Intrusion Detection System for Wireless Networks Based on User Mobility Profiling

by

Shaoying Zou

A thesis submitted to

the Faculty of Graduate Studies and Research

in partial fulfillment of

the requirements for the degree of

Masters of Computer Science

Ottawa-Carleton Institute for Computer Science

School of Computer Science

Carleton University

Ottawa, Ontario

August 2005

The undersigned recommend to

The Faculty of Graduate Studies and Research

acceptance of the thesis,

Intrusion Detection System for Wireless Networks Based on User Mobility Profiling

submitted by

Shaoying Zou

Dr. Doug Howe

(Director, School of Computer Science)

Dr. Michel Barbeau

(Thesis Supervisor)

Dr. Evangelos Kranakis

(Thesis Supervisor)

Carleton University

August 2005

Abstract

Intrusion detection is the art of detecting inappropriate, incorrect, or anomalous activities. There are two types of Intrusion Detection Systems (IDSs) such as: misuse detection systems and anomaly detection systems. When used in a wireless system, IDS is designed to capture the malicious use of available services so that it protects availability and security for legitimate users. Several intrusion detection technologies such as calling patterns on application layer, Radio Frequency Fingerprinting (RFF) on physic layer, and detection on the network layer are designed to protect wireless networks.

As a complement to the above technologies, employing User Mobility (UM) profiling, this thesis addresses the following open question: how to identify abnormal users efficiently with low false alarm rate in the anomaly detection system.

This thesis provides a feasible solution to this question with two classification frameworks, Instance Based Learning (IBL) and Hidden Markov Models (HMMs). It also describes details of design and implementation of the frameworks. The performance of two frameworks were evaluated by simulating the IBL with location data, and the HMMs with both location data and other mobility features (e.g., transmission time, speed, and course). The True Detection Rate (TDR), True Acceptance Rate (TAR), and False Detection Rate (FDR) were examined. The IBL framework has better success rate and is easy to implement. The HMMs framework could produce precise results if it has enough data from profiled users.

Moreover, this thesis analyzes a performance of the true detection rate and false alarm rate with authentic UM position data and other related mobility features.

Acknowledgements

First and foremost, my grateful thanks go to Dr. Michel Barbeau and Dr. Evangelos Kranakis, the supervisors of my Master thesis. Their invaluable and patient guidance, enthusiastic encouragement, and support, etc. accompanied me in every stage of my thesis work. What I have learned from them will provide me with a lifetime of benefits.

I would also like to offer my appreciation to my fellow students, especially Ms. Jen Hall, Mr. Paul Boone, and Ms. Xiaotao Shu, etc. The discussions I had with them were very helpful and enjoyable.

Last but not least, my appreciation goes to my parents for their love and understanding.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter starts by introducing the context in which this thesis has been written, followed by the motivation for this research. Then, the thesis contributions are summarized, and finally an outline of the thesis is presented.

1.1 Context of the Work

Mobile wireless networks, unlike wired networks which have several cables installed, can offer a multitude of services such as making calls using a mobile phone, receiving a message on a pager, or checking email on mobile devices. Besides these services for civilian use, wireless networks are also employed in many other fields such as the military or police force. For example, the military or police may use the network to relay information for situational awareness on the battlefield and to coordinate emergency disaster relief personnel after a natural disaster [JM96].

The wireless networks can be classified according to the following types based on their region of coverage. The first type of wireless network is called Wireless Local

Area Networks (WLANs). In this case, Local Area refers to a small geographic region such as a cafeteria, or an office, and can also refer to larger places such as a whole university campus. People can form a network or gain access to the Internet in these areas. Sometimes without an access point, several devices can form a temporary network if people only want to share the resources they already have and they do not need to access other networks such as Internet.

The second type of wireless network is called Wireless Personal Area Networks (WPANs). With two current approaches used, Infra Red (IR) and Bluetooth (IEEE 802.15), people can create a small network to connect personal devices within an area of about 30 feet. However, IR requires no barriers between any two connected devices and its link range is less than Bluetooth.

The third type of network is called Wireless Metropolitan Area Networks (WMANs). This is the technology that uses wireless connection to cover a geographic area such as a city or suburb. WiMAX (IEEE 802.16) standard is developed to solve this problem.

The forth type of network is called Wireless Wide Area Networks (WWANs). This type of networks can be used to cover large areas such as cities, via multiple satellite systems or antenna sites operated by an ISP.

Compared to wired networks, wireless networks have many advantages. First, they will never have the problems of cable. Second, people with wireless devices can easily take advantage of this service at any convenient place. The benefit is significant to the service providers who do not need to install cable. Instead, an ISP only needs to provide access points or stations in some places.

However, wireless networks have their own disadvantages. For example, the Quality of Service (QoS) is not guaranteed if there is any interference with the link. But the most serious problem with wireless networks is security. Unlike the wired networks for

which it is difficult to obtain physical access, the wireless signal can be intercepted easily by anybody with corresponding devices. That is, it is easy to eavesdrop on wireless network communications. Wireless devices are also vulnerable to hacking, due to the capability of the devices. In recent times, the media has reported an increase of the problem of personal information being stolen from cellular phones. One problem in particular is identity theft and intrusion in the cellular networks.

1.2 Motivation

With the development of mobile networks, it is clearly evident that the use of wireless devices and cellular phones is becoming a computing platform of choice. While this trend had been forecast some years ago, what is crucial today is the need for robust Intrusion Detection Systems (IDSs) in wireless and mobile networks. Although numerous commercial anomaly-based detection systems are currently available, they all suffer from a high false detection rate. This outstanding problem can be addressed by comparing an observed event or behavior against multiple profiles, namely calling patterns, hardware characteristics of devices (e.g., hardware fingerprint of transceivers) and others, prior to rendering a decision. This thesis mainly focuses on these aforementioned problems.

1.3 Thesis Contribution

This thesis contributes to the design, implementation, and performance evaluation of two IDSs with different classification framework such as: Instance Based Learning (IBL) and Hidden Markov Models (HMMs). In addition, we present a novel scheme to

integrate the User Mobility (UM) profiling with the IDSs. As we know, UM profiling is a technology that has been used by cellular network providers to minimize the cost of paging. To the best of our knowledge, it is the first time that UM profiling is being used for IDS.

The main contributions of this thesis are as follows:

1. Design of an IBL framework based on UM profiling. First, an implementation of the IBL model is developed with a simulation tool, MATLAB. Second, the performances of this model, such as the True Detection Rate (TDR) and the True Acceptance Rate (TAR), are provided.

2. Design of a framework for the detection of anomalous trajectories of mobile users in wireless communication networks. This approach is based on UM profiling and HMMs classification method. The implementation results for this model are also provided.

By analyzing the experiment results, we demonstrated that the use of mobility profiles of wireless users for anomaly-based intrusion detection is technically feasible. Second, we also provide a better solution for IDSs by adding more mobility features into UM profiles. Third, we also identified a performance bottleneck for the true acceptance rate in both IBL and HMMs implementation. For that problem, we provide a solution by padding location coordinates into the missing place. Through simulation this process demonstrated that the performance is improved.

1.4 Road Map

The rest of my thesis is organized in the following manner: Chapter 2 reviews and characterizes the background and the related work on intrusion detection techniques for wireless networks. We introduce our first experiment with the IBL framework, perform a simulation, and analyze results in Chapter 3. Chapter 4 also offers enhanced implementation with a more precise classification approach: that is HMMs. Detailed analysis of the results from HMMs method are included in Chapter 4 as well. The conclusion of the thesis and the opportunities for future work are addressed in Chapter 5.

Chapter 2

Background and Related Work

This chapter offers a survey of some existing intrusion detection strategies for wireless networks, and an introduction to our test environment. We start by introducing two main intrusion detection technologies, anomaly-based detection and misuse-based detection, in Section 2.1. In Section 2.2, we discuss some related IDS work for wireless networks on different layers, including using RFF, Network Layer Detection and Calling Patterns. Section 2.3 introduces two classification approaches that are used in our IDSs. Finally, we give a short introduction to our test environment in Section 2.4

2.1 Intrusion Detection Technologies

When any intrusion is attempted, intrusion prevention (e.g., via authentication) is the first line of defense against malicious activities. However, only relying on prevention is not sufficient to protect wireless networks. There are always exploitable weaknesses in the systems due to design and programming errors. For instance, if the authentication merely relies on a property of the hardware technology (e.g. IMEI code for cellular

phones), then intrusion based on cloned hardware will not be identified. So intrusion detection technologies were developed as the second line of defense to protect the resources of the legitimate users. Numerous intrusion detection technologies have been developed over the last few decades. The conventional technologies for IDSs can be classified in two groups: misuse detection and anomaly detection, each of which has its own methodology to identify intruders.

The misuse detection method [KS94] first finds all possible attack patterns, and then puts these patterns into a database. Whenever an observed action from a user is received, the misuse detection system compares this incoming observation with patterns stored in the database in order to classify it as normal or abnormal behavior. For example, a distinctive pattern for a guessing password attack can be "there are more than three failed login attempts within a short time". Rogers Communications Inc. uses this technique to protect a user's web account. If a person's activities match the pattern, the user's account is locked for 24 hours.

In contrast, anomaly detection [LB99] uses another approach, which stores the user's history behavior into a database, and compares any incoming observation with the patterns in the database to identify intruders from legitimate users. For example, the normal profile of a user may contain the daily operating sequence and the averaged frequencies of some system commands after his/her login. If these activities are monitored, and the sequence or the frequencies are significantly different than the profile, then an anomaly alarm is raised. Since misuse and anomaly detection have their own properties, some IDSs, for example, IDES [LTG+92] and NIDES [AFTV94], use both anomaly and misuse detection techniques.

2.1.1 Misuse Detection

Misuse intrusion detection has been clearly understood in the literature as the detection of specific, precisely representable attacking modes. In other words, misuse detection is basically a pattern matching method which is well developed for the detection of such offenses. A user's activities are compared with known signature patterns of intrusive attacks. Each specific mode of offense can be considered as a pattern and many of these can be matched simultaneously against the audit logs generated by system. Although research on anomaly-based detection for IDSs has been done for years [LSM99, JV91, LS00], most commercial systems focus on misuse detection (i.e. pattern matching for known attacks), requiring frequent updates when new attacking modes are developed. Many current commercial network IDSs [AT00, CS99, ISS99] are capable of automatically responding to network attacks through increased logging, firewall reconfiguration, termination of connections, and even automatic blocking of suspicious networks.

The main advantage of misuse detection is that it can accurately and efficiently detect instances of known attacks. The main disadvantage is that it lacks the ability to detect the truly innovative (i.e., 'zero day') attacks.

2.1.2 Anomaly Detection

Unlike misuse detection techniques, anomaly detection is based on an assumption that the behavior of an intruder is different from a normal user's behavior. Furthermore, people also assume that these differences can be measured quantitatively. Relying on these assumptions, [LSM99, JV91, FHSL96] studies many techniques to analyze different data sources. For example, [LSM99] uses data mining for network traffic,

[JV91] uses statistical analysis for audit records, and [FHSL96] uses sequence analysis for operating system calls. In particular, [LS00] described a framework for network intrusion detection, which offers some guidelines to extract related features from data records.

The main advantage of anomaly detection is that it does not require prior knowledge of intrusion and can thus detect new intrusions. The main disadvantage is that it may not be able to describe what the attack is and may have high false alarm rate because they are not capable of discriminating between abnormal patterns triggered by an authorized user and those triggered by an intruder [LJ00]. A successful anomaly detection system must overcome many challenges. As a rule of thumb, the user's behavior might change over time. Learning algorithms should track user behavior and adapt to a changing environment to allow for consistently evolving systems.

2.2 Related Works on Wireless Networks

In the previous section, we described two approaches for intrusion detection. We now review other related designs and implementations of intrusion detection on different layers of wireless networks.

2.2.1 Radio Frequency Fingerprinting

Due to the infrastructure of ISO/OSI networks, IDSs, which are developed for wired/wireless networks, can be applied on all of the seven layers from the application layer down to the physical layer. Hall, Barbeau and Kranakis [HBK03, HBK04] have developed an approach that exploits the phase characteristic of signals for intrusion detection at the physical layer with RFF technique.

Their proposals are based on the assumption that every piece of transceiver hardware (e.g., 802.11b transceiver) has its own unique radio frequency fingerprint (the transient portion of the signal the hardware generates), which can be used to identify each transceiver and protect network resources from intruders. In many other IDSs, Media Access Control (MAC) address is also used. However, unlike RFF technique, which reflects the hardware characteristics, The MAC address can be acquired over air and reused to gain access to the network (MAC address spoofing).

The key objective is to successfully detect the start and end of the transient and extract three components, such as amplitude, phase and frequency. A technique such as Discrete Wavelet Transform (DWT) [Mal99] can be used. [CPAH95, HP96] state that the Daubechies filter can be used to acquire the DWT coefficients with lower computational complexity. In order to improve results, [HBK03] also applied the Bayesian filter to increase TDR by reducing the impact of noise.

A performance evaluation of RFF shows that classification success rates range from over 90% to 100%. Based on simulation results, this anomaly based intrusion detection technology is feasible. However, we notice that the success rate can be improved by optimizing the composition of the transceiverprint. Future investigation is required to examine the scalability of such a system.

2.2.2 Network Layer Detection

Zhang and Lee [ZL00] proposed the first node-based anomaly-based intrusion detection architecture, in which a local anomaly detection engine is created on a rule based classification algorithm RIPPER [Coh95]. In this architecture, local response is activated when a node locally detects an anomaly or intrusion with high confidence.

Its primary goal is not only to make each node in a wireless network responsible for detecting the intrusion independently, but also neighboring nodes can collaborate to probe the malicious activities in the border area. However in a wireless ad hoc network, there may not be a clear separation between normalcy and anomaly. For example, a node that sends out false routing information might be compromised, or merely has temporarily outdated routing information.

In the system each node attempts to detect anomalies or intrusions in the wireless network. If any intrusion is identified with weak evidence, the node with the intrusion detection agent initiates a global intrusion detection procedure through a cooperative detection engine.

Huang and Lee [aHL03] extended their previous work on local anomaly detection and developed a cross-feature analysis technique to explore the correlations between features using a classification decision tree induction algorithm C4.5 [Qui93]. Their detection engine uses features extracted from the routing table, such as route add count, route removal count, route find count, etc. However, due to the use of network layer statistical features, their system is unable to localize the attack source unless the identified attack occurred within one hop.

2.2.3 Calling Pattern

Another user profiling method that uses calling patterns in cellular networks has been developed. This approach [BN01, BN02, FHK+95] classifies the mobile phone calls into two main groups (e.g., normal ones and anomalous ones) according to their log files. The assumption is that all users' phone calls can be characterized with time and location of the calls. Whenever the phone call is finished, the time and its

location are compared with the stored user pattern profile. If the probability of an intrusion is high, a warning message is sent to the client's own cellular phone. This immediate notification can help the legitimate users to reduce their losses if their phones have been cloned. Moreover, to satisfy this design goal, [BN02] used a Radial Basis Function (RBF) network to divide the users into different groups and build the log files.

2.3 Brief Introduction to Two Classification Methods

In this thesis, we applied two classification methods into our IDS. One is IBL, and the other is HMMs. Each classification method has its own property. Mainly with k-nearest neighbor method, IBL decides how similar two instances, which can be points in an n-dimensional space, are. IBL stores all examples in the training set. When a new example arrives, retrieves those examples similar to the new example and looks at their classes. In contrast, HMMs can calculate the probability with only one attribute (that is, the hidden state). Since HMMs show the relation between observe signals and hidden states, they are used in many fields, e.g., speech recognition and bioinformatics.

2.3.1 Instance Based Learning

When we employed the IBL [LB99] framework in our implementation, we also brought several important concepts such as: similarity measure, similarity measure to the profile, noise suppression, and decision rule.

1. Similarity Measure: Similarity measure is a mathematical value to describe how close or similar two sequences of location coordinates are.

   To calculate this similarity measure, we define the following formula:

$$sim\,(X, Y) = \sum_{i=0}^{l-1} \omega\,(X, Y, i) \tag{2.1}$$

   with:

$$\omega(X, Y, i) = \begin{cases} 0, & \text{if } x < 0 \text{ or } x_i \neq y_i; \\ 1 + \omega(X, Y, i-1), & \text{if } x_i = y_i \end{cases}$$

   Here X, Y are two mobility sequences, each of which consists of l (i.e., 10) location coordinates, and i is the index of the sequence of location coordinates. The similar value of each location coordinate at the same index between two sequences, $\omega(X, Y, i)$, is zero, while these two location coordinates are not the same. Otherwise, the similar value is the former similar value plus one. The purpose for applying this formula is that we want to represent the value connecting with the order of location coordinates in sequences.

2. Similarity Measure to the Profile: The UM profile stores a set of training sequences from the stream of the UM sequence. The former method only shows how we compute the similarity value between two sequences of location coordinates. So we need to apply it for the all sequences stored in the UM profile to calculate the similarity measure to the profile, which is the rule to describe how similar the test sequence and the sequences in the profile are. In other words, it is also a standard to classify the user's observation movement as normal or abnormal.

To achieve a test sequence of similarity measure to the profile, we simply calculate its similarity measure by comparing it with each sequence of location coordinates in the profile, and select one of the largest values to stand as its similarity measure to the profile. It is defined as:

$$sim_D(X) = max_{y \in D} sim(Y, X)$$

The maximum value of $sim_D(X)$ is:

$$\sum_{i=1}^{l} = \frac{l(l+1)}{2}$$

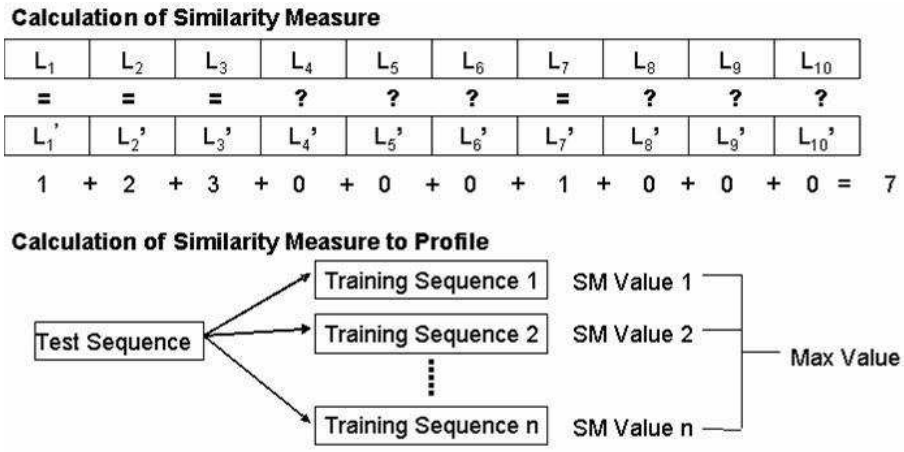

Figure 2.1: IBL Concept

3. Noise Suppression: In a realistic experiment, some degree of deviation of the UM sequence from the sequences in the profile (which is considered as noise) might happen for some unexpected reasons such as traffic conditions or weather.

However, this kind of noise can be suppressed. In doing so, the average of similarity values to profile of W number of test sequences is used. So, we give the following formula:

$$v_D(i) = \frac{1}{W} \sum_{j=i-W+1}^{i} sim_D(j)$$

4. Decision Rule: To make a decision whether a set of incoming observation sequences belong to a legitimate user or are from an intruder, we need to establish two thresholds for each legitimate user: the minimum threshold and the maximum threshold. Any degree of similarity values that fall between these two thresholds, which we named the acceptance region, represents the normal behavior. Otherwise, we would say that the mobility sequence is from an intruder.

2.3.2 Hidden Markov Models

Extended from the Markov Chain model, the HMMs [Rab90] which is a statistical model has a finite set of states, each of which is not visible directly to an external observer. However, with statistics, these states show some association that can be represented with a set of probability distributions with other observation symbols. For example, in the experiment in Chapter 3, the report location coordinates from user devices are observation symbols; user's location coordinates in real are hidden states. Since the wireless interference, or message delay during data transmission, the data received in database sometimes are not exactly the real location coordinates where users are at that transmission time. When we applied HMMs, we want to find out the probability distribution between the real location coordinates and the report

location coordinates. As well, we can use the probability distribution to evaluate how close or similar the incoming sequences and the sequences stored in the profile.

The HMMs is composed of the following components defined in Table 2.1:

N	The number of hidden states which are as $S = \{s_1, s_2, \ldots, s_N\}$
M	The number of observation symbols which are as $O = \{o_1, o_2, \ldots, o_M\}$
$A = \{a_{ij}\}$	The state transition probability matrix
$B = \{b_j(k)\}$	The observation symbol probability matrix
$\Pi = \{\pi_i\}$	The initial state probability distribution

Table 2.1: HMMs components

$a_{ij} = P\big[q_{t+1} = S_j | q_t = S_i\big], 1 \leq i, j \leq N$ is a transition probability from state i to state j, and the state at time t as q_t.

$b_j(k) = P\big[O_k \quad at \quad t | q_t = S_j\big], 1 \leq j \leq N, 1 \leq k \leq M$ is the observation probability with the observation symbol k in state j at that time t.

$\pi_i = P\big[q_1 = S_i\big], 1 \leq i \leq N$ is the initial probability of state i.

With these probability distributions, three problems can be solved with HMMs.

- The evaluation problem: for given HMMs and an observation sequence, to calculate the probability that the observation sequence was produced by the model.

- The hidden state sequence problem: for given HMMs and an observation sequence, to find the most probable hidden state sequence to represent by the observation sequence.

- The training problem: to determine the parameters of a given model.

Correspondingly, Rabiner [Rab90] shows three solutions to solve these problems. In this thesis, we are only interested in how to employ these solutions on the evaluation problem and training problem, which are relevant to our project, to build the IDS.

Solution to the evaluation problem

To calculate the probability of the observation sequence produced by a given model, a procedure called *the forward procedure* is applied. In the procedure, a forward variable $\alpha_t(i)$ is defined as the probability of the partial observation sequence, $O_1O_2 \ldots O_t$, and state S_j at time t produced by the given model λ.

$$\alpha_t(i) = P(O_1O_2 \ldots O_t,\ q_t = S_i|\lambda)$$

We can solve for $\alpha_t(i)$ inductively with following formulas:

$$\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N. \tag{2.2}$$

$$\alpha_{t+1}(j) = \left[\sum_{j=1}^{N} \alpha_t(i)a_{ij}\right] b_j(O_{t+1}), \quad 1 \leq t \leq T-1,\ \ 1 \leq j \leq N. \tag{2.3}$$

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \tag{2.4}$$

In initialization step, Formula 2.2 is used to initialize the forward probabilities as the joint probability of state S_i and initial observation O_1.

After that, Formula 2.3 is used to calculate the probability of the partial observation sequence until time $t+1$ and state S_j at time $t+1$ for the given model based on the former result. Figure 2.2 illustrates how state S_j can be calculated at time $t+1$ from the N possible states, S_i, $1 \leq i \leq N$, at time t. Because $\alpha_t(i)$ is the probability of the joint event that $O_1O_2 \ldots O_t$ are observed, and the state at time t is S_i, $\alpha_t(i)a_{ij}$ is the probability of the joint event that $O_1O_2 \ldots O_t$ are observed, and state S_j is reached at time $t+1$ via state S_i at time t. Adding all the products over N possible states S_i, $1 \leq i \leq N$ at time t obtains the probability of S_j at time

$t+1$ with the previous partial observations. When this result is multiplied with the probability $b_j(O_{t+1})$ (that is, observation O_{t+1} in state j), $\alpha_{t+1}(j)$ is obtained.

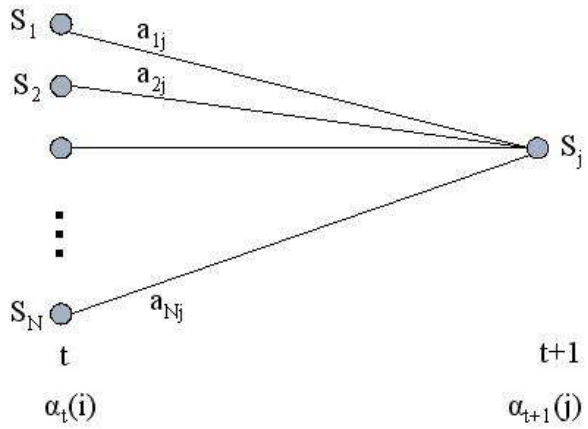

Figure 2.2: Illustration of the calculation of the forward variable $\alpha_{t+1}(j)$

Finally, Formula 2.4 gives the calculation of $P(O|\lambda)$ as the sum of the terminal forward variables $\alpha_T(i)$. In other words, $P(O|\lambda)$ is the sum of the $\alpha_T(i)$.

Solution to the training problem

Training problem is the most difficult part of HMMs. To determine the model parameters (A, B, π), one more auxiliary variable $\beta_t(i)$ is defined as the probability of the partial observation sequence from $t+1$ to the end, and state S_j at time t and the model λ.

$$\beta_t(i) = P(O_{t+1}O_{t+2}\ldots O_T|q_t = S_i, \lambda)$$

Then the probability of being in state S_i at time t, and state S_j at time $t+1$, given the observation sequence O and the model λ can be defined as $\xi_t(i,j) = P(q_t =$

$S_i, q_{t+1} = S_j|O, \lambda)$. The term $\xi_t(i,j)$ can also be written in a following form:

$$\begin{aligned}\xi_t(i,j) &= \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}\end{aligned} \tag{2.5}$$

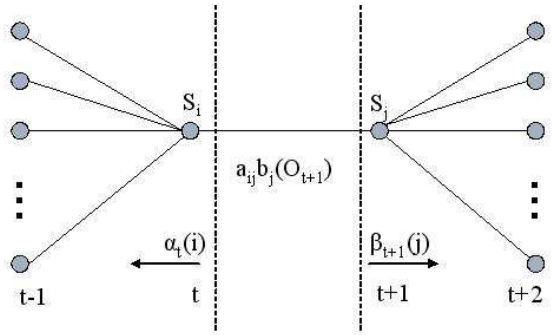

Figure 2.3: Illustration of the sequence of operations required for the calculation of the joint event that the system is in state S_i at time t and state S_j at time $t+1$

Also the probability of being in state S_i at time t, given observation sequence O, and the model λ, $\gamma_t(i) = P(q_t = S_i|O, \lambda)$, can be expressed in terms of the forward-backward variables. That is,

$$\begin{aligned}\gamma_t(i) &= \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)}\end{aligned} \tag{2.6}$$

The relationship between $\gamma_t(i)$ and $\xi_t(i,j)$ can be expressed as $\gamma_t(i) = \sum_{j=1}^{N}\xi_t(i,j)$.

So, summation of $\gamma_t(i)$ over the time index t can be interpreted as the expected number of times that state S_i is visited, or the expected number of transitions made from state S_i. Similarly, summation of $\xi_t(i,j)$ over the time index t can be interpreted

as the expected number of transitions from state S_i to state S_j. That is,

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of transitions from } S_i$$
$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{expected number of transitions from } S_i \text{ to } S_j$$

With the previous formulas, a set of reasonable reestimation formulas for determining the parameters of HMMs (A, B, π) are

$$\bar{\pi}_j = \text{expected number of times in state } S_i \text{ at time } (t=1) = \gamma_1(i) \tag{2.7}$$

$$\begin{aligned} \bar{a}_{ij} &= \frac{\text{expected number of transitions from state } S_i \text{ to } S_j}{\text{expected number of transitions from state } S_i} \\ &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \end{aligned} \tag{2.8}$$

$$\begin{aligned} \bar{b}_j(k) &= \frac{\text{expected number of times in state } S_j \text{ and observing symbol } v_k}{\text{expected number of times in state } S_j} \\ &= \frac{\sum_{t=1, O_t=v_k}^{T-1} \gamma_t(i)}{\sum_{t=1}^{T-1} \gamma_t(i)} \end{aligned} \tag{2.9}$$

Based on the above formulas, we iteratively use $\bar{\lambda}$ in place of λ and repeat the reestimation computation, we update the HMMs parameters until some limiting point is reached.

2.4 Introduction to Test Environment

We built our test environment based on the Automatic Position Reporting System (APRS) [Fil04] and collected experiment data from APRS. APRS was developed by Bob Bruninga for capturing and reporting radio operator's position and other

information such as weather reports and geographical information.

Figure 2.4 illustrates the APRS infrastructure. A station can receive messages broadcasted from users directly if the distance between users and station is within the transmission range of the radio. Alternatively, the messages can be relayed to a station by other user's radio, if the distance between users and station is larger than the transmission range of the radio. After a station receives the message updates, it stores these records into a database chronologically. Figure 2.5 shows the cables connected to receiver and antenna. The big black box is the power supply for the receiver. Figure 2.6 illustrates what receiver, the TNC, looks like. Figure 2.7 shows the trajectories belonging to different users who are identified with their call signs reflected on a map of Ottawa.

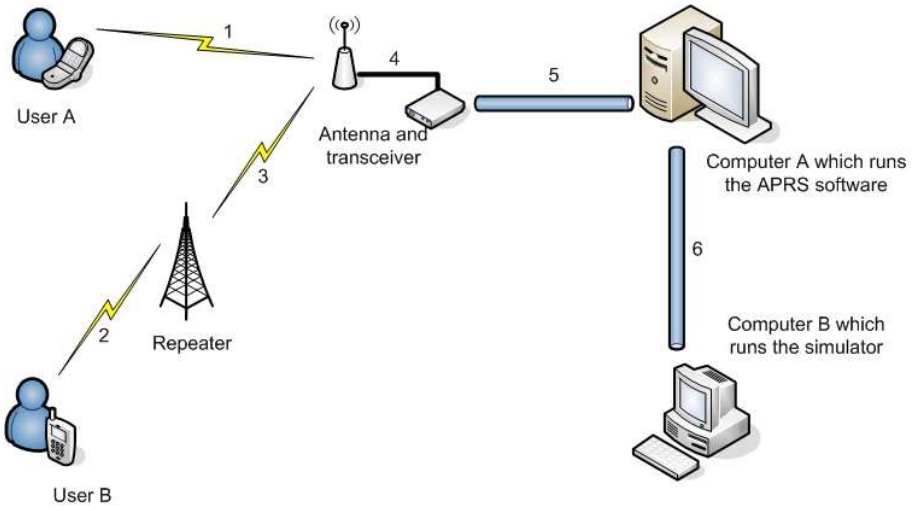

Figure 2.4: Infrastructure of APRS system

Figure 2.5: Antenna for receiving data

Figure 2.6: TNC to connect to the antenna

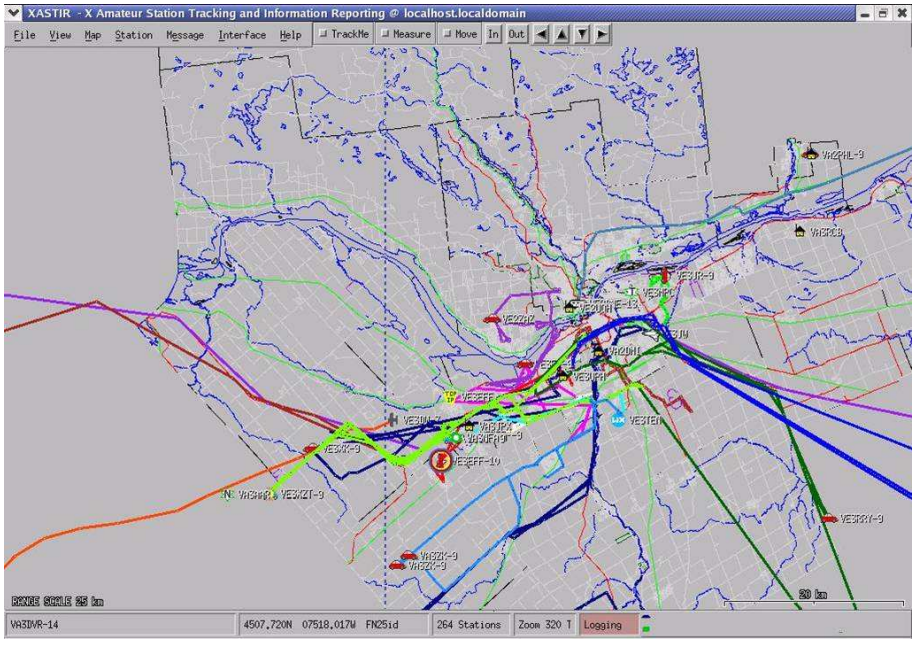

Figure 2.7: APRS system for Linux

Chapter 3

Intrusion Detection System with Instance Based Learning Framework

In this chapter, we mainly focus on how we design and simulate the IDS with the IBL method. Section 3.1 introduces our design architecture of IDS based on the IBL framework. Section 3.2 offers a description of collecting the experiment data. We also explore the method applied to minimize the deviation of data in Section 3.3. The details of creating UM profile is discussed in Section 3.4. In addition, during the preliminary testing, we present our performance measure in Section 3.5. Finally, conclusions are made in the last section.

3.1 Intrusion Detection System Design

In this section, we illustrate the architecture of our IBL-based IDS (see Figure 3.1). The whole system is composed of four components:

- Data Collection

- High Level Mapping
- Feature Extraction and Definition of UM Profiling
- Classification

Data collection is designed to gather the experiment data from mobile users. When this process is done, high level mapping is used to minimize the deviation of data caused by traffic jam, interfering in transmission, etc. After we conduct the mapping for experiment data, we extract features and make UM profiles based on these data. The last component of our IDS is to make classification based on these UM profiles.

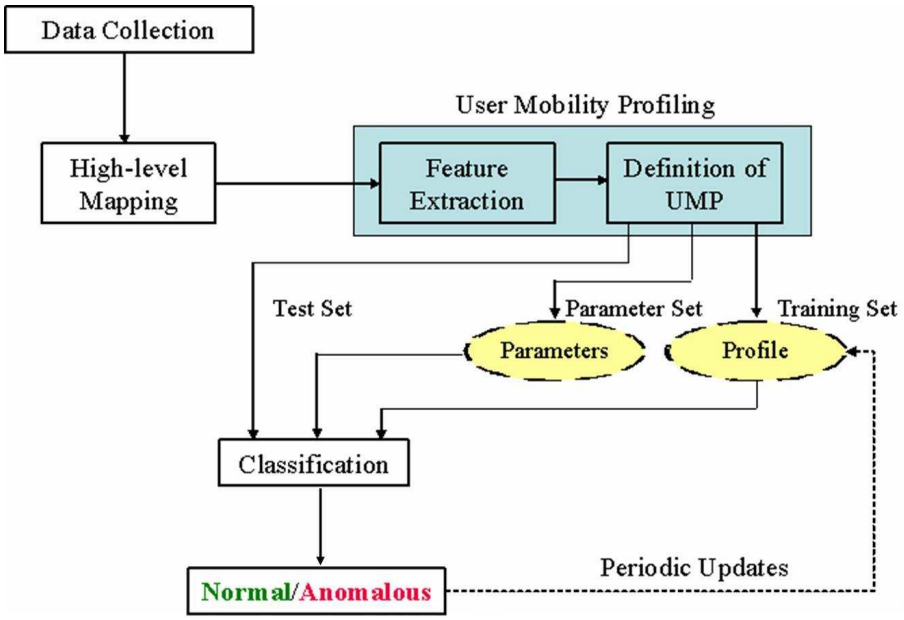

Figure 3.1: The architecture of IDS with IBL framework

3.2 Data Collection

Before we start this project, the first step is to collect data that shows the UM characteristics from different users. The purpose of this step is to obtain sufficient UM data and combine it with the following steps to build the UM profile and their mobility patterns. For collecting UM data, we use the APRS, which is introduced in Chapter 2.

To build an IDS for mobile networks, we use the assumption by Markoulidakis [MLTS95], which states that approximately 75% of mobile users can be well characterized in terms of their mobility patterns. These users, such as working people and housekeepers, have their unique daily routines which can be used to identify them from other users.

When we apply the data collection phase, we face two types of problems:

- irregular transmissions: Due to the interference or other reasons, the station might not receive all broadcasts from users.
- lack of information: Since user can choose different message formats for their radio, the broadcasts may not include all pieces of information.

However, these situations may not be bottleneck in wireless systems, since all transmission periodicity and message formats are obliged.

3.3 High Level Mapping

After we receive all location coordinates and related data from users, the second step is to conduct high level mapping, converting a location coordinate into a small grid

by rounding the last two digits from the raw data. Due to traffic, weather, and interference, location coordinates that users broadcast for the same trip might not be exactly the same. Even if we compare one user's testing set with location coordinate sequences in his/her own UM profile, we could not obtain too many highly matched sequences. This bottleneck prevents from achieving high performance for the TDR.

Figure 3.2 shows details of how high level mapping functions. It is possible that a user might take two similar paths but not the same ones, so that the broadcast locations are close. For example, after rounding these location coordinates, two different coordinates can be considered in the same grid with latitude as j+3 and longitude as i+2. As we explored, this process also minimizes the deviation between different users. Suppose the dashed line (see Figure 3.2) represents a trajectory from another user, the same logic is applied, which means that two trajectories would be thought of as the same one. In other words, a potential impersonation attempt is successful.

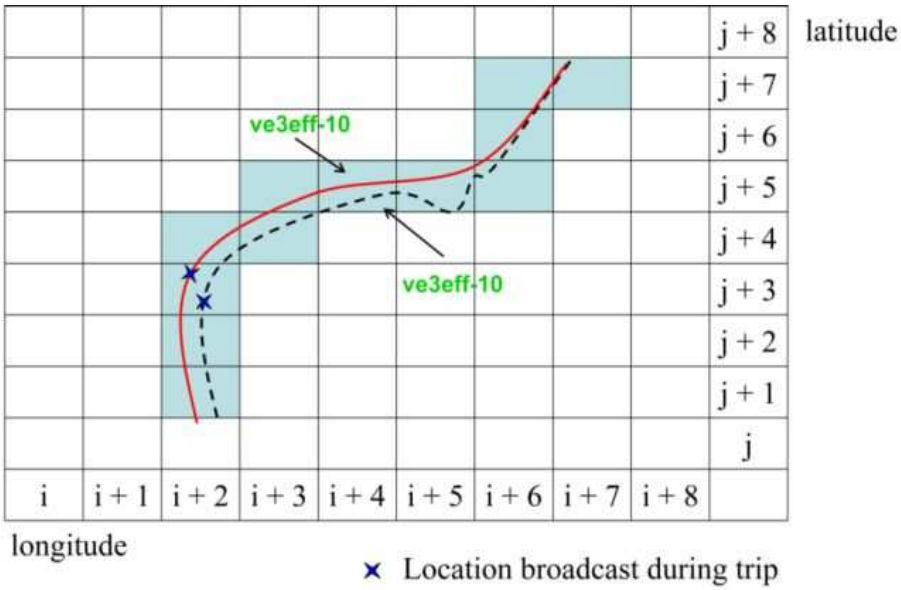

Figure 3.2: High level mapping

3.4 User Mobility Profiling

This section serves as an introduction of how we develop UM profiles using the modified data produced by high level mapping based on the location coordinate feature only.

3.4.1 Feature Extraction

Each row of messages in users' broadcast data set has a significant amount of information, including the user's identity, transmission time, receiving time, location coordinates, speed, and course (that is, direction). These messages are stored in a database according to the order in which the server receives them. In this implementation, however, we only extract location coordinates from the database following the order of receiving time.

After all location coordinates are extracted, they are stored in memory as a stream of location coordinates. Figure 3.3 shows the structure of stream of location coordinates in a logical view. To build UM profile, the next step is to transform the stream of location coordinates to a set of UM sequences, each of which contains ten location coordinates. The process of making UM profile takes the first ten location coordinates from the stream, and stores them into a UM profile as the first UM sequence (see Figure 3.3). To make the second UM sequence, the process shifts the starting point by one, and then takes another following ten location coordinates. This process is repeated until it exhausts all the location coordinates in the stream. All resulting UM sequences are saved as a component of the UM profile that serves as input to the profile and the classification phases. The purpose of using an overlapping window is to permit each location coordinate to become a starting point of a mobility sequence.

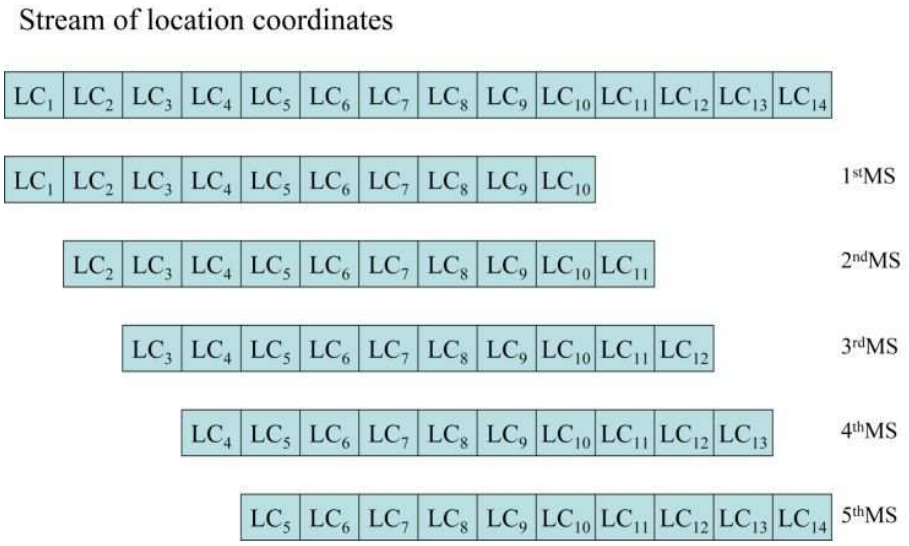

Figure 3.3: Structure of stream of location coordinates

3.4.2 Components of User Mobility Profile

Not only is a set of mobility sequences stored into the UM profile, but also other components are stored. The following is a list of components that compose a UM profile, and are employed during the classification phase.

- Call sign: specify the user's identification
- Training sequences: the UM sequences used for setting minimum threshold and maximum threshold
- Test sequences: the UM sequences used for classification
- Minimum threshold: the lowest similarity value obtained from comparing training sequences with the UM profile
- Maximum threshold: the highest similarity value obtained from comparing training sequences with the UM profile

- Window size: deviation on the UM sequences might happen during the user's moving due to some factors. To normalize this kind of deviation, the concept of window size (that is, the number of sequences to be used to normalize the sequences) is introduced into our implementation. By calculating the average of the similarity values, the deviation of sequences is reduced.

3.5 Simulations

This section describes how to classify the users as either legitimate or abnormal based on sequences of location coordinates in the test set. In doing so, we use the IBL classification method.

The objective of the simulation is to evaluate the performance of our IDS with two performance criteria such as: the TAR and the TDR. To obtain these two criteria, we select five users from the user database, and use these users to build the UM profiles and apply them in the experiment. The reasons we choose a small cluster size of users (five of them) are as following:

1. Not all of users in the database are moving. Some users who only report weather information are stationary. Some moving users broadcast few location coordinates, which are difficult to build UM profile to characterized these user's mobility patterns.

2. We apply these users not only on IBL framework, but also on HMMs framework. So we expect these users are suitable for both IDSs, and then we can compare the performance of two IDSs.

3. Besides the performance comparison, We also need to analyze the experiment

results. Five users can help me to identify the reasons that effect the TAR and TDR.

4. The goal of our experiment is trying to prove the feasibility that accurately characterized UM profiles can be used in IDS for wireless networks. More users but without correctly UM profile will not help us to make the right decision on what cause the low performance. Of course, after we find out the factors that effect the performance, we can try more users to see what results are for users with highly consistent mobility behaviord, or users with more chaotic mobility behavior.

3.5.1 Simulation Results

To obtain simulation results, another key performance parameter r, which is the pre-established acceptable false alarm rate, is used to obtain an acceptance region on the user's similarity distribution. In this experiment, r is set to 20%. To obtain the acceptance region for each user, $r/2$ quantizes is applied on the upper and lower distribution separately to achieve the minimum and maximum thresholds. If the similarity value of the sequence comparing to the UM profile falls in an acceptance region (between the minimum and maximum thresholds), it is classified as *normal*. Otherwise, it is considered as an *intrusion*. The following five plots show the minimum and the maximum threshold for five different users (see Figure 3.4 - 3.8). We also use these five users to identify the performance criteria considered.

True Acceptance Rate

To discuss the performance of IDS, we apply TAR to examine the level of reliability of the system. The TAR specifies how correctly a system is categorizing mobility

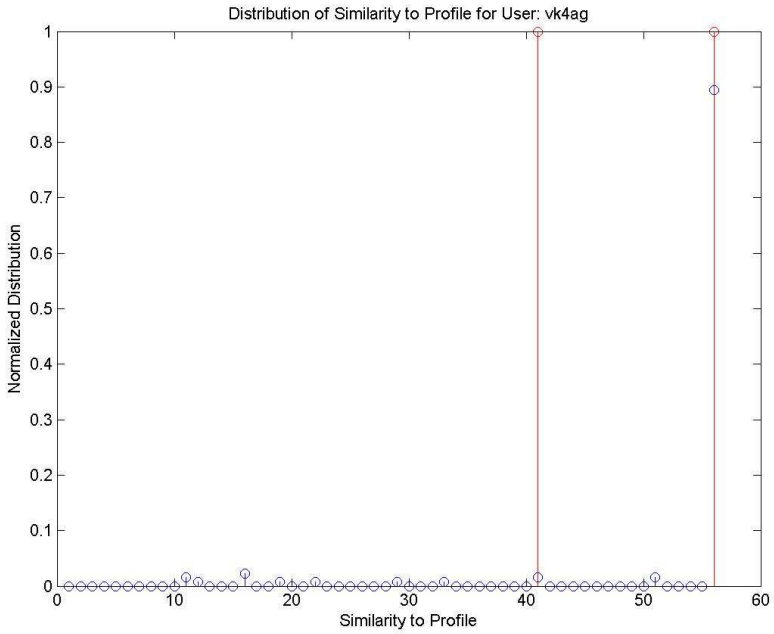

Figure 3.4: Distribution for user vk4ag

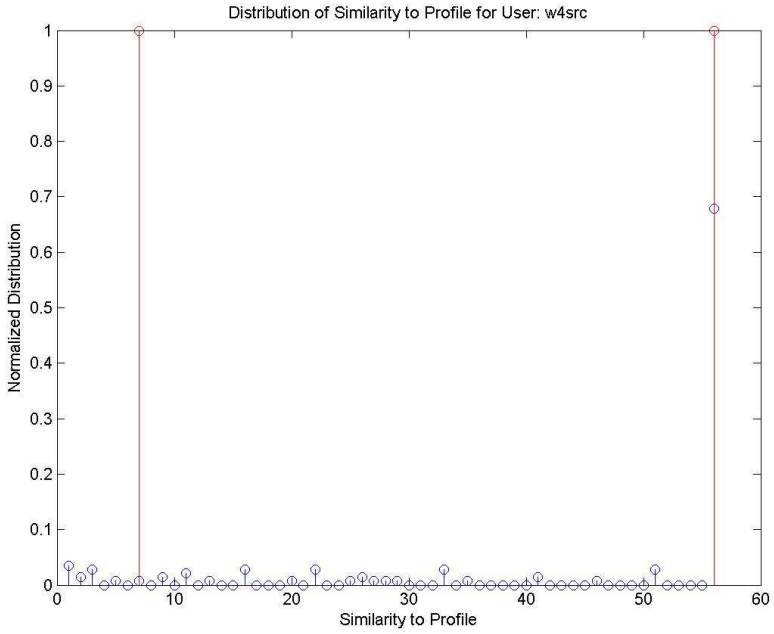

Figure 3.5: Distribution for user w4src

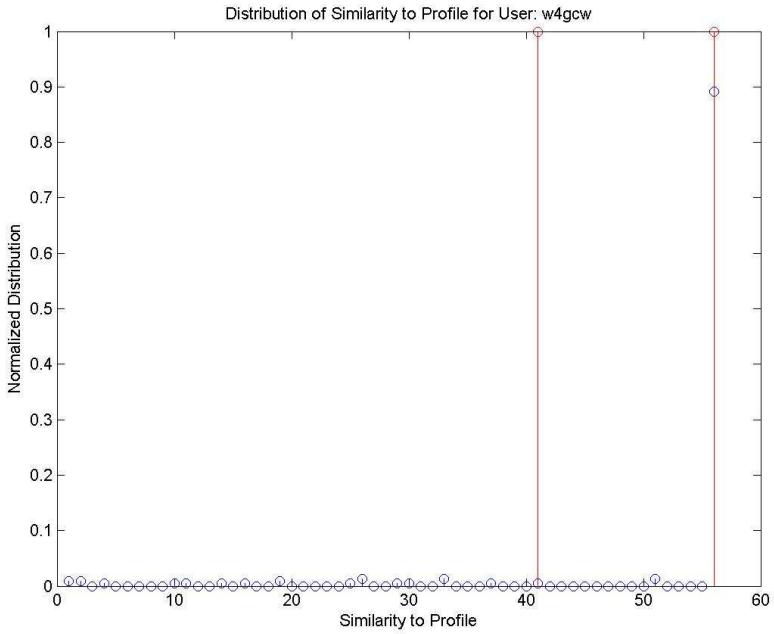

Figure 3.6: Distribution for user w4gcw

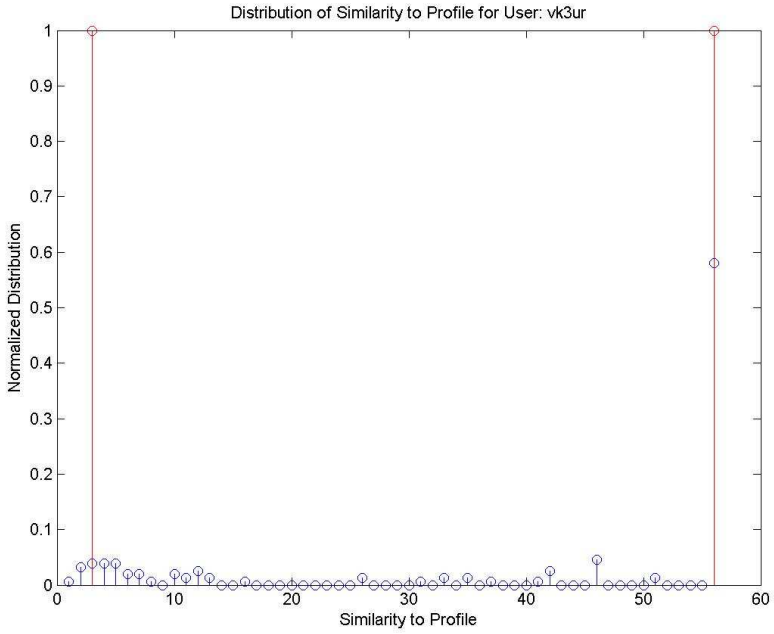

Figure 3.7: Distribution for user vk3ur

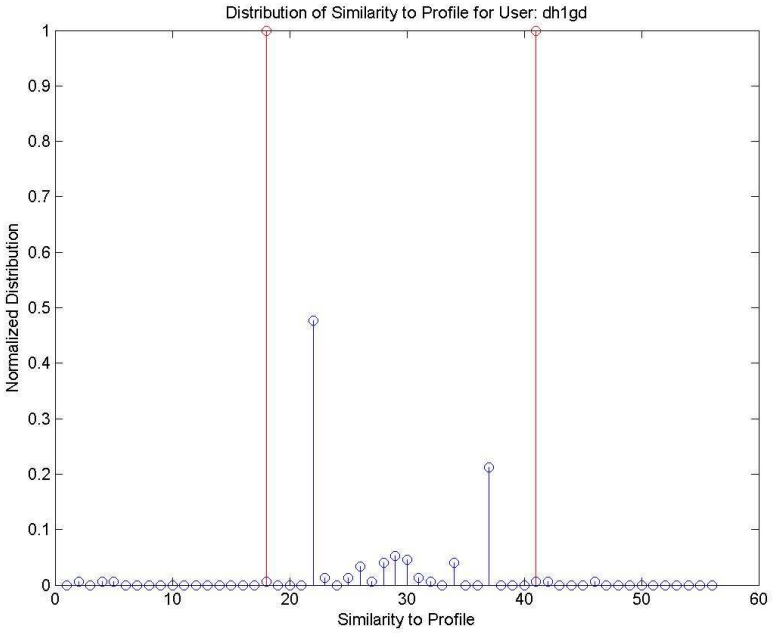

Figure 3.8: Distribution for user dh1gd

sequences as normal (originating with the profiled user). So the TAR is the proportion of the numbers of successfully identified incoming sequences and the total numbers of incoming sequences.

However, in our experiment, the data from users are broadcasted voluntarily. We don't know when we will receive the messages from users. To obtain the TAR, we take part of the history data as test sequences from one user and use them to compare with his or her UM profile. It can be written as:

$$TAR = \frac{\text{the numbers of mobility sequences correctly categorized as normal}}{\text{total numbers of mobility sequences from one profiled user}}$$

Figure 3.9 presents results of the simulation. The x-axis of the bar chart shows five users we use in this simulation, and the y-axis of the bar chart represents the

TAR for each profiled user.

True Detection Rate

Another metric we use to evaluate the system performance is TDR which specifies how successful a system is in detecting attacks when they happen. Compared to TAR, TDR can be written as:

$$TDR = \frac{\text{the numbers of intrusion successfully identified as abnormal}}{\text{total numbers of intrusion happened}}$$

However, in our circumstance, all experiment data are from profiled users. By definition, we don't have the really happened intrusions. To obtain the TDR results, we simulate the happened intrusion by comparing four users' test sequences against the remaining user's profile. If the similarity measure of a sequence from one of the four users' profile falls outside of the acceptance region (that is, less than the minimum

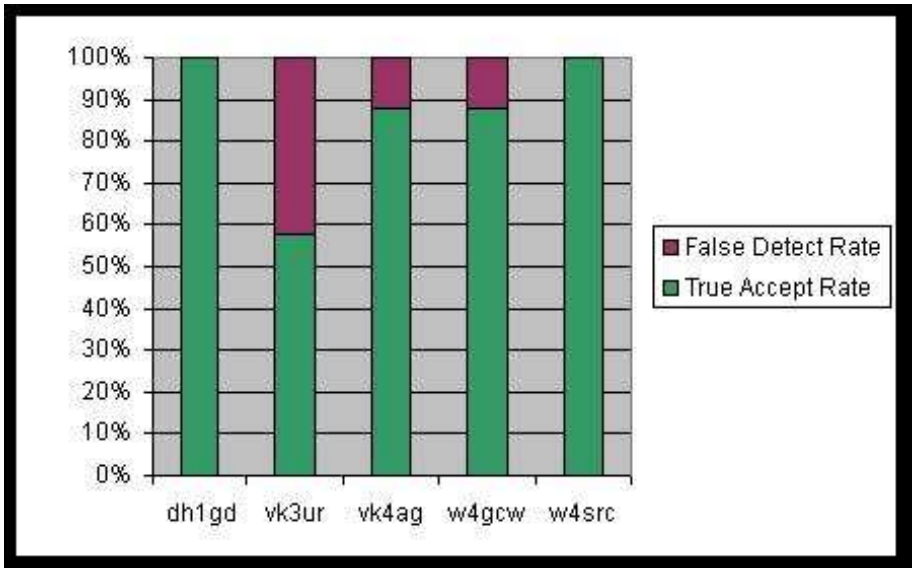

Figure 3.9: True acceptance rate for 5 users

threshold or greater than the maximum threshold), it is successfully detected as from an intruder. We can obtain the TDR by comparing the number of accurately detected sequences with the total number of the test sequences.

Figure 3.10 illustrates the TDR for the same five users. The x-axis of the bar chart shows five users we used, and the y-axis of the bar chart represents the TDR for each profiled user.

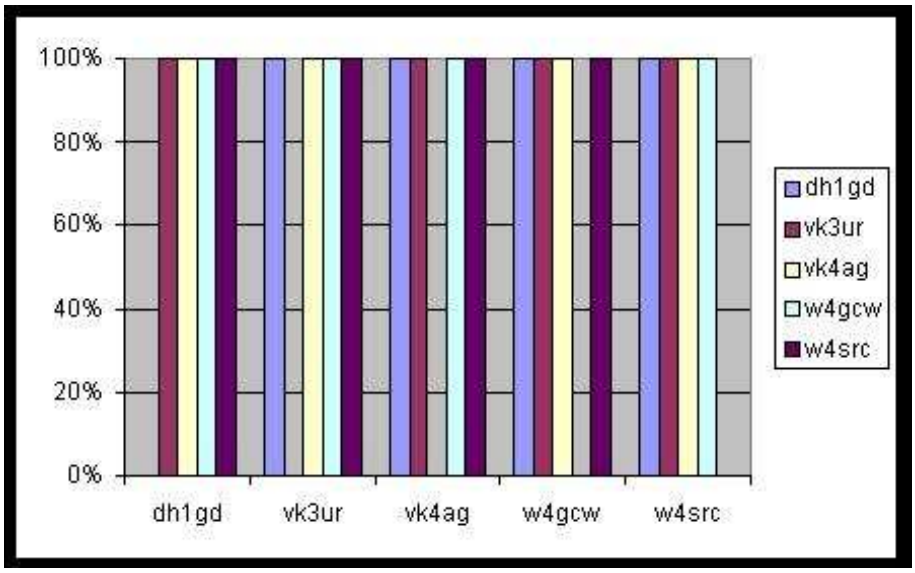

Figure 3.10: True detection rate for 5 users

3.6 Conclusions and Analysis

We illustrate the TDR and the TAR in Figure 3.9 and 3.10 for anomaly-based IDS based on UM profiling. Besides these two TAR and TDR we get form the experiment, we also check the performance of our IDS from another point of view. In Section 2.3.1, we have introduced how we calculate the simulation value between two mobility

sequences by comparing each item in the sequence and accumulate these sub-similar values. This step is the key operation to obtain comparison values which are used to classify intruders and profiled users. From the example in the Figure 2.1 and the Formula 2.1, we know this process is a linear computation. In other words, the time complexity for that process is $O(n)$.

The task of our IDS is to find out whether the claimed person is the profiled user; it doesn't perform the identification process. In other words, we are only interested in a yes/no question instead of to find out who the user really is. From this point of view, this IDS has a good scalability that can be extended from five users (in our simulation) even to thousands of users. However, we also realize when the IDS does the classification, it compares the incoming sequence with all the sequences stored in UM profile which grows with the time. The longer time we collect data from this user, the larger the UM profile is. A large profile definitely effects the performance time. So profile updating and refining is necessary for this IDS.

The following shows the advantages of our IDS framework that:

- performs 100% the TDR
- performs high TARs ranged from 80% to 100%
- offers an efficient comparison between the test sequences and the UM sequence stored in UM profiles
- is practical and easy to implement

Therefore, our anomaly-based IDS based on UM profiling is technically feasible.

When we check the TAR, the 4 of 5 users obtain a high success rate. However, for the remaining user, the TAR is less than 60%. The reasons caused that low

performance are the irregular transmission and a small data set for this user. Due to these factors, the UM test sequences could not match the UM sequence in the UM profile. Also our IDS obtained 100% TDR, for all of 5 users. Because these users have their particular UM sequences in their UM profile with respect to other users' UM profiles, our IDS can successfully identify them from other users.

Chapter 4

Intrusion Detection System with Hidden Markov Models Framework

In this chapter, we extend our discussion on how an IDS can be used in wireless networks by applying UM profiling. In the following sections, we give new definitions of UM profile and UM pattern. In addition, we modify the approach, which extracts UM patterns from this new type of UM profile. Finally, we conduct our simulation by using a new classification method based on a model called the HMMs, and make our conclusions based on the simulation results.

4.1 Framework of Intrusion Detection System

To distinguish abnormal users from legitimate users, we first need to build an UM profile for each legitimate user, and then extract UM patterns belonging to this user according to his/her own UM profile. In this chapter, we bring a totally different definition of UM profile and UM pattern, with respect to Chapter 3. Following this

new definition, we store the UM profile and UM patterns into two different files. We assume that we only have data from legitimate users. For simulation purposes, we apply one user's UM profile against the same user's UM patterns in our IDS to acquire the TAR. To obtain another performance metric of IDS, the TDR, we merely apply other users' UM profiles against one user's UM patterns in our IDS. These two rates are the performance criteria of the simulation. This IDS can also be extended to a real time situation. When a user moves, a new observed mobility sequence arrives in the system. Whenever the system receives a new mobility sequence, it automatically compares the incoming sequence with UM patterns, which belong to the associated user, by applying the HMMs classification method. If the incoming sequence matches any one of the mobility patterns stored in the UM profile, the system believes the observation sequence comes from the legitimate user; otherwise, the system is set to either trigger an alarm to report this as an intrusion, or just to raise a system alarm level to determine whether there is a real intrusion or this observation sequence is merely a new path that the legitimate user takes.

As shown in Figure 4.1, the architecture of our IDS consists of following components:

- Data Collection
- Data Normalization
- Building UM Profile and UM Pattern with Features
- Classification with HMMs

To satisfy these designed requirements, we implement a simulation by using C++ on Linux. From an implementation point of view, our program can be divided into

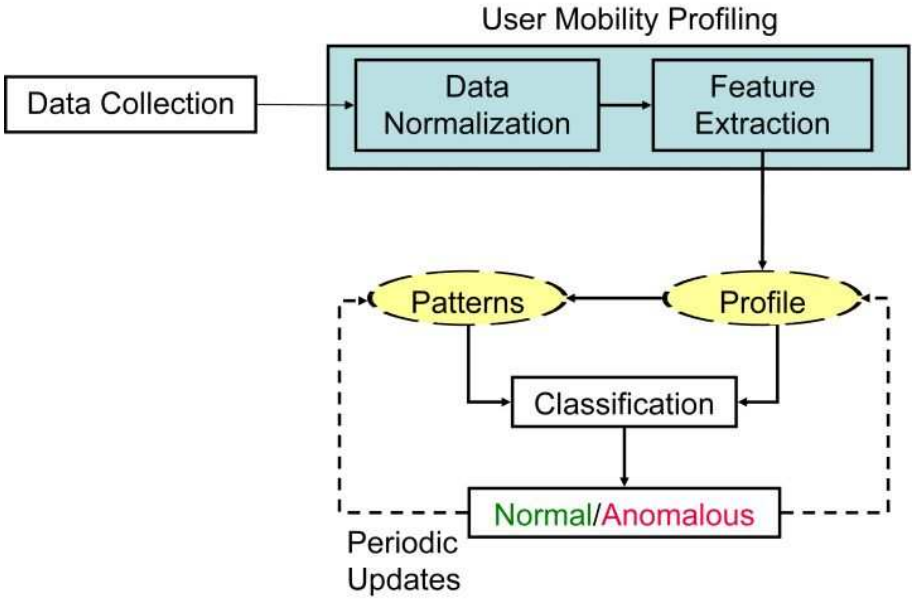

Figure 4.1: Architecture of IDS with HMMs framework

two parts:

- UM Profile and UM Pattern Maker
- Classification Part

The function of the UM profile and UM pattern maker is to read raw location updates from a stream of location coordinates along with other corresponding information, eliminate noise according to UM properties, extract other mobility features bound with location coordinates from the data stream, and then make the UM profile and UM patterns based on these features.

The classification component is used to identify users with the HMMs classification method according to the UM profile and UM patterns that are created in previous step. This component compares each observation sequence in the UM profile with

every UM pattern. If the observation sequence matches any one of the mobility patterns stored in the UM profile, in other words, if the probability of this observation sequence coming from the mobility pattern is larger than the pre-established threshold, then this observation sequence is considered to belong to one legitimate user. Otherwise, the IDS is trigged and report this as an intrusion.

Before we jump into the details about how the new IDS is designed, how the UM profile and UM patterns are defined, as well as how we implement system functions; we need to make the following assumptions:

1. If mobile devices are active, they will broadcast a signal periodically. But from the real data we collect, the interval duration between two sequential messages may be different from interval to interval. To deal with this irregularity, we assume that some messages are missing during the transmission. To improve performance, we pad the missing location coordinate message in intervals to make message transmission appears periodic.

2. If mobile users follow their normal mobility patterns, the time spent on the road from a start point to an end point will be roughly same. The normal mobility pattern here is the sequence of locations that occur most frequently each day. If things change too much, for example a big traffic jam, even if the user takes the same path from the same start point to the same end point, our system might still classify this sequence as an anomaly.

3. Because of the transmission speed of radio signals, we make the assumption that the receiving time $time\_rx$ is exactly the same as the transmission time $time\_tx$. We don't take into account the delay between transmission and reception.

4.2 Data Collection

The first step of this project is to collect data that show UM characteristics from different users is for the same purpose as in the previous chapter. However, in this experiment, not only location coordinates are collected from user broadcast messages, but also other related information (e.g., transmission time, speed, and course) is extracted.

The purpose of this step is to acquire enough UM data to build the UM profile and patterns.

4.3 Data Normalization

The data collection processing is done in the same manner as in the first experiment. All user message data are stored in a database (see Table 4.1). For each user the message data can be considered as one big data stream. Figure 4.2 shows a logical view of this structure. Each square represents a specific location coordinate in a chain linked with other squares in chronological order. Different square shows the mobile property at the current specific location coordinate. The square with a cross inside stands for moving, and the square without anything inside stands for stationary.

We choose the following features extracted from Table 4.1 to determine and eliminate noise. These features are also used to create the UM profiles and UM patterns.

- Location coordinates (lat & lon)
- Call (user identity)
- Receiving time ($time\_rx$)

- Speed
- Course (direction)

Field	Type	Field	Type
Cnt	int(11)	Icon	char(2)
time_rx	Timestamp	Speed	decimal(4,1)
time_tx	Datetime	Course	decimal(4,1)
Call	char(10)	Raw	int(11)
Lat	float(8,5)	Altitude	int(11)
Lon	float(9,5)		

Table 4.1: Data fields in database

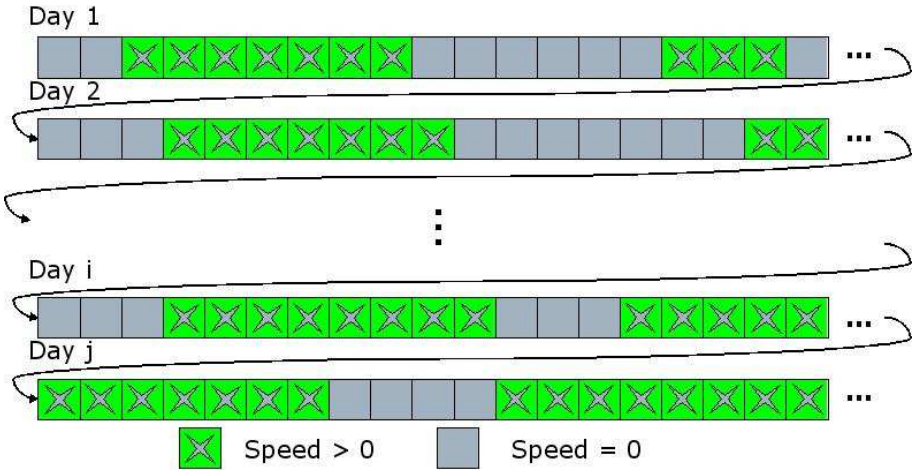

Figure 4.2: Stream of location coordinates in logical view

In this data preparation phase, we assume that all the data received is from legitimate users. When checking the location coordinates, none of the messages are considered from intruders. Either it is normal data we need, or it is some type of noise, which we need to eliminate from the data stream. When we analyze the data

collected from the previous process, there are three types of errors that influence the performance of our IDS.

- Irregular Transmission

  According to the design requirements, users should broadcast their location updates periodically. However, in the real-world situation, the broadcast messages coming from users are usually irregular. For example, if one user's broadcast periodicity is set as 60 seconds, the time interval sometimes occurs to be up to 10 minutes between two sequential location updates. Irregular message transmission affects the accuracy of UM profiles and extracting UM patterns, and is the main cause of high false detection rates.

- Information Lost during Message Relay

  Some location updates relayed by other radios may have lost part of the information, because not all users apply the same broadcast message format. If these radios relay messages, part of the messages, such as speed, course, and other information are lost with respect to the original. We use speed to determine UM status, timestamp to calculate the broadcast periodicity, and course to pad the missing location coordinate. Any lost information can affect the precision of the UM profile and UM patterns we create. This type of lost information limits the ability of IDS's to accurately classify legitimate users and intruders.

- Messages Duplication during Relay and Message Interference during Transmission

  Since other users' devices can relay the messages they receive, the server might receive two or more messages containing exactly the same information; only the

receiving time marked by the server will be different. This is one type of noise we defined. The other type of noise is interfering messages. We discuss these in detail in Section 4.3.1.

For the first two errors, irregular transmission and information lost during message relay, there is no a good solution, because these events have already happened. We can not set the clock back to see what actual data was sent at that time. However, to solve the third error, we can filter out the duplicate messages and interfering messages by browsing the raw data stream and comparing these messages with normal data.

4.3.1 Eliminating Data Noise

The UM data we received from the radios is full of noise. Quite obviously, if we can not remove noise from the raw data, this noise will interfere with the ability of IDS to correctly distinguish between legitimate users and intruders. In other words, the TDR will drop and the false alarm rate will rise. The purpose of data normalization is to improve the system performance by applying one of the mobility features, broadcast periodicity.

After we analyze the data that we obtain from database we define two types of noise for our IDS.

Type I Noise

Users can set their radios to relay other users' messages to help extend the range of these messages. After setting this property, when users' radios receive location updates from other users, the radios repeat these reports automatically. For example, when user A broadcasts his location updates, if another user B stands between the station and user A, and the distances between station and these two users are within

the transmission ranges of both users' radios, the station can receive the location reports of user A more than once. One copy is from user A directly, and the other copy is from user B. The station can not tell the difference between these two messages, but records them into the database. This situation is even worse, if more than one user stands between user A and the station. The resulting redundant data can be considered to be a type of noise, since this duplicate data alters user trajectory (see Figure 4.5 - 4.6).

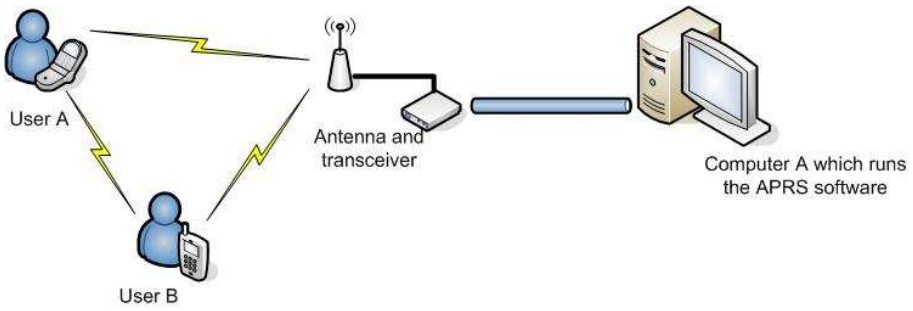

Figure 4.3: Type I noise

Receive Time	Location	Speed	Course
T1	L1	S1	C1
T2	L1	S1	C1
T3	L3	S3	C3
T4	L4	S4	C4
...	...	...	...

Table 4.2: Type I noise representation in database

Type II Noise

Because of the inaccuracy of users' GPS receiving devices or due to interference during the message transmission, part of the information in messages, such as location coordinates, speed and course, can be changed dramatically. As shown in Figure 4.4,

the user's trajectory may appear completely different due to this type of noise.

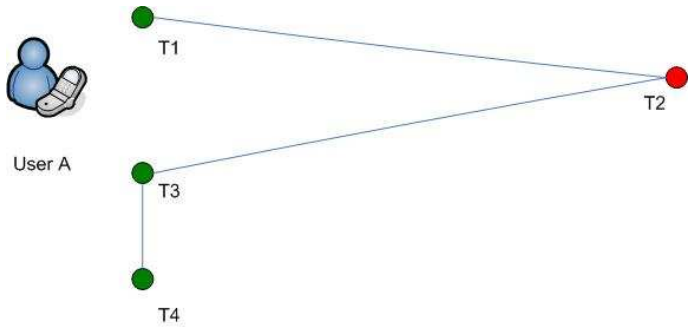

Figure 4.4: Type II noise

Receive Time	Location	Speed	Course
T1	L1	S1	C1
T2	L2	S2	C2
T3	L3	S3	C3
T4	L4	S4	C4
...	...	...	...

Table 4.3: Type II noise representation in database

Figure 4.5 and Figure 4.6 show the comparison between raw data and the data after eliminating the noise.

It is important to eliminate these two types of noise before we create the UM profile and UM patterns. First we need to calculate the user's broadcast periodicity. This is achieved by observing all the time intervals between the sequential broadcasts and determining the one most frequently appearing for each user. Different users may have different broadcast periodicity according to their own choice, so this interval time is an important feature used to classify different users. When we read each update from the stream of messages, which includes location coordinates, transmission time, speed and course, we record every interval time between two sequential updates and

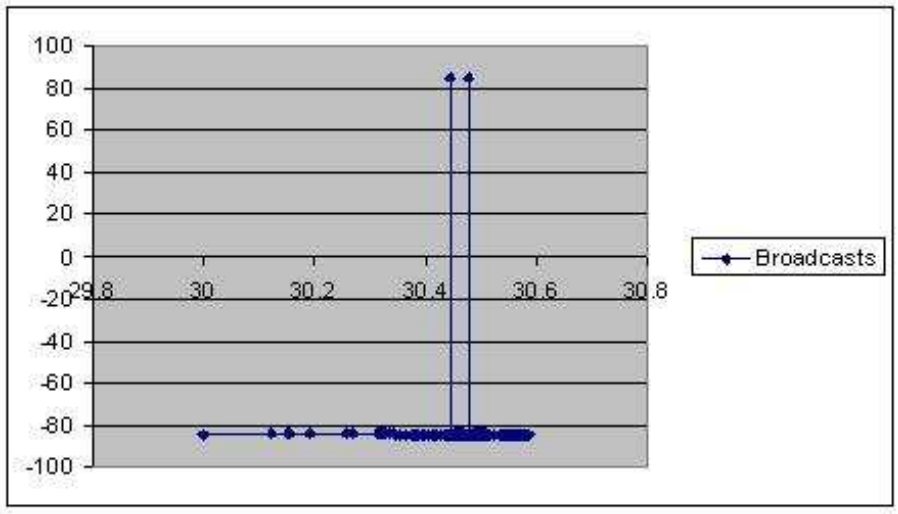

Figure 4.5: User trajectories before eliminating noise

count how often these intervals show up. After all location updates have been read, we select the interval time that has shown up most frequently as user's broadcast periodicity. In the rest of the implementation, this periodicity is used to build UM profiles and UM pattern files, to make user classification.

Once the broadcast periodicity has been determined, the next step is to eliminate noise from the raw data stream. To erase duplicate messages (type I noise), the process needs to go through each row of messages and check the interval time between two sequential messages. If any interval time is less than 75% of the broadcast periodicity, we review recent messages to see whether there is one message which contains the same location coordinate, speed, and course. If it exists in the data stream, we mark the second one as noise and delete it from the data stream. We limit the checking range to within 6 time periodicity or within the last 6 minutes, since the user may pass the same location coordinate at the same speed and course on different days, but it is improbable for this to occur within such a short period.

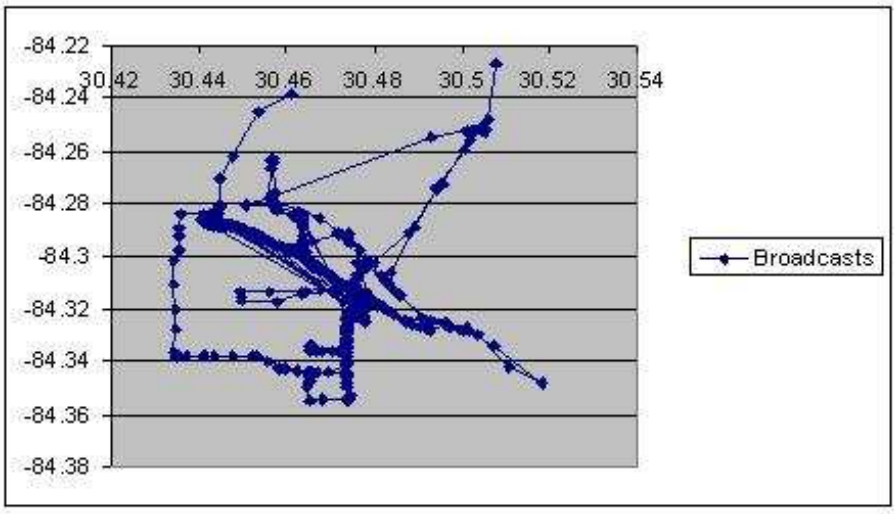

Figure 4.6: User trajectories after eliminating noise

This process is repeated until it reaches the last record in the data stream.

For type II noise, we can only determine that one message is noise after we do a calculation based on the broadcast periodicity and other mobility features, such as speed and interval time between these two sequential location updates. We can do the elimination process for type II noise in one of two possible places:

1. By processing type II noise right after finishing handling type I noise. After that we extract UM sequences from the data stream without any noise and store them into an UM profile.

2. We can eliminate type II noise while building the UM profile.

If we choose the first method, we need to perform extra calculations on the stationary location coordinates. Therefore, the second method was chosen to save processing time.

4.4 Building User Mobility Profile and User Mobility Pattern with Feature Extraction

After finishing type I noise elimination, in order to create an UM profile for each user, we extract mobility sequences from the data stream. Once created, we use this profile to build the UM patterns. Meanwhile, we also eliminate type II noise from the mobility sequences.

4.4.1 Definition of User Mobility Profile and User Mobility Pattern

A UM profile is defined as a set of mobility sequences containing location coordinates which represent user TRAJECTORIES with respect to other corresponding mobility features (e.g., transmission time, speed, and course). A UM pattern is defined as a general sequence of location coordinates, which is abstracted from all similar TRAJECTORIES to represent each unique PATH this user has taken. Each user has one mobility pattern file that stores a set of UM patterns. The broadcast periodicity is also stored in this file, which is useful to identify users.

Since the whole data stream for users is actually a long sequence of location coordinates with other corresponding information, to extract the UM profile we only need to find the start and the end points for each TRAJECTORY. All location coordinates between the start and end point are stored as one record in the profile with the start and end points. Other related features collected as the same time as the location coordinates are also stored.

The following tables (Table 4.4 - 4.5) describe the format of the UM profile and

the UM pattern.

Components		Description
Call sign		User's Identification, file name with suffix ".ptt"
Data record	Transmission time	The time when user sends the message out
	Location	The location coordinate. latitude/longitude
	Speed	The speed at that location coordinate
	Course	The direction at that location coordinate an angle from 0 to 360 degrees

Table 4.4: Structure of UM profile

Components		Description
Call sign		User's Identification, file name with suffix ".pan"
Broadcast periodicity		The time elapsed between messages
Data record	Location	The location coordinate. latitude/longitude

Table 4.5: Structure of UM pattern

The relationship between TRAJECTORY and PATH here is that each TRAJECTORY is a sequence of location coordinates which represents a single trip a user has made, and each PATH is a sequence of location coordinates which is compiled from data gathered from a series of similar trips. In other words, there can be more than one similar or even identical TRAJECTORIES in the UM profile, but only one unique PATH to represent these similar TRAJECTORIES.

From an implementation point of view, we extracted the user's TRAJECTORIES first, and then abstracted the specific PATH based on these related TRAJECTORIES. We combine the PATH and TRAJECTORY together to acquire the observations to be used as input to our HMMs. More detail will be included in the next section which introduces the relationship between hidden states and observations in the HMMs method.

4.4.2 Feature Extraction

The feature extraction process finds the first location coordinate where user velocity is greater than zero in the data stream. Then the process monitors the user's speed moving from one location coordinate to the next until the velocity again equals zero. Of course, users may have stopped half way to their destination due to traffic lights or stopping for gas, so we defined the end location coordinate of one mobility sequence to be the point when the duration of zero velocity exceeds a pre-established threshold. Hence, all location coordinates between these two points are considered as one record of a UM sequence. This process is repeated until all the data in the data stream was exhausted. All resulting sequences are stored in a temporary file and are applied as input to create the UM pattern and for the classification phase.

We extract each specific sequence of location coordinates with other features. In this project, we are trying to prove that normal mobile device subscribers have their own mobility patterns, and these patterns can be used to identify each wireless subscriber. However, users often stay in one place for a long time, and usually spend a short time on the road. In other words, if users' devices report their location updates during the whole day, most of the updates will be stationary. Table 4.6 shows the proportion of moving location coordinates and total numbers of location coordinates.

Call sign	Total number of location coordinates	Number of moving location coordinates (Speed > 0)	Proportion
w4gcw	13855	1419	10.24%
w4src	19534	2285	11.70%
vk4ag	10673	481	4.51%
vk3ur	14734	1300	8.82%

Table 4.6: Relationships between total number of location coordinates and moving location coordinates

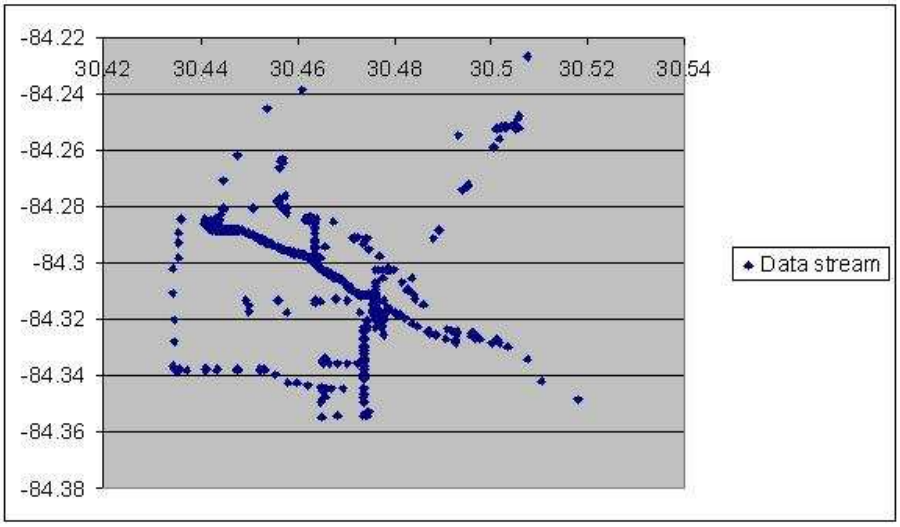

Figure 4.7: Raw data stream

The reasons behind eliminating the stationary location coordinates from the data stream are the following. First, we build our IDS based on UM patterns. Our framework can not make any claims about the legitimacy of a user based on stationary coordinates. Second, these repeated stationary location coordinates only offer one piece of information such as: from time A to time B the user stays at place C. We don't have to store redundant location data in the database.

The following figures (Figure 4.7 and Figure 4.8) show the comparison between the raw data stream and the data sequences after eliminating the stationary location coordinates. Notice that Figure 4.7 is identical to Figure 4.8. No useful information has been lost.

After we obtained all sequences of location coordinates that represent the users' TRAJECTORIES, the next step was to remove the type II noise from the sequences. The result generated by Formula 4.1 and 4.2 are compared. To identify type II noise, we first compute the distance between two consecutive location coordinates

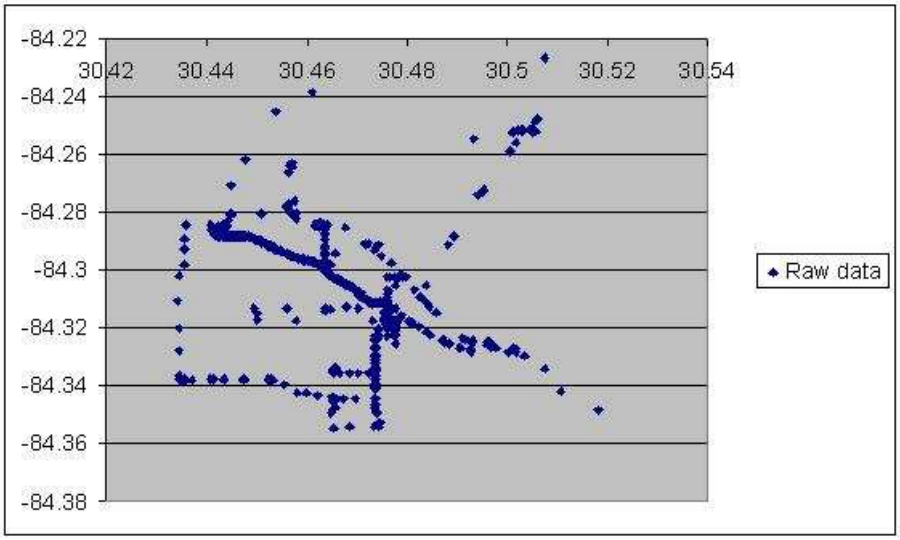

Figure 4.8: Data stream after eliminating redundancy data

using Formula 4.1 then compare this distance with the product of the speed and the interval time between these two location coordinates using Formula 4.2. If the two calculated distances differ by more than some threshold, the Data Normalization process marks this location coordinate as type II noise, and eliminates it from the sequences.

$$Dis = \left[\sin\left(\frac{lat_1}{57.2958}\right) \cdot \sin\left(\frac{lat_2}{57.2958}\right)\right] + \left[\cos\left(\frac{lat_1}{57.2958}\right) \cdot \cos\left(\frac{lat_2}{57.2958}\right) \cdot \cos\left(\frac{lon_2 - lon_1}{57.2958}\right)\right] \tag{4.1}$$

$$Dis = speed \cdot (time\_rx_2 - time\_rx_1) \tag{4.2}$$

4.4.3 Standardizing the End Points

So far, we eliminated noise and also extracted all mobility sequences of location coordinates from the data stream. Now, we can store these mobility sequences into one file as the UM profile. In order to create UM patterns and make classification,

one more step is needed. We need to identify which sequences of location coordinates can be represented as one PATH to abstract the UM patterns. In other words, we need to cluster similar sequences together. How can we determine whether sequences are similar? In this phase, we say two mobility sequences are similar, if the two have the same start and end points. More details are covered in the next section.

To abstract the same path, we first cluster similar sequences of location coordinates together. However, due to the inaccuracy of GPS receiver systems, location data in the message may not be correct. Even data collected from stationary users may contain errors. These errors, depending on the various GPS devices, can exceed 20 meters and can even reach 50 meters. For clustering purposes, when we acquire the set of all location sequences, we need to conduct the conversion processing on the start and end points of each sequence to eliminate these deviations.

This conversion process applied on each start and end point of a user's trajectory can be described as follows. The subroutine reads the start and end points from all the sequences, and stores them into two identical data structures. We take the start points as an example. The subroutine calculates the difference between any two points. If the distance is smaller than a pre-established threshold, (e.g., 20 meters) then these two points will be considered potentially to be the same point. When all points have been examined, the center point is determined from the set of points within the threshold distance. The same process is done for the end points. The distinction between start and end point is important since this distinction can be used to differentiate trips with similar trajectories but taken in the opposite direction.

Figure 4.9 shows raw start points and the grouped start points. Figure 4.10 is for the end points.

After we convert the start and end points, each sequence of location coordinates

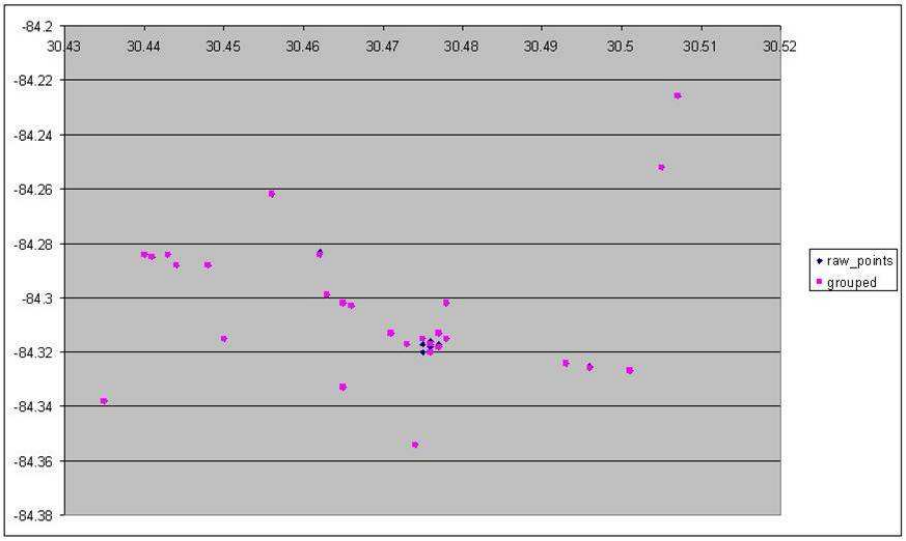

Figure 4.9: Grouped start points

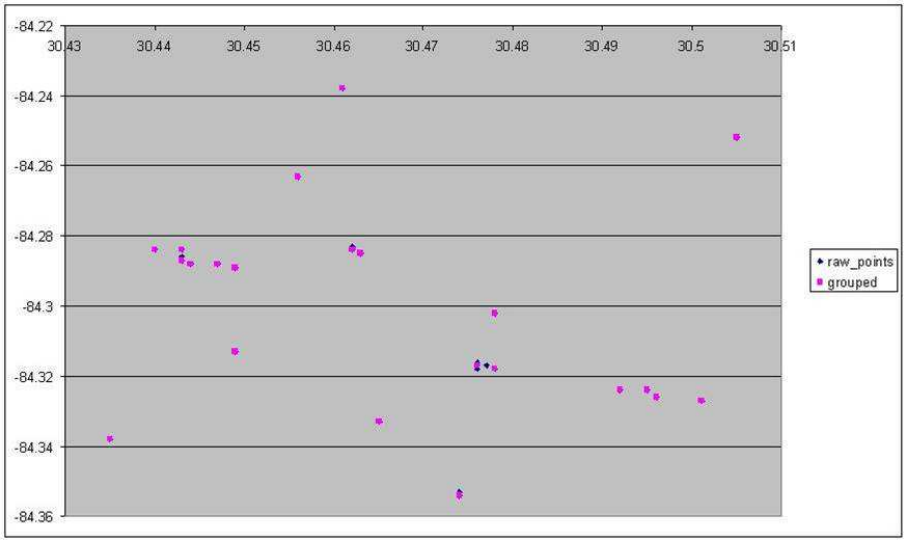

Figure 4.10: Grouped end points

can be stored in the UM profile with corresponding information, (e.g., transmission time, speed, and course). This UM profile will be used to create the UM patterns, and as input for the classification in the next phase.

4.4.4 Making User Mobility Patterns

As we mentioned in the previous section, each UM pattern is one abstraction for all similar UM sequences in the UM profile. The purpose of building an UM pattern is to use one single location sequence to represent all similar trips. There are two benefits to do that.

1. It can save storage. An IDS only needs few memory to store highly condensed patterns.

2. When a pattern has been created faithfully, it can help the IDS to make the correct classification decision.

The procedure of making UM patterns is to find all similar sequences from the UM profile and then extract a general sequence of location coordinates as a representative UM pattern. This is an iterative process, and the term *similar* here has two levels of meaning.

In the first step, the similar sequences are defined as any two sequences that have the same start and end points. If any number of sequences are clustered as similar sequences, the procedure also records all location coordinates of each sequence and employs the K-mean clustering method on this set of location coordinates to calculate K mean values. Here K is the mathematical mean value of the number of broadcasts for these similar sequences. It will also be used to describe the number of hidden states in the next section.

$$K = \lfloor \frac{B_1 + B_2 + B_3 + \ldots + B_n}{n} \rfloor$$

with:

$$B_i = \text{number of broadcasts in the sequence } i$$

We store this K-mean sequence together with the group of similar sequences, and use it as a representative for these grouped sequences. After all sequences are clustered, a set of K-mean sequences are retrieved, and then they are ordered by the number of the sequences stored in each group and the length (number of the location coordinates) of each K-mean sequence, or the K value. The greater the number of sequences in one group and the larger the K value of the K-mean sequence, the higher order we put it in a temporary file.

It is not precise to say that similarity of sequences is based on the same start and end points. Two mobility sequences that are similar but without the similar start point or end point will not be classified into the same group. In the second step, we solve this problem by changing the definition of term *similar*. If the number of location coordinates of a K-mean sequence that has fallen into a virtual region composed of another K-mean sequence is greater than 75% of its own K value, we say these two K-mean sequences are similar. This means that all of the mobility sequences represented by these two K-mean sequences are similar. The procedure starts comparing remaining of K-mean sequences with the first K-mean sequence. If any K-mean sequence can match the first K-mean sequence by more than 75%, the procedure will consider this K-mean sequence to be *similar* to the first one. Then the procedure merges the latter group of sequences into the former one, and does the same thing in the first step to make a new K-mean sequence. This procedure repeats itself until it goes through all remaining of K-mean sequences in the temporary file. The final K-mean sequence is saved as one UM pattern. Then the procedure starts

the same process from the next K-mean sequence available in the temporary file until there is no K-mean sequence left. All the final K-mean sequences are the UM patterns, which will be used for the user classification.

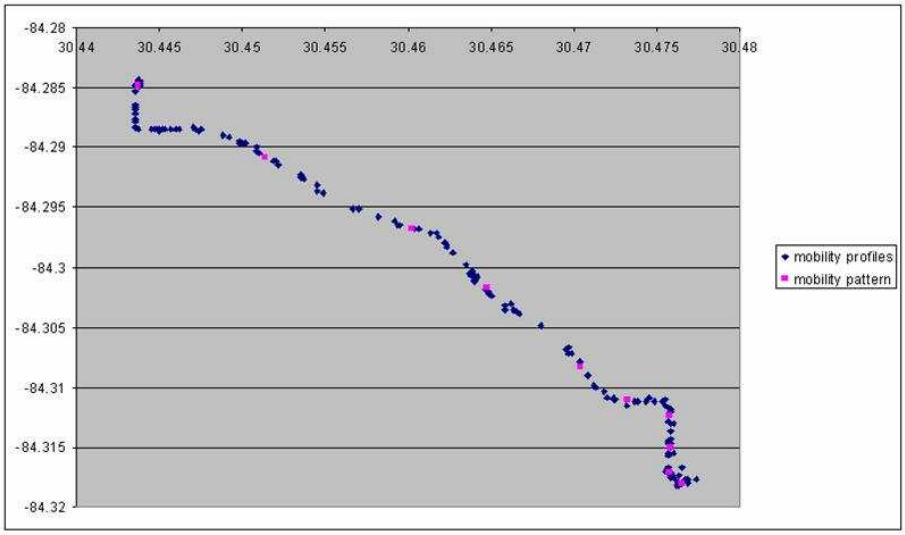

Figure 4.11: A UM pattern and the UM profile it represented

4.5 Classification

In the file preparation phase, we create the UM profile and UM patterns. The final step is to make classification between legitimate users and intruders based on these two files. In this experiment, we employ a new classification method, the HMMs.

4.5.1 Hidden Markov Models with the Project

In the ideal situation, we know a user broadcasts K message updates for each trip. Ideally, the broadcast sequence generated should be the same whenever the user takes the same trip. In this case, user's mobility states can be represented with

these location updates directly, and the transition probabilities among these states can also be obtained. After that, a Markov Chain model based on these transition probabilities can be created to calculate the probability of UM sequences produced by the given model.

In reality, the message broadcasts are distributed all along the entire path (see Figure 4.11) and the K broadcast locations can not be determined precisely. Therefore, it is difficult to make one to one mapping between broadcast locations and UM states as in the ideal situation. Since the UM states are not observable, they are called as hidden states. However, we can find the relationship between the UM states and the observation symbols that represents the region in which the broadcast locations fall (see Figure 4.12).

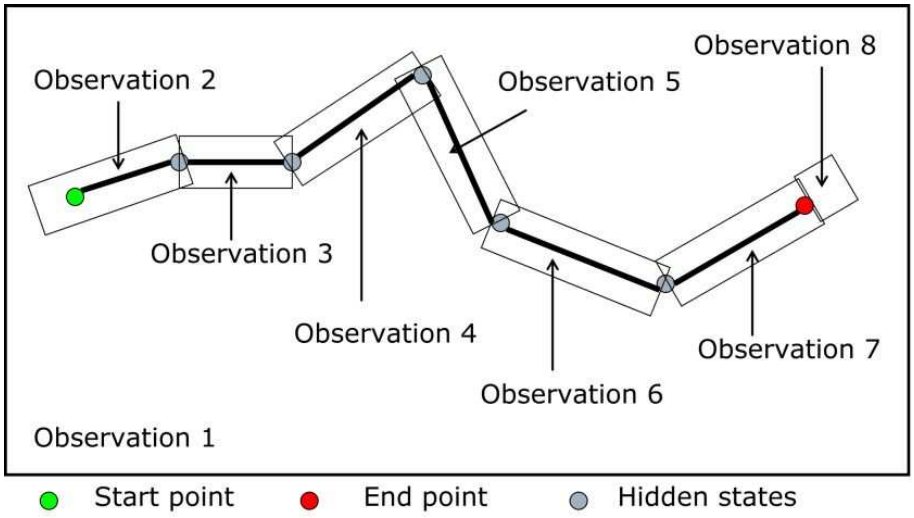

Figure 4.12: Making observation sequence

To obtain the observation symbols, any two consecutive location coordinates in the UM pattern are used to compose a virtual rectangular region that has a observation number according to the position in the UM pattern. If a location coordinate of

the UM sequence in the UM profile is within an observation region, the corresponding number will be used to represent this location coordinate. Otherwise, if this location coordinate falls outside of any regions composed of these K-mean location coordinates, it is assigned a value one as its observation symbol.

The type of HMMs we use in our project is called the left-right model or the Bakis model (see Figure 4.13), because the underlying state sequence associated with the model has the property that as time increases the state index increases (or stays the same).

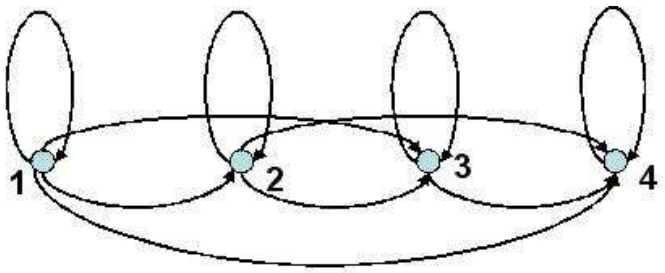

Figure 4.13: An example of four-state left-right model

Our classification procedure consists of two parts which are training the HMMs with UM patterns (Baum-Welch method in Section 2.3.2, Formula 2.7, 2.8, 2.9), and the evaluation of the observation sequences in UM profile (Forward-backward procedure in Section 2.3.2, Formula 2.2, 2.3, 2.4) with the HMMs we create. In this project, we are not interested in how to find the corresponding hidden state sequence for the observation sequence from the given model. We use the variants [Rab90] from Rabiner's HMMs notation. The number of hidden states for each HMM is K, which is defined as the average number of broadcast updates during a PATH.

We create HMMs which have K number of hidden states for each mobility pattern belonging to user. The sequences in the UM profile are the data we gathered from

user directly, so we combine these sequences with UM pattern to obtain observations by the same method we used in the previous section. Each location coordinate in the sequence from UM profile contributes for one observation symbol. After that, a UM sequence is converted to an observation sequence.

In the classification phase, if the calculated probability result of an observation sequence produced by the given HMMs (solution for evaluation problem) is smaller than a pre-established threshold, then our IDS claims that the observation sequence is not coming from this particular HMMs.

However, HMMs have a major shortcoming. To build HMMs (solution for training problem), a large amount of data is needed. Normally, over one hundred iterations are needed to make stable probability transition matrices, depending on the precision required. Each iteration needs one UM sequence as input. For our experiment, the time (one and a half months) was insufficient to collect enough data to satisfy this requirement.

Due to the lack of data, we can not design our IDS framework as we implemented (divide the UM profile to 3 parts) in Chapter 2. However, the goal of our project is to prove that the UM patterns extracted from the UM profile can be used to identify each distinct profiled user. Following this idea, we found it is not necessary to divide the UM profile to 3 parts; one for user profile, one for getting parameters, and the last one for testing in this simulation. The entire UM profile can be used as a test set. For the training set, we just selected one UM sequence from UM profile for each corresponding UM pattern, and then use this one as input for training HMM. The training process repeats as many times required with the input to make three stable transition probability matrices (e.g., A, B, Π). There are three reasons why we were doing that:

- First of all, we don't have enough data for training HMMs.
- Secondly, this action is based on the assumption that users repeat the daily routines. So it is reasonable that we take one UM sequence from UM profile and then use it many times.
- Thirdly, if we can prove these HMMs created on fake data can be used to identify users from the test set, we also can believe these HMMs can be used to identify users in the future.

To train HMMs for one UM pattern, firstly, we selected one UM sequence that can be converted to a sequential observation sequence without any missing or repeated observation symbol from the UM profile. For instance, if one UM pattern has nine hidden states, and after we apply this UM sequence on the pattern, we can obtain the observation sequence 2, 3, 4, 5, 6, 7, 8, 9, 10.

Secondly, we set initial values as random for three matrices: the state probability matrix, the observation probability matrix, and the initial state distribution, and then take the observation sequence as input to adjust these three matrices using the solution to the training problem in Section 2.3.2. This process is repeated until the difference between this iteration and last iteration is smaller than a pre-established small value ($|A_{i+1} - A_i| < \sigma$).

Finally, once the three matrices are obtained, we use them to set the threshold and perform the classification.

4.5.2 Obtaining Threshold and Probability Distribution

Setting a threshold for classification is another problem because of lack of data. As you may recall, to build HMMs for each UM pattern, we select a UM sequence which can be converted to an observation sequence without any missing or repeated observation symbols. To set a threshold for the HMMs, we reused this observation sequence, except that we modified 25% of the values in the observation sequence. Table 4.7 illustrates how to get a sequence for threshold setting.

Original Sequence	2, 3, 4, 5, 6, 7, 8, 9, 10	$P(O\|\lambda) = 0.99999$
Modification sequence	1, 1, 4, 5, 6, 7, 8, 9, 10	$P(O\|\lambda) = 3.45204E-12$

Table 4.7: Example of original sequence and modification sequence

Then we apply the modified sequence on the corresponding HMMs to acquire the probability value using the solution to the evaluation problem in Section 2.3.2 (Formula 2.2, 2.3, 2.4). This probability value is used as a threshold for the HMMs.

To obtain the probability distribution on the sequences in the UM profile, same methods are applied on the similar UM sequences. Figure 4.14 shows the probability distribution of 31 similar UM sequences applied on its corresponding UM pattern. The x-axis stands for the UM sequences, and the y-axis is the log of probability value for the UM sequence on that given UM model. If the probability value is smaller than the threshold, then this sequence is classified as from an intruder. In this case, it will be marked as the false detection.

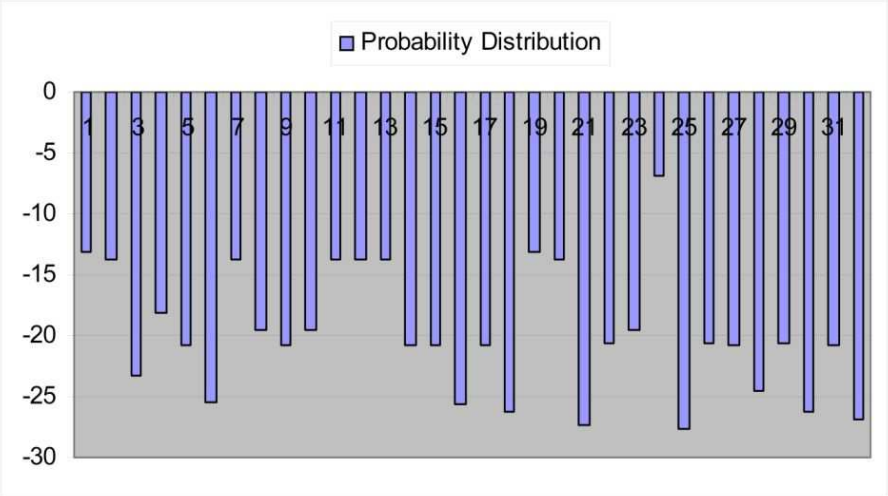

Figure 4.14: The UM profile probability distribution on its mobility pattern

4.5.3 Padding Missing Location Coordinates

At the beginning of this chapter, we mention the problem of missing message updates. So to improve the accuracy of results, it is helpful to pad missing location coordinates into the observation sequence by using speed, course, and other location coordinates.

Figure 4.15 shows how the padding of location coordinates is calculated. Suppose the interval time between location coordinate 3 and 5 is greater than 1.5 times the broadcast periodicity. According to our assumption given in Section 4.1, we determine that at least one location update was lost during transmission.

The padded location coordinate (L_4) is the mean value of two calculated location coordinates using Formula 4.8; one location coordinate ($L_{4'}$) is calculated with the previous speed and course using Formula 4.3, 4.4, and 4.5, and the other location coordinate ($L_{4''}$) is the orthogonal point on the line formed by the previous location coordinate (L_3) and the next one (L_5) using Formula 4.6 and 4.7.

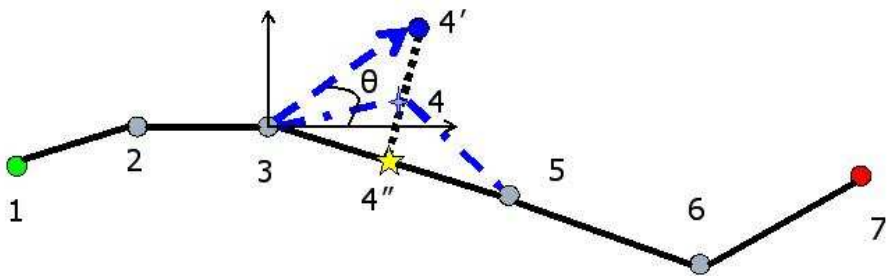

Figure 4.15: Padding a missing location coordinate

$$\triangle d_{4'3} = S_3 \times T_{periodicity} \tag{4.3}$$

$$L_{4'} = \triangle d_{4'3} + L_3 \tag{4.4}$$

$$\theta = C_3 = \arctan(\frac{L_{4'}.lon - L_3.lon}{L_{4'}.lat - L_3.lat}) \tag{4.5}$$

$$L_{4"}.lat = L_3.lat + k \times (L_5.lat - L_3.lat) \tag{4.6}$$

$$L_{4"}.lon = L_3.lon + k \times (L_5.lon - L_3.lon) \tag{4.7}$$

with:

$$k = \frac{(L_5.lat - L_3.lat) \times (L_{4'}.lat - L_3.lat) + (L_5.lon - L_3.lon) \times (L_{4'}.lon - L_3.lon)}{(L_5.lat - L_3.lat)^2 \times (L_5.lon - L_3.lon)^2}$$

$$L_4 = \frac{L_{4'} + L_{4"}}{2} \tag{4.8}$$

Here $\triangle d_{4'3}$ is the distance between L_3 and $L_{4'}$, S_3 is the speed at L_3, $T_{periodicity}$ is periodicity, and C_3 is the course at L_3.

Table 4.8 shows the probability changing before and after padding. From this table we can see if we pad two missing location coordinates into an observation sequence with eight observation symbols (the first and second row), the probability of

the observation sequence from a given model (the same as the given model in Section 4.5.2) would rise efficaciously from 10^{-10} to 10^{-5}. We also need to consider whether this padding process would increase the possibility that an observation sequence generated by an intruder is accepted as legitimate user. Suppose the process pads two coordinates into an observation sequence from another user (the third and fourth row), the probability increases from 10^{-24} to 10^{-21} that the observation sequence will still be identified as from a different user.

Sequence length	Observation sequence	$P(O\|\lambda)$
8	2 2 3 4 1 6 1 8	1.261348E-10
8	2 2 3 4 5 6 7 8	6.312796E-05
8	1 1 1 1 5 1 7 1	4.079319E-21
8	1 1 1 1 1 1 1 1	1.049020E-24

Table 4.8: Probability before and after padding location coordinates

4.6 Simulation

The objective of this simulation is to assess the performance of the HMMs framework. To illustrate the effectiveness of HMMs framework, we use the same principle to obtain the performance criteria, the TAR and the TDR, for the same five users from the last simulation.

4.6.1 Details of Simulation

The simulation is carried out for each user in the IDS using his/her profile and patterns.

1. Select a user and compare his UM profile against his UM patterns in the IDS to obtain TAR.

2. Select a user and compare his UM profile against other users' UM patterns in the IDS to obtain TDR.

3. Get the final statistics.

4.6.2 Results of Simulation

True Acceptance Rate

Figure 4.16 presents the results of the second simulation. In this simulation we use the same five users as in last one. The x-axis of the bar chart shows call sign of the five users, and the y-axis of the bar chart represents the TAR for each profiled user.

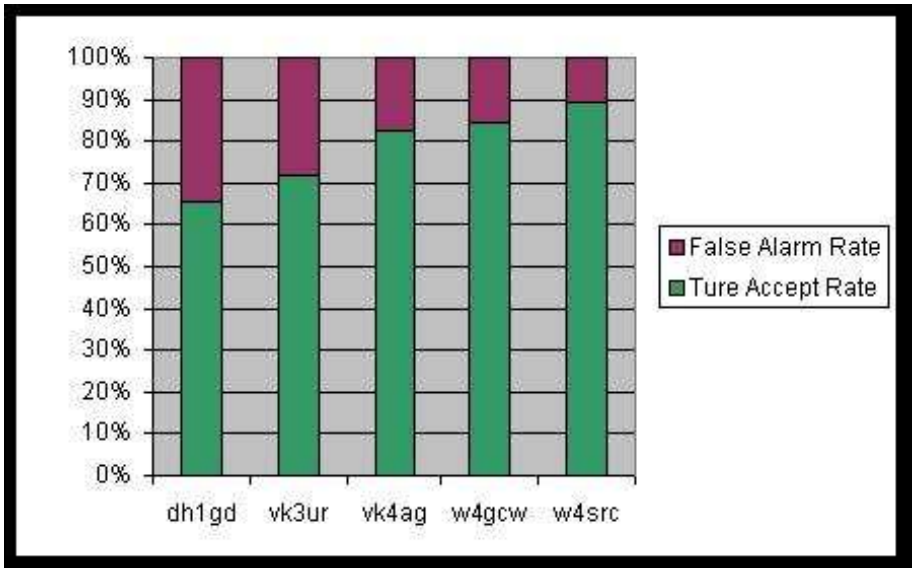

Figure 4.16: True acceptance rate

We apply one user UM profile against his own UM patterns to produce the TAR. From Figure 4.16 it can be seen that the TAR range from over 65% to 90%. The differences of the TAR among these users depends on how the UM profile made. The more specific UM profile is, the higher TAR we obtain.

True Detection Rate

Figure 4.17 illustrates the TDR for the same five users. Like the previous plot, the x-axis of the bar chart shows the call signs of five users, and the y-axis of the bar chart represents the TDR for each profiled user.

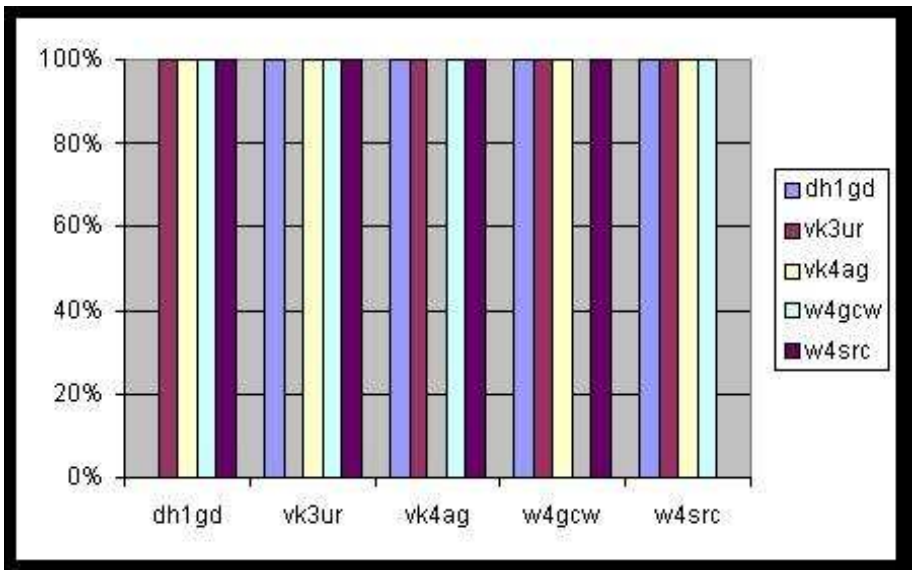

Figure 4.17: True detection rate

To obtain the TDR from each user, we compared the remaining four users' UM profile against this one user's UM pattern.

As in Figure 4.17 shown, the TDRs are 100% for all five users. These results are expected since, to each user, the UM patterns are different. Our IDS can correctly pick out the mobility sequences that do not belong to legitimate users.

4.7 Conclusions and Analysis

Same with the first experiment, we also check the complexity of the IDS with HMMs. Suppose we have already built HMMs for each UM pattern. The main computation for this IDS is to calculate the probability value of incoming UM sequence, a given model. As we mentioned in Section 2.3.2, to calculate the probability of a sequence, the forward-backward procedure (Formula 2.2, 2.3, 2.4) is applied. It requires on in order of N^2L calculations [Rab90]. Here N is the numbers of the hidden states, and L is the length of the sequence. In this experiment, L and N are in the same order of magnitude, so the time complexity is $O(N^3)$.

From former sections, we know only abstraction of UM sequences are stored into UM profile. So when IDS makes the classification, it only needs to compare the incoming sequence with these abstraction sequences, instead of comparing with all of the UM sequences the user has. However, UM patterns change sooner or later. It is necessary to update the UM patterns in the profile every period of time to reach better performance.

From the plots above (Figures 4.16 - 4.17), not surprisingly, we find the results from HMMs framework are worse than the results from IBL framework. To build a useful HMMs framework, a large amount of data is required. In our circumstances, lack of data causes the performance to be poor because the built models can not reflect the UM pattern faithfully. Also, in the HMMs framework, only mobile data can be used to create UM profiles and UM patterns. However, we analyze our experiment data for these five users, (see Table 4.6), only around 10% data can be employed in our simulation, which makes the situation worse.

Another factor which reduces performance is users' irregular location updates.

Although we have a procedure which helps to improve the TAR to some degree by padding location coordinates. The true missing location coordinate can not be recovered.

However, in real wireless system, these two problems might not occur because the location updates with the same format are transmitted periodically.

Chapter 5

Conclusions and Future Work

5.1 Concluding Remarks

With the increasing use of wireless networks, a proper mechanism is needed to identify intruders from legitimate users quickly and correctly so as to protect legitimate users' properties from malicious activities. Therefore, researches on IDS over wireless networks have been a popular topic. An efficient intrusion detection mechanism need to balance its performance against its cost in the context of wireless networks. The work of this thesis can be summarized in the following aspects:

- Design and implement the IDS framework by employing two different approaches: IBL and HMMs, respectively.
- Develope a UM profile and UM patterns for the IBL and the HMMs frameworks.
- Conduct simulations of these two frameworks based on the authentic data from five different users as input.

- Analyze performance in the TAR, the TDR, and the false alarm rate of IBL based and HMMs based implementation.

A summary of the performance evaluation for different strategies is given in Section 3.6 and Section 4.7.

5.2 Future Work

After we give the design and implementation of one IDS with an IBL classification method in Chapter 3, the other IDS with a HMMs classification method in Chapter 4, and carry out the analysis of all these two framework combined with the simulation results in Section 3.6 and Section 4.7, several interesting problems remain for further investigation.

1. In the current IBL framework, the process for calculating the similarity values between incoming sequence and sequences in the UM profile is not precise. It might be extended by computing the similarity value with transition probability between two states.

2. Presently, the IBL framework needs an effective method to make a UM profile, because only location coordinates are used. From the analysis in Chapter 3, we know location updates could be missing or duplicated. Therefore making an efficient UM profile and UM patterns would definitely improve the TAR and the TDR.

3. It worths finding a new approach to build HMMs using less data compared to the original HMMs framework which takes a significant amount of data to train stable probability matrices.

4. One assumption is that all intrusions that happened in networks follow a uniform distribution such that the probabilities of the intrusion are equal. It would be very interesting to study the case when intrusions have different probabilities. For example, it could be more likely that intrusion happens during the weekdays than on weekends.

Based on results from last two experiments, we know the TDR for both IDSs are 100%, which proves the feasibility of the system that, with different user mobility pattern, IDSs can identify users with different mobility patterns. However, the TAR for IBL and HMMs have difference in some degree; IBL has a better TAR results than HMMs'. But the UM profiles in IDS with IBL framework are built only with location coordinates which are not precise to describe user's mobility. That brings us another possibility if we can combine these two IDSs together, since each of them has its own property. As we mentioned before, IBL can make the identification very fast (time complexity is $O(n)$). On the contrary, in IDS with HMMs framework, the UM profiles are built with not only location coordinates but time stamp, speed, and direction. So, that will be an interesting question to join these two features together to see if we can obtain better performance for the IDS for wireless networks.

Appendix A

Glossary

APRS Automatic Position Reporting System

DWT Discrete Wavelet Transform

GPS Global Positioning System. Devices with a special receiver can determine their geographic location by triangulation with a series of orbiting satellites.

HMMs Hidden Markov Models

IBL Instance Based Learning

IDS Intrusion Detection System

IMEI International Mobile Equipment Identity

IR Infra Red

ISP Internet Service Provider

MAC Media Access Control

QoS Quality of Service

RFF Radio Frequency Fingerprinting

TAR True Acceptance Rate

TDR True Detection Rate

UM User Mobility

WLANs Wireless Local Area Networks

WMANs Wireless Metropolitan Area Networks

WPANs Wireless Personal Area Networks

WWANs Wireless Wide Area Networks

Bibliography

[AFTV94] Debra Anderson, Thane Frivold, Ann Tamaru, and Alfonso Valdes. Next-generation intrusion detection expert system (nides), software users manual, beta-update release. Technical Report SRI–CSL–95–07, Computer Science Laboratory, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025-3493, may 1994.

[aHL03] Yi an Huang and Wenke Lee. A cooperative intrusion detection system for ad hoc networks. In *SASN '03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 135–147, New York, NY, USA, 2003. ACM Press.

[AT00] Inc Axent Technologies, 2000.

[BN01] A. Boukerche and M.S. M. A. Notare. *Applications of neural networks to mobile and wireless networks.* In Biologically Inspired Solutions to Parallel and Distributed Computing, A. Zomaya(Ed.). Wiley, New York, 2001.

[BN02] Azzedine Boukerche and Mirela Sechi M. Annoni Notare. Behavior-based intrusion detection in mobile phone systems. *J. Parallel Distrib. Comput.*, 62(9):1476–1490, 2002.

[Coh95] William W. Cohen. Fast effective rule induction. In Armand Prieditis and Stuart Russell, editors, *Proc. of the 12th International Conference*

on Machine Learning, pages 115–123, Tahoe City, CA, jul 1995. Morgan Kaufmann.

[CPAH95] H. Choe, C.E. Poole, A.M.Yu, and H.H.Szu. Novel identification of intercepted signals from unknown radio transmitters. In Naval Surface Warfare Ctr Harold H. Szu, editor, *SPIE Proceedings*, volume 2491, pages 504–516, 1995.

[CS99] Inc. Cisco Systems. Cisco secure intrusion detection system., 1999.

[FHK+95] Y. Frankel, A. Herzberg, P. A. Karger, H. Krawczyk, C. A. Kunzinger, and M. Yung. Security issues in a cdpd wireless network. *IEEE Personal Communications*, 2(4):16–27, August 1995.

[FHSL96] Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff. A sense of self for Unix processes. In *Proceedinges of the 1996 IEEE Symposium on Research in Security and Privacy*, pages 120–128. IEEE Computer Society Press, 1996.

[Fil04] S. Filjar, R.; Desic. Architecture of the automatic position reporting system (aprs). In *Proceedings Elmar 2004*, pages 331 – 335, June 2004.

[HBK03] J. Hall, M. Barbeau, and E. Kranakis. Detection of transient in radio frequency fingerprinting using phase characteristics of signals. In Lambertus Hesselink, editor, *Proceedings of the 3rd IASTED Inter-national Conference on Wireless and Optical Communications (WOC)*, pages 13–18, Banff, Canada, July 2003. ACTA Press.

[HBK04] J. Hall, M. Barbeau, and E. Kranakis. Enhancing intrusion detection in wireless networks using radio frequency fingerprinting. In M.H. Hamza, editor, *Proceedings of the 3rd IASTED International Conference on Communications, Internet and Information Technology*, pages 201–206, U.S. Virgin Islands, Nov 2004. ACTA Press.

[HP96] Ralph D. Hippenstiel and Yalcin Payal. Wavelet based transmitter identification. In *In International Symposium on Signal Processing and its Applications Proceedings*, pages 740–743, Gold Coast Australia, August 1996.

[ISS99] Inc. Internet Security Systems. Realsecure 3.0., 1999.

[JM96] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[JV91] Javits and Valdes. The SRI IDES statistical anomaly detector. In *In Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, may 1991.

[KS94] Sandeep Kumar and Eugene H. Spafford. A pattern matching model for misuse intrusion detection. In *Proceedings of the 17th National Computer Security Conference*, pages 11–21, 1994.

[LB99] Terran Lane and Carla E. Brodley. Temporal sequence learning and data reduction for anomaly detection. *ACM Trans. Inf. Syst. Secur.*, 2(3):295–331, 1999.

[LJ00] E. Lundin and E. Johnsson. Anomaly-based intrusion detection: privacy concern and other problems. *Computer Networks*, 34:623–640, 2000.

[LS00] Wenke Lee and Salvatore J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Trans. Inf. Syst. Secur.*, 3(4):227–261, 2000.

[LSM99] W. Lee, S. Stolfo, and K. Mok. Mining in a data-flow environment: Experience in network intrusion detection, 1999.

[LTG+92] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. Neumann, H. Javitz, A. Valdes, and T. garvey. A real-time intrusion detection expert system (ides)-final technical report. Technical report, Computer Science Laboratory, SRI international, Menlo Park, CA, February 1992.

[Mal99] Stephane Mallat, editor. *A Wavelet Tour of Signal Processing*, San Diego, USA, 1999. Academic Press.

[MLTS95] J. G. Markoulidakis, G. L. Lyberopoulos, D. F. Tsirkas, and E. D. Sykas. Evaluation of location area planning scenarios in future mobile telecommunication systems. *Wirel. Netw.*, 1(1):17–29, 1995.

[Qui93] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[Rab90] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition*, pages 267–296, 1990.

[ZL00] Yongguang Zhang and Wenke Lee. Intrusion detection in wireless ad-hoc networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 275–283, New York, NY, USA, 2000. ACM Press.