Performance Evaluation

There are several metrics that can be used to determine how good the obtained classification results are. The main source for any classification performance metrics is the confusion matrix. In the confusion matrix, each column represents the instances in a predicted class, while each row represents instances in an actual class. Table 3.5 shows the layout of any confusion matrix.

Table 3.5: Confusion Matrix Layout

		Predicted Outcome	
		Positive	Negative
Actual values	Positive	TP	FN
	Negative	FP	TN

As shown in Table 3.5, there are 4 possible outcomes of any classification result. The following list describes these outcomes.

1. **True Positive (TP)**: refers to any test instance that actually belongs to the positive class and was predicted as positive;
2. **True Negative (TN)**: refers to any test instance that actually belongs to the negative class and was predicted as negative;
3. **False Positive (FP)**: refers to any test instance that actually belongs to the negative class and was predicted as positive; and
4. **False Negative (FN)**: refers to any test instance that actually belongs to the positive class and was predicted as negative.

The confusion matrix in Table 3.5 shows the possible outcomes when we have only two classes (binary classifier). In many classification experiments, there are n classes, with $n > 2$ to classify. In order to determine the four main counts (TP, TN, FP, and FN) for each class, n binary confusion matrices need to be generated. The binary matrix for a class X will have its positive class referring to instances from class X while the negative class refers to any instance that belongs to a class other than X. Assuming that the number of training samples is the same for each class and the number of testing samples is also the same, the final counts can be determined by calculating the mean of each count value.

Once the final counts are present (after calculating the means), the performance evaluation metrics can be calculated. The first evaluation metric that we used is the accuracy. The accuracy refers to the percentage of instances that were correctly classified into either positive or negative. The formula to calculate the accuracy is shown in Equation 3.5.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.5}$$

It should be noted that in the case of multi-class problems, TN does not necessarily mean a correct result. For example, if we have three classes to classify and we are building the binary confusion matrix for class 3, a TN means that an instance belongs to either class 1 or 2 was classified into either class 1 or 2. In other words, it does not matter if class 1 instances were classified as class 2 or if class 2 instances were classified as class 1, as long as they are not classified as class 3. In this case, they all belong to the TN count.

The second performance metric we are using is the true positive rate (TPR). TPR refers to the percentage of positive instances that are classified as positive to the actual number of positive instances ($TP + FN$).

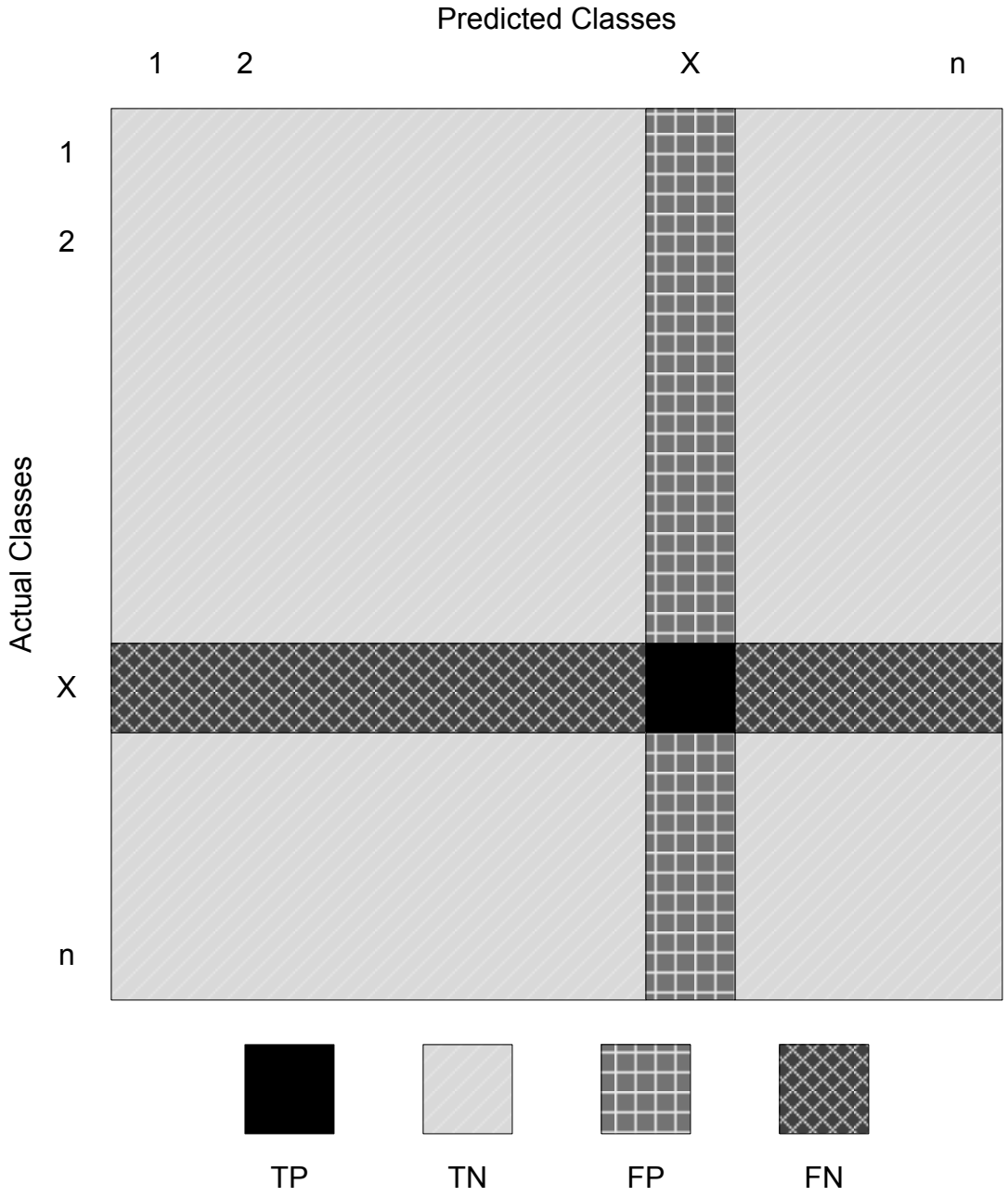

Figure 3.6: Multiple Classes Confusion Matrix

$$TPR = \frac{TP}{TP + FN} \tag{3.6}$$

The final performance metric which we are using that is derived from the confusion matrix is the false positive rate (FPR). The FPR metric refers to the percentage of negative instances that are classified as positive to the total number of negative instances ($FP+TN$).

$$FPR = \frac{FP}{FP + TN} \tag{3.7}$$

In order to show the significance of both the TPR and FPR, we would like to use a security system analogy where people who are authorized to enter belongs to the positive class and the ones who are not authorized to enter belongs to the negative class. TPR represents the percentage of people who are authorized to enter and can enter. In other words, TPR must be as high as possible. On the other hand, FPR must be as low as possible since it represents the percentage of people who are not allowed to enter but they can in-fact enter.

Another performance metric we are using is the area under the receiver operating characteristic (ROC) curve, also known as area under curve (AUC) [41]. ROC curves are a graphical plot of the TPR vs. the FPR, for a binary classifier system as the discriminant threshold is varied. ROC curves provide us with tools to select the best classification models and discard bad ones by calculating the AUC for the curves generated by each of the models.

3.6.2 Enrolled/Unenrolled Tag Identification

The first classification experiment is conducted to classify the data we have into two classes: enrolled and unenrolled. The purpose behind this experiment is to model a security system in which only authorized tags are allowed to access (enrolled) and the rest of the tags are not allowed access to the system (unenrolled).

We trained the system using 66% of the available data and we tested the system using the remaining 34%. We started by building an ROC graph that represents the performance of 5 classification algorithms, namely, 1-NN, 3-NN, 5-NN, Parzen windows, and SVM. The resultant graph is shown in Figure 3.7.

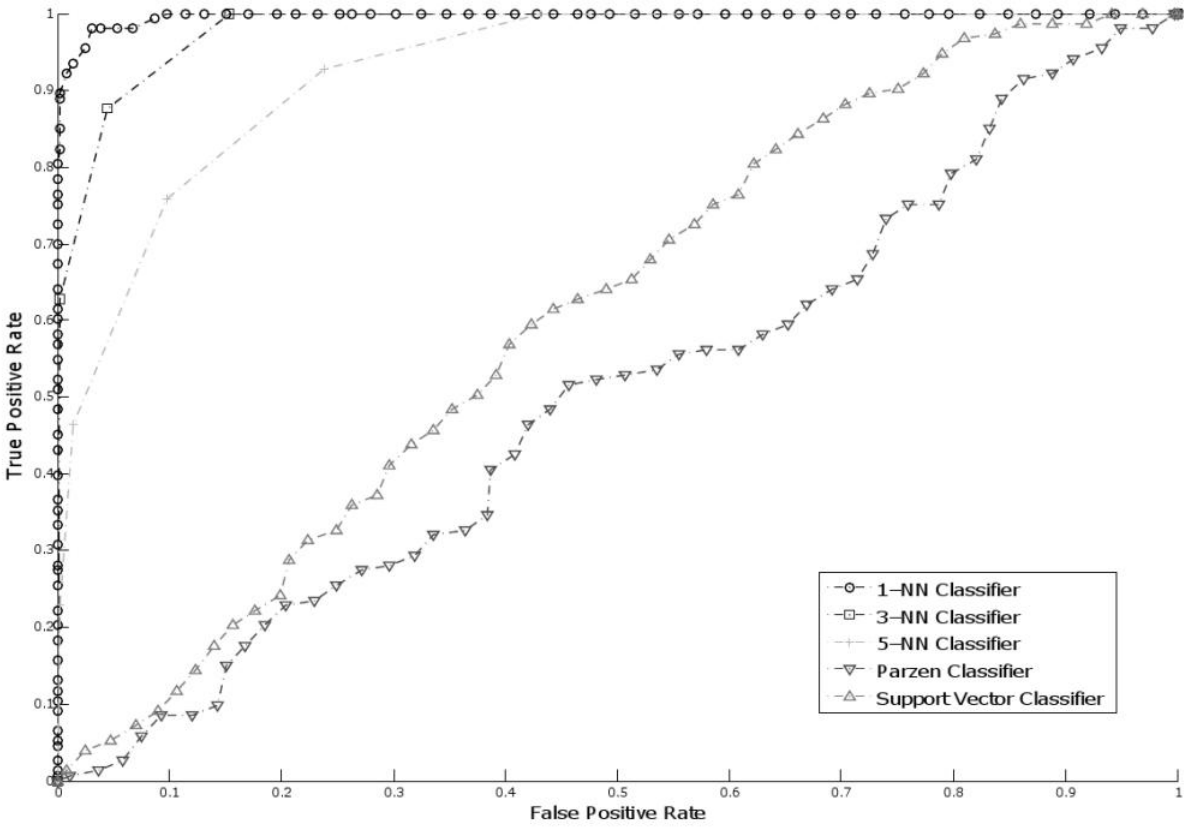

Figure 3.7: Enrolled/Unenrolled ROC Curves using 66/34

Figure 3.7 clearly shows that 1-NN out-performs every other classifier and in general k-NN is better than Parzen and SVM. To be sure and to quantify how good 1-NN is compared to the rest, the area under the curve is calculated. The AUC results are provided in Table 3.6.

The results from the AUC calculations agree with the initial observation shown in Figure 3.7. The results shows that the best classifier to use is 1-NN with an AUC of 99.29% while the worst classifier to use is Parzen windows with 49.18% AUC. Since 1-NN is the best classifier, we generated the confusion matrix which is shown in Table 3.7. The confusion

Table 3.6: Enrolled/Unenrolled AUC using 66/34

Classifier	AUC
1-NN Classifier	99.29%
3-NN Classifier	96.90%
5-NN Classifier	93.36%
Parzen Windows Classifier	49.18%
SVM	61.09%

matrix shows that 1-NN only miss-classified 15 samples out of 510 available test samples. Out of the miss-classified sample, 6 out of 150 belonged to the enrolled class and 9 out of 360 belonged to the unenrolled class.

Table 3.7: 1-NN Enrolled/Unenrolled Confusion Matrix using 66/34

	Predicted	
Actual	Enrolled	Unenrolled
Enrolled	144 (TP)	6 (FN)
Unenrolled	9 (FP)	351 (TN)

To further support the results, we generated the TPR from each of the classifiers and the results are shown in Table 3.8. In order to generate consistent data, 10-fold cross validation is used. With 10-fold cross validation we partitioned the data we have into 10 equal size parts. For 10 iterations, we train the classifier on 9 of the 10 parts and we test the classifier using the remaining part, while keeping in mind that the testing part will not be duplicated in any of the iterations. The reason behind using cross validation is to provide evidence that the results we obtained are consistent. The TPR values in Table 3.8 agree with the results we got from the AUC calculations. All the results show that 1-NN is the best classifier to use while Parzen windows and SVM will generate poor results.

3.6.3 Tag's Manufacturer Identification

The second classification experiment that was performed is to identify a tag's manufacturer. The data-set we collected has tags from three manufacturer (A, B, and C). We collected

Table 3.8: Enrolled/Unenrolled TPR Results using 10-Fold Cross Validation

Classifier	TPR
1-NN	97.20%
3-NN	96.73%
5-NN	94.93%
Parzen Windows	70.00%
SVM	70.00%

100 tags from each manufacturer. Each tag was measured five times for a total of 500 measurements per manufacturer.

We started the experiments by using 66% of the data for training the classification models and the rest of the data to test these models. In other words, 66/34 is the used training/testing ratio for all classifiers. After doing the classification experiments, we generated the confusion matrices for each classifier. Table 3.9 shows these confusion matrices. By looking at the data in the confusion matrices we notice that regardless of the used classifier, tags that belongs to model A will be always be classified correctly. Tags that belong to either model B or C will have some instances miss-classified as the case with Parzen windows and SVM.

In order to determine which of the classifier performs better, we repeated the classification experiment using 10-fold cross validation. The TPR results are shown in Table 3.10. The cross validation TPR results shows that 1-NN, 3-NN, and Parzen windows performs well with a 99.93% TPR. 5-NN also performs well with a 99.87% TPR. The algorithm that performed worst was SVM with a 95.00% TPR. Having a 95.00% TPR may not be bad in some systems, but we were able to get better results using k-NN and Parzen windows.

In order to further support the obtained results, we generated a set of ROC curves for each classifier. The ROC curves are shown in Figure 3.8. The ROC curves do not provide any information regarding which of the classifiers is better than the rest. The ROC curves shows an equal perfect performance for all the classifiers. In other words, all the classifiers

Table 3.9: Manufacturer Model Identification Confusion Matrices for Different Classifiers using 66/34

1-NN Classifier			
	Predicted		
Actual	Manufacturer A	Manufacturer B	Manufacturer C
Manufacturer A	170	0	0
Manufacturer B	0	170	0
Manufacturer C	0	0	170

3-NN Classifier			
	Predicted		
Actual	Manufacturer A	Manufacturer B	Manufacturer C
Manufacturer A	170	0	0
Manufacturer B	0	170	0
Manufacturer C	0	1	169

5-NN Classifier			
	Predicted		
Actual	Manufacturer A	Manufacturer B	Manufacturer C
Manufacturer A	170	0	0
Manufacturer B	0	169	1
Manufacturer C	0	0	170

Parzen Windows Classifier			
	Predicted		
Actual	Manufacturer A	Manufacturer B	Manufacturer C
Manufacturer A	170	0	0
Manufacturer B	0	170	0
Manufacturer C	0	0	170

SVM Classifier			
	Predicted		
Actual	Manufacturer A	Manufacturer B	Manufacturer C
Manufacturer A	170	0	0
Manufacturer B	0	166	4
Manufacturer C	0	16	154

Table 3.10: Manufacturer Model Identification TPR Results using 10-Fold Cross Validation

Classifier	**TPR**
1-NN	99.93%
3-NN	99.93%
5-NN	99.87%
Parzen Windows	99.93%
SVM	95.00%

have a 100% AUC.

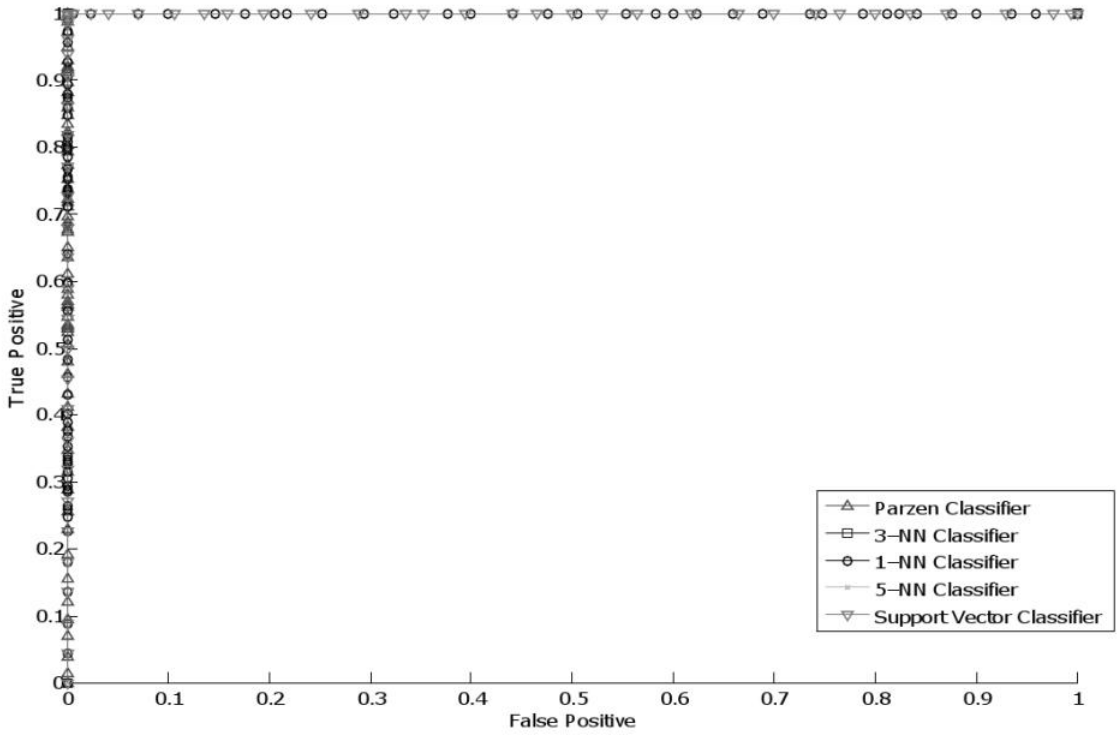

Figure 3.8: models ROC Curves using 66/34

3.6.4 Individual Tag Identification

The last experiment that was performed is to identify an individual tag. Since we have tags from three manufacturers, we performed the same experiment three times, each time for a different manufacturer.

3.6.4.1 Identifying Tags From Manufacturer A

In order to identify models from manufacturer A, only tags from manufacturer A were used for training and testing. Since we have 100 tags that represent each model, we trained our

models to distinguish between 100 classes. One issue we faced, is that only five instances are available as representatives of each class. We generated the ROC curves using 66% of the data for training (three instances) and 34% of the data for testing (two instances). The resultant ROC is shown in Figure 3.9. The AUC that is associated with each of the ROC curves is provided in Table 3.11.

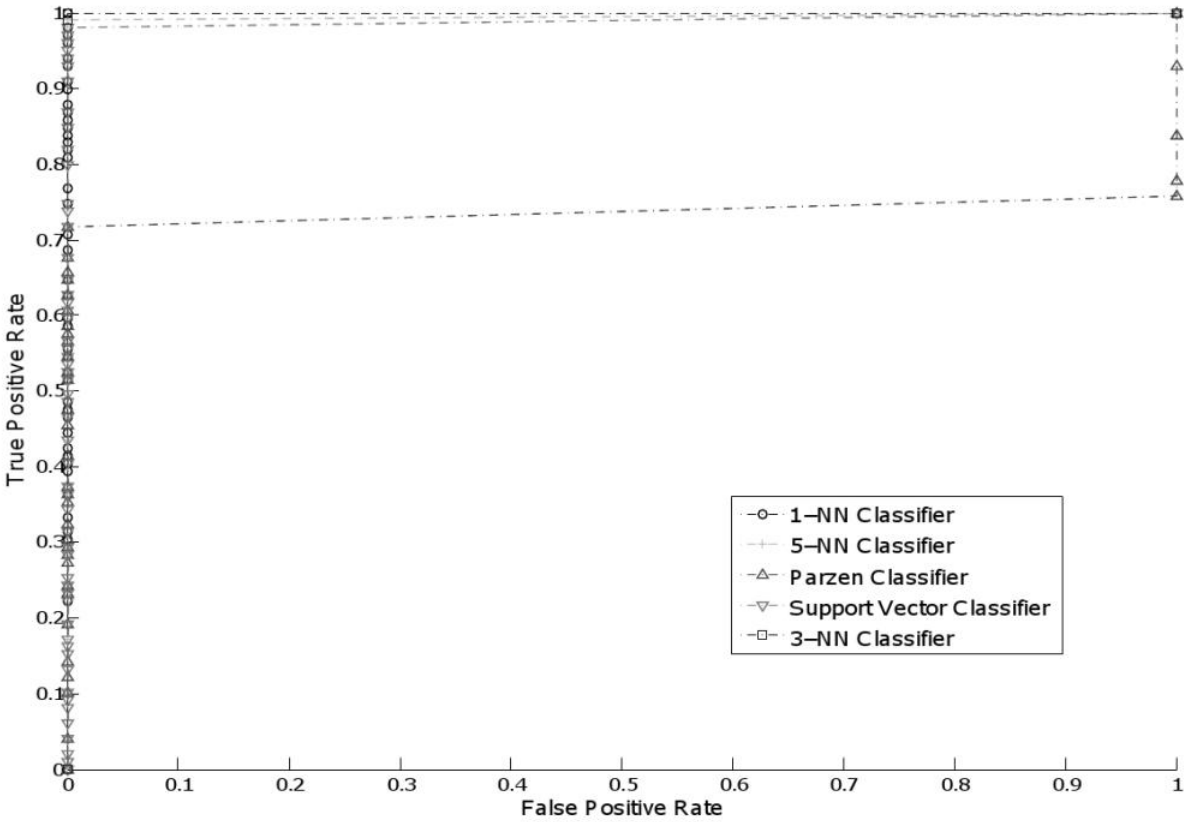

Figure 3.9: Manufacturer A ROC Curves using 66/34

Table 3.11: Manufacturer A AUC using 66/34

Classifier	**AUC**
1-NN Classifier	100.00%
3-NN Classifier	100.00%
5-NN Classifier	99.49%
Parzen Windows Classifier	73.74%
SVM	98.99%

Since only five instances are available, and in order to provide enough evidence of the

ROC results, we applied 5-fold cross validation. With 5-fold cross validation we divided the data for each class into five partitions. We train the model using four partitions and we test the resultant model using the fifth partition. The previous procedure is repeated five time while rotating the testing and training partitions. In other words, every partition will be used for testing exactly one time. The 5-fold cross validation TPR and accuracy results are shown in Table 3.12.

Table 3.12: Manufacturer A Tag Identification TPR and Accuracy Results using 5-Fold Cross Validation

Classifier	**TPR**	**Accuracy**
1-NN	90.2%	99.80%
3-NN	83.8%	99.68%
5-NN	77.0%	99.54%
Parzen Windows	59.0%	99.17%
SVM	0.0%	98.58%

3.6.4.2 Identifying Tags From Manufacturer B

The second individual tag identification experiment is like the first one with only one difference; the data used for training and testing belong to tags from manufacturer B. First, we generate the ROC curves that describe that classifiers performance. The ROC curves are shown in Figure 3.10. To convert the ROC results into numerical data that makes it easier to compare, we calculate the AUC for each of the ROC curves. The AUC results are shown in Table 3.13. The results show that the performance of k-NN is much better than the performance of the other classifiers.

Next, we determine the TPR and the accuracy of the classifiers when identifying tags of model B. The results for the 5-fold cross validation are shown in Table 3.14.

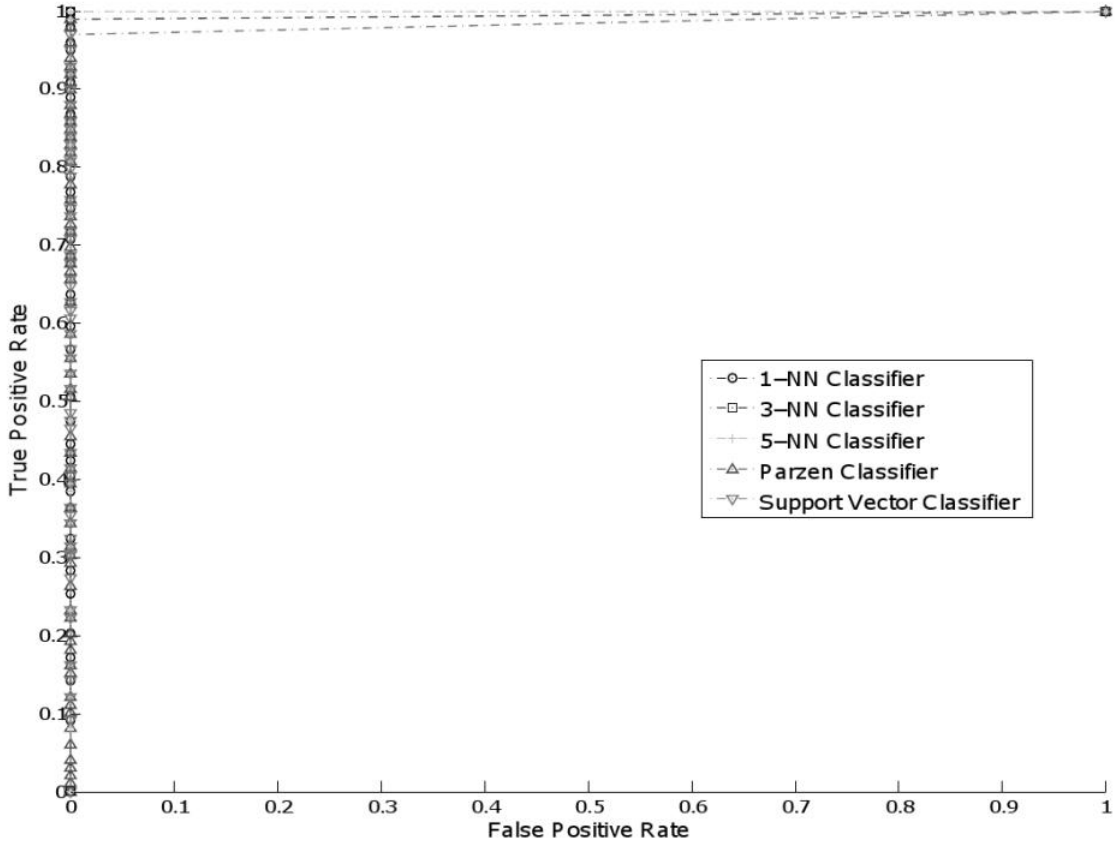

Figure 3.10: Manufacturer B ROC Curves using 66/34

Table 3.13: Manufacturer B AUC using 66/34

Classifier	AUC
1-NN Classifier	100.00%
3-NN Classifier	100.00%
5-NN Classifier	100.00%
Parzen Windows Classifier	99.49%
SVM	98.48%

3.6.4.3 Identifying Tags From Manufacturer C

Finally, we identify individual tags that belong to manufacturer C. As the case with the previous two experiments, we have 100 classes to classify. Each tag is represented by five instances which makes it hard to provide a good model description of a tag model. The

Table 3.14: Manufacturer B Tag Identification TPR and Accuracy Results using 5-Fold Cross Validation

Classifier	TPR	Accuracy
1-NN	90.6%	99.81%
3-NN	84.6%	99.69%
5-NN	77.0%	99.54%
Parzen Windows	58.6%	99.18%
SVM	5.0%	98.56%

generated ROC curves are similar to the ones that was generated for tags from manufacturers A and B. The ROC curves shows that k-NN has a better performance than Parzen windows and SVM. The ROC curves are shown in Figure 3.11 and its associated AUC is provided in Table 3.15.

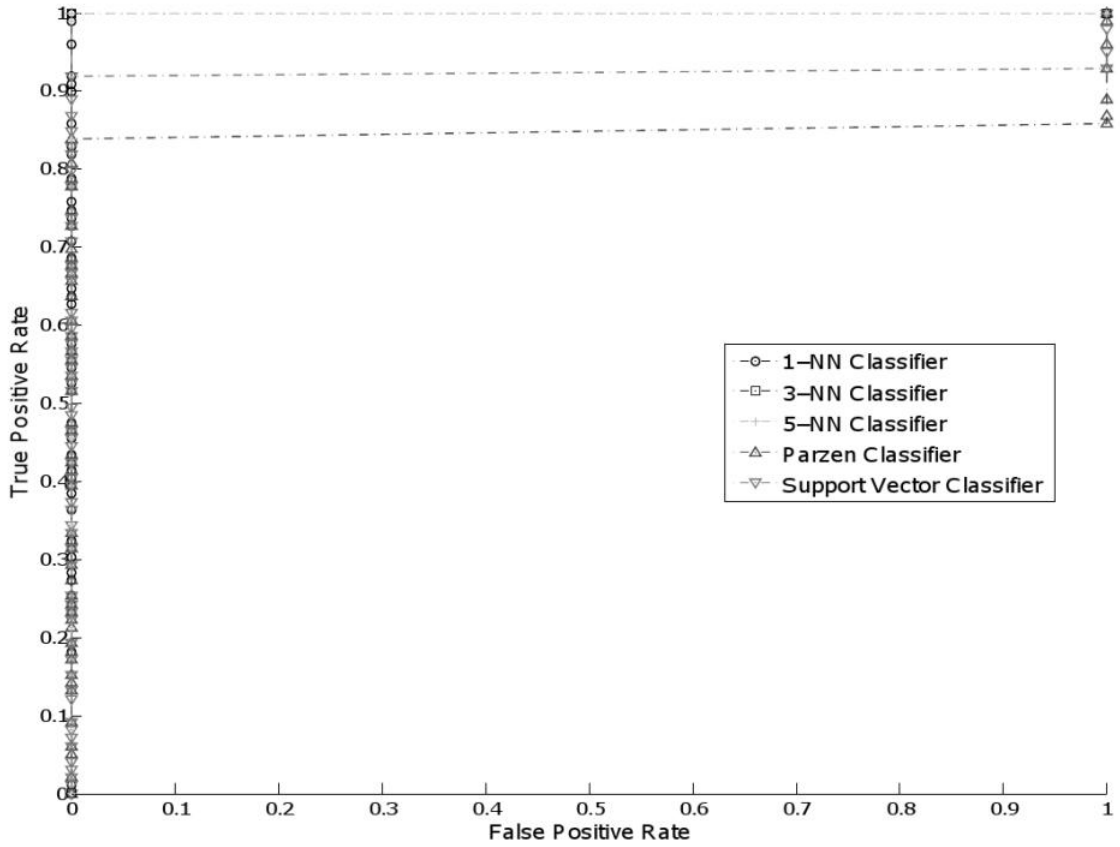

Figure 3.11: Manufacturer C ROC Curves using 66/34

Table 3.15: Manufacturer C AUC using 80/20

Classifier	AUC
1-NN Classifier	100.00%
3-NN Classifier	100.00%
5-NN Classifier	100.00%
Parzen Windows Classifier	84.85%
SVM	92.42%

The final test that was performed to determine the classifiers performance on the problem of identifying tags from manufacturer C is to determine the TPR and the accuracy using 5-fold cross validation. The results for the 5-fold cross validation are provided in Table 3.16.

Table 3.16: Manufacturer C Tag Identification TPR and Accuracy Results using 5-Fold Cross Validation

Classifier	TPR	Accuracy
1-NN	90.4%	99.81%
3-NN	84.4%	99.69%
5-NN	76.2%	99.52%
Parzen Windows	58.6%	99.17%
SVM	0.0%	98.53%

3.6.5 Results Discussion

In this section, we discuss the results that were obtained from the previously listed experiments. In the first experiment, we divided the data into either enrolled or unenrolled class. 66% of the data was used to train the classifiers and the remaining 34% was used to test the classifiers. The results show that k-NN performs better than the other classifiers. An average of 96.5% AUC was obtained when k-NN was used compared to Parzen windows and SVM with 49.18% and 61.09%, respectively. 10-fold cross validation was used to further support the AUC results. k-NN had an average of 96.28% compared to 70.00% for either Parzen windows or SVM.

In the second experiment, we used multi-class classifiers to identify tags based on their

manufacturer. The manufacturer identification results show almost perfect classification performance for all the used classifiers, which is clear by glancing at the ROC curve in Figure 3.8. The presented confusion matrix shows that instances which belong to manufacturer A can all be correctly identified, with occasionally miss-classification of manufacturer B or C. The 10-fold cross validation results show that an average of 99.91% TPR was obtained when k-NN is used. Parzen windows and SVM had a 99.93% and 95.00% TPR, respectively.

The final experiment that was conducted is to identify an individual tag. To perform the task of individual tag identification, we divided this experiment into three experiments. In each of the experiments, only tags from a particular manufacturer were used for training and testing the classification models. The major issue we faced when identifying individual tags is that only five instances are available as representatives of any tag class. Having only five instances to train and test the classifiers will create a large error margin since we are dividing the already small data set between testing and training. In other words, the classification models we are using are not converging. The ROC curves were generated using three instances per class to train a classifier models, which was tested with the remaining two instances. The ROC curves and their associated AUC calculations strongly recommend k-NN classifier to use over Parzen and SVM. To further support these recommendation we applied 5-fold cross validation and we calculated the average TPR. For manufacturer A, 1-NN, 3-NN, 5-NN, Parzen windows, and SVM had a 90.2%, 83.8%, 77.0%, 59.0%, and 0.0%, respectively. For manufacturer B, 1-NN, 3-NN, 5-NN, Parzen windows, and SVM had a TPR of 90.6%, 84.6%, 77.0%, 58.6%, and 5.0%, respectively. As for manufacturer C, 1-NN, 3-NN, 5-NN, Parzen windows, and SVM had a TPR of 90.4%, 74.4%, 76.2%, 58.6%, and 0.0%, respectively.

All the evidence from the previous results suggests that k-NN is the best choice for identification. k-NN works by comparing the test instance with all the training instances directly. It computes the difference between every training instance and the testing instance

and selects the k instances that are the closest to the testing instance. A weighting algorithm that is shown in Equation 1.1 is used to classify the presented testing instance.

Another point we would like to mention is how much the amount of available data influences the outcome results. TPR is the ratio between TP and $TP + FN$, while accuracy calculates the amount of correct classification from the perspective of one class. In the case of individual tag identification, each class has only one instance that belongs to it. Recall that we measured 100 tags for each manufacturer. This means that when calculating the TPR and the accuracy for class X, one instance does belong to X, while the remaining 99 instances belong to the remaining classes. For example, as long as the 99 instances are not confused with class X the TN count will be high, which drives the accuracy results to be high even if we have a 0.0% TPR. Therefore, the TPR can be low but the accuracy can be high. For example, when identifying tags from manufacturer A, the TPR for SVM is 0.0% while the accuracy is 98.58%. The previous example shows how easy it is to deceive a person who is reviewing the findings, if only partial results are provided.

Chapter 4

Hidden Markov Models

In this chapter, we disscus pattern recognition using Hidden Markov models (HMM). In order to identify tags using HMM, we focus on the time-voltage wave form that we extracted using [2]. A sample of the time-voltage waveform is shown in Figure 4.1. In order to be able to use HMM, the system to be modeled needs to generate some observable output. This output can be used directly as in Section 4.1, or it can be processed as in Sections 4.2 - 4.4.

The time-voltage waveform we are focusing on is the tag's transmission of the PC+EPC+ CRC data. As shown in Figure 4.1, the voltage values alternate between two main values, which is interpreted by the reader as binary code (ones and zeros).

In this section, five experiments are conducted. Each experiment is divided into two parts: manufacturer identification and individual tag identification. The difference between the five experiments is based on the type of observations we are using as inputs to the HMM. In the first experiment, we use the time-voltage waveform without any processing. The remaining four experiments involve some data processing. In the second experiment, we divide the time-voltage waveform into frames of ten observations each and we calculate the power in each frame using Equation 4.1, where V is the V_{rms} component of the frame's voltage sequence, while 50 refers to the used impedance. In the third experiment, the intervals that the voltage sequence stays in high or low are being calculated. Fig. 4.4 shows the transition points. The fourth experiment combines the power and timing. As in the third experiment, we calculate the intervals at which the voltage stays in the high and the low ranges, but we are adding the power in these intervals to the timing data. In the final experiment, we use the models generated by the second and third experiment and we combine

the results of these models by assigning weights to them. Thus, we are using both the time and power observations but instead of using a single model as in the fourth experiment, we are combining only the results of the models.

$$10log\left(\frac{V^2}{50} \times 1000\right) \tag{4.1}$$

All the HMM experiments were conducted using the Matlab toolbox [42] which was designed based on the theoretical knowledge presented in [54].

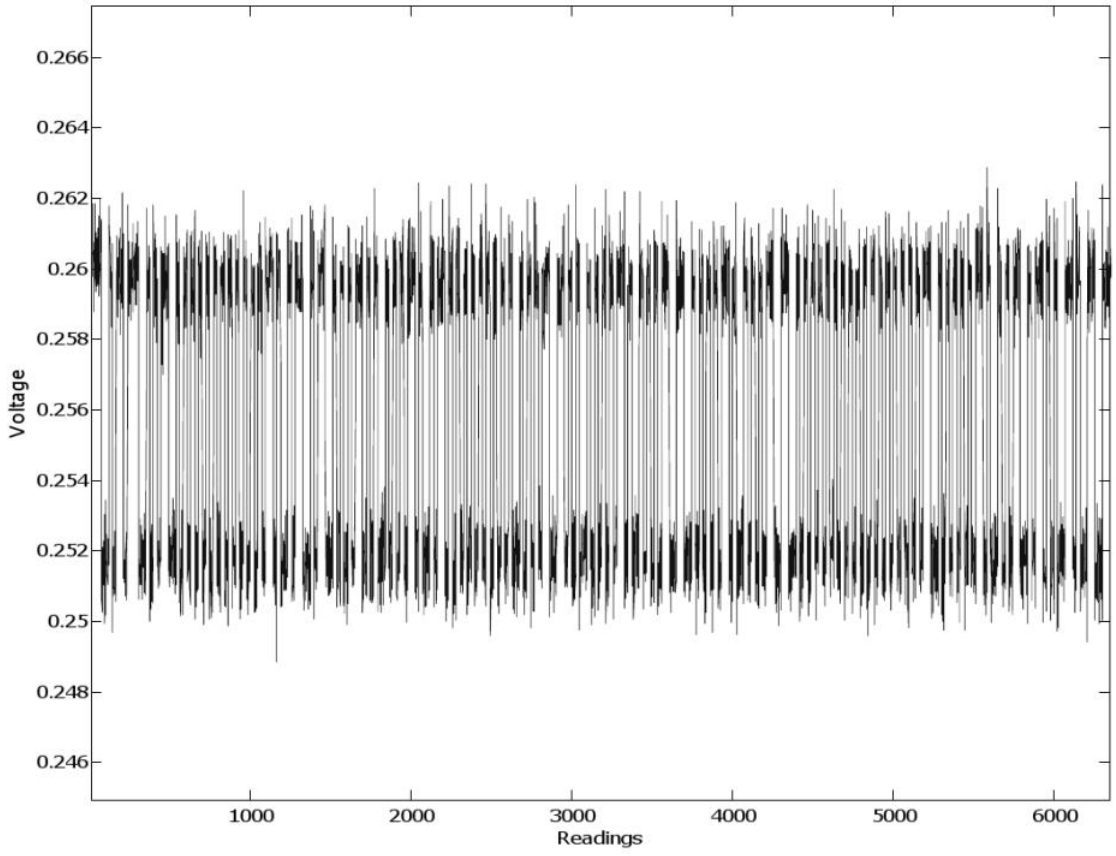

Figure 4.1: Time-voltage waveform sample

4.1 HMM Using Voltage Observations

In this section, we focus on the time-voltage waveform as the direct input to the HMM in order to identify a tag's manufacturer. No data processing is used. The data we have are from tags from three manufacturers (A, B, and C). We built the Markov models using 80% of the data (400 instances) and we tested the models using the remaining 20% of the data (100 instances).

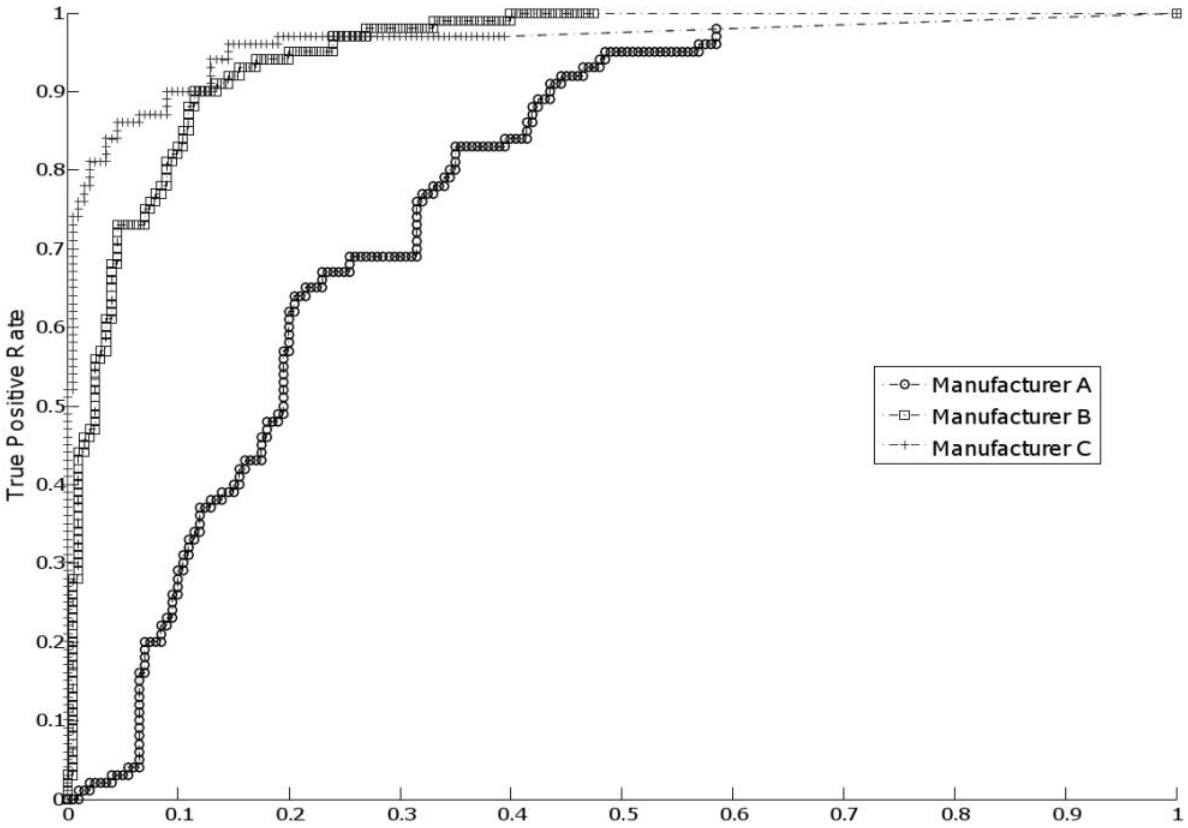

Figure 4.2: ROC curves for HMM applied to the Manufacturer Identification Problem using Raw Voltages as Observed Features

The confusion matrix that is in Table 4.1 shows that 28% of manufacturer A tags were confused as manufacturer B, 13% of manufacturer B tags were confused as manufacturer A, and 17% of manufacturer C tags were confused as manufacturer A. From the confusion

matrix, the TPR and the accuracy are calculated, and provided in Table 4.2. The results show that manufacturer B has the highest TPR with 87%, while manufacturer C has the highest accuracy with 92.67%.

Finally, the ROC curves that describe the performance of HMM in identifying each of the manufacturers are shown in Figure 4.2. The ROC curves matches the results obtained from the confusion matrix that show tags from manufacturer A with the poorest performance and tags from manufacturers B and C with close performance.

Table 4.1: Confusion Matrix for HMM applied to the Manufacturer Identification Problem using Raw Voltages as Observed Features

	Predicted Manufacturer		
Actual Manufacturer	**A**	**B**	**C**
A	68	28	4
B	13	87	0
C	17	1	82

Table 4.2: Manufacturer Identification TPR, Accuracy, and AUC Results Using Voltage Observations

Manufacturer	**TPR**	**Accuracy**	**AUC**
A	68.00%	79.33%	79.38%
B	87.00%	86.00%	94.87%
C	82.00%	92.67%	96.16%

4.2 HMM Using Power Observations

The second experiment we performed using HMM preprocesses the obtained time-voltage sequence. We divide the time-voltage sequence into frames. Each frame has 10 voltage readings (observations). We calculate the rms value of the frame's voltage sequence and use the V_{rms} values to calculate the power in each frame using Equation 4.1. The ROC curves in Figure 4.3 show an improvement on the performance compared with the ROC curves in Figure 4.2.

The confusion matrix for power observations is provided in Table 4.3 and the associated TPR and accuracy calculations are provided in Table 4.4. As expected, the performance for identifying all manufacturers improved. The TPR for identifying tags from Manufacturer A increased to 86% compared to 68% when using voltages directly without any preprocessing. Regarding Manufacturer B and C, the TPR jumped to 93% and 95% compared to 87% and 82%, respectively. The same improvement happened with the accuracy of the HMM. It jumped from 79.33%, 86.00%, and 92.67% to 92.00%, 93.67%, and 97.00% for manufacturers A, B, and C, respectively.

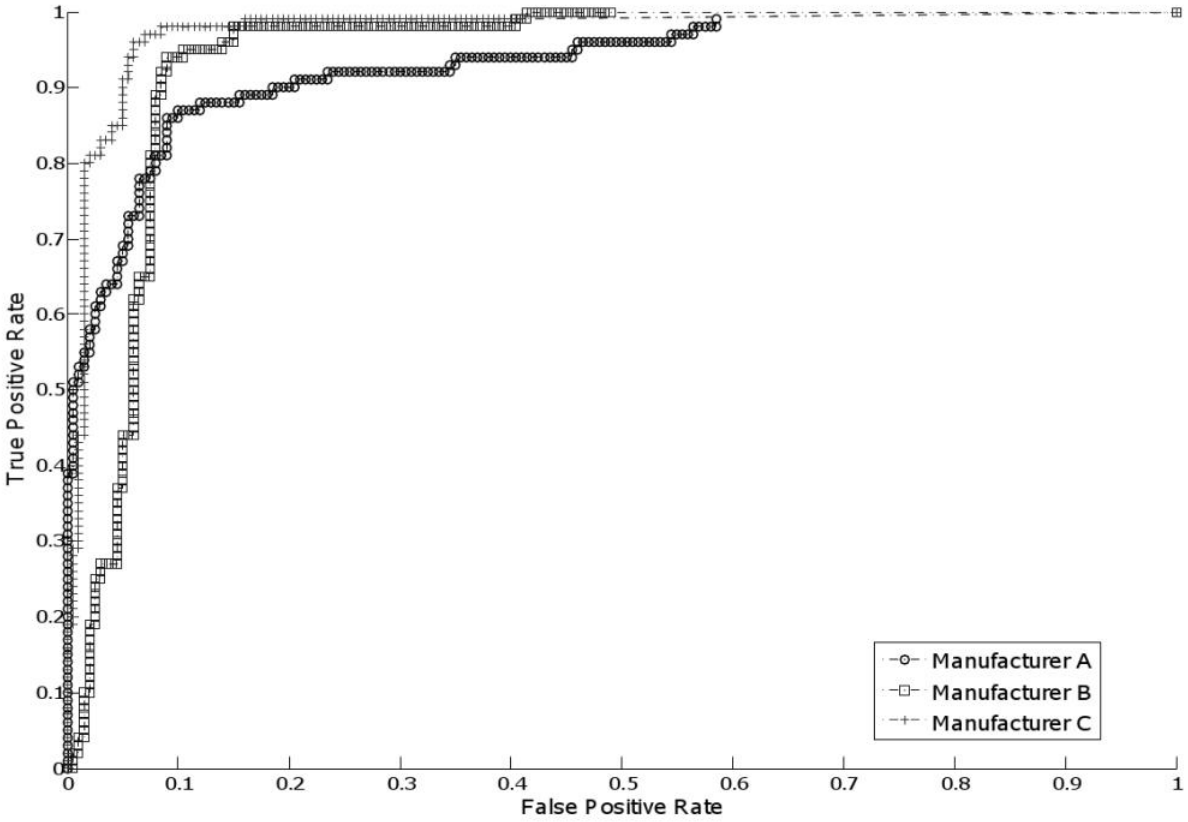

Figure 4.3: ROC curves for HMM applied to the Manufacturer Identification Problem using Power as Observed Features

The results from this experiment show that preprocessing the data will result in better performance. We expect that further preprocessing will improve the results even more. We

Table 4.3: Confusion Matrix for HMM applied to the Manufacturer Identification Problem using Power as Observed Features

	Predicted Manufacturer		
Actual Manufacturer	**A**	**B**	**C**
A	86	11	3
B	6	93	1
C	4	1	95

Table 4.4: Manufacturer Identification TPR, Accuracy, and AUC Results Using Power Observations

Manufacturer	TPR	Accuracy	AUC
A	86.00%	92.00%	93.89%
B	93.00%	93.67%	93.78%
C	95.00%	97.00%	97.50%

also expect that combining different data types will yield better performance as the case with classical pattern recognition algorithms.

4.3 HMM Using Time Observations

In this experiment, we use the transition points that are illustrated in Figure 4.4. We calculate the time between each of the points, and use the intervals as the observations input for the HMM. The ROC curves are shown in Figure 4.5 while the associated AUC are shown in Table 4.6. The generated ROC curve shows that identifying tags from manufacturer A performs much better than identifying tags from manufacturers B or C. By calculating the area under the ROC curves we found that the AUC for manufacturer A is 97.09% compared to 75.29% and 37.18% for manufacturers B and C, respectively. We must keep in mind that ROC is a technique to compare performance; more performance metrics are necessary to compare the work we did with other people.

The ROC curves clearly show that using timing for manufacturer A will yield much better results than tags from manufacturers B and C, but by how much? To answer this question

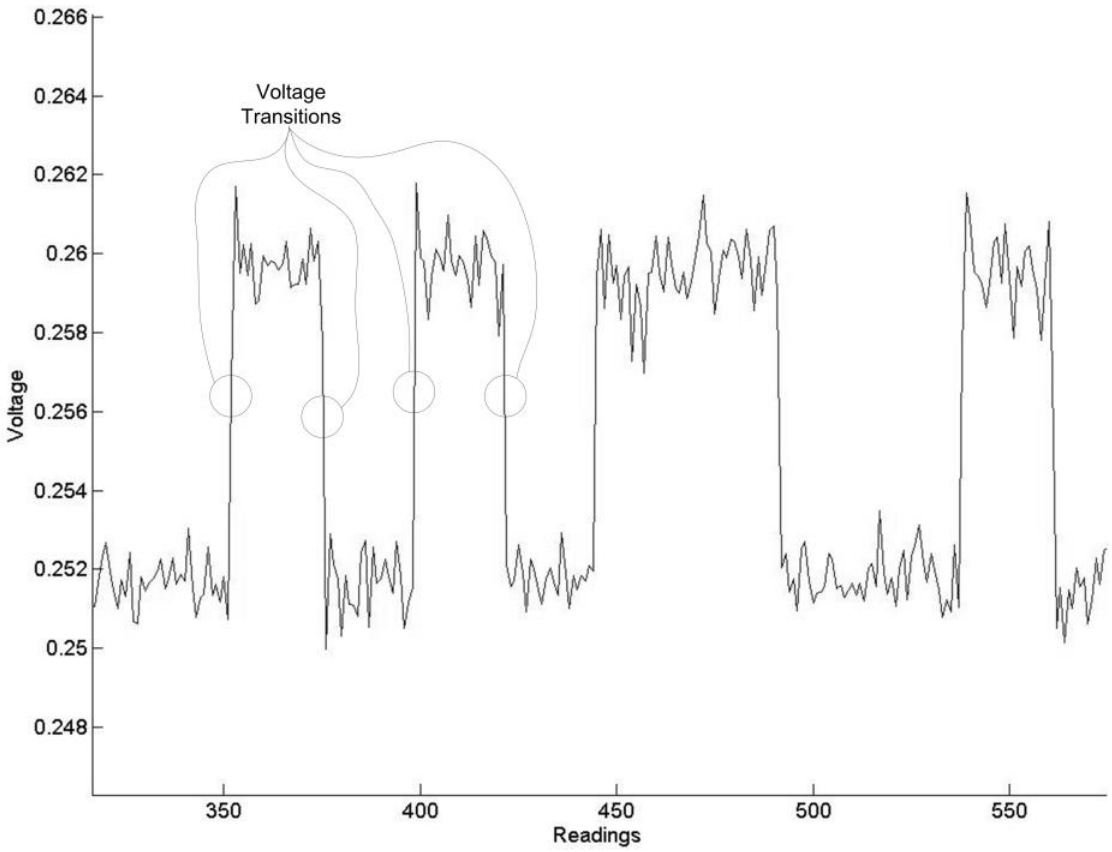

Figure 4.4: Voltage transitions between high and low ranges

we generate the confusion matrix, which is provided in Table 4.5. From the confusion matrix the TPR and the accuracy metrics are calculated and shown in Table 4.6. Although tags from manufacturer A have the best AUC, their TPR and accuracy are lower than that of B and slightly higher than C. The accuracy to identify tags from manufacturer A or C is 90.67% and it jumps to 97.33% if we are identifying tags from B. On the other hand, the TPR values are 91%, 94%, and 83% for tags from manufacturer A, B, and C, respectively.

Table 4.5: Confusion Matrix for HMM applied to the Manufacturer Identification Problem Using Timing as Observed Feature

	Predicted Manufacturer		
Actual Manufacturer	**A**	**B**	**C**
A	91	0	9
B	4	94	2
C	15	2	83

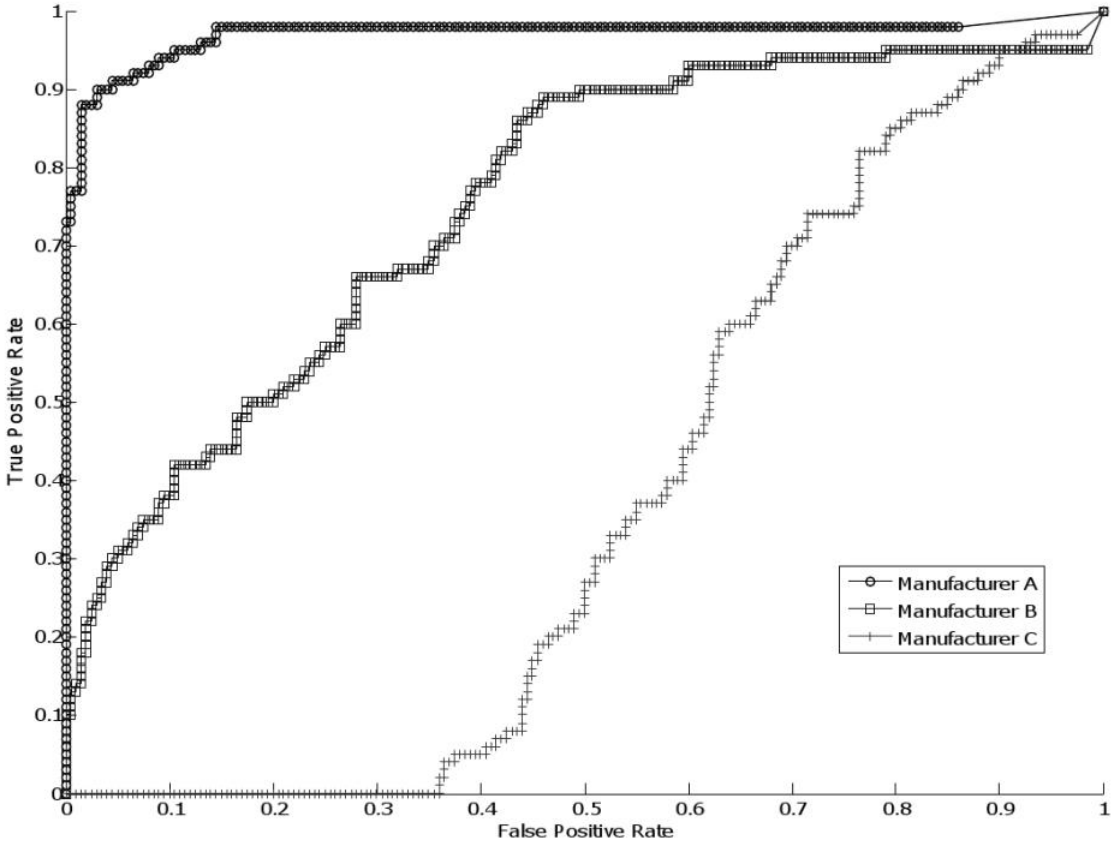

Figure 4.5: ROC curves for HMM applied to the Manufacturer Identification Problem using time as Observed Feature

Table 4.6: Manufacturer Identification TPR, Accuracy, and AUC Results Using Time Observations

Manufacturer	TPR	Accuracy	AUC
A	91.00%	90.67%	97.09%
B	94.00%	97.33%	75.29%
C	83.00%	90.67%	37.18%

4.4 HMM Using Time and Power Observations

By looking at the data we derived from the experiments presented in Section 4.2 and Section 4.3, we notice that tags from manufacturer A can be identified better using timing observations while tags from manufacturers B and C can be identified better using power

observations. In this experiment, we combine both the power and timing observations. As the case in Section 4.3, we divide the data we have into frames. Each frame consists of the voltage sequence that is between transitions (the transitions are illustrated in Figure 4.4). Each frame sequence is translated into two observations. The first observation represents the time it took to transmit the voltage sequence. The second observation represents the power that is contained within the voltage sequence. Power calculations are done using Equation 4.1.

First, we generate the ROC curves that give an indication of how good one model is compared to the other models. The resultant ROC curves are provided in Figure 4.6. By looking at the ROC curves, we notice that none of them are as good as we expected. It looks like combining power and timing will degrade the performance instead of improving it. Next, we generate the confusion matrix to further determine the performance of the models.

Table 4.7: Confusion Matrix for HMM applied to the Manufacturer Identification Problem using timing and power as Observed Features

	Predicted Manufacturer		
Actual Manufacturer	**A**	**B**	**C**
A	82	16	2
B	48	42	10
C	32	18	50

From the confusion matrix data that is provided in Table 4.7 and the performance metrics that are provided in Table 4.8, we notice that none of the manufacturer identification models provide acceptable results. Tags from manufacturer A have a TPR of 82%, an accuracy of 67.33%, and an AUC of 74.87%. Tags from manufacturer B have a TPR of 42%, an accuracy of 69.33%, and an AUC of 60.10%. Tags from manufacturer C have a TPR of 50%, an accuracy of 79.33%, and an AUC of 69.49%.

From this experiment, we notice that combining several measurements together into the same model decreases the performance. In the next section, we combine timing and power

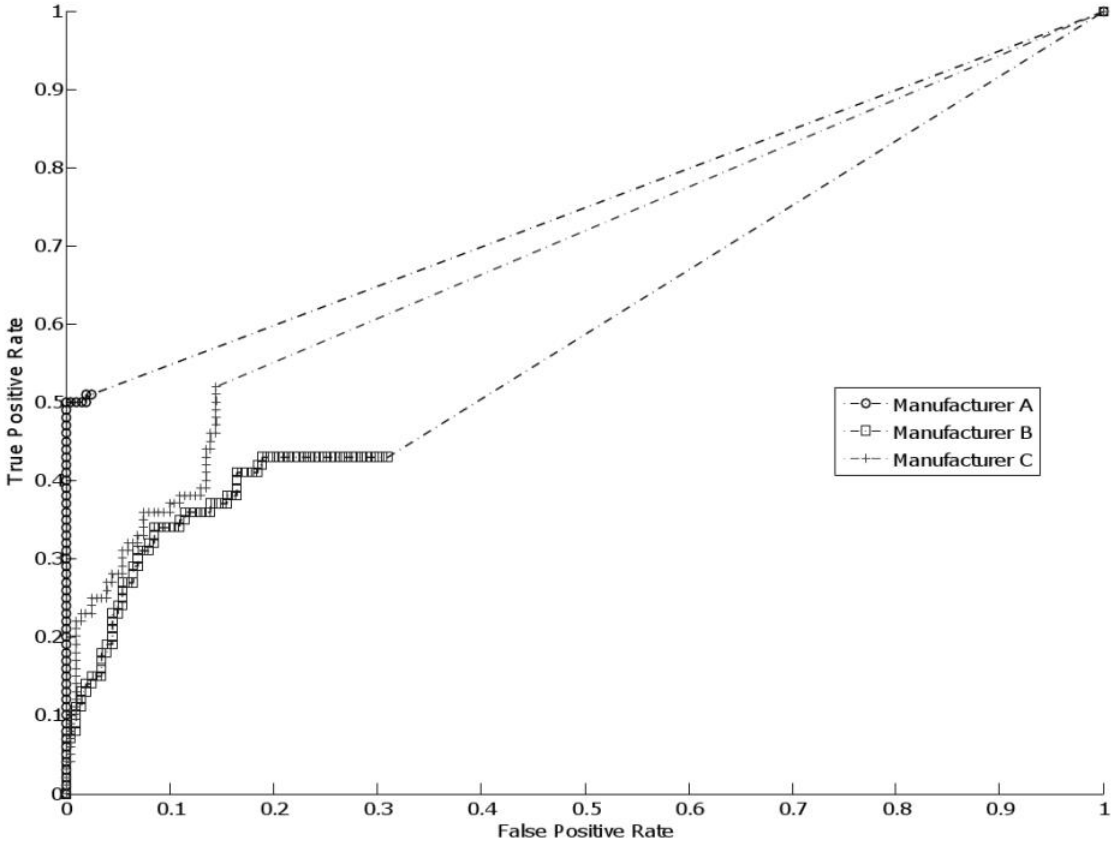

Figure 4.6: ROC curves for HMM applied to the Manufacturer Identification Problem using time and power as Observed Features

Table 4.8: Manufacturer Identification TPR, Accuracy, and AUC Results Using timing and power Observations

Manufacturer	TPR	Accuracy	AUC
A	82.00%	67.33%	74.87%
B	42.00%	69.33%	60.10%
C	50.00%	79.33%	69.49%

but we use separate models for each of them instead of one model.

4.5 HMM Using Time and Power Observations In Separate Models

The previous experiment showed that combining the power observations and the time observations into one stream will degrade the performance of the hidden Markov models. In this experiment, we use both the time observations and the power observations to identify tags. Instead of combining both observations into one model, we build two models: one model using power observations alone and the other model is built using time observations alone.

In order to formulate a final decision, we gather the decisions from both the time and power models using weights. Since we have two models to use and there are three classes to identify any given instance, a 3×2 weight matrix is used. In any row of the weight matrix, the relations shown in Equation 4.2 must be true where W_{it} is the weight given to instances from manufacturer i for the time model, and W_{ip} is the weight given to instances from manufacturer i for the power model.

$$W_{it} + W_{ip} = 1 \tag{4.2}$$

We use trial and error to determine the best weight matrix. Table 4.9 shows the weight matrix we used to generate the results shown in Table 4.11. The confusion matrix that is responsible for this performance is shown in Table 4.10.

Table 4.9: Weight Matrix

Manufacturer	**Power**	**Time**
A	1.0	0.0
B	0.4	0.6
C	0.1	0.9

The first thing we did is to combine the data we got by using testing instances on the

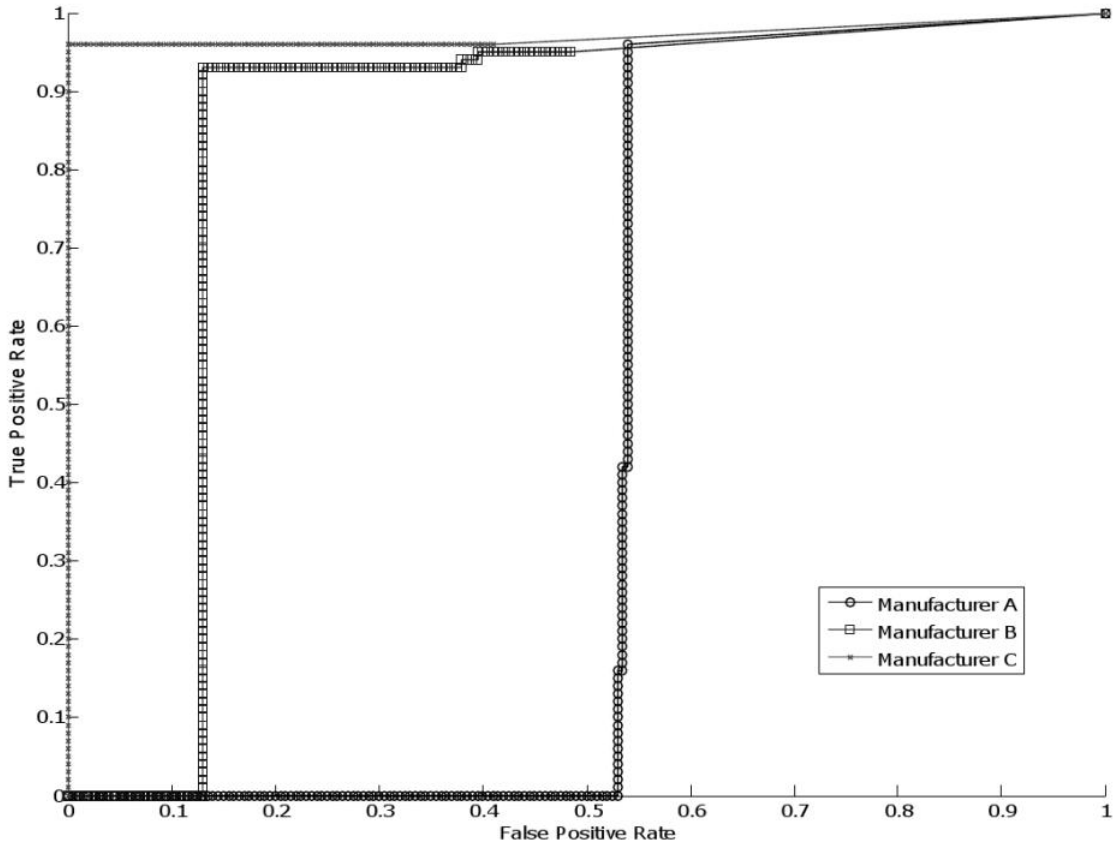

Figure 4.7: ROC curves for HMM applied to the Manufacturer Identification Problem using Time and Power as Observed Features in Separate Models

timing and power models. The Matlab code that uses the weights to combine the results from the time and power models is provided in Appendix C. The resultant ROC curves are provided in Figure 4.7. The extracted ROC data show that instances from manufacturer A does have an AUC of 45.37% compared to 83.42% and 97.18% for instance from manufacturer B and C, respectively.

Table 4.10: Confusion Matrix for HMM applied to the Manufacturer Identification Problem Using Timing and Power Observations in Separate Models

	Predicted Manufacturer		
Actual Manufacturer	**A**	**B**	**C**
A	91	0	9
B	0	97	3
C	0	4	96

Table 4.11: Manufacturer Identification TPR, Accuracy, and AUC Results Using Timing and Power Observations in Separate Models

Manufacturer	TPR	Accuracy	AUC
A	91.00%	97.00%	45.37%
B	97.00%	97.67%	83.42%
C	96.00%	94.67%	97.18%

After generating the ROC curves, we noticed that the performance improved compared to the performance of the models when the timing and the power observations were combined into a single observations stream. Encouraged with the results of the ROC curves, we generated the confusion matrix. The confusion matrix is provided in Table 4.10. The confusion matrix shows improvement. In fact, the resultant confusion matrix is much better than using either the timing observations or the power observations by themselves. From the confusion matrix, we calculated the TPR and the accuracy. The results are provided in Table 4.11.

4.6 HMM Manufacturer Identification Results Discussion

In the previous sections, we focused on using HMM to identify the manufacturer of a specific passive UHF RFID tag. Five experiments were performed. In all the experiments, we focused on the PC+EPC+CRC tag's transmission, which was extracted and processed to generate the observations for the identification process. The extracted observations include voltage, power, and timing data. Table 4.12 provides a summary of the manufacturer identification results.

In the first experiment, we used the time-voltage waveform "as is" without any pre-processing. The results showed moderate performance. Using time-voltage waveform as observations provided us with an average TPR of 79%, an average accuracy of 86%, and an average AUC of 90.13%. Tags from manufacturer A showed the worst performance, while

tags from manufacturer B showed the best TPR. Also, tags from manufacturer C showed the best accuracy. The results of using voltage observations are provided in Table 4.2.

After using raw voltage observations, we proceeded to use power observations. We divided the time-voltage observations into frames of 10 readings each. We calculated the V_{rms} values of each frame, and used Equation 4.1 to calculate the power. The results showed an improvement compared to when using raw voltages. An average TPR of 91.33%, an average accuracy of 94.22%, and an average AUC of 95.1% were obtained. Tags from manufacturer C had the best TPR, accuracy, and AUC. The detailed results are shown in Table 4.4.

Since preprocessing the PC+EPC+CRC transmission provides better results than using the raw observations, we explored other preprocessing techniques. By zooming into the time-voltage waveform, we noticed many transitions from low voltages to high voltages as shown in Figure 4.4. We processed the time-voltage wave sequence to find these transitions. We calculate the time between each of the transitions and used the resultant values as the observations. The results show an average TPR of 89.33%, an average accuracy of 92.89%, and an average AUC of 69.85%. Tags from manufacturer A had the best AUC. Tags from manufacturer B had the best TPR and accuracy. Surprisingly, tags from manufacturer C had the worst performance. Detailed results are shown in Table 4.6.

The fact that using either voltage or power yields acceptable performance for tags of manufacturer C while using time yields bad results for tags of same manufacturer led us to believe that combining features might provide us with better results than using any observation type by itself. Therefore, we used the same method to extract the timing information but we added to it the power between transitions. The results show an average TPR of 58%, an average accuracy of 72%, and an average AUC of 68.15%. Detailed results are in Table 4.8. Combining the time and power observations degraded the performance. Therefore, we created separate models for the power and the time and combined the results by applying

weights.

The final experiment we performed is based on the previous idea of using weights. The main rule to be enforced is that the sum of the weight must be equal to 1 as seen in Equation 4.2. We combined the models in Section 4.2 and Section 4.3. The weights are provided in Table 4.9. The reason behind using separate models rather than a single model with both observations is due to:

1. Combining observations will result in a degradation in the performance as seen in Table 4.8; and

2. Some manufacturer tags show a good performance when using time observations compared with time observations while instances from other manufactures show a good performance when power observations are used compared to using time observations.

The results when using separate models while applying weights to combine the results showed the best performance when using HMM. The results show an average TPR of 94.67%, an average accuracy of 96.45%, and an average AUC of 75.32%. With the exception of the AUC, we are getting the best TPR and accuracy among all performed HMM experiments.

4.7 Individual Tag Identification

In all of the previous sections, we focus on identifying a tag's manufacturer. We use different observations to determine which of them is the best for identifying a tag's manufacturer. In this section, we expand the previous work by identifying a specific tag rather than its manufacturer. We are identifying 100 tags of three manufacturers. In other words, each experiment will be repeated three times (since we have three manufacturers). Because of the large number of classes to identify, we will not be presenting the ROC curves and their

AUCs. We will only present the TPR and the accuracy measurements. Also, we will not be presenting the confusion matrices since each of them will have a size of 100×100. Since it is hard to present all the data for the 100 classes, we are calculating the average for the classes taken from each manufacturer model for both the TPR and the accuracy metrics. Also, because only five instances are available for each class, we are using four instances to build the class model and the remaining instance will be used to test the model. The data is shown in Table 4.12.

The results show that we are getting a very low TPR. The TPR ranges from 19% when using voltage observations on tags from manufacturer B, to only 1% when using power and timing observations in the same stream with tags from any manufacturer. On the other hand, an accuracy higher than 98% was obtained in all the tests. The fact that we are having a high accuracy and a very low TPR shows how important it is to find the correct ratio between the number of instances we used to build each of the class models, the number of instances we use to test each of the available models, and the number of models we have.

The results in Table 4.12 show the importance of balancing the number of test samples to the number of classes we want to classify. In the experiment we performed in this section, we only have one instance to test any of the models. Having one instance means that if we misclassify that instance a TPR of zero will result. Also having only four instance to build and train the models is not enough to have a good convergence of the models log-likelihood. Notice that even though we have an extremely low TPR, the accuracy is fairy high. The reason for this is because the accuracy calculations include the TN count. The formula to calculate the accuracy is shown in Equation 3.5. As shown in Figure 3.6, the TN count does not reflect accurate classification. The TN counts an instance that is not from class X and was not classified as class X. In other words, having one instance of each class while having a hundred possible classes will yield high accuracy and low TPR if we have a poorly trained models.

Table 4.12: Tag Identification Results using HMM

Manufacturer Identification Results						
	Manufacturer A		**Manufacturer B**		**Manufacturer C**	
Observations Used	**TPR**	**Accuracy**	**TPR**	**Accuracy**	**TPR**	**Accuracy**
Voltage	68.00%	79.33%	87.00%	86.00%	82.00%	82.67%
Power	86.00%	92.00%	93.00%	93.67%	95.00%	97.00%
Time	91.00%	90.67%	94.00%	97.33%	83.00%	90.67%
Power and time in the same observations stream	82.00%	67.33%	42.00%	69.33%	50.00%	79.33%
Power and Time in separate models	91.00%	97.00%	97.00%	97.67%	96.00%	94.67%
Individual Tag Identification Results						
	Manufacturer A		**Manufacturer B**		**Manufacturer C**	
Observations Used	**TPR**	**Accuracy**	**TPR**	**Accuracy**	**TPR**	**Accuracy**
Voltage	5.00%	98.10%	19.00%	98.38%	6.00%	98.12%
Power	8.00%	98.16%	16.00%	98.32%	18.00%	98.36%
Time	1.00%	98.02%	2.00%	98.04%	1.00%	98.02%
Power and time in the same observations stream	1.00%	98.02%	1.00%	98.02%	1.00%	98.02%
Power and Time in separate models	3.00%	96.06%	2.00%	98.04%	1.00%	98.02%

Chapter 5

Signal Authentication using Fingerprints

Previously, we discussed identifying an RFID tag based on its transmitted signals. Recall, identification is a one-to-many process in which an instance is mapped into a category that is already defined. In this chapter, we focus on authenticating an RFID tag based on its transmitted signals. Authentication is a one-to-one process in which we want to verify that an RFID tag is the one it claims to be.

In this chapter, we focus on tag and reader authentication. In tag authentication, we verify that the tag is the tag it claims to be. In reader authentication, we verify that the reader we are using to read and to communicate with the tags is not compromised. We develop an authentication system based on fingerprints. There are three parts of the authentication system: RFID tag(s), RFID reader(s), and a back-end server. A graphical representation of the system is shown in Figure 1.3. As Figure 1.3 shows, we only have one server, which is completely trusted. This server will have direct links to any RFID reader in the system. The readers have the responsibility of communicating with the tags, extracting the feature vector from these tags, and communicating with the server to determine if the tags are authentic or not. An authentication system can be divided into two main processes: the enrollment process and the verification process. In this chapter, we will use the words authentication and verification interchangeably.

5.1 Enrollment Process

During the enrollment process, tags are queried and the representative feature vector is measured. Figure 5.1 shows the flowchart of the enrollment process. During the enrollment process, each tag is queried X number of times. For each query, the tag's response is recorded, features are extracted, and the feature vector is stored. After collecting X feature vectors, several statistics of each of the features, such as mean, variance, and standard deviation are calculated. By calculating the means of the features, an average feature vector is created. This average feature vector will be used for any future authentication request. The received statistics, the feature vector, and the X feature vector readings are stored on the server. The hash of the mean feature vector is then calculated and stored on the tag. A lightweight hash function such as PRESENT-80 can be used [8]. PRESENT-80 is designed specifically to be compatible with RFID environment. It was designed with all the power, memory, and size constraints in mind. The server can either store one hash that describes the entire feature vector or it can store a single hash for each of the available features (a vector of N hashes, where N is the length of the feature vector).

We assume that the features are independent of each other and that they follow a Gaussian distribution. Since Gaussian distribution is assumed, the mean and the standard deviation are needed so that the distribution for each of the features can be estimated. Since each of the features' distribution parameters will be used in the verification process, a reference to them must be stored on the same tag that was used to extract those values. Upon request, the tag will send the stored hash value to the reader, which in turn will send it to the server that has the necessary data.

To summarize, during enrollment, multiple feature vectors of the tags to be enrolled are collected. For each feature in the feature vector, the mean, variance, standard deviation is calculated. The feature vectors and the other extracted values are stored on the server. This

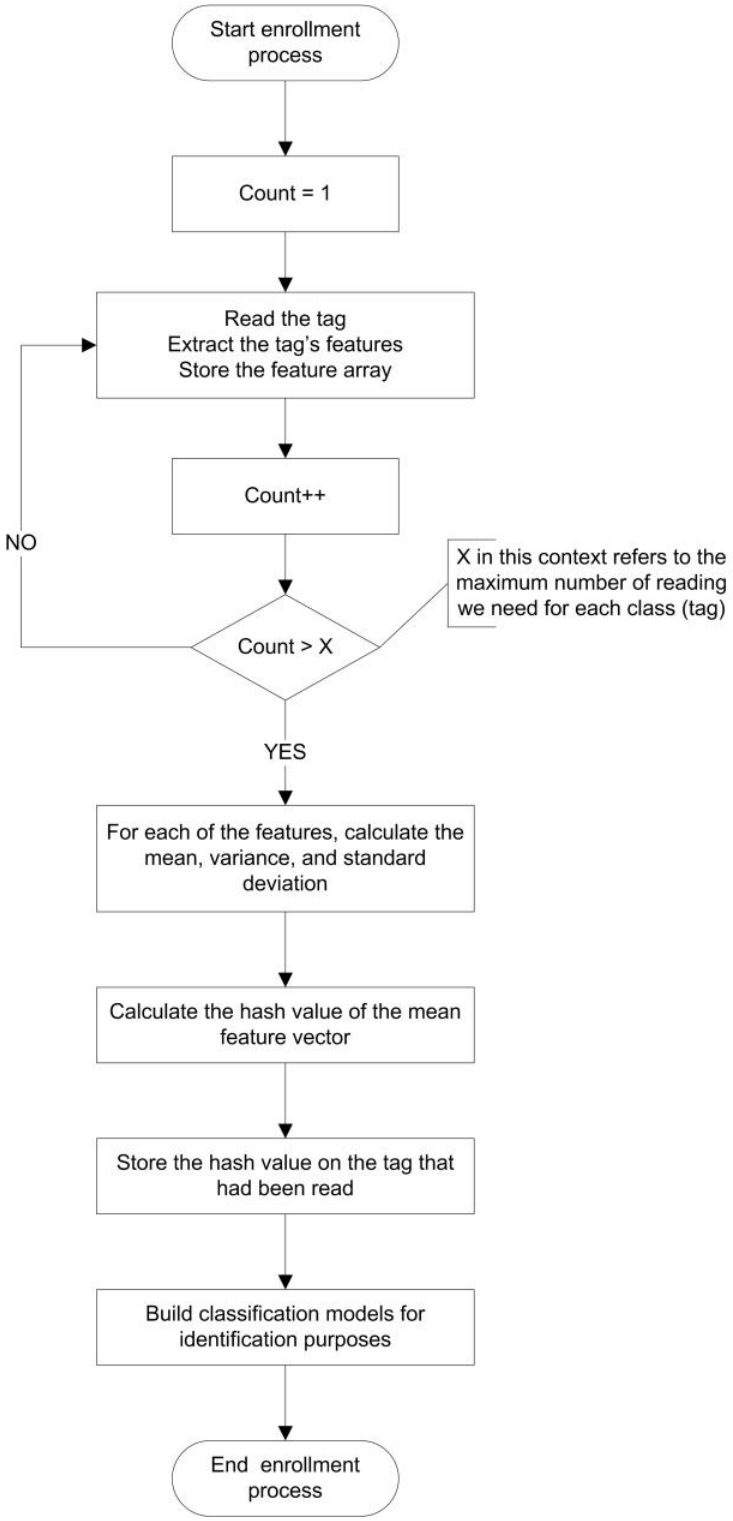

Figure 5.1: Enrollment process of the RFID authentication system

table is keyed using a hash of the mean feature vector. A copy of the hash is stored on the enrolled tag to be used as a reference for future authentication requests.

5.2 Verification Process

During the verification process, a tag that was previously enrolled is authenticated. Recall that during the enrollment process, a hash of the feature vector is stored in the memory banks of the tag. When the tag is to be authenticated, the stored hash value is retrieved and sent to the server. At the same time, the reader uses the data transmitted by the tag to extract a feature vector that describes this tag. Using the hash value, the server retrieves the average feature vector calculated at enrollment, including weights, and the standard deviation of each of the features. The server will then communicate with the reader to determine if the presented tag is authentic or not.

In order to determine the tag's authenticity, the distance between the newly read feature vector and the mean feature vector is calculated. A simplified authentication system based on the distance is presented in Figure 5.2. Different features have different impact on the tag's authenticity. Features with high variance are assigned lower weights than features with low variance, (the reason for the before-mentioned statement is due to the presence of noise when the tag is being read). Having high variance means that the feature reading will be less accurate, while having low variance means that we can maintain high confidence in the read feature. To calculate the variance, we normalize all the features so that they are between 0 and 1, as shown in Equation 5.1. In Equation 5.1, max and min refers to the maximum and minimum values of a certain feature for a particular tag, while x_i refers to an instance of the feature to be normalized. Once all the features are normalized, the variance is calculated using Equation 5.2.

$$normalized = \frac{x_i - min}{max - min} \tag{5.1}$$

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 \tag{5.2}$$

Using variance, we calculate the weight of each of the features. The weights must satisfy the condition in Equation 5.3. Since the feature with the less variance will have the highest weight, we are using Equation 5.4 to calculate the weight of each feature. The weights and variance calculations occur during enrollment and are retrieved using the hash value during the verification process. Using the weights, we verify each weighted feature based on distance as shown in Figure 5.2. The distance is compared to a threshold value that is based on the enrolled feature vectors. If the distance between the recently read feature vector (the one that was read for authentication) and the mean feature vector is less than the threshold, then the tag is authentic; otherwise, the tag is considered to be unauthentic.

$$\sum_{i=1}^{N} w_i = 1 \tag{5.3}$$

$$w_i = \frac{\frac{1}{var_i}}{\sum_{j=1}^{N}\frac{1}{var_j}} \tag{5.4}$$

Next, we introduce a complete verification process the uses distances between enrolled data and recently read data. A flowchart of this verification system is provided in Figure 5.3.

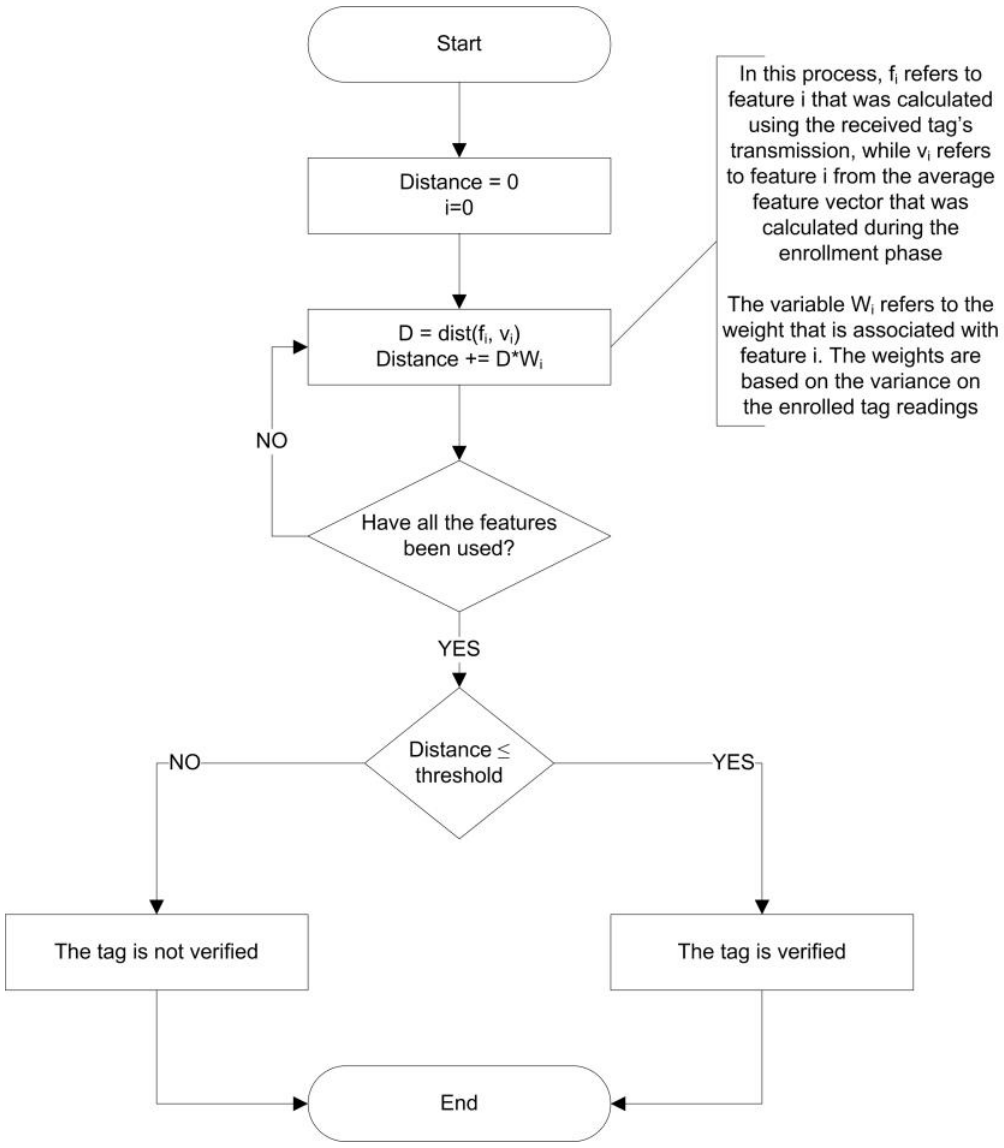

Figure 5.2: How to calculate the distance between feature vectors

First, we introduce a simple authentication system that optimistically assumes no attempts to fool the system, which is shown in Figure 5.3. The verification process starts with the reader collecting the tag's response for a specific query. The reader then extracts the feature vector from that response. The reader asks the tag to transmit the hash value stored in its memory banks. Upon receiving the hash value, the reader will send it to the back-end server, which uses it to retrieve the mean feature vector, and the thresholds associated with each of the features. Using the retrieved mean feature vector and the associated thresholds,

the server sends these values to the reader. The reader computes the distance between the received feature and the tag's extracted feature. Using the threshold, the reader determines if that feature can actually be generated from the tag or not. The decision is sent back to the server. Once the server receives the reader's response, it sends the next feature mean value. Once all the features' mean values are sent to the reader and the decisions are collected, the final decision of the tag's authenticity is made. The final decision is made using Equation 5.5, where D is the final decision score, N is the length of the feature vector, d_i is the reader's decision for feature i, and w_i is the weight of the i^{th} feature.

$$D = \sum_{i=1}^{N} d_i \times w_i \tag{5.5}$$

Once the final score is calculated, it is compared with a threshold. If that decision is greater that the threshold value, then the tag is authentic. If it is not greater than the threshold, then it is not authentic. The previously discussed authentication technique has its drawbacks.

1. The algorithm ignores the possibility of having a compromised reader; and
2. There are no precautions taken to guard against replay, eavesdropping, or man-in-the-middle attacks.

In this remainder of this section, we focus on solving the compromised reader problem. To solve the other problems, we assume that the reader and the server are using modern communication authentication protocols [10] [9] [31] [37] [50].

RFID readers are usually geographically separated and in enough numbers to monitor all passing objects with attached RFID tags. An intruder can theoretically attack one or

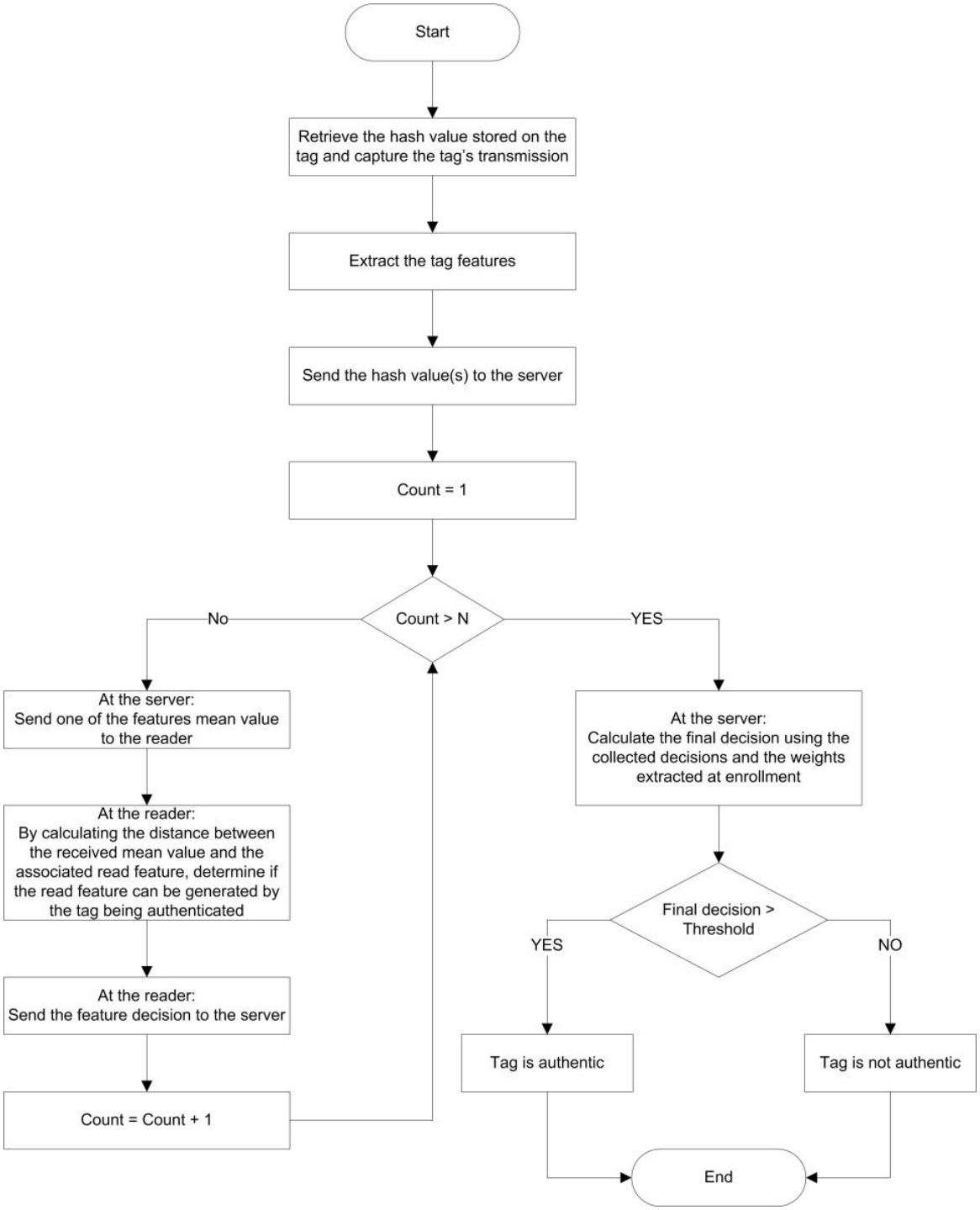

Figure 5.3: Simple tag authentication process

more of the readers and install special software to send false data to the server regarding the authenticity of a certain tag. (**We must be able to distinguish between compromised**

readers and normal readers). In order to do so, we introduce the concept of decoys. A decoy is a feature value that could not possibly be generated by the tag we are authenticating. For example, assume that a tag was enrolled that has a time feature with values between 400 microseconds and 500 microseconds. An example decoy time feature is 100 microseconds. A normal reader will respond to the server's decoy by saying that the tag failed that test, while a compromised reader may respond by saying that it passed the test. In other words, a compromised reader will always respond to the server by saying that the tag passed each of the tests. Once the server receives a positive response to a decoy, the probability that the reader is compromised increases. The error could be caused by noise or measurement problems. An authentication system using decoys is shown in Figure 5.4. A mixture of decoys and real measurements should be sent to verify.

Before sending the features to the reader, the server must determine how many and which of the features to replace by decoys. The number of decoys used must be such that the reader will have very small probability to guess the decoys. At the same time, it must be such that the remaining features can authenticate a tag with high certainty. In the system we propose, the server selects at random which of the features is to be replaced with decoys. Once the server determines the decoys position in the feature vector, it replaces the associated features with the decoys, and sends the new feature vector to the reader.

When the server sends a decoy, the normal response from the reader is that the tag did not pass that feature's test. Therefore, the reader sends a fail result to the server. A compromised reader may respond with a pass score for each of the received features (including the decoys). When the server receives a pass score for a decoy, the server knows that the reader may be compromised. In the system we present, the tests refer to the distance calculated between the current read feature and the mean of that feature (sent by the server). If the calculated distance is near zero, then the result of the test is assumed to be a pass; otherwise, the test result is a failure. Since we are using distance, there is no hard score that can be considered

as a pass. A passing score depends entirely on the distribution parameters of that feature (mean, variance, and standard deviation). There are two actions the server could follow when receiving the reader responses:

1. The server could continue to send features to the reader even if the reader sent a pass score for a sent decoy; or
2. The server could keep monitoring the responses sent from the reader. If the number of decoys with received pass responses exceeds a certain amount, the server could ignore the reader or consider it compromised.

For the first scenario, the server checks if the responses from the reader for the decoys were all as expected (failed score). If the responses are as expected, then the server combines the received scores of all the genuine features (the ones that are not decoys) into a final score that is compared to a threshold. If the final score is greater than the threshold, the tag is authentic; otherwise, the tag is not authentic. It is worth mentioning that if this scenario is used, the server can send the feature vector in one session and receive the final decision vector, thus, reducing the communication cost between the reader and the server.

If the second scenario is used, the reader builds a reputation profile for each of the available readers. When the server sends a decoy to the reader and receives an unexpected response (pass), the reader's reputation at the server decreases. As more decoys are answered with a passing score, the reader's reputation keeps on decreasing until it goes beyond a certain threshold. When the reputation of the reader drops below that threshold, the server assumes that the reader is compromised, and it takes appropriate actions such as ignoring the reader, notifying the system administrator, etc.

Next, we will introduce the methods to determine the threshold value, number and value of the decoys, and the tag score. All of the previous problems are solved using the same

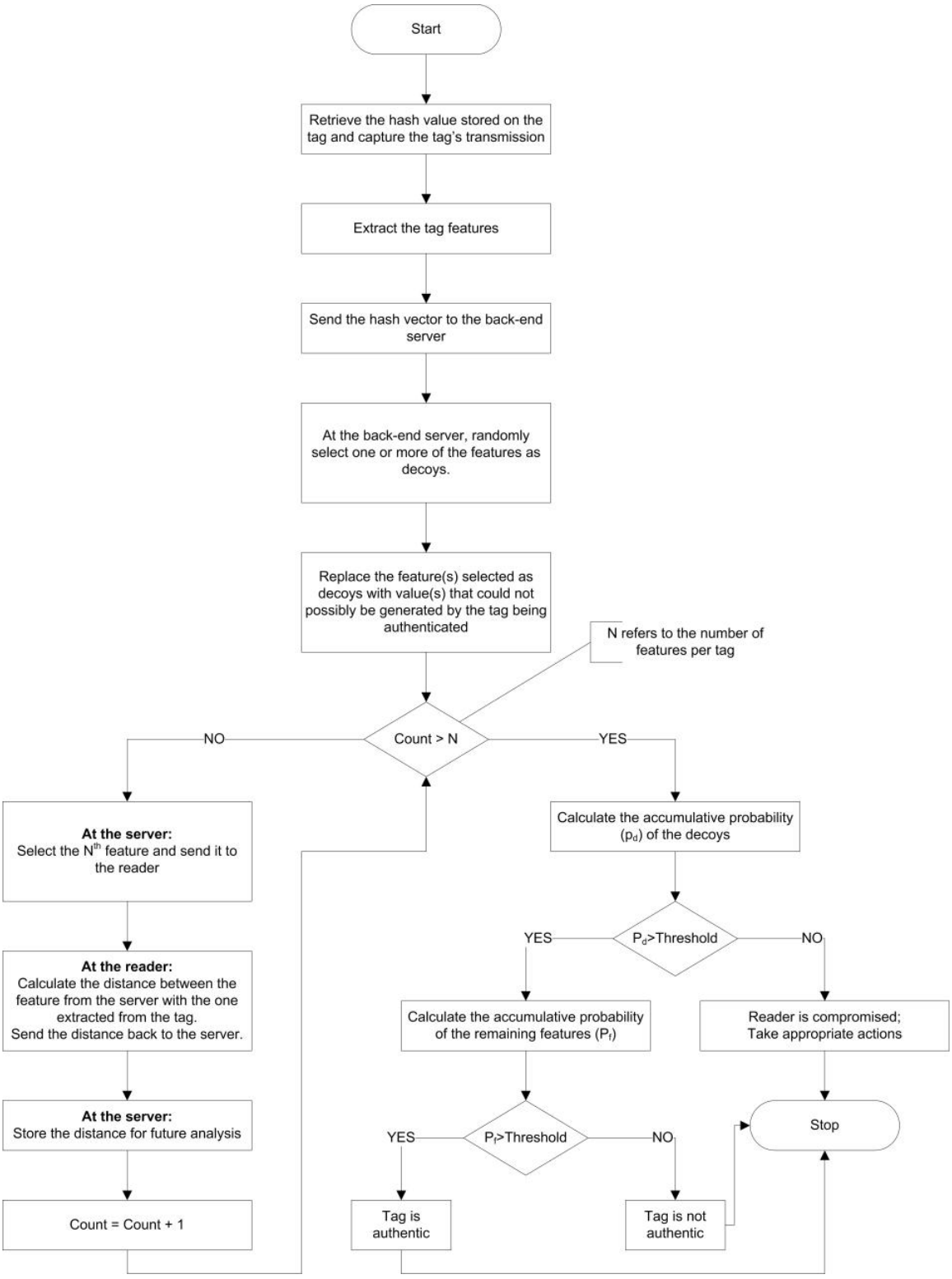

Figure 5.4: Proposed tag authentication process

principle. There are two assumptions that are related to the features extracted from the tags transmission.

- The features are assumed to follow a Gaussian (normal) distribution; and
- The features are assumed to be independent of each other.

The Gaussian distribution (also known as Normal distribution) is a continuous probability distribution that has a bell-curve shaped probability density function. The Gaussian function is shown in Equation 5.6, where x is the random variable, μ is the mean of the random variable, σ^2 is the standard deviation, and e is the Euler number ($e = 2.7182818284...$).

$$f(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{5.6}$$

As shown in Figure 5.5, the center of the bell-curve is located at the mean, while the spread of the curve depends upon the standard deviation of the random variable. Figure 5.5 shows three random variables that are in the interval [-40, 40]. The curve with the lowest peak has the highest standard deviation ($\sigma^2 = 8$), the curve with the highest peak has the smallest standard deviation ($\sigma^2 = 2$), while the middle curve has a standard deviation ($\sigma^2 = 4$). All the curves are centered around zero, which is the mean of the random variables.

After determining the distributions of each of the observed features for each of the enrolled tags, the threshold at which we will consider that a tag is authentic or not is calculated. The threshold we propose is a probability based threshold. Out of the enrolled values of a specific tag, for each of the features, we select the value that is furthest from the mean value of that feature. Using the Gaussian function, we determine the probability that this value occurs. (The reason behind selecting the furthest feature read is because the

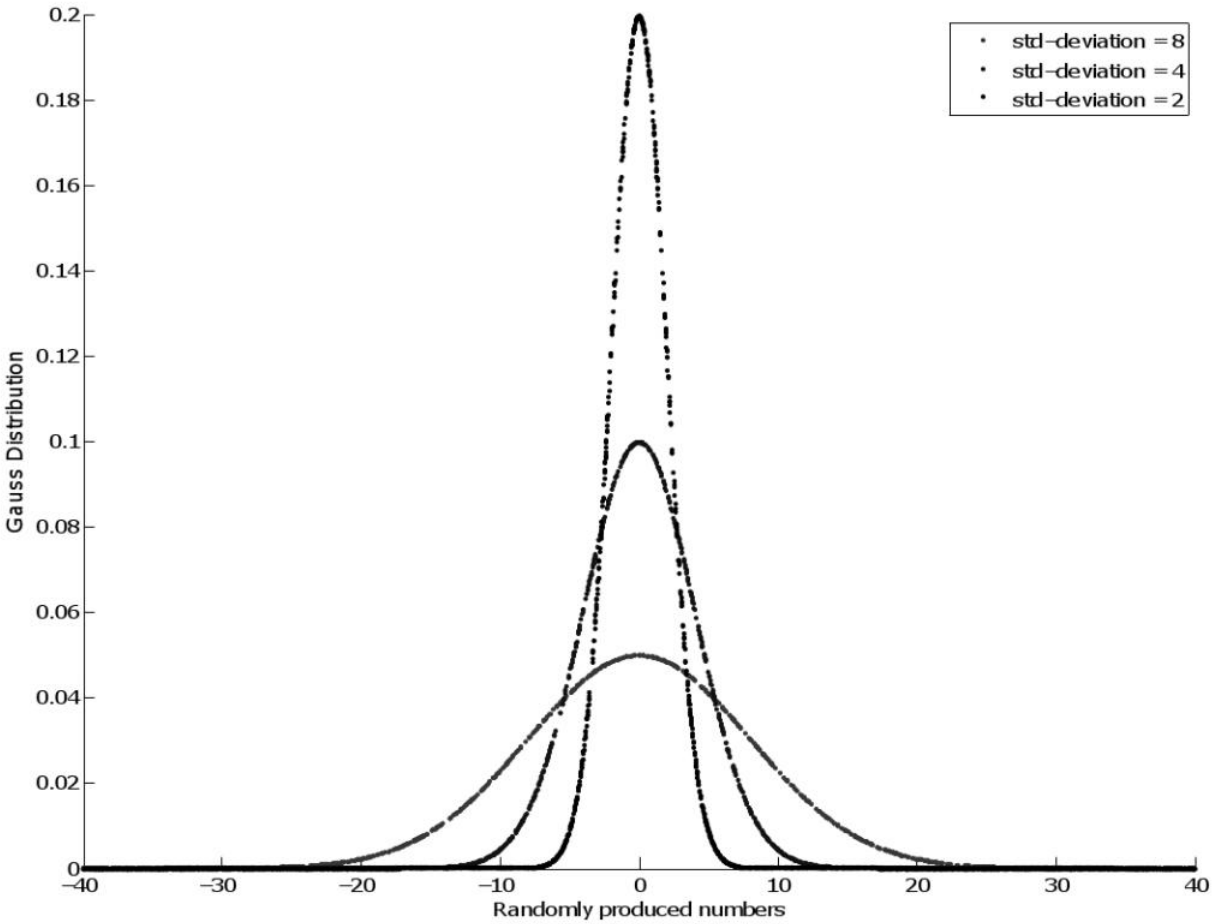

Figure 5.5: Gaussian Distribution Examples

furthest feature value happens when the noise is at its highest). Once all the probabilities are determined, we use Equation 5.7 to calculate the overall tag's threshold, where *dist* is the distance between the mean of the i^{th} feature and the measured values with the maximum distance of the i^{th} feature, and $f(x)$ is the Gaussian function introduced in Equation 5.6.

$$\prod_{i=1}^{N} \int_{\mu-dist}^{\mu+dist} f(x)dx \tag{5.7}$$

To use the proposed threshold, the server sends the mean values to the reader. The reader

calculates the distance between the mean value and the measured feature value, and sends the distance back to the server. The server calculates the probability that this distance can happen using Equation 5.8, where *dist* refers to the distance between the feature's current read and the mean of that feature. After sending all of the mean feature vector and receiving all of the calculated distances from the reader, the server compares the resultant cumulative probability $\left(\prod_{i=1}^{N} prob_i\right)$ and it compares it with the threshold calculate using equation 5.7. If the current cumulative probability is less than the threshold, then the tag is authentic; otherwise, the tag is unauthentic.

$$prob_i = \int_{\mu-dist}^{\mu+dist} f(x)dx \tag{5.8}$$

Figure 5.6 describes the choice of using Equation 5.8 as the threshold. In the rest of this section, the following symbols are extensively used:

- m, refers to the feature's observed read that is the furthest from the feature's mean value (only observations during enrollment are considered);
- d_m, refers to the distance between the mean of the feature being tested, and the furthest enrolled read from the mean;
- f, refers to the observed read of a feature;
- d_f, refers to the distance between the feature value sent from the server and the feature observation the reader just extracted from the tag to be authenticated;
- c, refers to a decoy sent from the server to the reader;
- d_c, refers to distance between the sent decoy and the feature observation the reader just extracted from the tag to be authenticated;

- d_{mc}, refers to the distance between the sent decoy and the furthest read from the feature mean (m);
- C, refers to the probability an observed feature occurs near the sent decoy (very small probability);
- M, refers to the probability that observations between the feature mean and the m observation occurs;
- N, refers to the number of available features; and
- D, refers to the number of decoys used.

By referring to Figure 5.6, the probability of a reading to occur that does not exceed the furthest one measured during enrollment for a certain feature (m) is equal to the area $A+2B$, which we refer to as M. On the other hand, the probability of the recently extracted feature is equal to the area A. For a tag to be authentic, the relation in Equation 5.9 must hold true. In other words, the area under the curve that is bounded by reads closer to the mean (A) should be less than the area bounded by the reads that are furthest from the mean (M).

$$\forall_i A_i \leq M_i \tag{5.9}$$

Equation 5.7 does not take into account the existence of decoys in the system. To accommodate decoys, we seperate the features selected as decoys from the genuine features. The authentication process is separated into two processes.

1. **Compromised reader test**: To perform the compromised reader test, we only focus on the responses for the decoys. When the server receives a response for a decoy, it

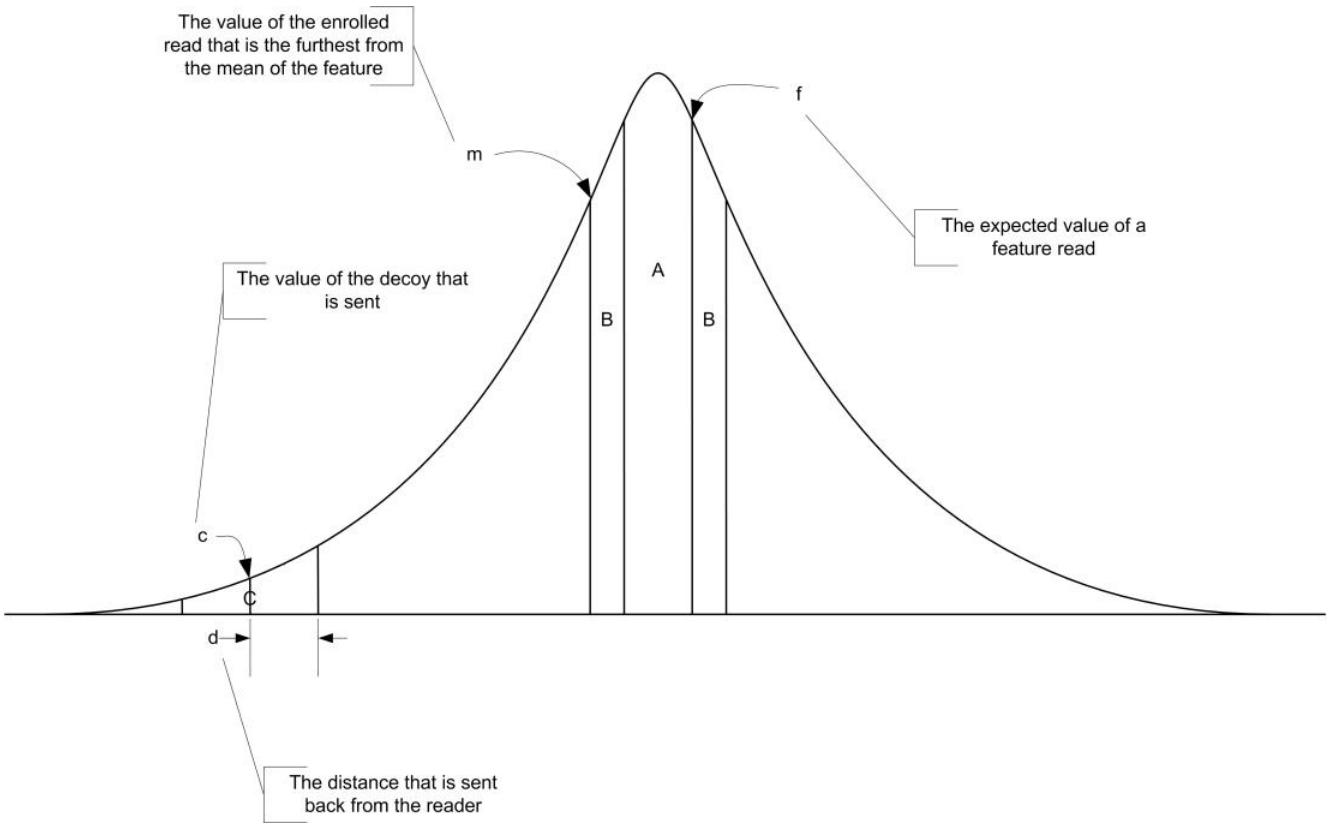

Figure 5.6: Threshold and Probability Relations

expects the distance to be larger than the distance between the decoy and furthest read from the mean (m). The probability of a feature to occur within the distance sent back by the reader can be calculated using Equation 5.10.

$$\int_{c-d_c}^{c+d_c} f(x)dx \tag{5.10}$$

If the reader tries to fool the server, a small distance will be sent back to the server, which results in a probability equal to area C shown in Figure 5.6. The reader is assumed to be compromised if the cumulative probability of the decoys is less than a certain threshold. The previous relation is outlined in Equation 5.11.

$$\begin{cases} \prod_{i=1}^{D} \int_{c-d_c}^{c+d_c} f(x)dx \ll \prod_{i=1}^{D} \int_{2c-m}^{m} f(x)dx, & \text{if } c < \mu \\ \prod_{i=1}^{D} \int_{c-d_c}^{c+d_c} f(x)dx \ll \prod_{i=1}^{D} \int_{m}^{2c-m} f(x)dx, & \text{if } c > \mu \end{cases} \tag{5.11}$$

2. **Tag authenticity test**: If the reader is assumed not to be compromised, we proceed with the tag authenticity test. We follow the same approach we followed before, but instead of using all the features (N), we will use only genuine features (not decoys) ($N-D$). The relation that outlines the previous argument is introduced in Equation 5.12.

$$\prod_{i=1}^{N-D} \int_{\mu-d_f}^{\mu+d_f} f(x)dx \leq \prod_{i=1}^{N-D} \int_{\mu-d_m}^{\mu+d_m} f(x)dx \tag{5.12}$$

Having decoys in the authentication system enables us to identify compromised reader(s). Some other questions arise from the use of decoys. What if the server stops sending features to the reader when enough decoys are sent and the reader's responses indicate that it is compromised? In addition, how many decoys should be sent to the reader? The solution for both questions is to have the server check the status of each of the decoys as soon as they arrive.

When the server receives the response of the first decoy, it checks the probability of the distance calculated by the reader using Equation 5.13, which holds true if the reader is compromised. If the resultant probability is less than the expected one, the server sends a decoy again after a number of sent features that are half the ones that preceded the first decoy. If the response for that decoy is still less than the expected response, another decoy is sent but with half the number of features that existed between the first and the second decoy. The previous process will continue until we start sending decoys without any genuine

features in-between.

$$\begin{cases} \int_{c-d_c}^{c+d_c} f(x)dx \ll \int_{2c-m}^{m} f(x)dx, & \text{if } c < \mu \\ \int_{c-d_c}^{c+d_c} f(x)dx \ll \int_{m}^{2c-m} f(x)dx, & \text{if } c > \mu \end{cases} \tag{5.13}$$

In the proposed system, the authentication process could be adapted to the number of decoys, and which features to use. In a non-adaptive solution, the server compares the accumulative probability with a precalculated threshold. In the adaptive solution, the system depends on the frequency of sent decoys, rather than their accumulative probability.

5.3 Simulations and Experiments

The authentication system is simulated in Matlab. Assumptions for the simulation are listed below.

- The features for the tags are assumed to be independent of each other;
- All of the features are assumed to follow a Gaussian (normal) distribution;
- The selected decoys are uniformly distributed throughout the feature set;
- For all the features, all the reads during the enrollment phase are within 1 standard-deviation of that feature's mean; and
- Decoys in any of the experiments do not exceed 50% of the available features.

In order to test the system we developed, we started by experimenting without the existence of decoys. After that we moved into testing the system with the existence of decoys.

5.3.1 Experiments without decoys

We first simulated the system, in which there are no decoys. We divided this experiment into three sub-experiments. The first sub-experiment is the effect of the number of features on the tag's score and threshold. The next sub-experiment was to simulate the authentication system when the correct data is retrieved from the server (for example, can we authenticate tag i if its enrolled data was retrieved). And finally, we simulated the case in which the wrong data is retrieved from the server (for example, does the authentication system returns unauthentic if we retrieved tag j data and we want to authenticate tag i, where $i \neq j$.

5.3.1.1 Effect of the number of features

In this experiment, we vary the number of features. The results are shown in Table 5.1. As Table 5.1 shows, no matter how many features we have, we can always authenticate a tag. The results also show that having more features will result in more authenticity confidence (the ratio between the tag's threshold and score increases when the number of features increase). For instance, if we have 2 features, the threshold will be 1.8947 times greater than the score. On the other hand, the threshold will be 8.0213×10^{42} times greater than the score when 128 features are being used.

5.3.1.2 Authentication when correct data is retrieved from the server

In this experiment, we model the correct behavior of the authentication system. We generate data for M tags, each of them has N features, and each of them was read R times during the enrollment phase. In the experiment, we assume that $M = 100$, $N = 64$, and $R = 100$. The simulation selects tags at random and for each of the tags it tries to authenticate it. The tag is assumed to be authentic when its score is less than the threshold value calculated during

Table 5.1: The affect of the number of features on the tag's threshold and score

Number of Features	Tag Threshold	Tag Score	Threshold/Score
2	0.4315	0.2275	1.8967
4	0.1573	0.0080	19.6625
8	0.0285	1.8288×10^{-05}	1.5584×10^{003}
16	7.1445×10^{-04}	4.1138×10^{-09}	1.7367×10^{005}
32	6.6531×10^{-07}	3.1823×10^{-23}	2.0907×10^{016}
64	1.7671×10^{-13}	2.3583×10^{-38}	7.4931×10^{024}
128	4.9488×10^{-26}	6.1696×10^{-69}	8.0213×10^{042}
256	4.1920×10^{-51}	1.1316×10^{-145}	3.7045×10^{094}
512	1.9599×10^{-100}	1.222×10^{-302}	1.6038×10^{202}
1024	5.4951×10^{-202}	0.0	———

Table 5.2: Simulation of authenticating 20 tags when correct data is retrieved from the server

Tag#	Tag Threshold	Tag Score	Tag#	Tag Threshold	Tag Score
1	8.5009×10^{-17}	1.9334×10^{-34}	11	4.4131×10^{-18}	1.2669×10^{-38}
2	1.0815×10^{-17}	6.7660×10^{-40}	12	3.2853×10^{-17}	1.2683×10^{-40}
3	1.7627×10^{-18}	6.1977×10^{-37}	13	1.6499×10^{-18}	3.9920×10^{-40}
4	2.3009×10^{-18}	5.9758×10^{-40}	14	2.7262×10^{-18}	5.4441×10^{-38}
5	3.7615×10^{-17}	6.5763×10^{-38}	15	1.7537×10^{-17}	1.2832×10^{-39}
6	4.0801×10^{-17}	5.1705×10^{-36}	16	3.9561×10^{-18}	2.8603×10^{-36}
7	1.5178×10^{-17}	4.3137×10^{-38}	17	3.7821×10^{-17}	1.5061×10^{-39}
8	2.8605×10^{-18}	1.8108×10^{-37}	18	1.8532×10^{-17}	9.7054×10^{-38}
9	2.8426×10^{-19}	5.7277×10^{-41}	19	4.6523×10^{-18}	1.3258×10^{-34}
10	2.5225×10^{-18}	8.7400×10^{-41}	20	1.0544×10^{-18}	7.8258×10^{-40}

the enrollment phase. The results show that we can authenticate 100% of the selected tags. Table 5.2 shows the results of the authentication process. The simulation chose to select 41 tags randomly out of the tag pool. Table 5.2 clearly shows that even though the same number of features is being used for all the tags, having different distributions will greatly affect each tag's threshold and score. Due to the large amount of data, we chose to present a subset of the data we have (the first 20 tags). The results show that for the test data, the ratio between the threshold and score ranges between 1.8645×10^{12} and 4.1020×10^{25}.

5.3.1.3 Incorrect authentication

In the final experiment without decoys, we test the system assuming the incorrect enrollment data was retrieved from the server. In other words, we are simulating a situation where an error occurred to the key sent from the reader to the server resulting in retrieving data for a tag different than the one being authenticated. As before, we are using 100 tags, each of them was read 100 times, and each of them has 64 features. The simulation selects a subset of the enrolled tags and it tests these tags with every other enrolled tag. The results should be that none of the tests are positive (authentic).

The results show that all the tags chosen as test instances had unauthentic result when compared with other tags. In other words, tag i was tested unauthentic when compared with tag j, where $i \neq j$. Table 5.3 shows a subset of the results obtained. Table 5.3 displays the results of three tags ,tag #25, tag #37, and tag#13, when compared with tags 1 through 10. Compared to Table 5.2, we notice that when the correct data is retrieved from the server, the tag's threshold is higher than the generated score. On the other hand, when incorrect data is retrieved from the server, the score will be higher than the threshold. The previous behavior is what we seek with the proposed authentication system. The results we obtained show that 100% of the tested tags have a negative (unauthentic) test result.

5.3.2 Experiments with decoys

In this section, we test the performance of the authentication system with decoys. In the first test, we assume the presence of a non-compromised tag or a reader. The simulation selects at random a set of tags to test and selects at random which of the features to be replaced by decoys. After that, the presence of a compromised reader is tested.

The results of this section experiments are presented in Table 5.4 and Table 5.5. In these

Table 5.3: Tag authentication with incorrect data retrieved

Tag#	Tag Threshold	Tag #25 Score	Tag #37 Score	Tag #13 Score
1	2.4759×10^{-18}	0.2561	0.3364	0.9483
2	4.1933×10^{-18}	1.0000	0.0324	0.4241
3	2.6997×10^{-18}	0.7813	0.0451	0.7480
4	1.5664×10^{-18}	0.9983	0.8221	0.5295
5	8.1903×10^{-18}	0.0381	0.0937	0.2731
6	2.4278×10^{-17}	0.1286	0.1117	0.4834
7	5.7622×10^{-17}	0.4828	0.0238	0.0247
8	1.4781×10^{-16}	0.4846	0.5914	0.8220
9	1.4092×10^{-19}	0.2173	0.2042	0.4106
10	1.3847×10^{-18}	0.9482	0.8960	0.5971

tables, two new columns are introduced to determine if the reader is compromised or not; the third and the fourth columns of each table. The third column titled **probability of max value**, represents the probability a feature has a value that is within d_{mc} distance of the decoy, while the fourth feature titled **probability of features**, represents the probability a feature has a value that is within d_c distance from the decoy. If the reader is not compromised then the value in the third column should be less than the value in the fourth column, while if the reader is compromised, the value in the third column should be higher than the value in the fourth column.

5.3.2.1 Authenticating tags in the presence of decoys and a normal reader

Unlike the previous experiments, in this experiment we include decoys. We test how the system reacts to the presence of decoys. In the simulation we generated 100 tags, each of which has 64 features. The simulation selected 69% of the tags to test. It also used 42% of the available features as decoys.

The results show that the system can authenticate 100% of the tags. The threshold vs. tag score results are shown in Table 5.4. Even though the number of features used are the same in all the performed experiments, the threshold and the score results are higher than

Table 5.4: Authenticating tags with the presence of decoys

Tag Threshold	Tag Score	Prob of max value	Prob of feature
2.6519×10^{-10}	8.1889×10^{-24}	7.1169×10^{-19}	5.8706×10^{-15}
9.7449×10^{-11}	7.8574×10^{-22}	8.0913×10^{-19}	2.0819×10^{-14}
3.5566×10^{-11}	2.3426×10^{-21}	6.7517×10^{-19}	1.5234×10^{-13}
2.4727×10^{-10}	4.6838×10^{-21}	9.2259×10^{-19}	2.7768×10^{-15}
2.0763×10^{-10}	1.9147×10^{-18}	6.5602×10^{-19}	3.1434×10^{-13}
4.2520×10^{-10}	6.9613×10^{-21}	7.6760×10^{-19}	1.0021×10^{-13}
1.4210×10^{-10}	3.3384×10^{-22}	1.1411×10^{-19}	9.1995×10^{-15}
4.6365×10^{-11}	9.3059×10^{-20}	1.3375×10^{-19}	1.6410×10^{-14}
2.5691×10^{-11}	4.5402×10^{-19}	7.9476×10^{-20}	2.4478×10^{-15}
3.8282×10^{-11}	2.4190×10^{-22}	2.9991×10^{-18}	1.4461×10^{-15}

the ones in Table 5.2. The reason for that is because we are excluding decoys and only using genuine features.

5.3.2.2 Authenticating tags in the presence of decoys and a compromised reader

In this experiment, we simulate the presence of a compromised reader. A compromised reader sends back to the server a small distance between the feature read from the tag and the mean value sent from the server. Because of the presence of decoys, the distance the reader should send back should be abnormally large.

As before, the simulator was instructed to generate data for 100 tags, each of which has 64 features. The simulator choose 7% of the tags to test. The simulator also chose 10 features and changed them to decoys, which are 2 standard deviations from each of the features' mean (the user will be asked to determine the distance between the feature mean and the decoy in terms of standard deviation). The results showed that the server was able to catch every attempt a compromised reader tries. 100% of the 7 attempts were caught and a compromised reader alert was issued. The results are shown in Table 5.5. If you notice the data in Table 5.5, it shows that the tag is authentic if the score and the threshold for each of the tags are compared. But since we are using decoys, the last two columns of the table

need to be compared before comparing the first two columns (threshold and score). Since the reader is compromised, it sends small distances in an attempt to fool the system. These small distances will lead to small cumulative probability that is shown in the last column of Table 5.5. If the reader is not compromised, then the data in column 4 must be larger then the data in column 3. But, as the experiment shows, the data in column 4 is less than the data in column 3, which leads us to believe that the reader is compromised.

Table 5.5: Authenticating tags in the presence of decoys and a compromised reader

Tag Threshold	**Tag Score**	**Prob of max Value**	**Prob of feature**
6.5991×10^{-10}	1.1028×10^{-28}	2.7019×10^{-18}	1.2841×10^{-47}
4.6687×10^{-10}	1.2691×10^{-21}	8.6397×10^{-19}	2.4215×10^{-46}
1.2713×10^{-10}	8.8701×10^{-23}	3.1241×10^{-18}	5.9741×10^{-45}
2.6211×10^{-10}	5.0280×10^{-24}	1.9485×10^{-18}	1.6536×10^{-47}
8.9660×10^{-12}	1.5748×10^{-22}	3.8511×10^{-18}	1.9575×10^{-45}
6.0950×10^{-11}	1.2868×10^{-31}	4.5505×10^{-18}	9.5575×10^{-46}
1.0448×10^{-11}	2.9443×10^{-26}	1.2095×10^{-18}	4.5280×10^{-49}

Chapter 6

Conclusions and Future Work

6.1 Summary

In this work, a study of electromagnetic signals and the ability to distinguish among them based on a specific set of chosen features is presented. RFID technology has been selected as a representative of electronic devices with wireless communication capabilities. Using the extracted features, we demonstrated that it is possible to identify the manufacturer of a specific tag as well as the individual tag with high accuracy. To identify RFID tags, we used classification techniques such as k-NN, Parzen windows, and SVM. Also, we used HMM technique to build models that describe the tag's identity. To determine the performance of each of the techniques, metrics such as TPR, accuracy, and AUC were calculated.

To identify RFID tags using regular classification techniques, a methodology to extract and select the features was introduced. After extracting the features from the tag's transmission, a subset of the features is selected to use as a training set for the classification techniques. To select the features, we started by testing all features combinations to determine if a particular feature pattern reveals itself. Once all feature combinations have been tested, we applied data correlation techniques to eliminate highly related features. Finally, we ranked the features using Fisher scoring to indicate which of the features should be removed if the feature set needs to be scaled down. The performed classification experiments showed that we can identify tag's manufacturer with almost 100% TPR and a 100% AUC. The experiments also showed that we can identify individual tags with 90.2% TPR, 99.8% accuracy, and 100% AUC if 1-NN is used.

In addition to classic classification techniques, HMM was used to identify tags (both manufacturer identification and individual tag identification). To use HMM, we extract a stream of observations from the transmission of the RFID tag(s). The observations were used in their raw format for the first experiment, while we processed them for the rest of the experiments. By processing the time-voltage waveform, timing and power observations were extracted. A combination of timing and power observations lead to the best performance if they are weighted. The best performance provided us with a 97.00% TPR, 97.67% accuracy, and 97.18% AUC. HMM has that advantage over classic classification techniques of being portable due their small size. Also, HMM has the ability to generate identification decisions very fast compared to classification techniques that degrade in performance when the data set becomes large.

After proving that using the appropriate observations (or feature set) we can identify a specific tag, we moved our attention to signal authentication problem. To authenticate RFID tags (or transmitted signals in general), we devised a technique in which we used decoys. A decoy is a feature value that could not possibly be generated by the tag we wish to authenticate. The authentication system we developed has two stages of operations: enrollment phase and verification phase. During enrollment, the tag (or signal) to be authenticated in the future is queried multiple times. Each time, the feature vector is recorded and stored for future use. After the enrollment is completed, the set of collected feature vectors is processed to extract statistics for each of the features (such as: mean, standard deviation, weight, etc.). When the authenticity of a tag (or signal) needs to be verified, a new feature vector is extracted and compared with the feature vectors extracted during enrollment. Using the extracted statistics, we can determine if the new feature vector was generated by the same tag (signal) that was used during enrollment.

One of the issues that we focused on is the possibility of having a compromised reader. A compromised reader will attempt to trick the system into treating unauthentic tag (signal)

as an authentic one. The decoys in the system are present to mitigate against such problems.

A simulation was devised to test if the proposed authentication system can in-fact authenticate tags. Based on the simulation, the system can authenticate 100% of the tags when the reader is not compromised. The system can also detect any compromised reader in the system.

6.2 Contributions

The contributions of this work can be summarized as follows:

1. Developed a methodology to determine the set of features from which we can identify signals with high accuracy;
2. Identified features to identify passive UHF RFID tags;
3. Investigated k-NN, Parzen windows, and SVM and we were able to identify a tag's manufacturer with 100% TPR and AUC (manufacturer is equivalent to tag IC chip). also, we were able to identify individual tags with 90.2% TPR, 99.8% accuracy, and 100% AUC when 1-NN is used;
4. Investigated the usability of hidden Markov model to identify signals; and we were able to achieve a 97% TPR, 97.69% accuracy, and 97.18% AUC;
5. Created a probabilistic system that can authenticate signals based on the distribution of the extracted signal features; and
6. Introduced the concept of decoys in the signal's authentication system from which we can identify and authenticate wireless signals even with the presence of compromised signal readers.

6.3 Future Work

In the future, we would like to expand this work. One of the issues we ran into is the limited size of the data we have. We would like to measure a larger set of RFID tags to provide a more accurate statistics. Also, we would like to expand the data set in which we obtained an accuracy of 97% using only timing features [4]; the advantages of using timing include being easier to measure and more stable.

Regarding the feature set, we would like to investigate additional sets of features to determine if better performance can be achieved. The problem of selecting decoys will also be of future interest to us; instead of selecting the decoys at random, we would like to determine a way to select which features to be replaced by decoys. Also, we would like to experiment with the way a compromised reader sends its distance to the server. Instead of sending a fixed distance, a compromised reader could change the distance based on the current feature value.

Adaptive authentication system is another idea we would like to test. In adaptive authentication system, the server will keep on monitoring the responses from the reader. The server will terminate the authentication process if it had enough evidence that the reader is compromised, which means that the server does not need to send all of the modified feature vector to the reader to detect a compromised reader. Finally, instead of simulations to determine the performance of the authentication system, actual tag's measurements could be used.

Glossary

Accuracy: The percentage of instance that were correctly classified into either positive or negative;

Authentication: The process of confirming or denying that a claimed identity is correct by comparing the credentials of user/object with those previously proven, stored, and associated with the identity being claimed;

Classification: Set of methods from which an object is mapped to a cluster of similar objects;

Confusion Matrix: A specialized matrix that allows visualization of the performance of a classification algorithm;

Cross Validation: A technique for assessing how the results of a statistical analysis will generalize to an independent data set;

Curse of Dimensionality: A phenomenon that happens when using more features for classification results in a degradation of the classifier performance;

Energy Harvesting: A technique in which energy from the reader is gathered by the tag, stored briefly and transmitted back to the reader;

False Negative: A count that describes the number of positive test samples that were wrongly classified as negative;

False Positive: A count that describes the number of negative test samples that were wrongly classified as positive;

False Positive Rate: The percentage of negative classes that are classified as positive to

the total number of negative instances ($FP + TN$);

Feature Extraction: The process at which features that describe a certain object will be extracted from the behavior of that object;

Feature Selection: The process at which a reduction in the number of features we have will happen. Features that can accurately describe the object will be selected while redundant features and unhelpful features will be eliminated;

Fingerprinting: The process of creating a unique key that can be used to accurately identify and/or authenticate an object;

Gaussian Distribution: also known as Normal distribution, is a continues probability distribution that has a bell-curve shaped probability density function, which is known as the Gaussian function;

High Frequency (HF): Radio frequencies that are between 3 and 30 MHz, with a wavelength that is between 10 to 100 meters;

Histogram: A graphical representation showing visual impression of the distribution of data;

Identification: The process of discovering the true identity of an item from the entire collection of similar items, which requires a one-to-many matching;

Radio Frequency (RF): A rate of oscillation in the range of about 3 kHz to 300 GHz, which corresponds to the frequency of radio waves, and the alternating currents which carry radio signals [65];

Receiver Operating Characteristic Curve: A graphical plot of the TPR vs. the FPR, for a binary classifier system as the discriminant threshold is varied;

True Negative: A count that describes the number of negative test samples that were correctly classified as negative;

True Positive: A count that describes the number of positive test samples that were correctly classified as positive;

True Positive Rate: The percentage of positive instance that are classified as positive to the actual number of positive instances ($TP + FN$); and

Ultra-High Frequency (UHF): Radio frequencies that are between 300 MHz and 3 GHz, with a wavelength that is between 10 cm to 1 meter;

Bibliography

[1] EPC radio-frequency identity protocols class-1 generation-2 UHF RFID protocol for communication at 860 mhz - 960 mhz version 1.2.0. `http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2/uhfc1g2_1_2_0-standard-20080511.pdf`, Oct. 2008.

[2] NEXJEN systems. `http://www.nexjen.com/`, September 2011.

[3] N. Abramson. The aloha system: another alternative for computer communications. In *Proceedings of the November 17-19, 1970, fall joint computer conference*, AFIPS '70 (Fall), pages 281–285, New York, NY, USA, 1970. ACM.

[4] B. A. Alsaify, D. R. Thompson, and J. Di. Identifying passive UHF RFID tags using signal features at different tari durations. In *Proceedings of the 6th Annual International Conference on RFID*, pages 40 – 46, April 2012.

[5] J. Arnbak and W. van Blitterswijk. Capacity of slotted aloha in rayleigh-fading channels. *Selected Areas in Communications, IEEE Journal on*, 5(2):261 – 269, feb 1987.

[6] S. P. Awate. Parzen-window density estimation. `http://www.cs.utah.edu/~suyash/Dissertation_html/node11.html`, February 2007.

[7] L. E. Baum and J. A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bull. Amer. Math. Soc.*, 73:360 – 363, 1967.

[8] A. Bogdanov, L. R. Knudsen, G. Le, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In *the proceedings of CHES 2007*. Springer, 2007.

[9] C. Bosley, K. Haralambiev, and A. Nicolosi. HB^N: An HB-like protocol secure against man-in-the-middle attacks. Cryptology ePrint Archive, Report 2011/350, 2011. `http://eprint.iacr.org/`.

[10] J. Bringer, H. Chabanne, and E. Dottax. HB^{++}: a lightweight authentication protocol secure against some attacks. In *Security, Privacy and Trust in Pervasive and Ubiquitous Computing, 2006. SecPerU 2006. Second International Workshop on*, pages 28 –33, june 2006.

[11] R. Brunelli and T. Poggio. Face recognition: features versus templates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(10):1042 –1052, oct 1993.

[12] M. Bustillo. Wal-Mart radio tags to track clothing. `http://online.wsj.com/article/SB10001424052748704421304575383213061198090.html`, July 2010.

[13] P. Chen, C. Lin, and Bernhard S. A tutorial on v-support vector machines: Research articles. *Appl. Stoch. Model. Bus. Ind.*, 21:111–136, Mar. 2005.

[14] Y. Chen and F. Zhang. Study on anti-collision Q algorithm for UHF RFID. In *Proceedings of the 2010 International Conference on Communications and Mobile Computing - Volume 03*, CMC '10, pages 168–170, Washington, DC, USA, 2010. IEEE Computer Society.

[15] H. Choi, K. Choi, and J. Kim. Fingerprint matching incorporating ridge features with minutiae. *Information Forensics and Security, IEEE Transactions on*, 6(2):338 –345, June 2011.

[16] H. Chu, G. Wu, J. Chen, and Y. Zhao. Study and simulation of semi-active RFID tags using piezoelectric power supply for mobile process temperature sensing. In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2011 IEEE International Conference on*, pages 38 –42, march 2011.

[17] W. Clarkson, T. Weyrich, A. Finkelstein, N. Heninger, J.A. Halderman, and E.W. Felten. Fingerprinting blank paper using commodity scanners. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 301 –314, may 2009.

[18] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995. 10.1007/BF00994018.

[19] B. Danev, T. S. Heydt-Benjamin, and S. Capkun. Physical-layer identification of RFID devices. In *Proceedings of the USENIX Security Symposium*, 2009.

[20] J. Daugman. How iris recognition works. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(1):21 – 30, jan. 2004.

[21] S. A. Dudani. The distance-weighted k-nearest-neighbor rule. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-6(4):325 –327, April 1976.

[22] R. P. W. Duin. *Patern Recognition Tools (PRTools)*, 4.0 edition, 2010.

[23] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179 – 188, 1936.

[24] Andrew G. Analysis of variance?why it is more important than ever. *Annals of Statistics*, 33(1):1–53, 2005.

[25] Q. Gu, Z. Li, and J. Han. Generalized fisher score for feature selection. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*, Barcelona, Spain, 2011.

[26] J. Hall, M. Barbeau, and E. Kranakis. Detecting rogue devices in bluetooth networks using radio frequency fingerprinting. In *Proceedings of the IASTED International Conference on Communications and Computer Networks (CCN '06)*, October 2006.

[27] X. He, D. Cai, and P. Niyogi. Laplacian score for feature selection. In *NIPS*. MIT Press, 2005.

[28] A.K. Jain, R.P.W. Duin, and Jianchang Mao. Statistical pattern recognition: a review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):4 –37, Jan. 2000.

[29] B. Jiang, J.R. Smith, M. Philipose, S. Roy, K. Sundara-Rajan, and A.V. Mamishev. Energy scavenging for inductively coupled passive RFID systems. *Instrumentation and Measurement, IEEE Transactions on*, 56(1):118 –125, feb. 2007.

[30] RFID Journal. What's the difference between passive and active tags? `http://www.rfidjournal.com/faq/18/68`.

[31] A. Juels and S. A. Weis. Authenticating pervasive devices with human protocols. pages 293–308. Springer-Verlag, 2005.

[32] V. Kanhangad, A. Kumar, and D. Zhang. A unified framework for contactless hand verification. *Information Forensics and Security, IEEE Transactions on*, 6(3):1014 – 1027, Sept. 2011.

[33] B. Kinsella. What do RFID tags cost? Sept. 2010.

[34] T. Kohno, A. Broido, and k. claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, (2):93–108, May 2005.

[35] V. Lakafosis, A. Traille, H. Lee, E. Gebara, M.M. Tentzeris, G. DeJean, and D. Kirovski. RFID-CoA: The RFID tags as certificates of authenticity. In *IEEE Intl Conf. RFID*, pages 207 –214, Apr. 2011.

[36] C. Lee and D. Landgrebe. *Feature Extranction and Clasification Algorithms For High Dimensional Data*. PhD thesis, Electrical Engineering, Purdue Univesity, West lafayette, IN, January 1993.

[37] Y. Lee, L. Batina, D. Singelée, and I. Verbauwhede. Low-cost untraceable authentication

protocols for RFID. In *Proceedings of the third ACM conference on Wireless network security*, WiSec '10, pages 55–64, New York, NY, USA, 2010. ACM.

[38] Voyantic Ltd. `http://www.voyantic.com/`, 2011.

[39] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[40] A.M. Martinez and A.C. Kak. Pca versus lda. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(2):228 –233, feb 2001.

[41] C. Marzban. A comment on the roc curve and the area under it as performance measures. June 2004.

[42] K. Murphy. *Hidden Markov Model (HMM) Toolbox for Matlab*, June 2005.

[43] E.W.T. Ngai, K.K.L. Moon, F. J. Riggins, and C. Y. Yi. RFID research: An academic literature review (19952005) and future research directions. *International Journal of Production Economics*, 112(2):510 – 520, 2008.

[44] D. S. Pallet. Performance assessment of automatic speech recognizers. *J. J. Res. Natl. Inst. Stand. Technol.*, 90:371–387, September 1985.

[45] E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):pp. 1065–1076, Sept. 1962.

[46] S. Pasanen, K. Haataja, N. Paivinen, and P. Toivanen. New efficient rf fingerprint-based security solution for bluetooth secure simple pairing. In *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, pages 1 –8, jan. 2010.

[47] K. Pearson. Contributions to the mathematical theory of evolution. ii. skew variation in homogeneous material. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 186:343–414, January 1895.

[48] S. Chinnappa Gounder Periaswamy. *Authentication of radio frequency identification devices using electronic charactaristics*. PhD thesis, University of Arkansas, December 2010.

[49] S. Chinnappa Gounder Periaswamy, D. R. Thompson, and J. Di. Fingerprinting RFID tags. *IEEE Trans. Dependable and Secure Computing*, 8(6):938 –943, Nov./Dec. 2011.

[50] S. Piramuthu. RFID mutual authentication protocols. *Decis. Support Syst.*, 50:387–393, January 2011.

[51] S. Prabhakar, A. Ivanisov, and A.K. Jain. Biometric recognition: Sensor characteristics and image quality. *Instrumentation Measurement Magazine, IEEE*, 14(3):10 –16, june 2011.

[52] S. Qing-Yun and K. Fu. A method for the design of binary tree classifiers. *Pattern Recognition*, 16(6):593–603, 1983.

[53] Murray R. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):pp. 832–837, 1956.

[54] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. chapter Readings in speech recognition, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[55] M.J. Riezenman. Cellular security: better, but foes still lurk. *Spectrum, IEEE*, 37(6):39 –42, jun 2000.

[56] H.P. Romero, K.A. Remley, D.F. Williams, and C. Wang. Electromagnetic measurements for counterfeit detection of radio frequency identification cards. *Microwave Theory and Techniques, IEEE Transactions on*, 57(5):1383 –1387, May 2009.

[57] I. K. Sethi and G. P. R. Sarvarayudu. Hierarchical classifier design using mutual information. *IEEE Trans. Pattern Anal. Mach. Intell.*, 4:441–445, April 1982.

[58] G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th annual Design Automation Conference*, DAC '07, pages 9–14, New York, NY, USA, 2007. ACM.

[59] J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):193–300, 1999.

[60] C. Swedberg. High demand keeps tag prices steady. Sept. 2010.

[61] S. Thirumuruganathan. A detailed introduction to k-nearest neighbor (knn) algorithm. `http://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/`, May 2010.

[62] F. Turroni, D. Maltoni, R. Cappelli, and D. Maio. Improving fingerprint orientation extraction. *Information Forensics and Security, IEEE Transactions on*, 6(3):1002 –1013, Sept. 2011.

[63] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260 – 269, Apr. 1967.

[64] Q. Wang and C. Suen. Analysis and design of a decision tree based on entropy reduction and its application to large character set recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6:406–417, April 1984.

[65] Wikipedia. Radio frequency, 2012. [Online; accessed 27-January-2012].

[66] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1-3):37 – 52, 1987.

[67] H. Wu and Y. Zeng. Efficient framed slotted aloha protocol for RFID tag anticollision. *Automation Science and Engineering, IEEE Transactions on*, 8(3):581 –588, july 2011.

[68] D. Zanetti, B. Danev, and S. Capkun. Physical-layer identification of UHF RFID tags. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, MobiCom '10, pages 353–364, New York, NY, USA, 2010. ACM.

[69] H. Zhao and S. Ram. Entity identification for heterogeneous database integration: a multiple classifier system approach and empirical evaluation. *Inf. Syst.*, 30:119–132, April 2005.

Appendix A: Building Confusion Matrix Using HMM Results

```
indexSize = size(index);
confusionMatrix = zeros(indexSize(1), indexSize(1));

for i=1:indexSize(1)
    for j=1:indexSize(2)
        if (index(i,j) == i)
            confusionMatrix(i,i) = confusionMatrix(i,i) + 1;
        end
        if (index(i,j) ~= i)
            confusionMatrix(i, index(i,j)) = ...
                confusionMatrix(i, index(i,j)) + 1;
        end
    end
end

confusionSize = size(confusionMatrix);

for i=1:confusionSize(1)
    TP(i) = confusionMatrix(i,i);

    FN(i) = 0;
    for j = 1:confusionSize(2)
        if(j~=i)
           FN(i) = FN(i) + confusionMatrix(i,j);
        end
    end

    FP(i) = 0;
    for j=1:confusionSize(2)
        if(j~=i)
            FP(i) = FP(i) + confusionMatrix(j,i);
        end
    end

    TN(i) = 0;
    for j = 1:confusionSize(2)
        for k = 1:confusionSize(2)
            if(j~=i && k~=i)
                TN(i) = TN(i) + confusionMatrix(j,k);
            end
```

```
        end
    end
    TPR(i) = TP(i)/(TP(i) + FN(i));
    accuracy(i) = (TP(i) + TN(i))/(TP(i) + TN(i) + FN(i) + FP(i));
end
```

Appendix B: Plotting ROC Curves Using HMM Results

```
T = zeros(3,300);
Y = zeros(3,300);

for i=1:100 %T represnt the target classes
    T(1,i) = 1;
    T(2,i+100) = 1;
    T(3,i+200) = 1;
end

count = 1;
for i=1:3 %Y represnt the output classes
    for j=1:100
        Y(i,j) = prob(1,j,i);
        Y(i,j+100) = prob(2,j,i);
        Y(i,j+200) = prob(3,j,i);
    end
end

[tpr,fpr,th] = roc(T,Y);

x = size(tpr{1});
y = size(tpr{2});
z = size(tpr{3});

expandTo = max([x(2) y(2) z(2)]);

for i = x(2):expandTo-1
    tpr{1}(1,i+1) = 1.0;
    fpr{1}(1,i+1) = 1.0;
end
for i = y(2):expandTo-1
    tpr{2}(1,i+1) = 1.0;
    fpr{2}(1,i+1) = 1.0;
end
for i = z(2):expandTo-1
    tpr{3}(1,i+1) = 1.0;
    fpr{3}(1,i+1) = 1.0;
end

tpr{1}(1,expandTo) = 1.0;
```

```
fpr{1}(1,expandTo) = 1.0;
tpr{2}(1,expandTo) = 1.0;
fpr{2}(1,expandTo) = 1.0;
tpr{3}(1,expandTo) = 1.0;
fpr{3}(1,expandTo) = 1.0;


hold on

plot(fpr{1}, tpr{1},'Color', [0, 0, 0], 'LineWidth', 1.0);
plot(fpr{2}, tpr{2}, 'Color', [0, 0, 1], 'LineWidth', 1.0);
plot(fpr{3}, tpr{3}, 'Color', [0, 1, 0], 'LineWidth', 1.0);

legend('Manufacturer A',...
    'Manufacturer B',...
    'Manufacturer C');

xlabel('False Positive Rate');
ylabel('True Positive Rate');
```

Appendix C: Using Weights to combine HMM models' results

```
%To generate the index array for confusion matrix calcaulations
for i=1:indexSize(1)
    for j=1:indexSize(2)
        index(i,j) = ceil(weights(i,1)*index_time(i,j) + ...
            weights(i,2)*index_power(i,j));
        if(index(i,j) > indexSize(1))
            index(i,j) = indexSize(1);
        end
    end
end

%To generate the prob array for ROC calculations
probSize = size(prob_power);
for i=1:probSize(1)
    for j=1:probSize(2)
        for k=1:probSize(3)
            prob(i,j,k) = weights(i)*prob_time(i,j,k) + ...
                weights(i)*prob_power(i,j,k);
        end
    end
end
```

Appendix D: Calculate Threshold for the Authentication System

```
function [ threshold, prob, values ] = CalculateThreshold(data, dataStat)

    dataSize = size(data);
    numTags = dataSize(1);
    numReads = dataSize(2);
    numFeatures = dataSize(3);

    for i = 1:numTags
        threshold(i) = 1;

        for j = 1:numFeatures
            distance = 10*dataStat(i,j,2);
            minBound = dataStat(i,j,1) - distance;
            maxBound = dataStat(i,j,1) + distance;

            %To construct the pdf of the feature
            count = 1;
            step = dataStat(i,j,2)/10;

            for k = minBound:step:maxBound
                pdfValues(count) = pdf('norm', k, dataStat(i,j,1), ...
                    dataStat(i,j,2));
                featurevalues(count) = k;
                count = count + 1;
            end

            [ furthestDistance, readIndex ] = findFurthestDistance(i, j...
                , data, dataStat);

            for k = 1:length(featurevalues)
                distanceArray(k) = abs(dataStat(i,j,1) - ...
                    featurevalues(k));
            end

            for k = 1:length(distanceArray)
                closestDistance(k) = abs(furthestDistance - ...
                    distanceArray(k));
            end

            [closestValue, minIndex] = min(closestDistance);
```

```
            if(minIndex > length(featurevalues)/2)
                bounds(1) = length(featurevalues)/2 - ...
                    (abs(minIndex - length(featurevalues)/2));
                bounds(2) = minIndex;
            elseif(minIndex < length(featurevalues)/2)
                bounds(1) = minIndex;
                bounds(2) = length(featurevalues)/2 + ...
                    (abs(minIndex - length(featurevalues)/2));
            else
                bounds(1) = minIndex - 1;
                bounds(2) = minIndex + 1;
            end

            areaUnderPDF = trapz(featurevalues(bounds(1):bounds(2)),...
                 pdfValues(bounds(1):bounds(2)));

            threshold(i) = threshold(i) * areaUnderPDF;
            prob(i,j,:) = pdfValues;
            values(i,j,:) = featurevalues;
        end
    end
end
```