11-20-2008

802.11 Fingerprinting to Detect Wireless Stealth Attacks

Aravind Venkataraman

802.11 FINGERPRINTING TO DETECT WIRELESS STEALTH ATTACKS

by

ARAVIND VENKATARAMAN

Under the Direction of Raheem A. Beyah

ABSTRACT

We propose a simple, passive and deployable approach for fingerprinting traffic on the wired side as a solution for three critical stealth attacks in wireless networks. We focus on extracting traces of the 802.11 medium access control (MAC) protocol from the temporal arrival patterns of incoming traffic streams as seen on the wired side, to identify attacker behavior. Attacks addressed include unauthorized access points, selfish behavior at the MAC layer and MAC layer covert timing channels. We employ the Bayesian binning technique as a means of classifying between delay distributions. The scheme requires no change to the 802.11 nodes or protocol, exhibits minimal computational overhead and offers a single point of discovery. We evaluate our model using experiments and simulations.

INDEX WORDS: 802.11 MAC protocol, Distributed coordination function, Rogue access points, MAC misbehavior, Covert channel.

802.11 FINGERPRINTING TO DETECT WIRELESS STEALTH ATTACKS

by

ARAVIND VENKATARAMAN

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

in the College of Arts and Sciences

Georgia State University

2008

802.11 FINGERPRINTING TO DETECT WIRELESS STEALTH ATTACKS

by

ARAVIND VENKATARAMAN

ACKNOWLEDGEMENTS

I acknowledge the patience and commitment from my advisor, Dr. Raheem A. Beyah. I would like to thank him for his support, in terms of motivation, direction and time in reviewing the work, as well as infrastructure.

I appreciate the help and company of fellow members of Dr. Beyah's research group - the Communications Assurance & Performance group.

I am also grateful to the faculty of the Department of Computer Science for providing an excellent academic and research base, and to Georgia State University for prolonged financial assistance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES

LIST OF FIGURES

1. PROLOGUE

As corporate and university stance increasingly shifts towards enterprise wireless local area networks (WLANs) and the scope of wireless-fidelity (commonly known as Wi-fi) improves in terms of location, speed, range and deployability, malicious interest follows suit. The renewed popularity of WLANs owing to the indispensability of mobile computing, combined with the relative ease of hacking into such networks has attracted security research and start-up attention. However, as is invariably the case with security, defenders have to keep pace with attackers, and hence the state-of the art market-wide wireless intrusion detection systems (WIDSs) often lack the necessary defense against the latest breed of attacks. Moreover, the open environment for communication in WLANs together with the hasty development of architectural standards and independent vendor-specific product implementations lead to challenges previously unseen, and introduce the potential for novel attacks that encompass a diverse range of design flaws.

We address the problem of covert attacks in WLANs the motivation for which is multi-fold. Given the shared unconstrained nature of operation of WLANs, the attacker is able to sneak under the radar undetected while continuing to inflict some form of damage to the network or beyond. The fruits of such a process could include unauthorized access to resources, network service disruption, increased channel access on a common medium, information leakage, among others. Since all of the above directly affect either the confidentiality, authentication or integrity of a system, monitoring schemes that address specific wireless network security issues are required outside of the encryption standards that have been proposed, such as 802.11i and encryption algorithms, such as WEP, WPA, WPA2, etc.

In particular, we seek to arrive at associated solutions for three such covert attacks based on the fundamental nature of operation of the 802.11 MAC protocol. The attacks include unauthorized or rogue access point insertion, MAC layer misbehavior and covert timing channels at the MAC layer. The consequences of the concealed nature of functioning of each of these attacks can be anywhere between minor and serious, because the attackers would be strategically located behind the intrusion detection system, if any.

A key motivation for coming up with a solution that can be easily implemented is that existing WIDSs do not perform well against covert attacks. There are simple ways, that will discussed later, to defeat rogue access point detection schemes widely used today in the industry. Moreover, the current WIDSs do not monitor for protocol manipulation, and hence do not offer sufficient defense against MAC layer misbehavior and MAC layer covert timing channels.

We propose a common architecture that can be implemented on the wired side of the network and can detect each of the attacks, by sampling incoming traffic streams and seeking expected behavioral traits. To this end, a study of the working of the 802.11 MAC protocol is necessary so as to create a foolproof approach that is centered on the inherent mechanism. The system employs its knowledge of the procedure followed by 802.11 compliant terminals in identifying events of legitimate operation as per protocol or lack of the same.

An important attribute of the MAC standard is the collision avoidance functionality by means of which a wireless node may operate in cooperation with its neighbors. Unlike a switched network, the shared medium in a wireless setting poses a restriction on the number of concurrent transmissions. As a consequence, on an average, each node spends a significant portion of time yielding to competing traffic before it can transmit its own. To avoid synchronization of waiting periods, apart from a fixed time interval, the protocol also requires the wireless node to back-off for a random duration. Such delays are however bounded by well-specified upper limits, and along with the physical transmission times, make up the time interval between a pair of successful transmissions. From the specifications, the expected values of a sequence of such inter-packet arrival times can be determined analytically for an average transmission, in the form of a finite random variable. A classifier may be trained on such baseline distributions and made to seek anomalous behavior. Additionally, it may be necessary to create individual attacker profiles depending on the type of attack.

The first attack that we deal with is the insertion of rogue access points. Attacks on wireless networks can be classified into two categories: external wireless and internal wired. In external wireless attacks, an attacker uses a wireless device to target the access point (AP) or other wireless nodes on the network. In internal wired attacks, an attacker or authorized insider inserts an unauthorized (or rogue) AP into the wired backbone for malicious activity or misfeasance. This proposal addresses detecting the internal wired attack of inserting rogue APs in a network by monitoring on the wired side for characteristics of wireless traffic. We focus on two 802.11 MAC layer features as a means of fingerprinting wireless traffic in a wired network. In particular, we study the effect of the Distributed Coordination Function (DCF) and rate adaptation specifications on wireless traffic by observing their influence on packet inter-arrival times (IATs). By focusing on fundamental traits of wireless streams, unlike existing techniques, we demonstrate that it is possible to extract wireless components from a flow without having to train our system with network-specific wired and wireless traces. Our approach is generic as it does not assume that the wired network is inherently faster than wireless network, is effective for networks that do not have sample wireless traffic, is independent of network speed/type, protocol and application, and is immediately deployable. We evaluate our approach using experiments and simulations. Using a Bayesian classifier we show that we can correctly identify wireless traffic on a wired link with 86-90% accuracy.

The second attack that we deal with is MAC layer misbehavior. We propose a simple, deployable scheme for classifying selfish behavior achieved by manipulating the 802.11 MAC protocol. Specifically, attacks that alter the DCF parameters and data rate are addressed by employing a combination of supervised and unsupervised learning techniques that extract differences in the delay patterns of protocol-abiding and illegitimate traffic. We apply an anomaly-based categorization which obviates the need to train on traces from different network instances. Further, we employ a rule-based system that uses analytically created attacker profiles to provide us with granularity about the degree of cheating. Our method requires no change to the 802.11 nodes or protocol, exhibits minimal computational overhead and offers a single point of discovery. Since the approach is holistic and does not rely on a feature selection using individual parameters, the technique is free of adaptive cheating. Additionally, the accuracy of classification is independent of number of terminals in the network, number of colluding attackers, protocol, rate adaptation and higher layer transmission behavior. Accuracy measures are evaluated with the help of simulations and a naïve Bayes classifier.

The third attack is covert timing channels. We propose a wireless network specific covert timing channel at the 802.11 medium access control (MAC) layer by configuring the protocol's delay parameters to values not used under normal behavior. Currently familiar to the research community as a means for selfish behavior and network-wide denial of service, this approach can be leveraged to create a timing channel outside the legitimate frequency bands. This way, our model is based on the previous attack. The ease of manipulation of the 802.11 driver, combined with the broad range of unused parameters gives an increased freedom of choice with the implementation by means of operating at several reduced or larger delays, as well as scope for trade-off between covertness and accuracy of extraction. Patterned to shadow the randomness in the temporal characteristics of wireless traffic, the channel design introduces the use of multi-packet representation per symbol and synchronization-less operation, that create an environment of increased decoding accuracy and immunity to regularity test based detectability respectively. We also show how anomalous components in traffic, exhibited by both the channel and the underlying MAC layer misbehavior can be detected using simple stateless monitoring. Though meant for extracting information from machines compromised on the wireless side, both decoding and detection schemes are such that they may also be run from the wired side, which is important considering that covert channels are likely to steal across the Internet. Apart from information leakage, another timely application for the channel in a wireless setting is stealthy communication within a wireless ad hoc bot-net. Appropriate experiments are performed to validate the concepts behind both the channel and its detection.

Each of the above described attacks and the proposed detection schemes will be presented in detail in terms of analysis, model design, validation and discussion in the following sections - rogue access point detection in Section 2, MAC layer misbehavior in Section 3, and MAC layer covert timing channel in Section 4. Finally, we conclude in Section 5 and discuss our plans for future work in Section 6.

2. ROGUE ACCESS POINT DETECTION

2.1 Introduction

A dangerous insider attack is one where cheaply available APs are illicitly plugged in with the motivation of extending the network. Like other insider attacks, the AP stays invisible to a firewall as it is actually behind it, thus making it difficult to detect. Hence the AP creates a back door for attackers obviating the need to go through the firewall. This proposal presents a practical solution for this attack which can happen in one of the two scenarios shown in figure 1 - wired networks *with* or *without* existing legitimate wireless APs.

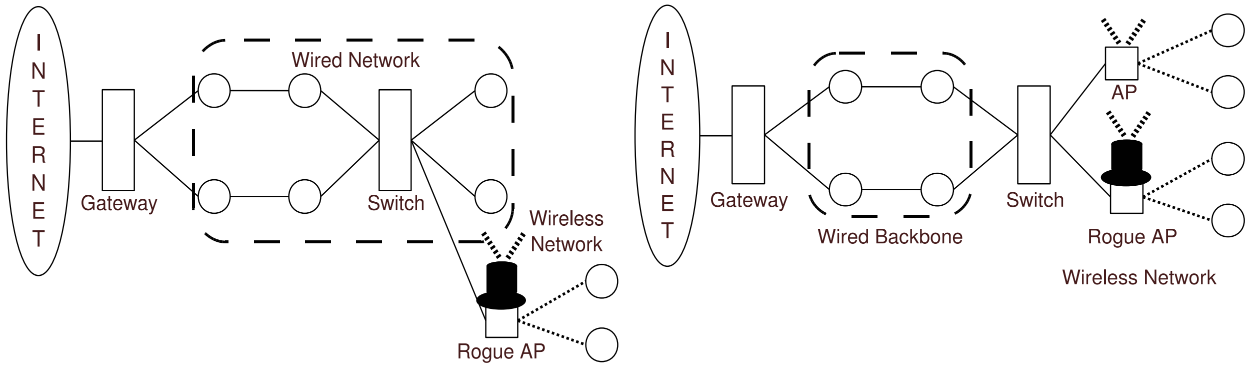

Figure 1. Wired networks (a) with, and (b)without existing legitimate wireless APs.

The core of our detection scheme is an agent sitting atop a switch, or a separate monitoring device that is connected to the mirror port of a switch that passively sniffs passing traffic streams on the wired side. Using inherent differences in wireless characteristics as compared to wired traffic, this agent is able to deem the originating link as being wired or wireless. The idea is that discovery of traffic with wireless characteristics in an otherwise wired network infers the presence of a rogue AP on the network.

While similar designs exist in some current solutions, this proposal is centered on a procedure more fundamental, offering two distinct approaches based on elements native to the 802.11 protocol. Though some of the existing methods work with proven efficacy, they do not try to exploit the underlying facets of the wireless MAC protocol to detect rogue APs, but instead attempt to classify wireless traffic based on the greater delay observed in network statistics (e.g., round-trip-time (RTT), inter-packet arrival time (IAT)). This is based on an assumption that the wireless link capacity will never reach that of wired. A more general solution is needed as this may not always be the case. Also, since many of the previous algorithms need to be trained on both wired and wireless traffic for a given network, they cannot be used in networks without existing APs as there would be no prior wireless trace available.

Our first approach exploits the collision avoidance process of the DCF in the 802.11 MAC. To avoid collisions while transmitting, a wireless node has to sense the channel prior to an attempt at sending. Once the channel is clear, the node will wait for a random time period (chosen from 0 time units to a fixed upper bound) before attempting to transmit. If the node senses that the channel is occupied or in case of a collision, the node has to back-off exponentially before retransmitting (i.e., the fixed upper bound increases exponentially, increasing the probability of choosing a higher back-off value). This procedure, carrier sense multiple access with collision avoidance (CSMA/CA), of the DCF has both fixed components and bounded random components that can be artificially produced and used as a baseline for wireless traffic.

The second approach exploits the process of rate adaptation in the 802.11 MAC. Rate adaptation algorithms allow wireless hosts to alter their encoding scheme (transmission rate) to account for channel interference during transmission. When interference is detected, the node adapts its rate and transmits at a slower rate in an attempt at reducing packet loss. As the rate adjusts (lower or higher), there are noticeable and unique ‘jumps’ in the packet IAT. These ‘jumps’ can be artificially produced and used as a signature for wireless traffic. For both of the above techniques, we show that the signature created stays intact and can be detected on the wired side allowing us to deem specific traffic as originating from a wireless node.

Each of the two approaches work best in specific cases. The first approach works best when there is little interference and the transmission rate essentially stays constant. Intuitively, the second approach works best when the network is more volatile as more ‘jumps’ are produced during that period. Since network stability is unpredictable, we combine the two schemes and present a solution that accounts for realistic, unpredictable network conditions.

To quantify the accuracy of the aforementioned techniques, we use a Bayesian classifier. The goal is to segregate link classes based on the differences observed in the IAT patterns of monitored traffic. To identify these patterns, we compare them to artificially constructed signatures of each network type and rate, allowing us to classify traffic without measured traces of the classes the traffic are expected to be placed into.

Details of the above mentioned design and validation of the model will be subsequently. The remainder of this section is organized as follows. Section 2.2 outlines previous work broadly classifying them into three categories based on the techniques used. In Section 2.3 we briefly illustrate why magnitude-based approaches are not optimal. An introduction to the 802.11 MAC protocol’s DCF and a breakdown of the delay pattern induced by it on wireless traffic is presented in Section 2.4. An analytical representation is derived from the inherent mechanism of the DCF following which we validate the model using a Bayesian classifier. A similar pattern of presentation is taken in Section 2.5 as in Section 2.4 where we perform an analysis of the manner in which rate adaptation occurs, followed by accuracy measures of our model. In Section 2.6, we perform a comparative study of the two techniques in an attempt to come up with a bridged solution. Finally, we present the scalability of our techniques in Section 2.7.

2.2 *Related work*

Current work on rogue AP detection can be classified into three categories. The first two categories contain techniques that use the magnitude of statistics (mean, median, entropy, etc.) of IATs and RTTs as the primary metric for classification respectively. The third category contains industry work that primarily make use of radio frequency scanning to discover wireless activity within a network.

References [1 - 6] fall in the first category. Beyah R., et al [1] were among the earliest to suggest the possibility of using temporal characteristics, such as IATs, for rogue AP detection. They used the IATs of data packets and TCP ACK packets to identify the type of traffic flow. The authors in [2] take a similar approach as that taken in [1] but extend the work by creating an automated classifier. Wei W., et al in [3, 4] present two similar proposals that examine IATs of TCP ACK pairs to identify the type of traffic flow. However, the use of ACK pairs limits this technique to TCP traffic only. A noteworthy effort in the area of traffic classification is [5] which attempts to categorize different types of access links using median and entropy of packet IATs. The approach is however not applicable for detecting rogue APs because it is active (requires probing) and requires cooperation (probe responses) from malicious nodes. In [6], the authors create a spectral profile for WLANs based on the entropy of IATs. They assume link quality and unpredictability of the wireless medium as the cause for greater wireless 'uncertainty' and do not study the effect of the DCF.

In the second category, [7 - 9] make use of RTT as a metric for classification. Since these methods rely on RTT, they cannot accommodate traffic streams other than TCP. Though [7] briefly mentions the effect of the DCF, it does not go into detail to study its mechanics. Reference [8] uses a distinctive approach for segregating network types, complete with traffic conditioning to eliminate noise. However, it demarcates wired and wireless traffic with the help of mean and deviation of the RTT dataset which is not advisable as these parameters differ with varying types, speeds, and congestion levels of networks. Their approach is claimed to be non-intrusive. However, since it involves conditioning of traffic it is still, at minimum, pseudo-active. In [9], although for a disparate motive and in a dissimilar context, Cheng L., et al were among the first to work on identifying wireless traffic for the purpose of access link type recognition. However, their model employs a probing process to gain information about nodes in the network and thus not likely to be of assistance in the "rogue AP" problem space for the same reason that [5], as mentioned above, falls short.

The third category includes several industry implementations [10 - 17], many of which exhibit non-scalability and limited effectiveness because of the use of either radio frequency (RF) scanning and/or MAC address based authentication. The use of RF scanning is not practical as the malicious user can use directional antennas, can adjust the power of the AP as to not be detected, and in large networks it becomes analogous to finding a needle in a haystack. The use of the MAC address as a parameter for authentication is not appropriate because of the ease of spoofing.

Outside of the three categories, [18-20] propose frameworks consolidating the above mentioned wired and wireless side detection models and inherit the flaws from each type.

As previous schemes primarily compare the *relative* behavior of traffic on each link, they require traces of each class of network traffic for their scheme to be effective. This approach is limiting, as a network without existing legitimate APs would not be able to easily provide a wireless trace.   Further, because many use threshold-based separation metrics, another limiting assumption made is that wireless networks will always be slower than their wired counterparts. As will be shown in subsequent sections, our method is free of each of the above mentioned assumptions.

2.3 Problem with magnitude based classification

As mentioned in Section 2.1, many of the existing works focus on the difference, in some form, of the magnitude of the IAT or RTT to differentiate wireless from wired traffic. In this section, we illustrate, via simulation, the challenge with these approaches as wireless speeds begin to approach and exceed that of wired traffic.

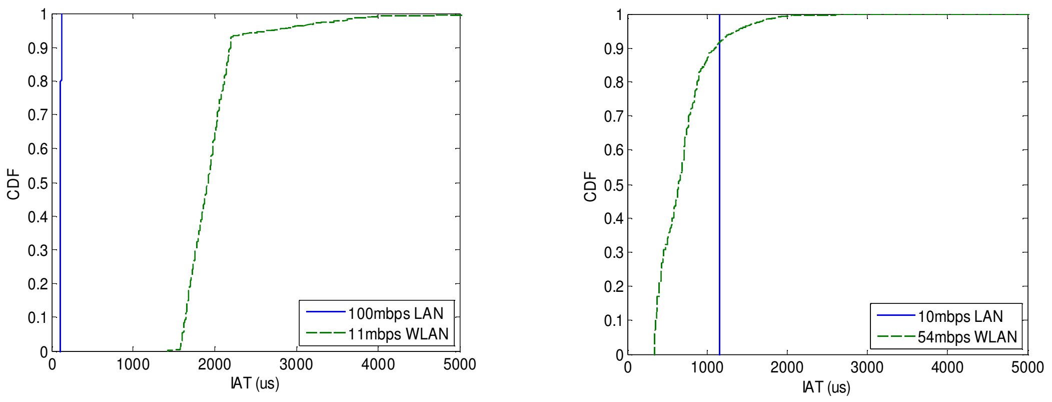

Figure 2. IAT distribution for (a) slower WLAN, (b) faster WLAN.

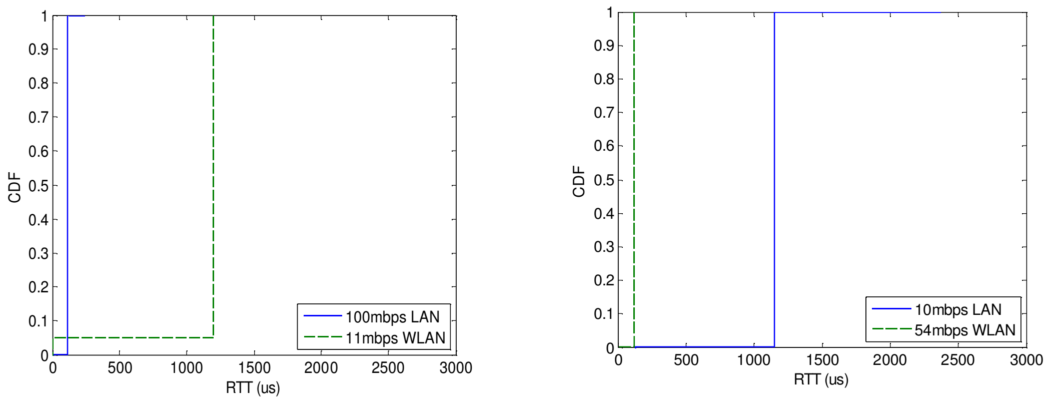

Figure 3. RTT distribution for (a) slower WLAN, (b) faster WLAN.

Simulations were performed using *ns2* [24], and cumulative distribution functions (CDFs) of the IAT and RTT are given in Figures 2 and 3 respectively. Figures 2a and 3a illustrate why the magnitude-based approaches work when the assumption is that WLANs are slower than LANs ($Wireless_{IAT,RTT} > Wired_{IAT,RTT}$). However, as shown in Figures 2b and 3b, these schemes will breakdown if WLAN speed exceeds that of the LAN *($Wireless_{IAT,RTT}$ < $Wired_{IAT,RTT}$)*. It should be noted that Figures 2b and 3b show why a threshold-based approach does not work *in theory*. At first glance it appears that the schemes could merely adjust the threshold to account for the inverse characteristics of the links. However, *in practice*, since the wireless bridge (AP) would be connected to a slower wired network, the traffic would be slowed such that it closely resembles the wired traffic, making it difficult to adjust the threshold to distinguish the two. Partially motivated by this fact, we propose an adaptable solution that makes no assumption about the link speed.

2.4 Scheme I - DCF based classification

A wireless node's packet transmission mechanism is dictated by the specifications of the 802.11 MAC layer protocol, the Distributed Coordination Function (DCF). The DCF employs a carrier sense multiple access with collision avoidance (CSMA/CA) distributed algorithm for collision avoidance. In this method, a node inclined to transmit on a wireless link has to wait for a fixed duration, namely a Distributed Inter Frame Space (DIFS) and a bounded random amount of time (σ) before using the channel. On receiving the data, the node at the other end waits for a fixed period called the Short Inter Frame Space (SIFS) before answering with a MAC-level acknowledgment (MAC-ACK), and the cycle follows thereon. Further, if the channel is sensed busy or if a collision is detected the originating node backs-off before trying again. The bounded random delay (Contention Window) has an exponentially increasing upper bound to reduce the chances of collisions. Accordingly, the DCF has both fixed components and bounded random components that can be artificially produced and used as a signature for wireless traffic. The process employed for transmission in a wireless medium and the delay between packet arrivals (IAT_{wl}) as observed at the receiver end are shown in Figure 4.

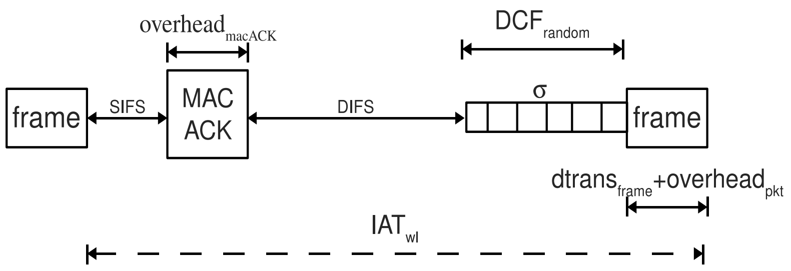

Figure 4. Illustration of the DCF in 802.11 networks

Drawing from the DCF's basic mode of operation, we deduce a pattern unique to wireless streams that allows one to anticipate packet arrivals at known intervals. This property will be derived analytically and later exploited to algorithmically construct profiles for use in a classifier.

2.4.1. *Analysis*

First, in order to demonstrate the effect of the DCF on the delay, we arrive at representations for the IATs of wired and wireless networks (IAT_{wd} and IAT_{wl} respectively). Here *dtrans* and *dprop* are the transmission and propagation delays for a network respectively. Since *dtrans* >> *dprop*, the propagation delay is neglected in our analysis.

$$IAT_{wd} = dtrans_{wd} + dprop_{wd} \tag{1}$$

$$IAT_{wl} = dtrans_{wl} + dprop_{wl} \tag{2}$$

$$dtrans_{wd} = dtrans_{frame} + dtrans_{overhead_{wd}} \tag{3}$$

$$dtrans_{wl} = dtrans_{frame} + dtrans_{overhead_{wl}} + DCF_{constant} + DCF_{random} \tag{4}$$

In Equation 4, $dtrans_{\text{frame}}$ is the transmission time per frame; $dtrans_{overhead}$ is the overhead incurred in transmitting the packet header in the wired case, and packet header and MAC-ACK in the wireless case. Note that $dtrans_{wl}$ is essentially representative of the waiting time incurred because of the DCF, the constituents of which are as follows.

$$DCF_{constant} = DIFS + SIFS \tag{5}$$

$$DCF_{random} = \sigma \tag{6}$$

$$dtrans_{overhead_{wl}} = overhead_{pkt} + overhead_{macACK} \tag{7}$$

The back-off (σ) is the random period a node has to wait in addition to the DIFS, one which is repeated for each unsuccessful transmission attempt. In other words, the back-off for the *i*th retransmission (σ_i) is randomly chosen from within the contention window (CW_i) which is an increasing function of the number of retransmission attempts and the number of times the channel was sensed as busy by the sender. The DCF uses a Binary Exponential Back-off (BEB) algorithm where for each retry the contention window size (starting at a lower bound (CW_{min}) is doubled until a maximum value (CW_{max}) is reached.

$$\sigma_i \in \left(0, CW_i\right) * slot\ time \tag{8}$$

$$CW_i = min\left[2\mathrm{CW}_{i-1}, CW_{max}\right] = min\left[2^i CW_{min}, CW_{max}\right] \tag{9}$$

$$\sigma_i \propto CW_i \propto CW_{min} \tag{10}$$

Hence arrival times can be predicted as a function of Cw_{min}. This is an important result which shows that the DCF provides us with an increasing trend for wireless links, one whose base frequency (σ) is presented below. Since the rogue AP problem space would often involve a single client node (the malicious intruder), we consider the case where there are minimal collisions in the network, and thus assume that σ varies between 0 and Cw_{min}.

$$\sigma = 1/\left(dtrans_{frame} + DCF_{constant} + \left[0, CW_{min}\right]\right) \tag{11}$$

This forms the basis for our scheme. Specifically, we seek to discover a wireless segment by extracting a basic recurring pattern that exists in all wireless streams. Further, a wireless series can be determined analytically which spares us from having to train a classifier with real traces.

Also, from Equation 4, it is important to consider two traffic types - TCP and UDP. Figures 5 and 6 show how the IAT distribution would look for the two different classes.

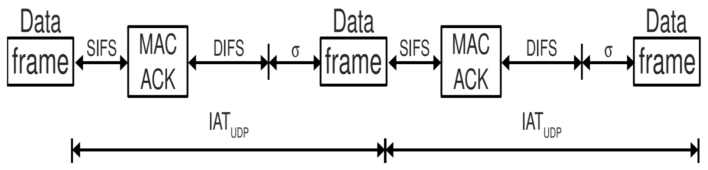

Figure 5. Packet arrival pattern - UDP .

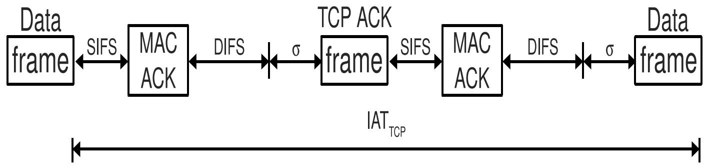

Figure 6. Packet arrival pattern – TCP.

The frame transmission time for each case would differ as shown in Hypothesis I.

TABLE I. Hypothesis I

```
1:  if traffic_UDP then
2:      $dtrans_{frame} = dtrans_{data}$
3:  else if traffic_TCP then
4:      $dtrans_{frame} = dtrans_{data} + dtrans_{tcpACK}$
5:  end if
```

Because of the difference in characteristics, considering an 802.11b network as an example, the transmission delay for the two classes would follow from the information in Table II (taken from [21]) as shown below.

$$\begin{aligned} IAT_{wl_{UDP}} &\simeq dtrans_{wl_{UDP}} \\ &= DCF_{constant} + DCF_{random} + dtrans_{frame} + dtrans_{overhead_{wl}} \\ &= DCF_{constant} + DCF_{random} + dtrans_{data} + dtrans_{overhead_{wl}} \\ &= 60 + \sigma + 1018 + 215 + 10 \\ &= 1303 + \sigma \end{aligned} \quad (12)$$

$$\begin{aligned} IAT_{wl_{TCP}} &\simeq dtrans_{wl_{TCP}} \\ &= 2\mathrm{DCF}_{constant} + 2\mathrm{DCF}_{random} + dtrans_{frame} + 2\mathrm{dtrans}_{overhead_{wl}} \\ &= 2\mathrm{DCF}_{constant} + 2\mathrm{DCF}_{random} + dtrans_{data} + dtrans_{tcpACK} + 2\mathrm{dtrans}_{overhead_{wl}} \\ &= 120 + 2\sigma + 1018 + 30 + 2(215 + 10) \\ &= 1618 + 2\sigma \end{aligned} \quad (13)$$

Figures 7a and 7b display the CDF of the IAT of TCP and UDP flows generated via experimentation, simulation, as well as those constructed synthetically using Equations 12 and 13. The figures illustrate how closely the experimental and simulated values follow the ones synthetically constructed.

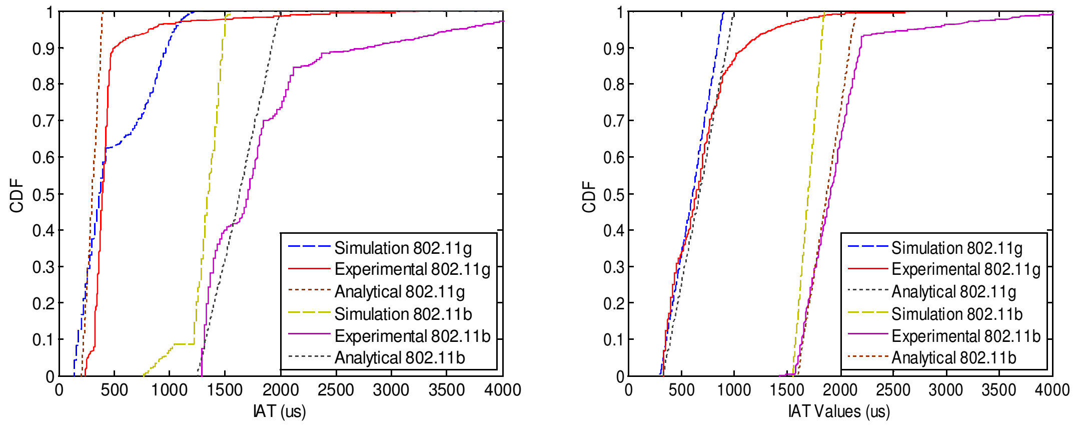

Figure 7. (a) CDF of IAT for UDP. (b) CDF of IAT for TCP.

Note that TCP does not always have to wait for an ACK before transmitting the next packet. Also, when a node is transmitting TCP traffic with a congestion window size greater than one (W>1), it is likely to exhibit UDP-like behavior (multiple sequential packets) except for the time when it is waiting for ACKs. In fact, in the case of traffic going to the Internet, a node is highly likely to transmit in bursts. Thus, TCP's IAT distribution would resemble that of UDP for the most part. Hence, having taken into account the frequency of packet arrivals for both UDP and the extreme-case TCP (W = 1), our model is scalable for all traffic types.

As part of our groundwork, we used the expression from Equation 4 - which repeats with the frequency shown in Equation 11, combined with the expected values for each type of WLAN (for example, the data from Table II was imported for 802.11b WLAN) to synthetically construct a profile set. To this end, we used a pseudo-random number generator to emulate the DCF back-off. Values were generated from within a range equivalent to the initial contention window. Also from Figure 7b, while more than 90% of the sample set follows a uniform random

TABLE II.
802.11b MAC Transmission overhead

Variable	**Parameter**	**Time (*μs)***	**Formula**
$DCF_{constant}$	DIFS	50	2*Slot time + SIFS = 50
	SIFS	10	SIFS
DCF_{random}	Average σ	310	(Number of slots * Slot time)/2 = (31*20)/2 = 310
$dtrans_{frame}$	$dtrans_{data}$	1018	Packet size/data rate = (1400*8)/11 = 1018
	$dtrans_{tcpACK}$	30	TCP ACK/data rate = (40*8)/11 = 30
$dtrans_{overhead}$	$overhead_{pkt}$	215	(Preamble + PLCP hdr.)/data rate + MAC hdr./data rate + MAC CRC bits/data rate = (144 + 48)/1 + (30*8)/11 + (4*8)/11 = 192+21+2 = 215
	$overhead_{macACK}$	10	MAC ACK/data rate = (14*8)/11 = 10

dispersal over the window size, a fraction of the flow tends to deviate out of bounds. We attribute this to the overhead in the network caused by $dprop_{wl}$ and possible limited link-layer retransmissions.

This shows that it is possible to independently conjecture how a wireless stream would behave in different types of networks. A Bayesian classifier was used for classification based on comparison of an incoming stream’s IAT values with the synthetically created IAT profile set.

2.4.2. Classification scheme

We build a Naïve Bayes classifier which bins the IAT datasets (the analytical profiles and experimental/simulation traces used for the purpose of testing the system), calculates for each dataset the number of occurrences in each bin, compares the bin frequencies of each profile with those of the trace and predicts the trace as being akin to the profile whose frequency distribution closest resembles that of the trace. The inputs are binned into ‘*n*’ number of bins, where *n* depends on the bin width and input data size. For both the bin width and input data size, different values are tried with the goal of optimizing *n* to furnish maximum accuracy.

Profiles f_i are compared with an unknown sample f_x based on frequency of occurrences in each bin.

Because the nature of incoming traffic cannot be predicted, prior probability is unknown and is assumed equally distributed over the n profiles.

$$PriorProbability \;\; P\left(f_i\right) = 1/n \tag{14}$$

$$Likelihood \;\; P\langle f_x | f_i \rangle \tag{15}$$

$$PosteriorProbability \;\; P\langle f_i | f_x \rangle = \mathrm{P}\langle f_x | f_i \rangle . P\left(f_i\right) \tag{16}$$

Since f_x is a random variable $\{x_1, x_2, \ldots x_d\}$,

$$P\langle f_i | f_x \rangle = P\langle \left(x_1, x_2, \ldots x_d\right) | f_i \rangle . P\left(f_i\right) \tag{17}$$

$$P\langle f_i | f_x \rangle = P\left(f_i\right) . \prod_{k=1}^{d} P\langle x_k | f_i \rangle \tag{18}$$

Likelihood (measure of how similar the unknown trace is to a given profile) is calculated for each profile using a two-sample Chi-square test which is run independently on all sample-profile bin frequency pairs. Posterior probability (measure of how likely a profile is the closest match for the unknown) is derived by aggregating the Likelihood measures (Chi-square values) each of which is calculated as shown below.

$$\chi^2 = \sum_{i=1}^{k} \frac{\left(S_{1_i} - S_{2_i}\right)^2}{S_1 + S_2} \tag{19}$$

S_1 and S_2 are bin frequencies of the two samples to be compared. They represent an unknown and a profile sample. *k* is the number of bins.

2.4.3. Experimental setup

An experimental testbed was built using three Lenovo laptops, three Dell desktops, a Netgear 10/100 Mbps Fast Ethernet switch and a Linksys 2.4Ghz wireless-b/g AP. To ensure that our technique for wired side detection is viable, we first determined whether the temporal characteristics of the IAT observed on the wireless link were in tact on the wired side. The arrival times on the wired side were recorded at the receiver node. On the wireless side, a laptop acting as sniffer was used in promiscuous mode to capture traffic from the wireless sender. We observed that the arrival rates were retained albeit with a uniformly witnessed lag (as a result of router queuing, etc.) as shown in Figure 8.

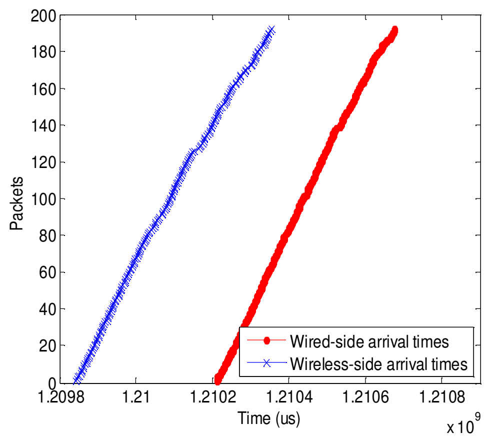

Figure 8. Packet arrival times on wired and wireless sides.

Next, for the purpose of testing the classifier's False Positive Rate (FPR), it was trained on traces from a simple wired connection between two computers. The analytically created 'wireless' profiles were used to train the classifier to test its True Positive (TPR).

The classifier was then tested on traces from both wired and wireless TCP/UDP data transfers. Experiments were performed on the WLAN for both 802.11b and 802.11g specifications by configuring the AP to operate in the required mode. For each network type and protocol, 50 sets of 1000 data packets each were fed into the classifier. The detections from the 50 trials were used in determining TPR/FPR measures for the classifier.

In a rogue AP attack, the attacker often hops on the connection for short bursts of time to avoid detection. Given the attacker's short-lived stay online, it is important that the classifier be able to work on minimum data. Accordingly, we tested our classifier with different input sizes and observed that it works with optimum accuracy for a minimal input trace of 1000 data packets (Figure 9).

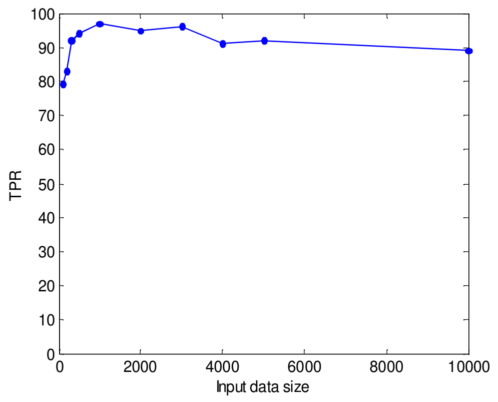

Figure 9. Input data size tuning.

2.4.4. *Accuracy measures*

As a preliminary measure towards testing the precision of detection, the width of the bins used in the Bayesian approach was tuned in an effort to determine the optimal width - one that yields peak accuracy. In Figure 10, note that with an increase in bin width the accuracy drops, which makes sense as the classifier works better with a higher number of bins.

The optimal bin width of 30μs was chosen, as it gives the minimum FPR of 0.1 and maximum TPR of 96. On testing the system with the chosen parameters for a total of 12 additional trials, it was observed that the technique is accurate in detection approximately 98% of the time, as can be seen from Figure 11.

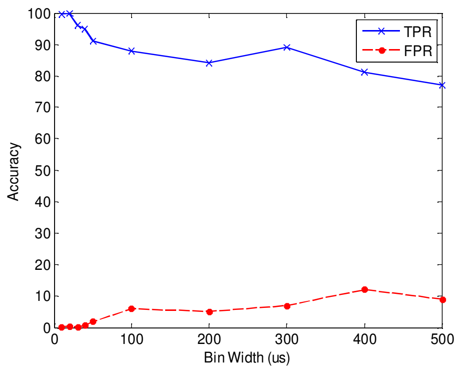

Figure 10. Bin width tuning.

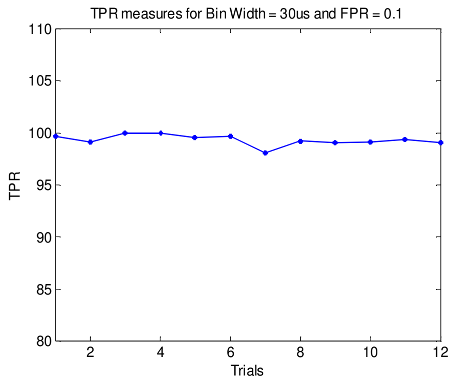

Figure 11. TPR for chosen bin width and FPR.

However, this scheme is optimal when there is no interference on the channel and the link is stable. As will be shown in Section 2.6, its performance degrades as rate adaptation occurs in response to poor link quality. Therefore, in the next section we present a scheme that thrives during rate adaptation.

2.5 Scheme II – Rate adaptation based classification

The 802.11 MAC protocol provides wireless entities with the ability to change their encoding scheme (data transmission rate) when the need arises. Using automatic rate fallback (ARF), when a node reaches a threshold of not receiving MAC-layer ACKs, it reduces its rate to one that corresponds to a stronger encoding algorithm in order to ensure more robust transmission.

The switching of the data rate creates a variation in throughput in a wireless transmission that is rarely found in wired. We exploit the unique behavioral characteristics at the time of a rate switch to identify wireless traffic.

2.5.1. Analysis

As shown in [21], rate switching occurs regularly in wireless networks because signal and link-layer interference are common phenomena. Given that rate switching is common, we seek to exploit this property, unique to wireless streams, to distinguish them from their wired counterparts. Figure 12 is an example of the expected packet arrival sequence for a given wireless transfer. Note that the IATs of packet pairs vary for each rate as slower rates trigger greater delays.

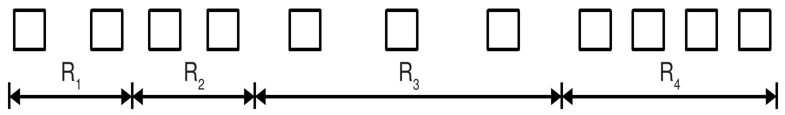

Figure 12. Packet arrivals during rate adaptation.

The probability of the event R_i occurring (P_i) depends on what we call the Signal Interference index (SI) which has a range $\{0 \leftrightarrow 1\}$.

$$IAT_{wl} = \sum_i IAT_i P_i \tag{20}$$

$$\begin{aligned} &If\,(i<k \wedge SI \rightarrow 0)\, P_i \leqslant P_k \\ &If\,(i<k \wedge SI \rightarrow 1)\, P_i \geqslant P_k \end{aligned} \tag{21}$$

In other words, the probability of occurrence of a lower rate is inversely proportional to signal interference and collisions. Our model safely assumes that the measure of interference is not known prior, hence Pi is unknown. This being the case, unlike Scheme I which assumed minimal rate adaptation, we choose to look not at sets of IAT_i (IAT between two packets transferred at same rate) but at IAT_j (IAT between two packets transferred at different rates). As shown in Figure 13, IAT_j is the delay during the 'jump' from one rate to the next.

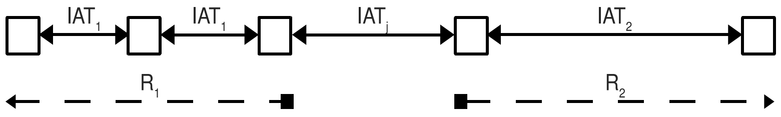

Figure 13. IAT pattern during a rate switch.

Accordingly, in our classifier, we associate IAT_j with IAT_{wl}. To determine which link type the test data (IAT_x) belongs to, we use the basic premise given in Hypothesis II. An initial set of experiments were performed on an 802.11b wireless network and IATs for packet pairs transmitted at the same rate and different rates were extracted. Investigating the behavior exhibited during the jump, it can be seen in Figure 14 that the IAT in this stage falls in between those of the stable rate phases before and after. This leaves us with Hypothesis III.

The rationale behind this (as shown in Figure 15) is that during the transition from R_1 to R_2, the MAC-level ACK is transmitted at R_1 and the subsequent frame at R_2. In other words, a node which decides to reduce its data rate transmits at the new rate but the MAC ACK for the previous data packet would still be sent from the AP at the old rate. Also, as can be seen from Figure 14,

TABLE III.
Hypothesis II

```
1:  if IATx ≅ IATj then
2:      Report wireless
3:  else
4:      Report wired
5:  end if
```

TABLE IV.
Hypothesis III

```
1:  if $R_i < R_{i+1}$ then
2:      $IAT_i > IAT_j > IAT_{i+1}$
3:  else
4:      $IAT_i < IAT_j < IAT_{i+1}$
5:  end if
```

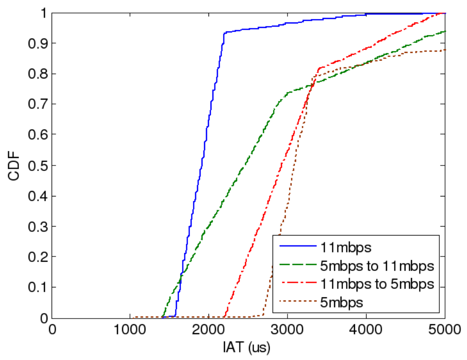

Figure 14. IAT behavior during a rate switch – TCP.

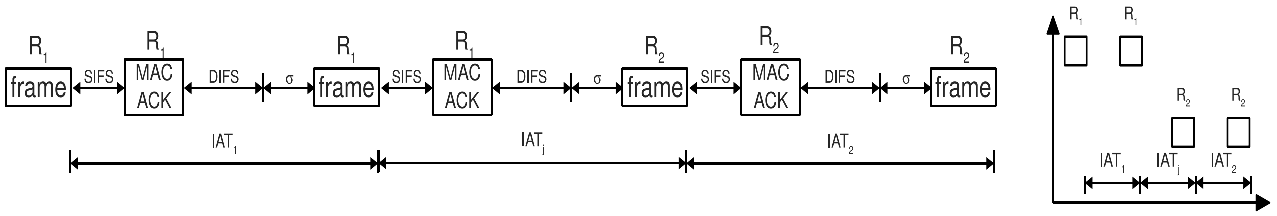

Figure 15. DCF behavior during a rate switch.

because of the difference in frame and MAC ACK sizes, the IAT during the jump (IAT_j) is biased towards that corresponding to the rate following the jump. That is, since the frame size >> MAC ACK size and the data frame is sent at the new rate, IAT_j is closer to the IAT associated with the new rate.

This difference in behavior during a rate switch can be exploited by studying how it reflects on the corresponding IATs. Of particular interest is $dtrans_{overhead}$ which looks different during a *jump* as shown below.

$$dtrans_{overhead_{(1,2)}} = overhead_{macACK_{(1,2)}} + overhead_{pkt_{(1,2)}} \tag{22}$$

$$dtrans_{overhead_j} = overhead_{macACK_1} + overhead_{pkt_2} \tag{23}$$

Similarly, $dtrans_{frame}$ (packet size/data rate) would be 207µs (1400*8/54) during a jump from 36mbps to 54mbps and 311µs (1400*8/36) for a jump from 54mbps to 36mbps.

Using Equation 4 as the base for our synthetic profiles again, substituting jump-specific $dtrans_{frame}$ and $dtrans_{overhead}$ values, our classifier can be trained as shown in Figure 16.

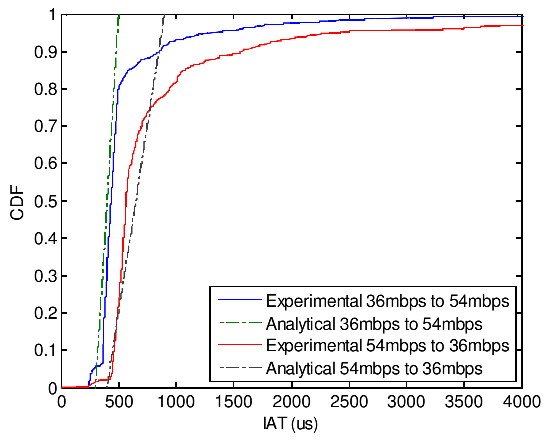

Figure 16. TCP Analytical vs. Experimental Signatures - 802.11g WLAN.

2.5.2. Classification scheme

The classifier used for this method is similar to the one employed for the previous scheme with appropriate changes to incorporate the fact that only the IAT values during ‘jumps’ in rates are considered as opposed to the values during a stable period. Hence though a Bayesian classifier is used, instead of block comparison of a trace of IAT readings with the profiles, individual values are inspected for possible ‘jumps’. Since a comparison of two datasets is not required, it is sufficient to check individual values to see which IAT jump signature it is closest to.

2.5.3. Experimental setup

To determine whether a node is switching rates when capturing packets on the wireless side is simple, as its physical layer header contains the actual transmission rate. However, the rate in the wireless frame is not carried over to the wired side. Accordingly, on the wired side, we have to infer the rate by observing the packets’ IATs. We verified that this approach is viable by capturing traffic on the wireless side and on the wired side. We observed packets that switch rates on the wireless side with a laptop acting as a wireless sniffer capturing promiscuously (by looking at the *radiotap* header in the wireless frame) and captured the IATs of the same packets on the wired side. From this we were able to determine that specific IAT values on the wired side correlated to confirmed rate adaptation on the wireless side.

Figure 17 gives a representative sample of the rates of the packets extracted on the wireless side and inferred on the wired side, illustrating the correlation of rates of the same packets observed at both points. A total of 6000 packets were transmitted with 81% of the rates seen on the wireless side accurately inferred by using the IAT on the wired side. This indicates the viability of inferring rates of wireless packets on the wired side.

It is important to note that though the accurate classification on the wired side of the rates of the packets was only 81% (Figure 17), we are still able to obtain a TPR of 97 % (Figure 20). This is because even the incorrectly inferred rates are closer to the synthetic 'jumps' that the classifier was trained on as opposed to the wired IAT values.

The experimental setup is similar to that used for Scheme I. As in [20], we use a synthetic means (microwave interference) to force rate switching to investigate Scheme II.

Ten trials were performed, in each of which the classifier was tested on 1000 TCP/UDP data packet pairs. TPR/FPR were generated from the share of the 1000 pairs accurately classified each time.

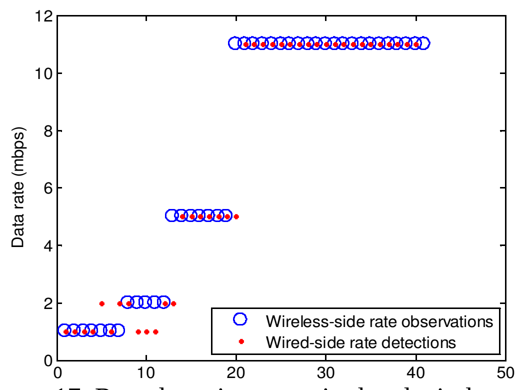

Figure 17. Rate detections on wired and wireless sides.

2.5.4. *Accuracy measures*

As in Scheme I, to validate the system, the bin width used in the classifier was tuned as before to first determine an optimum value. Though as anticipated, an increase in bin width caused a decrease in the TPR, a corresponding decrease was observed in the FPR. In order to deal with this inconsistency, we calculate what we call the Effective Accuracy (TPR-FPR) and find the optimum value that maximizes the difference between TPR and FPR in an attempt to make a balanced trade-off between the two metrics. For the chosen bin width (20 μs), 12 additional trials are run over the network to observe the accuracy distribution. The results are shown in Figures 18-20.

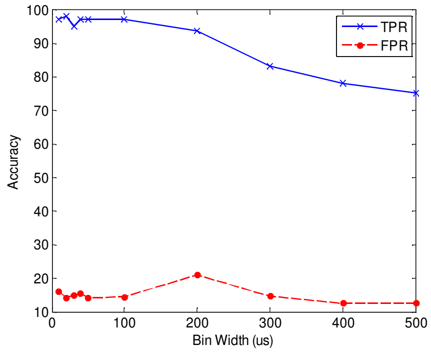

Figure 18. Bin width tuning.

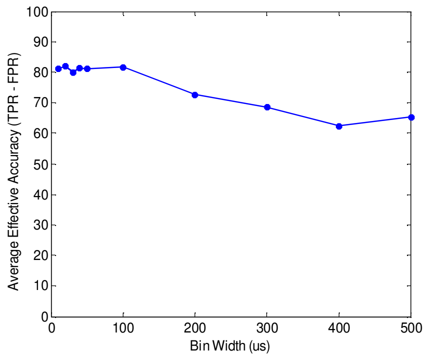

Figure 19. Effective Accuracy.

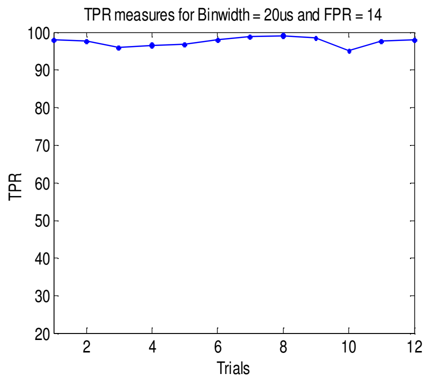

Figure 20. TPR for chosen bin width and FPR.

2.6 *Consolidated model*

While Scheme I compares samples as a whole with the profiles, Scheme II checks individual IATs of each packet pair for a switch in data rate. This implies that since the sample trace compared may encompass several rates, Scheme I's accuracy (TPR) is likely to subside with increased rate adaptation as shown in Figure 23.

2.6.1. *Analysis*

In an effort to present a general solution that works when the link is stable as well as when rate adaptation is occurring, we consider the Signal Interference index (SI) defined as:

$$SI \propto \frac{Accuracy_{Scheme\ II}}{Accuracy_{Scheme\ I}} \tag{24}$$

This essentially captures the inverse relationship between Schemes I and II. Scheme I works better when there is no interference, while Scheme II works better during interference. Specifically, Scheme I's accuracy (TPR) decays with increased signal interference while Scheme II has a high FPR during less rate adaptation. Thus, it is important to consolidate the benefits of the two approaches in a way that the resulting system is effective regardless of the stability of the link.

2.6.2. Classification scheme

To work around these problems and in an attempt at combining the two schemes, we partition the data set into blocks of a basic unit with the expectation that each block will be comprised of data at a specific rate. Of course this need not be the case. So, in addition to this, to bridge the two methods, we exploit the fact that Scheme I detects stable rate periods better and Scheme II detects the jumps. For the combined solution, Scheme I contributes the network type/speed observation for the partitions and Scheme II tells where two stable rate periods intersect (jumps), the aggregation of which gives us the temporal distribution of rates for a series of packet pairs (Figure 21).

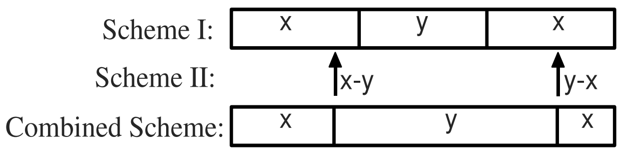

Figure 21. Depiction of combined scheme.

2.6.3. Experimental setup

The experimental setup used for the first two schemes were employed. Similar to Scheme

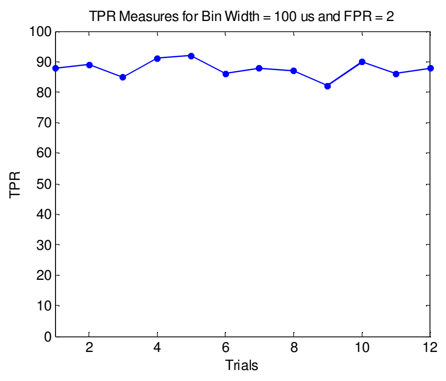

Figure 22. TPR for chosen bin width and FPR.

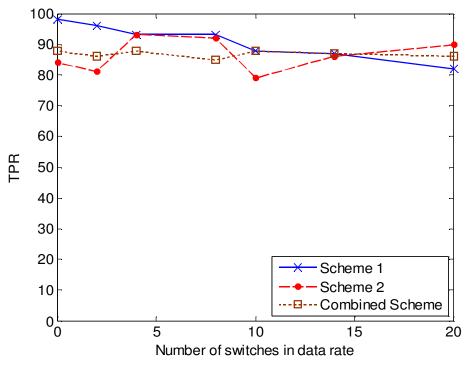

Figure 23. Scheme accuracy comparison.

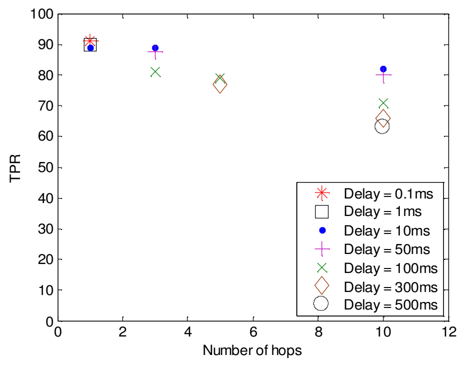

Figure 24. Multi-hop accuracy.

I, 50 trials of 1000 packets each were fed into the classifier which performed the aggregation.

2.6.4. *Accuracy measures*

The accuracy measures of the consolidated system are shown in Figures 22 and 23. Note that the accuracy of the combined scheme is not as high as Scheme I. However, this technique is still effective, and unlike Scheme I and Scheme II, the combined technique is more realistic as it makes no assumption about the link quality.

2.7 *Scalability study*

Finally to test the combined system's scalability, simulations were performed where detection would be performed several hops downstream as opposed to just the switch immediately after the AP. Ideally the system should sit on the gateway since it is the last hop before the Internet. We consider the effect of different access-link and bottleneck delays including the best-case (1ms, 10ms, and 50ms) as well worst-case (300ms and 500ms). The measures we observed (Figure 24), indicate that despite the decrease in accuracy with multiple hops, even in the worst case the system averages above 60% accuracy.

3. MAC MISBEHAVIOR DETECTION

3.1 Introduction

As a considerable portion of 802.11 wireless driver functionality shifts to software with the goal of increased customization, it becomes easier to cheat at the MAC layer by breaking the inherent behavioral fairness. Noted ways of performing this include tweaking DCF parameters (contention window, slot time, SIFS, NAV, CTS/RTS thresholds), scrambling frames and intentionally impinging on external CTS/RTS frames.

We do not consider attacks that target specific frames. We present an abstract methodology for detecting DCF parameter manipulation as a whole. We also introduce the possibility of MAC layer cheating by switching off rate adaptation and scale our model to include this type of cheating.

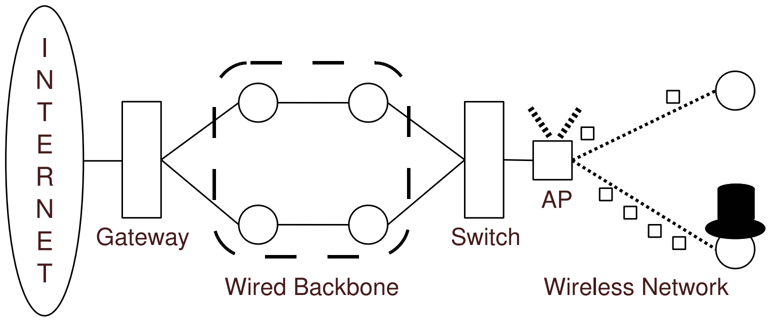

Figure 25. MAC layer misbehavior scenario.

The most familiar motive for cheating at any layer is bandwidth gain when sharing a medium (for example, access to the Internet) with others. Hence, we address the problem in an infrastructure wireless setting as opposed to an ad hoc network, strategically performing the detection on the wired side. Like the previously proposed Rogue AP detection solution, the core of our detection scheme is an agent sitting atop a switch, or a separate monitoring device that is connected to the mirror port of a switch, that passively samples passing traffic streams on the wired side and observes the influence of misbehavior on IATs.

Our model is primarily based on an anomaly-based classifier that monitors for exceptions from the normal delay of nodes that do not cheat on DCF. The classifier makes no assumption on the distribution of the normal values and is not trained on a predefined behavior. Instead it works by seeking a deviation in the closeness value of incoming IAT sequences. It runs a Bayesian test at runtime to do so. As a second line of defense, analytical profiles that represent attacker delay distributions help refine the detection by providing the degree of misbehavior. This helps administrators to decide on an appropriate punishment scheme.

This proposal is centered on a simple solution that does not require procedurally intensive functionality to be implemented on wireless terminals/access points or modifications to the 802.11 standard. As all wireless traffic through the base station passes (or can be routed) into the wired backbone where detection takes place, the scheme is centralized and is not affected by issues that accompany wireless-side detection, such as, interference, collisions, visibility and scalability. Since the classifier performs a relative, basis-less comparison of legitimate and misbehavior traffic, attackers cannot make subtle adaptations to their routine and sneak under the radar. Also, it is not limited to 'available' signatures and can detect patterns that may be missed by analytical feature set generators. In other words, a set of inputs need not entirely represent all types of behavioral traces. Depending on the number of nodes, collision probability, protocol, rate adaptation and other similar influential factors, there may be more that a purely supervised classifier does not account for. We also take into account the scenario of colluding attackers, where a group of malicious individuals or single-user controlled bots could target a well-behaved network in an attempt to cause a network-wide denial of service.

The remainder of this section is organized as follows. Section 3.2 outlines previous work broadly classifying them into three categories based on techniques used. Section 3.3 provides an analysis on the effect of cheating at the MAC layer. The proposed model is discussed in Section

3.4. Our simulation setup is explained in Section 3.5. We discuss some of the scheme's scalability in Section 3.6. Accuracy evaluations are given in Section 3.7.

3.2 Related work

Current work on MAC misbehavior can be broadly classified into three categories. The first category consists of approaches that analytically reproduce “random” back-off in an attempt to emulate the idle time between legitimate transmissions. They try to extract a deterministic behavior model from a stochastic system in order to recreate expected base-line profiles. The second category assumes that fabricating back-off values is not scalable and proposes changes that incorporate detection in nodes. The third category focuses on the effect of misbehaving senders in the absence of an arbiter in ad hoc networks.

References [26 - 27] fall in the first category. In [26], the proposed method requires nodes to monitor the idle time between an RTS and the subsequent CTS from and to their immediate neighbors. Based on collision probability (p_c), nodes analytically construct profiles for legitimate terminals' distributions, to be compared against unknown traffic. Nodes calculate p_c from the frequency of collisions, as observed in their vicinity. The work introduces the *collision factor* - average number of collisions in a network with *n* nodes. The method in [27] assumes that the network operates at saturation point. It uses a Markov chain based model to determine the IAT distribution, complete with consideration for p_c. It introduces the *greedy factor* - an indication of how often a node is expected to transmit. However [26, 27] do not account for the fact that apart from being a function of the system state (number of terminals at a given instance), p_c is also a function of frequency and duration of transmissions, which depend on higher layers. Further, [27] assumes that nodes transmit at a constant rate and does not take into account the influence of rate adaptation on legitimate traffic while emulating it.

In the second category, [28-30] suggest changes to be made to either the driver or protocol in order to include detection into the 802.11 wireless architecture. The authors in [28] introduce the Predictive Random Back-off algorithm which involves tuning the Binary Exponential Back-off algorithm to generate a reproducible back-off that can be monitored for. In [29], the receiver assigns back-off values to sender. The receiver assigns an initial back-off following which the sender calculates a new back-off as a function of the assigned back-off and number of retransmissions. On receiving data from the sender, the receiver calculates the new back-off based on the number of retransmissions to check for misbehavior. This active method requires changes to be made to the driver, and induces computational overhead as well as redundancy on the receiver and sender sides in calculating the new back-off. In [30], detection is done at the access point and involves a series of tests to check for misbehavior on different levels. However, it uses the magnitude of statistics, such as mean of back-off, as the primary metric for classification. This is not advisable as the attacker may adjust the back-off sequence in a way that the effective mean equals the expected. Also, it supervises the number of idle slots, which means that if the attacker cheats on slot time and not on contention window, he would not be detected because the mean number of idle slots stays constant.

The third category [31-36] includes studies that analyze the consequence of misbehavior in ad hoc networks. References [31-33] work on similar lines where individual nodes monitor their neighbors' back-off. The model in [31] assumes that the monitoring node follows the same back-off sequence as it's neighbors and anticipates synonymous behavior from them. In [32], the sender and receiver strike a mutual agreement on the expected back-off values at the beginning. This model works on the logic that as long as one of the two nodes is honest, the system is free of cheaters. In [33], *tagged* nodes announce the state of their pseudo-random generator (PRNG) based on which monitoring neighbors determine the expected sequence. Since the MAC address is used as seed for the PRNG, MAC spoofing can result in others getting caught while the attacker remains undetected. The above solutions do not account for interference and hidden terminal problems. Game theoretic ideas are utilized in [34,35] to conceptualize the working of a setting of colluding attackers. They conceive a Nash equilibrium and seek to determine the existence of a Pareto optimal point of functioning baring which the network would collapse under the pressure of the combined greedy operation. A new type of misbehavior is discussed in [36], where a receiver forces a timeout at the sender by choosing a large SIFS, thereby causing the corresponding frames to be dropped. Since they involve different scenarios and misbehavior models, [33-36] fall out of the scope of this work.

Our model is free of the above mentioned problems as it performs a proportion study of unidentified traces as opposed to a comparison with analytically constructed profiles. It is independent of rate adaptation and higher layer behavior unlike the first category, does not require changes to the protocol/driver unlike the second category, and does not seek to address misbehavior in ad hoc networks as the related work in the third category does.

3.3 Analysis

This section illustrates the working of the three kinds of misbehavior addressed in this proposal, highlighting the consequence on legitimate terminals.

Figure 26 outlines the effect of cheating on DCF parameters has on the fair working of CSMA.

Cheating on DIFS reduces the minimum delay incurred by a wireless node (Figures 26b and 27a). Cheating on contention window (CW) shrinks the random period that follows DIFS (Figures 26b and 27b). Cheating on rate adaptation involves transmitting at a constant data rate

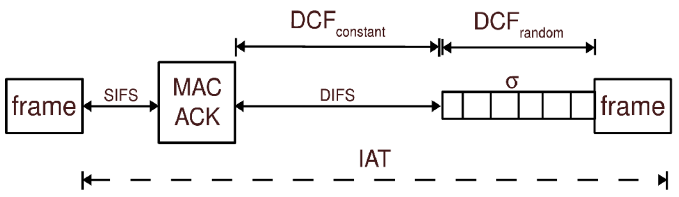

Figure 26. (a) DCF working - legitimate.

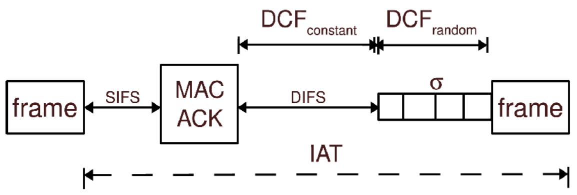

Figure 26. (b) DCF working - attacker.

by switching off Auto Rate Fall-back (ARF), while the legitimate nodes adapt to lower data rates to handle signal interference (Figure 27c). Note that the constant bit rate referred to here is the link-layer rate and is different from the application layer terminology used in the context of UDP. Given the sub-optimal operation of ARF [37], it is tempting to switch off rate adaptation to transmit faster.

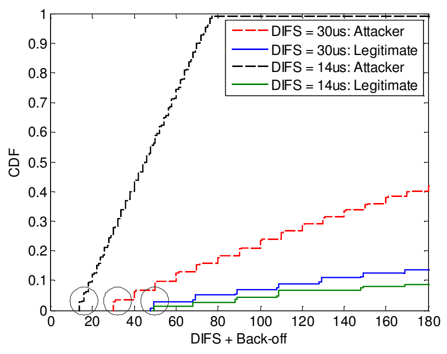

Figure 27. (a) DIFS cheating.

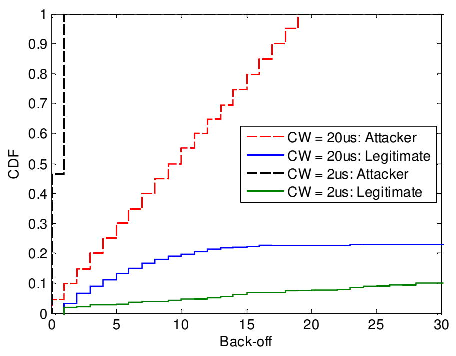

Figure 27. (b) CW cheating.

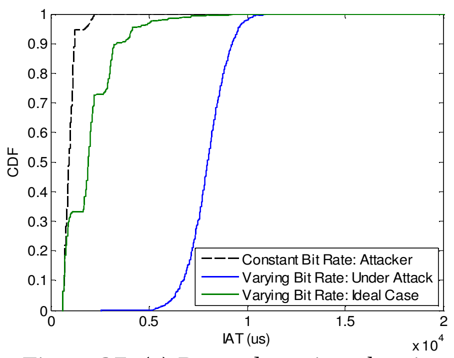

Figure 27. (c) Rate adaptation cheating.

3.4 *Classification scheme*

The working of our model is two-fold. First, we identify the cheaters in the group. Next, we obtain more detail about the stream in order to be able to better deal with the cheater.

3.4.1 Misbehavior detection

While some related work perform a signature-based categorization, our classifier compares unknown sequences with each other to find the c*loseness 'c'*. Each subsequent trace is examined for likeness to the first incoming trace. A hypothesis test is performed over a threshold *(c_{thresh})* that is initially fed to the classifier, but is dynamically updated as the classifier learns. The motivation behind using an anomaly-based detection scheme is that during a given time window, all nodes in a network are expected to converge to a similar profile.

3.4.2 Degree of misbehavior

Following the detection of an attacker, to extract more details from a misbehaving distribution, we need baselines to compare with. To this end, while some existing work create profiles for legitimate transmissions, we emulate attacker behavior analytically.

A preliminary screening is performed to check for DIFS cheating. This is done by extracting from the IAT the portion of the delay contributed by DCF. It is checked to see if the minimum value of the extracted sequence equals the default DIFS. Degree of misbehavior *m* is determined as a function of the difference between observed and expected DIFS.

Next, to check for CW cheating, we derive analytical back-off representations for various attacker choices of CW.

We consider three classes of misbehavior, as shown in Hypothesis IV.

Table V. Hypothesis IV

1: **Class A:**	$(Cwmin, Cwmax) \in \{(2,2),(5,5)\}$ $DIFS = \{1-10\}$
2: **Class B:**	$(Cwmin, Cwmax) \in \{(7,7),(10,10)\}$ $DIFS = \{11-25\}$
3: **Class C:**	$(Cwmin, Cwmax) \in \{(15,15),(20,20)\}$ $DIFS = \{26-40\}$

Note that the default values for (CW_{min},,CW_{max}) and DIFS in the 802.11b standard are

(32,1024) and 50 μs respectively.

The attacker profiles are not created beforehand. They are created at runtime as a function of $dtrans_{frame}$ and $dtrans_{overhead}$ (refer Section 2.4.1). The above parameters are calculated from packet size and transmission rate. This being the case, it is important to account for different transmission rates as a result of rate adaptation.

Given that packet data rates are not available on the wired side where detection is performed, we employ the mechanism we introduced in Section 2.5 to infer the rates of a sequence of packets.

Also, unlike legitimate profiles, we do not need to consider p_c for misbehavior profiles because attackers avoid exponential back-off to steal more (that is, $CW_{min} = CW_{max}$).

Note that there is no degree of misbehavior pertaining to cheating on rate adaptation because an attacker would and hence neglected.

3.4.3 Bayesian classifier

We employ the Bayesian classifier for classification - one with a similar template as that introduced in Section 2.4.2.

In the case of misbehavior detection, it compares the bin frequencies of the first trace with those of every other trace to calculate *c*. This is done using a two-sample Chi-square test.

In the case of misbehavior degree prediction, it compares the bin frequencies of each attacker profile with those of the unknown trace and predicts the trace as being akin to the profile whose frequency distribution closest resembles that of the trace.

3.5 Simulation setup

A generic simulation template of up to 50 nodes was created in *ns2*, each of which could choose from different traffic models, protocols, WLAN speeds, locations, mobility patterns, packet sizes, etc. Of course, MAC parameters would be altered by selfish nodes. Though simulations were performed for a longer duration, individual 10 second time windows were monitored for anomalies.

3.6 *Scalability study*

In the simulation setup, several parameters were varied to study their effect and to check the robustness of our concept. Parameters changed include number of nodes, data rate, protocol, location, mobility, traffic model, etc.

The model's foundation is such that since the attacker steals irrespective of the number of nodes at any given instance of time, we should be able to see the difference in distributions (Figure 28).

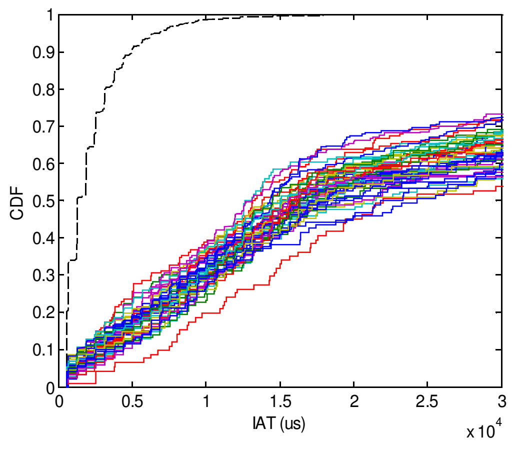

Figure 28. Larger network.

We simulate a group of attackers to check the robustness of our scheme. As seen in Figure 29, when the attackers perform the same changes to the protocol, the model should see a cluster of good and bad nodes.

Similarly, the gap between attacker and other delays persists independent of the transport protocol, as shown in Figure 30.

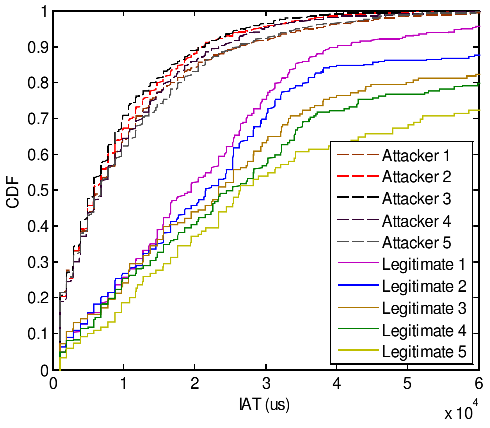

Figure 29. Colluding attackers scenario.

Further, Figure 30 and 31 show how flows that follow different traffic models compare. Figure 30 shows saturated constant rate transmissions. Figure 31 shows exponential flows with varying frequency of transmissions (that is, with on/off periods). This is an important result as it shows that our model supports varying higher layer behavior. It is important to note that the *closeness* between TCP attacker and UDP legitimate traffic is minimal, but that becomes a trivial concern if traces from a traffic class are compared with those from the same class. It is possible to extract the transport protocol from the IP header of incoming traffic flows even if the data is encrypted because the detection is performed on the wired side.

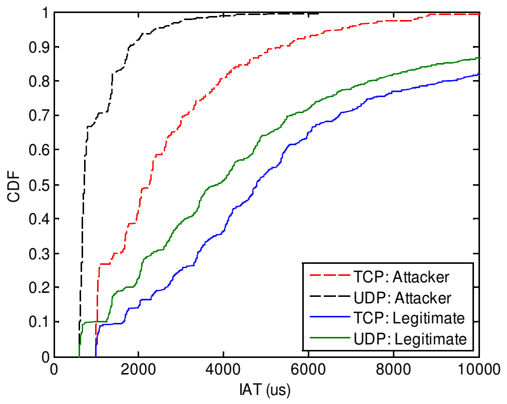

Figure 30. UDP vs TCP - Saturated.

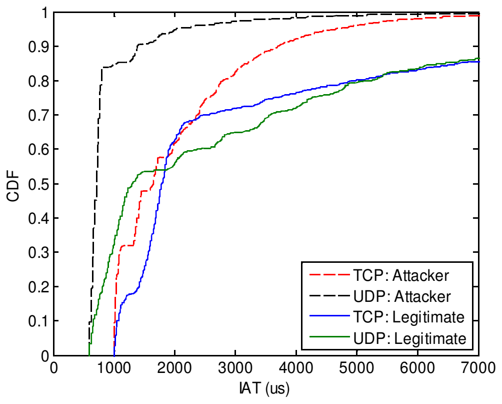

Figure 31. UDP vs. TCP - Bursty.

Note that the scenarios outlined in this section were used in measuring the system's accuracy, the results of which will be presented in the next section.

Also, as one of the scenarios to test the system, we introduced mobility in the nodes and varied their locations. Different combinations of node placements were tried within a given topology. The attacker could be placed close to or far from the other nodes. He could be close to a few and far from the others. He could be close to (worst-case misbehavior) or far from (best-case misbehavior) the base station. The idea behind this type of testing is to see how the *closeness* varies for different scenarios. As noticed in the results shown in this section, there is minimal overall variance in accuracy.

Further, our scheme is free of adaptive cheating because the bad node's influence over good nodes shows up in a comparison of distributions. Since the model does not *expect* specific parameters, it is free of parametric adaptive cheating where a clever attacker may choose a different parameter from the one being monitored for. Also, since we do not look at magnitude based metrics, we do not suffer from effective-mean adaptive cheating.

3.7 Accuracy measures

As a preliminary measure towards testing the precision of detection, the width of the bins used in the Bayesian approach was tuned in an effort to determine the optimal width - one that yields peak accuracy. Traces from 10 windows of 10 seconds each were fed into the classifier. The detections from the 10 trials were used in determining True Positive Ratio (TPR) and False Positive Ratio (FPR) measures for the classifier. This procedure was followed for different bin widths.

In Figure 32a, note that with an increase in bin width the accuracy drops, which makes sense as the classifier works better with a higher number of bins.

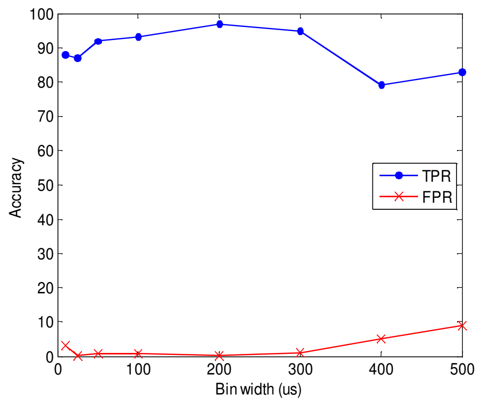

Figure 32. (a) Classifier bin width tuning.

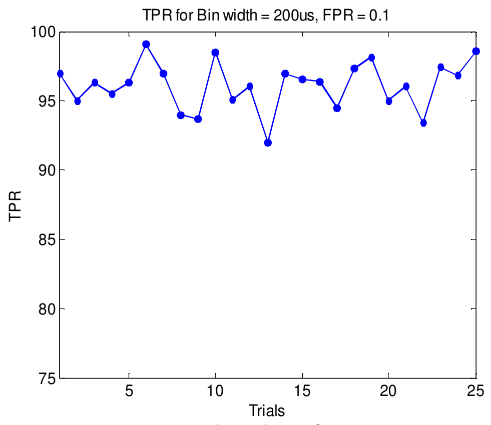

Figure 32. (b) Classifier accuracy.

An optimal bin width of 200μs was chosen, as it gives the minimum FPR of 0.1 and maximum TPR of 98. On testing the system with the chosen parameters for a total of 10 additional trials, it was observed that the technique is accurate in detection approximately 96% of the time, as seen in Figure 32b. For the trials in Figure 32b, traces from different scenarios were tested, in each of which network parameters were changed (as will be shown in the next section) in an attempt to test the scheme's robustness.

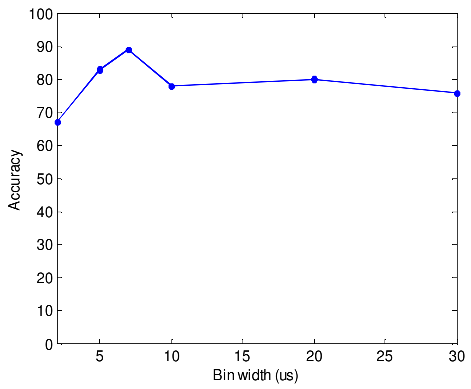

Figure 33. (a) Degree prediction bin width tuning.

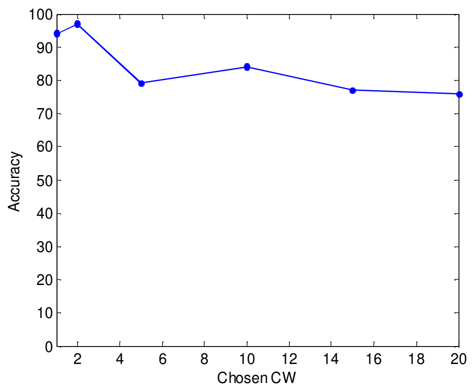

Figure 33. (b) Degree prediction accuracy.

A similar round of bin width tuning was performed to optimize the prediction of the degree of misbehavior. For each bin width, a total of 6 tests were performed, each corresponding to a CW value from Hypothesis IV. An optimal bin width of 7us (Figure 33a) was chosen, which gives an accuracy of 89%. Note that relatively low bin widths provide considerable accuracy in this scheme. This is because unlike classification, only the back-off portion of IAT is considered to perform the degree prediction. With the chosen bin width, each CW was tested to arrive at a pattern of accuracy as a function of degree of misbehavior (Figure 33b).

4. MAC LAYER COVERT TIMING CHANNEL DETECTION

4.1 Introduction

Most of the WIDSs in market today do not monitor for protocol misuse. In this section, we introduce the possibility of creating a stealthy timing channel for communication in a wireless network, through misbehavior at the native protocol level, and also present a detection scheme to counter the attack. This way, we exploit the attack discussed in the previous section to create the foundation for another.

Owing to its stochastic nature, a WLAN provides scope for an increased degree of information hiding. To maximize the strategy of utilizing the randomness in such a network to leverage covertness, we focus our channel design on the random portion of the inter-packet arrival delay between successive data transmissions. To this end, we use the random delay produced by the collision avoidance functionality of the 802.11 MAC protocol as *cover* for the intended covert channel.

While covert timing channels have long been studied at higher layers, and covert storage channels, using 802.11 headers, have been discussed recently, we propose a timing channel at the MAC layer that is based on misbehavior and normal behavior at the protocol stack. We study the fundamental operation of the 802.11 MAC protocol in an attempt to arrive at a platform that can generate distinct attacker profiles out of varying degrees of protocol manipulation. We show that despite such channels being distinct enough to be extracted at the receiver end, third party detectability is difficult because the working of the model is such that it contributes minimally to net throughput and does not exhibit regularity in transmission.

Ideally, we must consider the case where an attacker has compromised a machine, but not necessarily to the extent that he has uprooted the current user off the machine. The current user of the machine may still be using it not knowing that it has been compromised. In such a case, if an attacker tries to open a traffic stream of his own, he may be recognized. So, for him to operate stealthily, not just on the network, but also at the host, his program should be able to encode covert delays on to existing legit user generated traffic. The receiver, that may be on either side of the network, is equipped with suitable knowledge of the 802.11 MAC protocol's delay generation process and decodes accordingly. We show that despite being on the wired side, a receiver would be still be able to extract traces of the 802.11 MAC protocol by observing the delay pattern in traffic.

Two of the most important application scenarios for such a channel include covert information relay from a wireless node to the other side, and secretive command exchange between the bot-master and zombies in an ad hoc wireless bot network.

Such a channel may theoretically be used not just for malicious reasons, but also as a means for authentication without being overheard by sniffers [42, 43]. Alternatively, it may be used to create individual traffic classes in a Multi-level Security (MLS) system where various levels of confidentiality and integrity are required depending on the security clearance. However, given the malicious nature of MAC misbehavior, we consider the non-generic case where an attacker implements the channel to get information out of a compromised machine unnoticed.

We underline the importance of fingerprinting wireless traffic on the wired side, especially in the context of covert channels, because often the attacker's intent is to withdraw information to his own machine which may be on the other side of the Internet.

We analyze the effect of changing the contention window, which is the upper bound on the random back-off process, in such a way that it produces a relatively anomalous behavior when compared to the cooperative behavior exhibited by regular terminals on the network. This action has been shown in the past to lead to network performance gain at the attacking node as

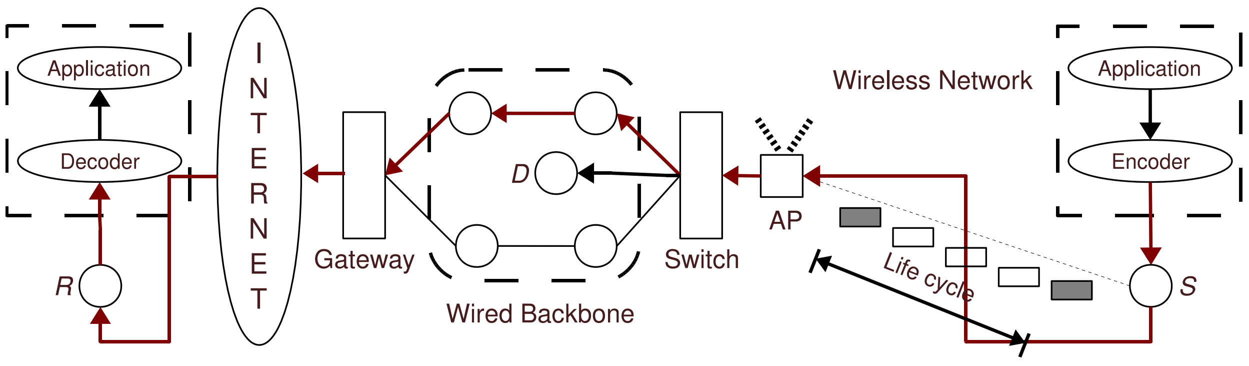

Figure 34. MAC covert timing channel.

well as network disruption throughout. We focus on studying how the overall delay in a wireless transmission is a function of the back-off and how changing the contention window can affect the arrival pattern of a process at the receiver end, in the context of its ability to create localized channels for secret communication.

The owner of the covert channel may transmit at different levels of back-off. The process at the receiving end would decode the incoming traffic by comparing the extracted delay to the analytically created 'expected delay' feature set.

Each varied degree of misbehavior, that is, each choice of an unused value for the contention window, can be used to represent a symbol in covert data. We introduce the possibility of creating channels out of delays that are both smaller and greater than the expected delay for wireless transmission, in such a way that the dual nature neutralizes the effect of the faster or slower transmission respectively on system and network performance. This improves the covertness of the model, if the system administrator or an intrusion detection system were to be performing basic traffic analysis - for instance, monitoring the system or network performance in terms of bandwidth utilization.

A second design choice for the channel is that a chunk of packets is used for each symbol of covert data to be transmitted. The reasoning behind this is that in order to be able to expect a delay at a particular selection of contention window, the receiver must account for the randomness over a range of values. Also, unlike using a magnitude of delay (and hence using a single packet pair) to constitute covert data, this approach is not affected by adverse network conditions (packet loss, delayed out of order arrival, etc).

The fact that the receiver would monitor traffic a block of packets at a time negates having to resort to covert transmission at predefined intervals. In other words, the sender and receiver do not settle on a frequency of embedding of covert data, but instead on the amount of data sent each time. This aspect greatly improves the channel's covertness.

Additionally, the covert block is sent at a pre-determined data rate for the sake of convenience at the receiver side. This will be elaborated in a later section. Note that this is not an attribute a detector can easily leverage while searching for the covert channel because if detection were performed on the wired side, the rates would not be available, and if detection were performed on the wireless side, the attacker may cycle through several predefined rates, rotating rates in pseudo-random intervals (the seed would be known only to the attacker at the receiving end apart from his covert channel application on the compromised machine).

Detection of such a channel is non-trivial given that crucial knowledge such as block size, data rate and chosen degree of misbehavior is limited to the owner of the channel. In such a case, the only way to detect the attack would be to monitor in chunks for block-wise anomalies in delay. Accordingly, our detection model is primarily based on an anomaly-based classifier that monitors for exceptions from the normal delay of nodes that do not cheat on DCF. The detection scheme is similar to the one introduced in Section 3.

The above aspects of the model will be discussed in the subsequent sections, backed up by experimental results. The remainder of the section is organized as follows. The next subsection introduces existing covert channel proposals and their detection models. In Section

4.3, we propose a model for MAC layer covert timing channels and Section 4.4 presents the results from the decoding accuracy tests. Finally, in Section 4.5, we present a detection scheme to counter the attack.

4.2 Related work

Covert timing channels have a prolonged history of research, but primarily with an emphasis on the wired network and higher layers of the TCP/IP protocol stack. Known covert channels include timing, side and storage channels. A simple, efficient timing channel at the IP layer is proposed in [38] that involves transmission or the lack of during regular intervals to represent covert information. An entropy-based detection method is presented in [39] that is shown to detect several common timing channels using a conditional rareness test. A TCP timing channel is discussed in [40] where covert relay is strategically delayed till periods of TCP burst in order to avoid the effect of jitter. The work in [41] discusses statistical measures, such as mean and variance to detect regularity based timing channels. The authors in [42] introduce a timing based authentication procedure that involves choosing two delay values larger than the delay observed under normal operation, to represent two covert symbols. A similar authentication method is proposed on the wireless side in [43], by means of modulating the watermark using different transmission rates. Also, on the wireless side, [44, 45] discuss storage channels on the wireless side by manipulating the 802.11 header. In [46], the AODV routing protocol is exploited to embed covert data in an ad hoc network setting.

We argue that using single packet pairs to represent covert data (like most of the above methods) is not suitable for noisy mediums, such as wireless networks, because of potential packet loss and out of order arrival. Hence, we suggest the use of multiple packet pairs for each bit of covert information. Also, our channel is hard to detect using regularity tests, because it does not rely on transmitting in regular intervals. Note that though our covert channel is based on MAC layer misbehavior, it would be hard to detect the channel using existing misbehavior detection schemes, most of which assume that attacker (misbehaving) traffic is saturated, and hence perform continuous monitoring. On the other hand, our detection scheme performs generic anomaly monitoring on partitioned time slices of traces.

4.3 Proposed model

The covert channel created at the MAC layer is a subtle timing based design whose existence is not observed easily at the TCP/IP layers because the existing detection schemes work with the information available at the higher layers.

The model is centered around the random delay period in DCF and requires nodes at both ends of the channel to be aware of the delay sequence. Though the back-off period constitutes a succession of random delays, the range is bounded, and this way, it is possible to predict a distribution for each chosen *CW*. We create analytical profiles pertaining to different degrees of misbehavior as shown in Figure 35.

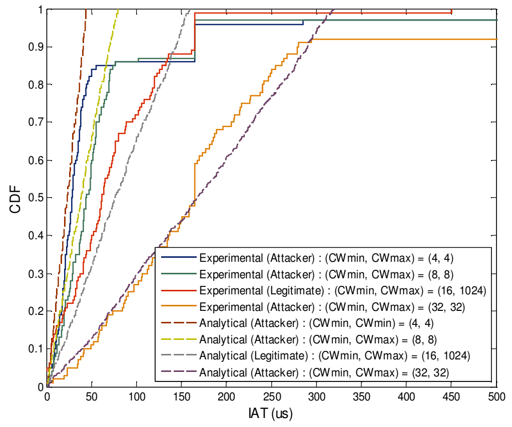

Figure 35. Analytical vs Experimental profiles for different *CW*.

The application that modulates higher layer transmissions with an appropriate delay works at the sender *S.* At the receiver end *R,* a decoder extracts the DCF components from each IAT and compares the resulting distribution with sample profiles. The DCF_{random} portion is the only part of IAT that is influenced by modifying *CW* and hence negating the contribution from the rest of IAT makes our study clearer and decoding more effective. Observe from Equations 12 and 13 that the share of $DCF_{constant}$ and DCF_{random} is minimum compared to that of $dtrans_{frame}$ and the rest of IAT. Hence it is important to set apart the uninfluenced portion so as to zoom in to the relatively negligible *DCF* part. It is important to note that the data rate of attacker traffic is set constant. This way, $dtrans_{frame}$ and $dtrans_{overhead}$ (both of which are functions of data rate) will not vary for each packet. This is necessary so that *R* can subtract out the same chunk each time. Given that *R* may not be on the wireless side, and hence cannot see the rates in the wired side, this assumption is required. Following the DCF extraction, attack classification is performed using the Bayesian binning technique presented for the previous two attacks .

Design decisions on how an appropriate value of *CW* may be chosen for the channel are discussed with experimental results in a later section. *Covert CW* values are proportionally chosen from each side of the *legitimate* (CW_{min} , Cw_{max}) pair. Such *low* (α) and *high* (β) values represent 0 and 1 in case of a two-symbol embedding strategy. The attacker may also use multiple symbols to speed up the transmission of covert data. In such a case, α_1 and α_2 would represent bit symbols '00' and '01'; β_1 and β_2 would represent '10' and '11'. Also, MAC misbehavior works best if the same values are chosen for both CW_{min} and CW_{max} so that there is minimal variation in delay. This also negates the effect of competing terminals in the network which makes sure that profile matching in larger networks take place just as good. That is, repeated backing off due to a significant number of nodes in the network can lead to distributions that exceed the expected delay pattern from a network with minimal cross-traffic. By canceling out the property of continuing to back-off despite the *CW* reaching CW_{max}, we can expect traffic that does not vary beyond a point proportional to the number of neighboring nodes.

As mentioned before, *S* transmits all information (covert and non-covert) in blocks. Note that when we state that we transmit in a block of packets, we refer to encoding covert bits of data in the packets from higher layers in blocks, by means of choosing different delay values. For a given block size *b*, depending on when covert data is required to be transmitted, *R* chooses whether or not to encode the bits in each successive block of packets sent out of the compromised machine. Accordingly, *R* monitors in blocks of size *b* and inspects each incoming sequence of packets for a match with one of the stored signatures. Block size *b* acts as a kind of secret key in such a communication model.

The reason for using a 'block' to represent each bit of covert information is as follows. In the case of *high* delay, regular traffic is less likely to fall into the covert frequency band. However, the reduced delay bands used to represent the other half of the covert channel *(low)* are not free from interference from the legitimate traffic band. This is because the back-off in regular traffic is chosen from a random window in the range of (*0*, *CW*). While there is no upper bound on the high delay that can be chosen (*high* can be anywhere above CW_{max} depending on the number of nodes in the network), there exists a lower bound on the *low* delay. For example, since the default CW_{min} is 16*µs* for an 802.11g network, in order to produce a delay pattern lower than that of legitimate traffic, *low* can only be chosen in the range (0, 15). That being the case, profile overlap issues would arise if a magnitude was used for *low* and *high* because the filter at *R* will not be able to tell the difference between *low* (purposely transmitted at a reduced *CW*) and a reduced legitimate back-off (incidentally chosen randomly between 0 and 15). Hence, it is important to handle traffic in 'blocks' (instead treating packets individually) for the purpose of accuracy when filtering out, because a sequence of packets cannot be transmitted continuously at reduced back-off values unless it were made to do so on purpose. That is, over a sequence of packets, it is likely that legitimate delay can be expected to be distributed uniformly over a large *CW*, say (0, 15) as opposed to say, (0, 3). Furthermore, using a block of packets to represent a bit of information means that unlike single packets, the channel is not influenced by unfavorable network conditions, including packet loss, delayed out of order arrival, etc. An immediate consideration for such case is that utilizing a group of packets to send out a bit of covert data could be expensive in terms of covert channel speed. We show in a later section that the number of packets required per covert piece of information is minimal.

An important attribute of our scheme is that since a block of secretly known number of packets is transmitted each time, it is not necessary to transmit in regular intervals. *S* and *R* need not synchronize on the period of covert data transmission, but instead on the amount of covert data sent out at a stretch. Since *S* and *R* do not settle a priori on an interval of transmission (constant or pseudo-random), the channel exhibits increased covertness. Apart from the fact the using a 'block' to represent covert data negates the requirement to settle on a frequency of embedding, it also obviates the need to synchronize on modulation start and end points because the whole block represents only one bit of covert data.

Since *R* is not waiting on *S* to transmit at regular intervals, *S* can transmit at his convenience. With this luxury comes the freedom for *S* to dynamically adapt to channel conditions. Instead of transmitting at static predefined intervals, he may improve the covertness of his channel by reducing the cycle length during times of increased traffic density on the network. In other words, he may transmit more frequently during rush hour and ease out when the compromised host is idle. He can also increase the amount of covert data being sent per cycle during times of high traffic (by increasing the block size). It is especially important to increase the block size during such times, because by nature of our model, since every block of packets represents only 1 bit of covert data, the channel could use the speedup.

TABLE VI
Class of misbehavior

Degree of misbehavior (*m*)	***(CW_{min},CW_{max})***
0	(16, 1024)
1	(1, 1)
2	(2, 2)
3	(4, 4)
4	(8, 8)
5	(32, 32)
6	(64, 64)
7	(128, 128)

The degree of misbehavior *m* may be chosen according to the channel operator's preference of level of covertness and decoding accuracy at *R*. Higher degree of misbehavior means greater accuracy of receiver extraction and faster transmission of covert data but reduced covertness. We distribute the class of misbehavior as shown in Table VI. Degrees 1-4 are the options to choose *low* from; degrees 5-7 fall in the category of *high.*

Given that the chosen block size *b,* frequency of embedding *f*, and degree of misbehavior *m* are not public, an intrusion detection system *D* would not be able to reproduce the operation of *R*. Additionally, the channel's existence cannot be observed using regularity based tests. For this reason, a different detection strategy is required. To this end, we present an anomaly based monitoring scheme in Section 4.5.

4.4 Validation

The experimental testbed used for validating the model is the same as the one used for testing Rogue AP detection. The laptops act as clients sending data to the desktop. One of the laptops is the compromised machine where the covert channel is implemented. Other laptops exhibit cooperative behavior.

MAC misbehavior is performed by modifying the CW values in the *madwifi* driver. A simple method for doing this is by taking advantage of the wireless QoS provisioning in 802.11e compatible wireless routers. In such a setting, it is possible to configure the Enhanced Distributed Coordination Access (EDCA) parameters (including CW) for each QoS traffic class - best effort, background, voice and video.

We briefly tested the effect of MAC misbehavior on both 802.11b and 802.11g networks. For each network type, a TCP trace of close to 10000 packets for both misbehaving and legitimate nodes was collected. The misbehaving node was configured with a (CW_{min}, CW_{max}) of (1,1) in an attempt to study the share of throughput being stolen from the network. It was observed that the effect of misbehavior is more severe in 802.11g networks. As seen in Figure 36, the magnitude gain in average throughput achieved by an aggressive attacker in an 802.11g network is up to three times as much than that in an 802.11b network. Hence, for our accuracy measurement experiments in the next section, we choose to work with a 802.11g network, as it presents the worst case. Note that in the above experiment, both misbehaving and legitimate nodes were made to transmit equal amount of data at the same time. Hence, since the frequency of embedding (proportion of covert data to legitimate data) is 100%, the large variation.

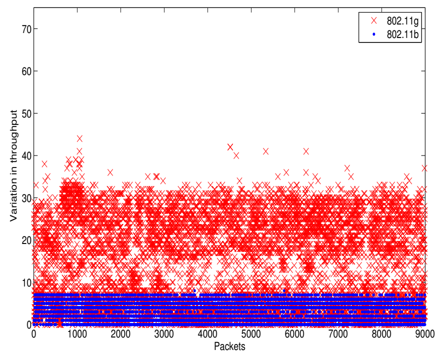

Figure 36. Throughput variation: 802.11g vs 802.11b.

To validate the working of our model, we performed a step-wise accuracy examination, starting with the accuracy of decoding at *R*.

First, we seek to tune the bin width *w* of each of the bins in the Bayesian classifier. Given the number of parameters that have to be tuned, we set out by assuming base values for the other parameters - *b*, *f and m* for the sake of validating *w*. Following this step, the optimum value of *w* would be used to correct the others. We assume *b* = 10 packets. This is the lowest upper bound on back-off possible in an 802.11g network. That is, if (CW_{min} , Cw_{max}) takes the lowest possible combination (1, 1), it performs a back-off in the range (0, 10) [Refer to Equation 6]. This represents worst case in that it is the minimum amount of information required for each bit of covert data, and hence it must give a true picture of the accuracy. Since each 'block' is inspected individually, the decoding accuracy of each block of data is independent of that of others. Hence, *f* does not influence accuracy and only affects covertness (measured as a function of variation in network throughput), the initial value for *f* does not concern bin width tuning. We randomly decide on a ratio of 1:9 for covert and legitimate data. Hence *f* = 10%. For the sake of convenience and the purpose of full visibility into the process, we choose two extreme values for *low* and *high.* We configure *S* with *m* = (1, 6), that is, *low* = (1, 1) and *high* = (128, 128) [Refer to Table VI]. 50 trials each for TCP and UDP were performed, where each trial is a transfer of 1000 packets (10 cycles of 100 packets each) Since *b* = 10 packets and *f* = 10%, a block (10 packets) of covert data was transmitted for every 9 blocks (90 packets) of legitimate data. In other words, each life cycle comprises a total of *(b* + 9*b)* packets*;* 10 such cycles comprise of 1000 packets in total. The resulting True Positive Ratios (TPR) and False Positive Ratios (FPR) for varying *w* are shown in Figure 37. An optimum bin width of 20*µs* that gives an FPR of 0.2 was chosen, for which 10 additional trials each were performed for TCP and UDP to study the variation of TPR, as shown in Figure 38.

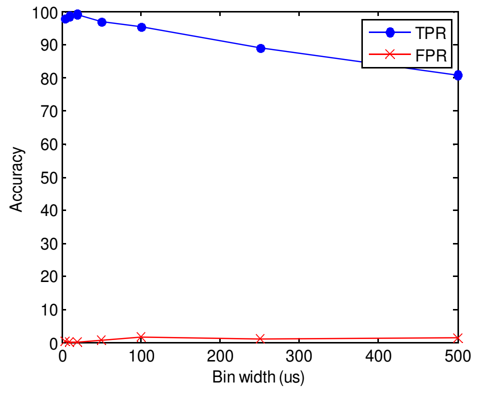

Figure 37. Bin width tuning.

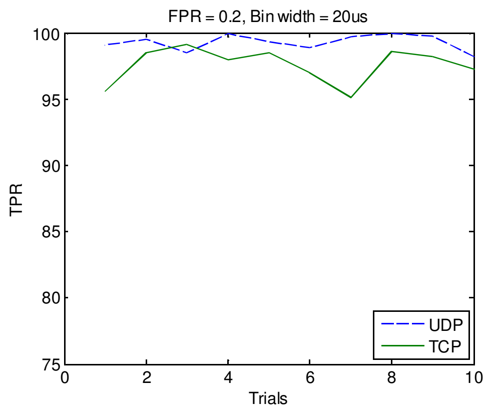

Figure 38. True positive ratio (TPR) measure.

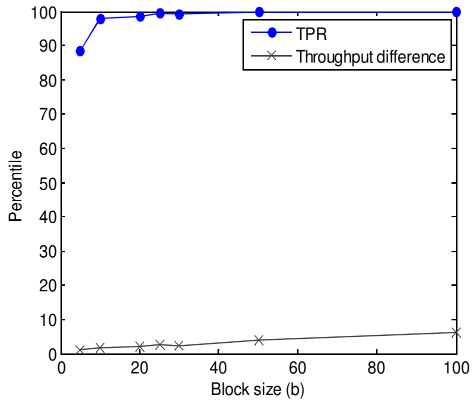

Figure 39. Block size tuning.

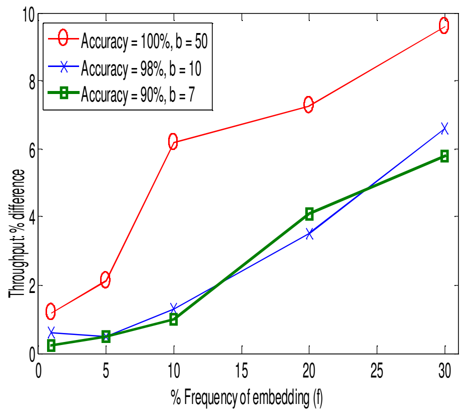

Figure 40 Embedding frequency tuning.

In the second step, block size b was tuned with $w = 20\mu s$, $f = 10\%$ and $m = (1, 6)$. The block size was varied to study the effect on TPR and network throughput variation. The difference in the average throughput of legitimate and attacker traffic is employed as an indication of covertness, where covertness is inversely proportional to throughput variation. Note in Figure 39 that with an increase in b, there is improvement in decoding accuracy and a corresponding increase in variation of throughput as well, meaning reduced covertness. The experiments performed were similar to the trials from the previous phase. The amount of data transmitted in each trial varies as a function of b and f ; it is 1000 packets (10 cycles of 100 packets each) for $b = 10$ packets and 10000 packets (10 cycles of 1000 packets each) for $b = 100$ packets.

In the third phase, we tune *f* - the proportion of covert data in a life cycle. We assume *m* as before and set $w = 20\mu s$. From Figure 39, we select three values of *b* that provide a suitable trade-off between decoding accuracy and covertness. With these parameters, we tune *f* to determine its effect on covertness. Ten trials of 10 cycles each were performed. We notice from Figure 40 that to achieve 100% accuracy, one needs to compromise on covertness, as it exhibits a relatively high deviation in bandwidth usage. However, note that the average variation is minimal across the range of *f*. Note that *f* is tuned only as a means of measuring the system's performance. The attacker would ideally want to adapt *f* proportional to the current state of network traffic, because he has to option of transmitting with non-regular periods.

Next, we seek to study the effect of different combinations of *low* and *high* on covertness and accuracy. We set $w = 20\mu s$, and set *b* and *f* in such a way that an appropriate trade-off between accuracy and covertness is achieved. For 98% accuracy, $b = 10$ packets, and $f = 5\%$, note from Figure 38 that the trade-off is optimal. With these parameters, m_{low} was varied between 1 and 4, and m_{high} between 5 and 7. Figures 41 and 42 provide the channel owner a neat idea of covertness and accuracy respectively for different *m*.

Note that in each of the above cases, 100% accuracy would still possible even in lossy networks by means of using forward error correction codes on top of the regular covert data.

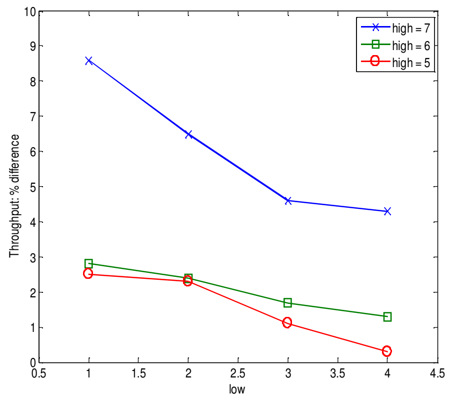

Figure 41. Misbehavior degree tuning: Covertness.

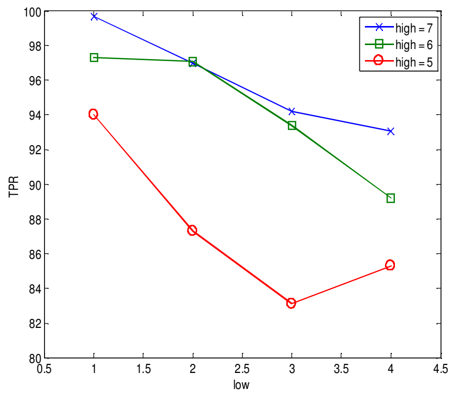

Figure 42. Misbehavior degree tuning: Accuracy.

4.5 Detection

A third party detector will not be able to use existing MAC misbehavior detection schemes to detect our covert channel, most of which assume saturated attacker traffic. In fact, we cannot use our MAC misbehavior detection method discussed in Section 3 for the same reason. Hence, we adapt our detection scheme to monitor in chunks of time, so as to give better visibility into the anomalies when they occur. This is necessary considering that the attacker would likely transmit covert data with reduced frequency.

A detector *D* lacks information such as block size, degree of misbehavior and data rate; hence, he cannot use the same classification scheme used by *R*. We suggest the use of an anomaly-based intrusion detection scheme, monitoring on time chunks of traces, where chunk size *t* is chosen by *D* after tuning to decide on the size that is optimum for accurately sensing the deviation. If chosen *t* is too small, detector may not sense the deviation because of inadequate packets to judge whether or not there is a deviation (might result in false positives); if *t* is too large, the detector may not see the deviation because the proportion of covert data compared to legit data is negligible and also because of possible ON/OFF transmission activity over a large interval of time.

The reason for monitoring chunks of time instead of blocks of packets is as follows. *R* compares each stream's DCF with a baseline. Each stream's classification process is independent of that of other streams. Hence, he waits for a block of sequentially transmitted packets to compare. On the other hand, *D* performs a relative comparison of streams with each other. Since the classification of one stream is dependent on the that of others, he cannot compare blocks of packets, because the individual traffic streams are not symmetric - that is, data can be transmitted at different times for each stream. To perform relative comparison, *D* requires traces of more than one stream transmitted at the same time. Thus, *D* monitors for chunks of time instead of a sequence of packets, and compares the IAT sequences of traces from different streams transmitted during each chunk. In the absence of more than one stream, *D* may compare the trace from one chunk of time with those from subsequent time frames in the same stream.

Given sufficient knowledge, one should be able to extract further accuracy of detection out of *D* by in turn processing each chunk of time in blocks of packets. This might be necessary considering that the chosen block size *b* would likely be negligible compared to the amount of data sent even in a chunk of a fraction of a second. In other words, $b << pkts_t$, where $pkts_t$ is the number of packets transmitted per chunk of time ($0.1s < t < 1s$). However, even if *D* assumes worst case attacker block size $b = 10$ packets (that provides maximum attacker covertness), a choice of a different *b* hampers detection accuracy because *D* would end up looking at the wrong places for an anomaly. Hence, we stick to monitoring data in chunks of time, as opposed to blocks of packets. Of course, this implies decreased degree of detection accuracy because of a large number of sampled packets per chunk of time, as will be shown.

Also, since the detector does not know the data rate, he cannot extract the DCF portion. So in our model, *R* extracts DCF while *D* uses IAT as a whole for classification.

For detection, we again use the Bayes classifier, except with appropriate changes to reflect the type of classification: supervised vs unsupervised. In the case of decoding at *R*, it compares the bin frequencies of each *known* profile with those of the unknown trace and predicts the trace as being akin to the profile whose frequency distribution closest resembles that of the trace. In the case of detection at *D*, it compares the bin frequencies of the first trace with those of every other trace in parts (one chunk of time at a time) to calculate *c*.

To validate the detection scheme, we used the same experimental setup as in the previous section, with one of the laptops acting as the detector, one acting as attacker and one as a legitimate node. We tested the anomaly-based detection concept by varying *t*, where *t* is a chunk of time in seconds. For every value of *t*, 500 traces each of purely legitimate and covert traffic each were collected to test *D* on. As before, we create a covert channel with parameters configured in a way that maximizes the optimality. Referring to Figures 40, 41 and 42, we choose values that provide a decent trade-off between decoding accuracy and channel covertness. The channel was set with b = 10, f = 5, m_{low} = 2, and m_{high} = 6. Accuracy tends to decrease, as shown in Figure 43, with an increase in *t*. This makes sense because as *D* monitors for longer time periods, given that the attacker is transmitting at very small values of *b* and *f*, the covert portion of traffic is hidden easily amidst the relatively large trace and anomalies are hard to detect. Having said that the scheme is able to detect with up to 80% accuracy even for increasingly stealthy attacker operation.

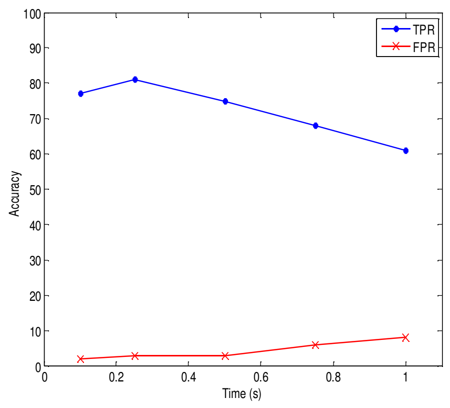

Figure 43. Detection accuracy.

5. CONCLUSION

In this thesis, we have proposed a solution for three covert attacks in a wireless network. The motive for addressing this problem space is that the state-of-the-art in market WIDSs do not possess strong defenses against the stealth attacks addressed herein - unauthorized APs, greedy behavior at the MAC layer, and MAC layer covert timing channels. Our model relies on fingerprinting wireless traffic from the wired side by extracting traces of the native (802.11 MAC) link-layer protocol from the observed traffic delay patterns. Since it does not require changes to be made to the existing wireless network infrastructure or protocol, and acts as a single point of discovery, it is an immediately deployable alternative to existing WIDSs. We have shown the efficiency of the model by means of experiments on a network testbed and network simulations.

6. FUTURE WORK

As a part of our future work, we plan to deploy and validate the scalability of the system on a real network as opposed to a testbed. Such scalability measures will be in terms of accuracy of detection in larger networks, that include both longer bridge distances between the wireless network and wired backbone, as well as increased competing traffic on the wireless and wired sides. Particularly, we would study the effect of such an increase in network size on the inter-packet arrival delay at the receiver end on the wired side, so that our analytical model may be tuned with a derived offset.

In the rogue access point detection problem space, we plan to extend our model with functionality to differentiate between legitimate and unauthorized APs through traffic analysis. To this end, we will study the differences in traffic patterns exhibited by different commercial APs, in an attempt to arrive at a range of distinct signatures. Such a profile set may be used in identifying APs of a make not used within a network. This logic is especially valid assuming corporate networks would employ high-end APs or wireless routers, whereas an attacker would likely bring in a cheaply available one.

We plan to study types of MAC layer misbehavior other than basic DCF parameter manipulation and extend our solution to detect any generic deviation from the protocol by means of misuse.

We will perform tests to show that our covert timing channel cannot be detected by regularity tests as well as existing MAC layer misbehavior detection techniques. Also, we will check if our detection model is able to detect other existing timing channels, including ones at the TCP and IP layers. We will measure the capacity and reliability of our covert channel. We will take into account an implementation of the channel that exploits delay parameters to represent multiple symbols, instead of just 0s and 1s. We will work towards formalizing our theory that usage of reduced and larger delays results in stabilization of throughput, and seek to analytically arrive at trade-off threshold points that can be used in designing the channel. We will study in detail how link-layer retransmissions and out of order packet arrivals affect our channel. We will address the covert channel prevention scheme of radio frequency jamming, and see how that affects our channel. As network throughput variation is not the only measure of covertness, we will address other means an intrusion detection system may use to monitor the network.

REFERENCES

1. Beyah, R.; Kangude, S.; Yu, G.; Strickland, B.; Copeland, J., “Rogue access point detection using temporal traffic characteristics,” *IEEE GLOBECOM* 2004.
2. Shetty, S.; Song, M.; Ma, L., “Rogue Access Point Detection by Analyzing Network Traffic Characteristics,” *IEEE MILCOM* 2007.
3. Wei, W.; Suh, K.; Gu, Y.; Wang, B.; Kurose, J., “Passive online rogue access point detection using sequential hypothesis testing with tcp ack-pairs,” *ACM IMC* 2007.
4. Wei, W.; Jaiswal, S.; Kurose, J.; Towsley, D., “Identifying 802.11 Traffic from Passive Measurements Using Iterative Bayesian Inference”, *IEEE INFOCOM* 2006.
5. Wei, W.; Wang, B.; Zhg, C.; Kurose, J.; Towsley, D., “Classification of access network types: Ethernet, Wireless LAN, ADSL, Cable Modem or Dialup?,” *IEEE INFOCOM*, 2005.
6. Baiamonte, V.; Papagiannaki, K.; Iannaccone. G., “Detecting 802.11 wireless hosts from remote passive observations,” *IFIP/TC6 Networking* 2007.
7. Beyah, R.; Watkins, L.; Corbett, C., “A Passive Approach to Rogue Access Point Detection,” *IEEE GLOBECOM* 2007.
8. Mano, C.; Blaich, A.; Liao, Q.; Jiang, Y.; Cieslak, D.; Salyers, D.; Striegel, A., “RIPPS: Rogue Identifying Packet Payload Slicer Detecting Unauthorized Wireless Hosts Through Network Traffic Conditioning,” *ACM TISSEC*, Vol. 11, 2007.
9. Cheng, L.; Marsic, I., “Fuzzy reasoning for wireless awareness,” *International Journal of Wireless Information Networks*, Vol. 8, 2001.
10. Bahl, P.; Padhye, J.; Ravindranath, L., “Enhancing the Security of Corporate WI-FI Networks Using DAIR”, *ACM MobiSys* 2006.
11. http://www.netstumbler.com

12. http://www.wimetrics.com
13. http://www.proxim.com
14. http://www.airdefense.net
15. http://www.airmagnet.com
16. http://www.airwave.com
17. http://www.cisco.com
18. Chirumamilla, M. K.; Ramamurthy, B., “Agent based intrusion detection and response system for wireless LANs,” *IEEE ICC* 2003.
19. Ma, L.; Cheng, X., “A Hybrid Rogue Access Point Protection Framework for Commodity Wi-Fi Networks,” *IEEE INFOCOM* 2008.
20. Songrit, S.; Kitti, W.; Anan, P., **“**Integrated Wireless Rogue Access Point Detection and Counterattack System,” *IEEE ISA* 2008.
21. Beyah, R.; Corbett, C.; Copeland, J., “A Passive Approach to Wireless NIC Identification,” *IEEE ICC* 2006.
22. Bianchi, G., “Performance analysis of the IEEE 802.11 distributed coordination function,” *IEEE Journal on Selected Areas of Communications*, Vol. 18, 2000.
23. Bing, B., “Measured Performance of the IEEE 802.11 Wireless LAN,” *LCN* 1999.
24. Chatzimisios, P.; Vitsas, V.; Boucouvalas, A. C., “Throughput and Delay analysis of IEEE 802.11 protocol,” *IEEE IWNA*, 2002.
25. http://www.isi.edu/nsnam/ns
26. Toledo, A. L.; Xiaodong Wang, "Robust Detection of Selfish Misbehavior in Wireless Networks," *IEEE Journal on Selected Areas in Communications,* Vol. 25, 2007.
27. Rong, Y.; Lee, S. K.; Choi, H. A., "Detecting Stations Cheating on Backoff Rules in 802.11 Networks Using Sequential Analysis," *IEEE INFOCOM* 2006.

28. Guang, L.; Assi, C., "Mitigating Smart Selfish MAC Layer Misbehavior in Ad Hoc Networks," *Wireless and Mobile Computing, Networking and Communications*, 2006.

29. Kyasanur, P.; Vaidya, N. H., "Selfish MAC layer misbehavior in wireless networks," *IEEE Transactions on Mobile Computing*, Vol. 4, 2005.

30. Raya, M.; Aad, I.; Hubaux, J. P.; El Fawal, A., "DOMINO: Detecting MAC Layer Greedy Behavior in IEEE 802.11 Hotspots," *IEEE Transactions on Mobile Computing*, Vol. 5, 2006.

31. Radosavac, S.; Moustakides, G.; Baras, J.; Koutsopoulos, I., "An Analytic Framework for Modeling and Detecting Access Layer Misbehavior in Wireless Networks," *ACM TISSEC*, Vol. 11, 2008.

32. Radosavac, S.; Cárdenas, A. A.; Baras, J. S.; Moustakides, G. V, "Detecting IEEE 802.11 MAC layer misbehavior in ad hoc networks: Robust strategies against individual and colluding attackers," *Journal of Computer. Security*, Vol.15, 2007.

33. Lolla, V.N.; Lap Kong L.; Krishnamurthy, S.V.; Ravishankar, C.; Manjunath, D., "Detecting MAC Layer Back-off Timer Violations in Mobile Ad Hoc Networks," *IEEE ICDCS* 2006.

34. Cagalj, M.; Ganeriwal, S.; Aad, I.; Hubaux, J. P., "On selfish behavior in CSMA/CA networks," *IEEE INFOCOM* 2005.

35. Youngmi J.; Kesidis, G., "Distributed Contention Window Control for Selfish Users in IEEE 802.11 Wireless LANs," *IEEE Journal on Selected Areas in Communications*, Vol.25, 2007.

36. Guang, L.; Assi, C., "A Self-Adaptive Detection System for MAC Misbehavior in Ad Hoc Networks," *IEEE ICC* 2006.

37. Pang, Q.; Leung, V. C. M.; Liew, S. C., "A rate adaptation algorithm for IEEE 802.11

WLANs based on MAC-layer loss differentiation," *Broadband Networks,* 2005.

38. Cabuk, S.; Brodley, C. E.; Shields, C., “IP covert timing channels: Design and Detection,” *ACM CCS* 2004.
39. Gianvecchio, S; Wang, H., “Detecting covert timing channels: an entropy-based approach.,” *ACM CCS* 2007.
40. Xiapu L.; Chan, E.; Chang, R., "TCP covert timing channels: Design and detection," *IEEE/IFIP DSN* 2008.
41. Berk, V.; Giani, A.; Cybenko, G., “Detection of covert channel encoding in network packet delays,” Technical Report, Dartmouth College, 2005.
42. Newman, R.; Beyah, R., “On the performance of using covert timing channels for node authentication,” Accepted for publication, *Security and Communication Networks Journal,* 2008.
43. Calhoun, T.; Newman, R.; Beyah, R., “ Authentication in 802.11 LANs using a covert side channel,” In submission, *IEEE ICC* 2009.
44. Krätzer, C.; Dittmann, J.; Lang, A.; Kühne, T., “WLAN steganography: a first practical review,” *ACM MM&Sec* 2006.
45. Frikha, L.; Trabelsi, Z.; El-Hajj, W., “Implementation of a covert channel in the 802.11 header,” *IEEE IWCMC* 2008.
46. Song L.; Epliremides, A., “A network layer covert channel in ad hoc wireless networks,” *IEEE SECON* 2004.