

Detecting Spoofing and Anomalous Traffic in Wireless Networks via Forge-Resistant Relationships

Qing Li and Wade Trappe

Abstract—Many wireless networks are susceptible to spoofing attacks. Conventionally, ensuring the identity of the communicator and detecting an adversarial presence is performed via device authentication. Unfortunately, full-scale authentication is not always desirable as it requires key management and more extensive computations. In this paper, we propose noncryptographic mechanisms that are complementary to authentication and can detect device spoofing with little or no dependency on cryptographic keys. We introduce forge-resistant relationships associated with transmitted packets, and forge-resistant consistency checks, which allow other network entities to detect anomalous activity. We then provide several practical examples of forge-resistant relationships for detecting anomalous network activity. We explore the use of monotonic relationships in the sequence number fields, the use of a supplemental identifier field that evolves in time according to a reverse one-way function chain, and the use of traffic statistics to differentiate between anomalous traffic and congestion. We then show how these relationships can be used to construct classifiers that provide a multilevel threat assessment. We validate these methods through experiments conducted on the ORBIT wireless testbed.

Index Terms—Authentication, intrusion detection, security, spoofing attacks.

I. INTRODUCTION

ONE serious threat facing wireless networks is spoofing, where one radio device can alter its network identifiers to that of another network device. Spoofing attacks are very easy to launch in many wireless networks. For example, in an 802.11 network, a device can alter its medium access control (MAC) address by simply issuing an `ifconfig` command. This weakness is serious, and there are numerous attacks, ranging from denial-of-service attacks [1] to session hijacking [2] to attacks on access control lists [3] that are facilitated by spoofing.

Although full-scale authentication mechanisms are the natural solution for coping with issues of identity verification, there are several reasons why it is desirable to explore other, complementary security measures for wireless networks. First, the use of authentication requires the existence of reliable key management/maintenance. This is often not possible. For example, wireless local-area network (WLAN) hotspots are often deployed in local coffee shops or book stores without any way to check the identities of their constantly evolving user

base and without any means to distribute authentication keys. Further, many wireless devices can be easily pilfered, their memory scanned, and their programming altered. This can lead to authentication keys becoming compromised and, in order to maintain the integrity of authentication checks, it is necessary to periodically refresh these keys through methods that are not subject to compromise [4]–[7]. Altogether, full-scale authentication [8]–[10] runs the risk of authentication keys being compromised and requires an extensive infrastructure to maintain the integrity of authentication methods.

In this paper, we take the viewpoint that it is desirable to have a lightweight security layer that is separate from conventional network authentication methods. Toward this objective, we propose a suite of lightweight security solutions for wireless networks that complement the use of conventional authentication services and are intended to detect multiple devices using the same network identity. Our approach involves introducing detectors of anomalous behavior within the MAC layer. If no authentication service is available, then our approach can serve as an effective antispoofing mechanism, while if authentication is available, then our approach can further mitigate the risk of compromised keys while lightening the load on authentication buffers, which are often the target of resource consumption attacks. Our approach does not replace authentication since it does not rely on the explicit use of authentication keys to identify entities. Instead, our strategy involves the verification of forge-resistant relationships between packets coming from a claimed network identity. Our methods are generic, operate locally, and are suitable for a broad array of wireless networks.

The rest of this paper is organized as follows. We first overview the basic strategy by describing a formal model for relationship-based security in Section II. There are two varieties of relationships that we explore in this paper: 1) relationships that are introduced through auxiliary fields in packets (Section III) and 2) relationships that result from the use of intrinsic properties associated with the transmission and reception of packets (Section IV). We support our strategies with theoretical analysis and provide guidelines for appropriately selecting associated parameters. We then provide an example of how our schemes can activate security responses through multilevel classification in Section V. We validate our detection methods in Section VI, where we have conducted experiments using the ORBIT wireless testbed. Finally, Section VII concludes this paper.

II. STRATEGY OVERVIEW

Typically, binding approaches are employed in order to defend against network identity spoofing [11]–[13]. The general scenario we are concerned with involves two or more devices

Manuscript received January 25, 2007; revised September 8, 2007. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Mohan S. Kankanhalli.

The authors are with Wireless Information Network Laboratory (WINLAB), Rutgers University, Piscataway, NJ 08854 USA (e-mail: qingli@winlab.rutgers.edu; trappe@winlab.rutgers.edu).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TIFS.2007.910236

claiming a particular network identity u_j . We refer to these devices as A , B , C , and so on. In general, we will only need to refer to three devices: 1) A ; 2) B ; and 3) X . Device X is a monitoring device and is responsible for detecting anomalous network behavior. The monitor X may be another wireless network participant or it may be part of a separate security infrastructure. Associated with identity u_j will be a particular sequence of packets $\{P_j(k)\}$. The packet $P_j(k)$ consists of the payload, which comes from a higher-layer service. In addition to the payload, each packet has a state $S_j(k)$ associated with it. The state $S_j(k)$ may itself be a field contained within $P_j(k)$ or may be a property measured at the receiver.

For each identity u_j , we require that there is a rule \mathcal{R} that specifies the relationship between a set of observed $S_j(k)$ states. Suppose that we define an observation $\omega_j(k)$ to be a collection of N states corresponding to the k th packet from u_j . For example, one choice for $\omega_j(k)$ might be to take $\omega_j(k) = \{S_j(k), S_j(k-1)\}$, which is simply two consecutive states. A relationship consistency check (RCC) is a binary function $R_j(\omega_j(k))$ that returns 1 if the states in $\omega_j(k)$ obey the rule \mathcal{R} with respect to each other (i.e., they are consistent), or returns 0 if they do not (i.e., they are inconsistent and suspicious). As we shall explore later, one possible relationship that we might require is that $S_j(k) = S_j(k-1) + 1 \pmod{M}$ for some modulus M . A potential RCC would return 1 if $S_j(k)$ is close to $S_j(k-1)$ or return to 0 if they are far apart.

Simply using any relationship \mathcal{R} and checking the corresponding RCC at a monitoring device is not enough to provide reliable security. We must have the guarantee that an adversary cannot easily forge a set ω_j that would pass the RCC. Therefore, just as it is necessary to add security properties to hash functions, in order to have cryptographically useful hash functions, we need to add forgeability requirements to the relationship \mathcal{R} . This leads to the following definition.

Definition 1: An ϵ -forge-resistant relationship \mathcal{R} is a rule governing the relationship between a set of states for a node j for which there is a small probability ϵ of another device being able to forge a set of states $\tilde{\omega}$ such that a monitoring device would evaluate the corresponding RCC as $R(\tilde{\omega}) = 1$.

The definition of a forge-resistant RCC (RRCC) then corresponds to defining a detector capable of identifying anomalous network traffic resulting from the spoofing of another node's identity. The output $R_j(\omega_j(k))$ is either 1 or 0, and is a declaration by a monitoring device of whether suspicious network behavior has occurred. The output of the RRCC may be viewed as deciding between two different hypotheses based upon an observation vector $\omega_j(k)$. Hypothesis testing is thus the appropriate framework for defining an RRCC [14]. Suppose the observation vectors $\omega_j(k)$ for identity u_j are N -dimensional, then $R_j(\omega_j(k))$ partitions N -dimensional space into two critical regions Ω_0 and Ω_1 . The region Ω_0 corresponds to those observation vectors that will be classified as nonsuspicious data (the null hypothesis \mathcal{H}_0), while Ω_1 corresponds to those observation vectors that are suspicious or anomalous (the alternate hypothesis \mathcal{H}_1).

There are several measures for quantifying the effectiveness of an R . The probability of false alarm $P_{FA} = Pr(\mathcal{H}_1; \mathcal{H}_0)$ is the probability that we decide $\omega_j(k)$ is suspicious when it was legitimately created. The probability of missed detection $P_{MD} = Pr(\mathcal{H}_0; \mathcal{H}_1)$ is the probability of deciding the vector

$\omega_j(k)$ is legitimate when it was actually created by an adversary. The power, or probability of detection, of R is simply $P_D = 1 - P_{MD}$. An RRCC with a forge resistance of ϵ therefore corresponds to an RRCC with $P_{MD} = \epsilon$. We note that it is often useful to characterize an RRCC via the pair $(\epsilon, \delta) = (P_{MD}, P_{FA})$, as there might be several choices for R that have equivalent probability of missed detections, yet drastically different false alarm characteristics. We therefore have the following definition:

Definition 2: An RCC R , corresponding to an ϵ -forge-resistant relationship \mathcal{R} , is an (ϵ, δ) -resistant RCC, or (ϵ, δ) RRCC if $P_{FA} = \delta$.

The (ϵ, δ) RRCC will operate at a monitoring device to detect anomalous behavior. Based upon the choice of ϵ and δ , anomaly detection might be used to drive various security responses. For example, in a WLAN, if an access point detects suspicious network behavior with low δ values, the network might automatically switch to activate higher-layer authentication services. Or, if an RRCC with moderate δ values is used, the network might only respond by issuing a message to the network administrator, warning of a potential network intrusion.

III. FORGE-RESISTANT RELATIONSHIPS VIA AUXILIARY FIELDS

There are several sources for forge-resistant relationships. We now look at two approaches that involve the use of auxiliary state fields contained in the packet to detect identity spoofing in wireless networks. We note that other strategies are possible, such as those that might arise from using specific signatures associated with the RF transmitter itself [15].

A. Anomaly Detection via Sequence Number Monotonicity

The first family of rules that we explore is based upon requiring packet sequence numbers to follow a monotonic relationship. This property is motivated by an observed behavior of the firmware of many 802.11 devices, and has been proposed for detecting spoofing [16]–[18]. In the discussion that follows, we shall formalize these earlier works by placing the sequence number method in the context of hypothesis testing, and deriving forge-resistance properties under different network conditions.

In 802.11 wireless networks (WLAN or ad hoc), before transmitting each packet, an 802.11 header is appended. One field of the 802.11 header is the sequence control field, which is inserted directly by the firmware into the header. There are two parts that comprise the sequence control field: 1) the fragmentation control and 2) the sequence number. The sequence number is a 12-bit field, providing sequence numbers with the range between 0 and 4095. Following each packet transmission, the firmware increments the sequence number field by 1. This monotonic relationship allows us to define a rule \mathcal{R}_{seq} . Except where noted, the remaining discussion regarding \mathcal{R}_{seq} shall focus on the case of a 12-bit sequence number field, as is used in 802.11. The extension to arbitrary cases is straightforward.

Before we discuss forge resistance, we will examine the behavior of the sequence number field for two important cases: 1) a single node using a specified MAC addresses transmitting packets to a receiver and 2) two nodes using the same MAC address (one spoofing the other) to transmit packets. We present the results of a simple experiment using 802.11 devices in Fig. 1, where the sequence number of consecutive received packets at

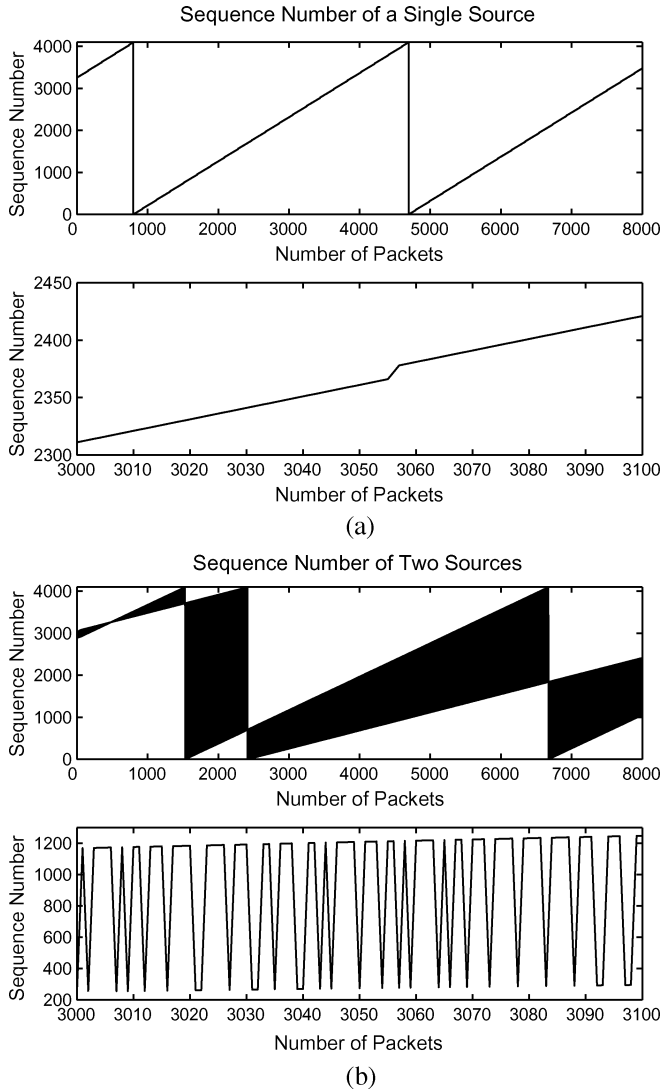


Fig. 1. Sequence number of consecutive received packets at a receiver in case of different number of sources. (a) Only one source is present in the network. (b) There are two sources sending out packets to the same receiver. The lower figures in parts (a) and (b) are the zoomed-in versions of the corresponding upper figures.

the receiver is presented for these two cases.¹ When there is only one source, the sequence number is a monotonic function of the number of packets sent, and wraps back to 0 after the sequence number reaches 4095, as shown in Fig. 1(a). We provide a magnified depiction of the monotonicity in the lower part of Fig. 1(a). The jump near packet number 3055 is due to packet loss. When there are two sources sending packets to the receiver, each device follows its own monotonic relationship, and the observed sequence number pattern is a mix of these two patterns, as shown in Fig. 1(b). The mixing results in a discontinuous trace, which is an anomalous behavior that we wish to detect.

The forge resistance of \mathcal{R}_{seq} is not dependent on whether another device can alter its sequence number. Although it is true that it is somewhat difficult for an adversary to set the sequence

¹We note that this behavior reflects both data and management packets. Even during transmission of management packets, the sequence number field is incremented. Additionally, we note that all sequence number experiments have retransmission disabled, as would correspond to a multicast/broadcast mode.

number field to an arbitrary value, it is possible for an adversary to bypass firmware and set an arbitrary value for the sequence number field. There have been several efforts recently to develop a software MAC layer for 802.11 that bypasses firmware restrictions and facilitates arbitrary packet generation [19], [20]. Instead, forge resistance follows from the fact that even if an adversary changes its sequence number based upon the last observed sequence number used by the legitimate device, the legitimate device will follow its own sequence and, hence, duplicate sequence numbers will be transmitted. As we will show, this duplication is detectable, and forge resistance follows from the fact that an adversary cannot stop the legitimate device from transmitting.

Our discussions will consider the following traffic patterns:

- Normal operations with packet loss. Under benign conditions, a single node will transmit a stream of packets, where an occasional packet will be lost due to packet loss. This is not considered anomalous traffic.
- Nonadaptive adversaries. One or more adversaries will blindly spoof the MAC address of a device without adjusting their sequence numbers. Highly discontinuous traces, such as those depicted in Fig. 1(b), are the result.

However, we note that more complex adversarial models are possible, where the adversary not only alters its MAC address, but also adapts its sequence number based on what it witnesses from the legitimate device. We note, however, that there is generally little advantage to the adversary in this case as, even if the adversary chooses to transmit a specific sequence number to pass a short term check, the legitimate device will eventually choose to transmit its own packet with that sequence number, thereby revealing an anomaly. Based on this observation, we focus our discussion on the two traffic cases before.

We now examine the normal operations with the packet-loss case. When there is only one source sending out packets to the receiver, the sequence numbers associated with the source's MAC address should increase by precisely one (modulo 4096) for consecutively received packets under ideal network conditions. In practice, packet loss will cause the increment between successively received packets to be more than 1. Further, there is no case where the increment should be 0. The likelihood of the difference being greater than 1 depends on the packet-loss rate of the link. Let $\{x_n, x_{n+1}, x_{n+2}, \dots\}$ denote the sequence numbers of consecutive received packets, and $x_n \in [0, 4095]$. We assume that the packet loss occurs with a rate p , and that packet loss is independent between successive packets. We now note that the difference $x_{n+1} - x_n = \tau$ will lie between 1 and 4096 (A difference of 0 is considered to be an anomalous behavior and, hence, mapped to 4096). Hence, the probability of $x_{n+1} - x_n = \tau$ is $(1-p)p^{\tau-1}/(1-p^{4096}) \approx (1-p)p^{\tau-1}$ for $1 \leq \tau \leq 4096$. The approximation $(1-p)p^{\tau-1}$ is extremely close when p^{4096} is very close to 0, which is valid for any link of interest (e.g., packet loss $p < 0.5$). The approximation $(1-p)p^{\tau-1}$ gives $E[\tau] = 1/(1-p)$, while the variance of τ is $\sigma_\tau^2 = E[\tau^2] - E[\tau]^2 = p/(1-p)^2$. By examining the formula for $E[\tau]$ and σ_τ^2 , it is clear that the mean and standard deviation are relatively small (compared to 4096), even for low-quality links. For example, for 50% packet loss, $E[\tau] = 2$ and $\sigma_\tau = 1.41$. In general, we can infer that, even for wireless networks with poor connectivity, the difference between successive packets will not be large.

We now turn to the case of one or more devices spoofing another's identity and discuss appropriate classification policies. We start with the nonadaptive adversaries case and look at the case of only one attacker. Let y denote the sequence number from the real source, and x the sequence number of the attacker. Assuming these two random variables are independent and uniformly distributed in $[0, 4095]$, then their difference $z = x - y$ follows a triangular distribution from -4095 to 4095 (the convolution of the distributions of x and y). If we consider the gap, $\tau = z \pmod{4096}$ and map a difference of 0 to 4096, then τ is a uniform distribution over $[1, 4096]$, and we have $E[\tau] = 2048.5$. In this case, the standard deviation is $\sigma = 1182$. Comparing the statistical behavior of the gap for the dual-source case with the gap for a single-source case, we see a large difference between the τ values, suggesting that the gap is a powerful statistical discriminant between normal network behavior and the behavior when the network is under a spoofing attack. Since τ is calculated using consecutive sequence numbers, the case of more than one adversary will appear similar to a dual-source case as it will involve at least two sources following their own sequence number progression. We may thus consider the multiple-source case as analogous to the dual-source case.

A natural question that arises is the issue: What if the adversarial device B transmitted first? Then, in this case, the detection rule will follow B 's sequence numbers, and when the legitimate device A finally communicates, the discontinuity will be detected. On the other hand, if the legitimate device never communicates, then the sequence number detector will track the adversarial sequence number and never identify an anomaly. Thus, anomalous traffic detection is only possible in the presence of heterogeneous sources: we must have the legitimate device transmit. This is a reasonable assumption since the legitimate device will at least transmit periodic control packets, such as beacons. For example, in 802.11 WLANs beacon frames are periodically broadcast by access points to facilitate AP discovery by client NICs.

We now turn to the issue of building a detector using sequence numbers, and quantify its probability of detection and false alarm. We shall first focus our discussion on the more general nonadaptive case. Using \mathcal{R}_{seq} , we may define an RRCC detection scheme as follows. Rather than operate strictly on two consecutive packets, the detection scheme should operate on a window of packets coming from a specific MAC address u_j . Our detector uses a window $\omega_j(k) = \{S_j(k), S_j(k-1), \dots, S_j(k-L+1)\}$, consisting of L consecutive sequence number state fields $S_j(k)$. The detector calculates the $L-1$ sequence number differences $\{\tau_1, \tau_2, \dots, \tau_{L-1}\}$, where $\tau_l = S_j(k-l+1) - S_j(k-l) \pmod{4096}$. A difference of 0 is considered to be an anomalous behavior and, hence, mapped to 4096. By using a window of data points, we may define a family of detectors with varying sensitivity levels. The basic detector determines that anomalous behavior has occurred if $\max_{l=1}^{L-1} \{\tau_l\} > \gamma$, where γ corresponds to a threshold that governs the probability of false alarm and missed detection. Alternative detector strategies might declare anomalous behavior if two or more differences τ_l are greater than a threshold.

Based on the probability calculations earlier and the window size L , we may calculate the ϵ and δ values for this RRCC. The null hypothesis \mathcal{H}_0 for this problem corresponds to a single

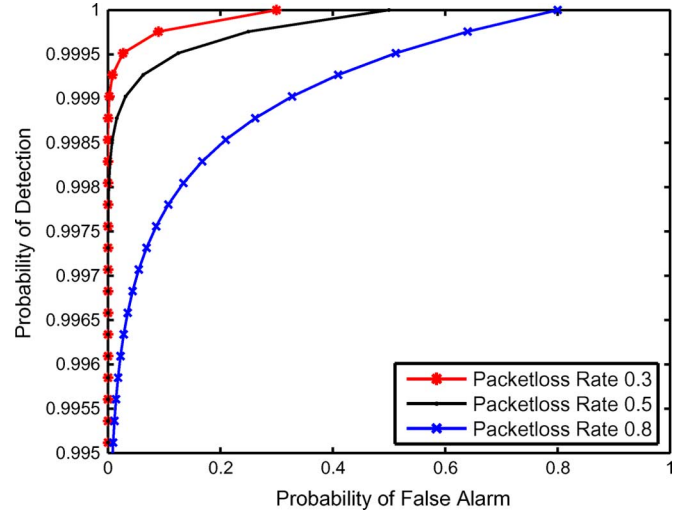


Fig. 2. Receiver operating curve for the sequence number monotonicity detector while $L = 2$.

source using a specific network identity. In order to calculate the probability of false alarm $P_{\text{FA}} = Pr(\mathcal{H}_1; \mathcal{H}_0)$, we must calculate the probability $Pr(\tau_l < \gamma; \mathcal{H}_0)$ for a threshold γ . Using the approximation, it is easy to see that $Pr(\tau_l < \gamma; \mathcal{H}_0) = (1 - p^\gamma)$. Since we assumed that packet loss applies independently to each packet transmitted, we obtain $\delta = P_{\text{FA}} = 1 - (1 - p^\gamma)^{L-1}$ for a window of size L . The calculation for $\epsilon = P_{\text{MD}} = Pr(\mathcal{H}_0; \mathcal{H}_1)$ involves calculating $Pr(\tau_l < \gamma; \mathcal{H}_1)$ for a threshold γ . As discussed earlier, under the adversarial hypothesis \mathcal{H}_1 , the distribution of τ_l is uniform over $[1, 4096]$ and, hence, $Pr(\tau_l < \gamma; \mathcal{H}_1) = (\gamma - 1)/4096$. The probability of missed detection is thus $\epsilon = Pr(\mathcal{H}_0; \mathcal{H}_1) = ((\gamma - 1)/4096)^{L-1}$. The choice of γ may be determined by setting a desired probability of a false alarm rate δ and solving γ from $\delta = 1 - (1 - p^\gamma)^{L-1}$. In order to illustrate the performance of the sequence number monotonicity detector, we plot the receiver operating characteristic (ROC) curves for $p \in \{0.3, 0.5, 0.8\}$ and for $L = 2$ in Fig. 2. In particular, we note that there is a rapid improvement in the probability of detection (i.e., $1 - \epsilon$) as packet loss p decreases. In fact, since most operational networks have a packet loss that is smaller than 0.1, we can infer that the probability of detection will be very close to 1. Additionally, by referring to the equations for ϵ , it is clear that increasing L also rapidly increases the probability of detection.

The monitor should not observe any out-of-order sequence number packets. However, throughout the experimental validation, we witnessed that roughly 0.01% of the time packets came out of order, and that these were always beacon packets yielding a gap 4095. The reason lies in the fact that the device driver tries its best to send out periodic beacons on time. The detection algorithm can be easily extended to include the out-of-order transmission situation by accepting an occasional gap of 4095 as a "single" source pattern. We now briefly discuss the performance of the detector in the presence of an adaptive adversary. We note that, for a window of packets, the adaptive adversary is likely to produce a cluster of sequence number gaps of 4096, which are easily detectable. As long as the adversary does not prevent the legitimate device from transmitting, the only factor that could prevent the detection of anomalous traffic in

the adaptive adversary model is the ambient packet loss. However, the packet loss will affect both the legitimate sender and the attacker, equally and unpredictably for both. This packet loss implies that the adversary cannot control which packets it sends will get through. Further, since the adversary cannot predict which source packet will successfully arrive, the adversary cannot do better than forging a packet with a sequence number that is one greater than the previous packet that was witnessed. This observation also implies that the adversary is most detectable if it is transmitting at roughly the same rate as the source and that, in order to avoid detection, the adversary should either inject very few packets, or flood the network with packets. For the first case, this has the effect of lessening the severity of the attack, while the second case would have the ultimate effect of making the adversary the only transmitter, thereby preventing the legitimate sender from transmitting. Later, in Section V, we shall examine a detector that quantifies the severity of a spoofing attack scenario by quantifying both the size of the sequence number gaps and the number of gaps present in a window.

B. One-Way Chain of Temporary Identifiers

In the previous section, the sequence numbers followed a publicly known pattern, and ultimately forge resistance required a guarantee that a legitimate device would frequently communicate. We now explore an alternative, where the state field is also a temporary identifier that changes for each transmitted packet. However, unlike the sequence number case, for this method, the temporary identifiers are difficult for the adversary to predict—the adversary must solve a cryptographic puzzle in order to figure out the next value of the temporary identifier. Thus, the puzzle must be solved before the adversary can transmit a single packet with a state field that would pass the RRCC. For every packet that the adversary wishes to send, he or she must solve another puzzle or randomly guess the state field value.

The advantage of this strategy is that it places an extra burden on the adversary in order to prevent him or her from injecting traffic. Even if the legitimate source does not communicate, the adversary must solve the cryptographic puzzle for each packet it wishes to create in order to not be detected.

In the one-way chain of temporary identifiers method, a temporary identifier field (TIF) serves as the state field for anomaly detection. For identity u_j , the relationship between TIFs $S_j(k)$ will follow a reverse one-way function chain F [10], [21]–[23]. The source first chooses a number n that is larger than the number of packets that will be sent during the total period of communication. The source chooses a final TIF $S_j(k)$, and then calculates earlier TIFs via the reverse one-way chain $S_j(k-1) = F(S_j(k))$ for $1 \leq k \leq n$.

Suppose that at an arbitrary time during the operation of the protocol, a monitoring node associates a TIF $S_j(k)$ with identity u_j . The next packet that the monitor receives which claims to have come from utility u_j has TIF S'_j . We would like to test to see whether S'_j could have produced $S_j(k)$ by the one-way function. Ideally, the verification process would check whether $S_j(k) = F(S'_j)$. If this fails, then the RRCC would declare that spoofing had occurred. However, in practice, there is packet loss, and this verification might fail for legitimate reasons. Therefore, similar to what is done in TESLA [22], [23], we use the loss-tolerant property of one-way chains. We

declare that spoofing has occurred if the test $S_j(k) = F^l(S'_j)$ fails for all $1 \leq l \leq L$, where $F^l(x) = F(F(\dots(x)\dots))$ is the l compositions of F applied to x . Under this strategy, we test the chain up to L times in case there was severe packet loss.

We now examine the issue of forge resistance. First, we examine the complexity associated with an adversary, successfully creating a valid sequence of L TIFS. In order to succeed in this task, the adversary uses a previous observed TIF value $S_j(k)$ and must solve for the L consecutive $S_j(k+1), S_j(k+2), \dots, S_j(k+L)$ by $S_j(k+j) = F^{-1}(S_j(k+j-1))$. Even though the function F is publicly known, its cryptographic one-way pseudorandom properties imply that an adversary must essentially “brute-force” search for the inverse. Suppose that $F : \{0, 1\}^b \rightarrow \{0, 1\}^b$ is a one-way pseudorandom function then, on average, finding a single inverse will require 2^{b-1} applications of F , and similarly finding L inverses will, on average, require $L2^{b-1}$ applications of F . Since, by definition, F is itself a polynomial (in b) time algorithm, the $L2^{b-1}$ factor dominates the adversary’s complexity. In particular, even modest values of b are sufficient. In practice, such pseudorandom functions can be created using encryption functions, such as AES.

Next, we examine the forge resistance of this method by calculating P_{MD} and P_{FA} assuming that the adversary does not have the computational capabilities to invert the one-way function. Let us suppose that S'_j has been witnessed. The case \mathcal{H}_0 is that S'_j is legitimate, while the case \mathcal{H}_1 is that S'_j is forged. The probability of missed detection $P_{MD} = Pr(\mathcal{H}_0; \mathcal{H}_1)$ is the probability that the adversary was able to make an S'_j such that $S_j(k) = F^l(S'_j)$ for at least one l . The probability $P_{FA} = Pr(\mathcal{H}_1; \mathcal{H}_0)$ corresponds to the likelihood that at least L consecutive legitimate packets were lost due to packet loss. P_{FA} may be calculated using the packet loss rate p to be $\delta = P_{FA} = p^L$. Calculating P_{MD} , however, involves the properties of F . Again, suppose that $F : \{0, 1\}^b \rightarrow \{0, 1\}^b$ is a one-way pseudorandom function mapping b input bits into b output bits. Following the properties of cryptographic one-way functions, with good pseudorandomness, the output distribution will be uniformly distributed over the 2^b possible outputs. Under this assumption, $\epsilon = P_{MD} = 1 - (1 - 2^{-b})^L$. Thus, we have an (ϵ, δ) RRCC that is parameterized by the network packet loss p , the bit-length b of the TIF, and the length L of the verification window.

We illustrate the performance of the TIF detector by examining ROC curves for $p \in \{0.3, 0.5, 0.8\}$ and for $b = 10$ and $b = 16$ in Fig. 3. From these curves, we see that the probability of false alarm increases as p increases, and that the probability of detection increases as b increases. The TIF detector achieves a high probability of detection with a small probability of false alarm. By comparing the ROC curves for TIF and the sequential detector, we see that even a small value of b can be quite powerful. Moving from $b = 10$ to $b = 16$ leads to the TIF ROC curves having better detection performance than the sequential ROC curves, assuming that the adversary does not conduct a brute force attack on the one-way function. In order to overcome a computational attack on the one-way function, we recommend $b > 64$.

Finally, we note that for our purpose, the initial TIF $S_j(0)$ need not be distributed to the monitoring nodes. If $S_j(0)$ is

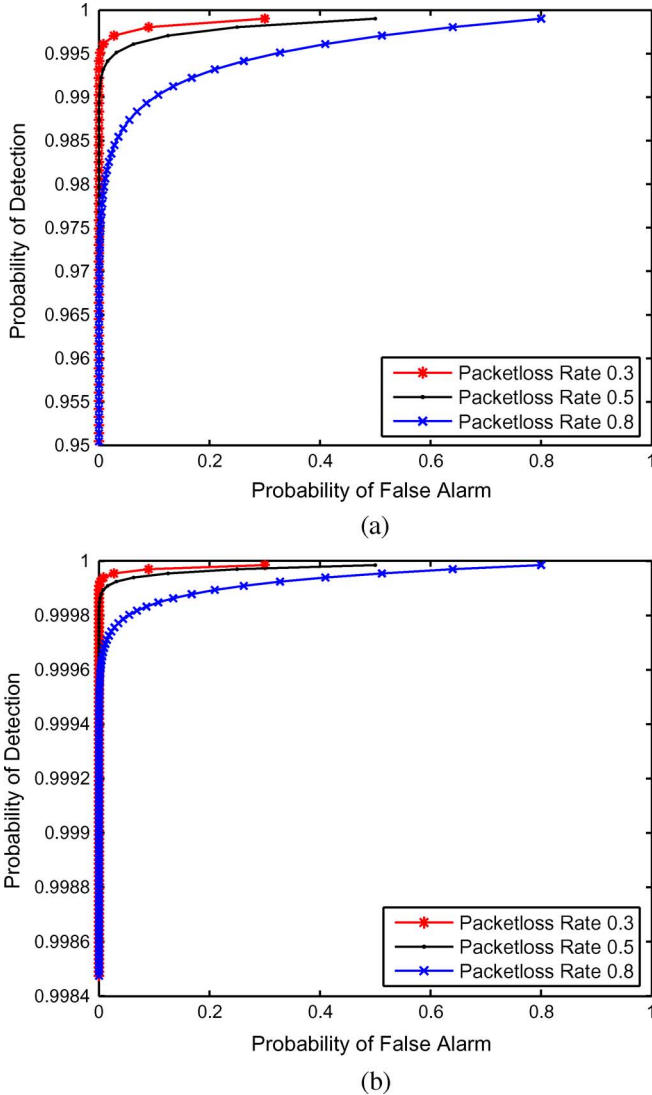


Fig. 3. (a) Receiver operating curve for a one-way chain of temporary identifiers detector with $b = 10$. (b) Receiver operating curve for a one-way chain of temporary identifiers detector with $b = 16$.

distributed to the monitor using an authenticated channel, then we will be able to perform entity authentication, similar to the strategy proposed in LHAP [8], [9]. This bootstrapping incurs extra overhead associated with authentication, and is contrary to our objective of detecting whether there is more than one source present. We emphasize that our objective is not conventional authentication, but rather to detect whether there is more than one source present. This means that if the adversary is spoofing another device's identity, and the other entity never communicates, then we will not detect this. However, if the legitimate device (or any other device attempting to spoof that identity) communicates, the TIF strategy will capture this anomalous traffic scenario without requiring authentication.

Finally, we discuss the storage overhead of TIFs. If the number of packets within a communication session is large, then the required one-way key chain could be long. One may employ efficient one-way function chain constructions to reduce the storage requirements [24]. Further, the sources may use a single TIF for a fixed amount of packets at a time. If an

adversary uses the same TIF for the transmission of its forged packets, the monitor can detect anomalous traffic by observing that the amount of packets sharing the same TIF is larger than the maximum amount allowed. The detection of anomalous traffic follows from the fact that an adversary cannot stop the legitimate device from using the same TIF. We note though that for network scenarios with packet loss, the receivers may receive fewer packets with the same TIF than the threshold amount, but this is a situation that an adversary cannot control or seek to exploit. In this case, the monitor can transition to the next TIF (even though the previous TIF may still be capable of supporting some more packets) by verifying the next TIF according to the one-way chain property.

IV. FORGE-RESISTANT RELATIONSHIPS VIA INTRINSIC PROPERTIES

In the previous section, we introduced an extra field into the packets to create relationships for anomaly detection. A potential drawback of those methods though is that they introduce additional communication overhead. In this section, we seek to exploit properties that are inherent in the transmission and reception of packets and, hence, do not require additional communication overhead.

A. Traffic Arrival Consistency Checks

Our basic strategy is to use traffic shaping to control the single-hop interarrival times observed by a monitoring device within the radio range of the source. The interarrival statistics are then used to discriminate between spoofing and nonspoofing scenarios.

Suppose we have a limited network scenario involving only two nodes A and B using the same network identity, and a monitoring device X that records the times at which it witnesses packets coming from that network identity. In particular, this network scenario does not involve any background traffic and, hence, the traffic that will be monitored by X is only a result of A and B 's behavior.

Now suppose, without loss of generality, that device A is the initial device using a particular network identity. For the traffic arrival consistency check, device A will send out packets such that the time between packets being transmitted follows a fixed distribution. For example, A might act as a Poisson source with rate λ or a constant rate (CBR) source with a fixed period. Throughout the operation of this consistency check, A will maintain this behavior. Regardless of A 's behavior, we assume the monitor X knows that the interarrival times τ follow a specified distribution $f_T(\tau)$.

Now, when the second source B starts communicating, a new distribution will be observed. The traffic arrival relationship \mathcal{R}_{tr} corresponds to the distribution $f_T(\tau)$ that X witnesses for A . When the RRCC corresponding to \mathcal{R}_{tr} is used, a test distribution $f_E(\tau)$ is measured and the objective is to decide whether $f_E(\tau)$ corresponds to $f_T(\tau)$ or not.

For the purpose of our discussion, we shall use the χ^2 -test for goodness of fit. We note, though, that other tests, such as the Kolmogorov-Smirnov test [25], are also suitable. Let us suppose the distribution $f_T(\tau)$ is divided into M adjacent intervals to yield the discrete distribution f_i for $1 \leq i \leq M$. The test distribution $f_E(\tau)$ will then be measured as a histogram using the same partitions. For n samples of the test distribution, we will

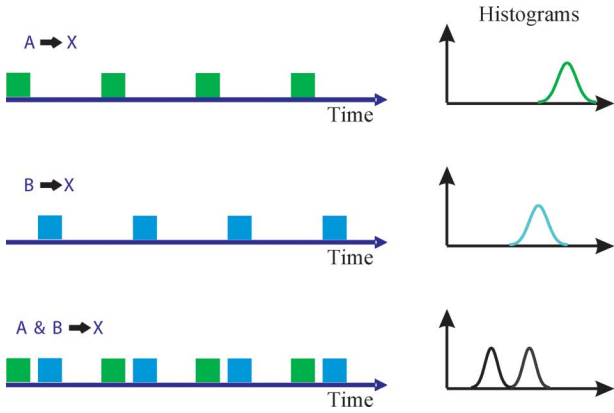


Fig. 4. Basic phenomena underlying traffic shaping and traffic relationship methods for anomaly detection. Here, A transmits with specific traffic distribution. When A and B both communicate, the resulting histogram will appear quite different from either distribution.

denote N_i to be the amount of occurrences of the test events in the i th partition and, thus, $N_1 + N_2 + \dots + N_M = n$. Then, the χ^2 -test statistic is

$$\chi^2 = \sum_{i=1}^M \frac{(N_i - n f_i)^2}{n f_i}. \quad (1)$$

Our null hypothesis for this problem \mathcal{H}_0 is that the N_i follows the same distribution as f_i . If the new observations N_i match the original distribution f_i , then we expect the test statistic χ^2 to be small. We reject \mathcal{H}_0 if $\chi^2 > \chi_{M-1, 1-\delta}^2$, where $\chi_{M-1, 1-\delta}^2$ is the $1 - \delta$ critical point for the chi-squared distribution with $(M - 1)$ degrees of freedom (DOF). This threshold yields a probability δ of false alarm (i.e., rejecting \mathcal{H}_0) when the measurements, in fact, do follow f_i .

We now explore the basic behavior that we can expect regarding \mathcal{R}_{tr} . In particular, suppose that A produces $f_A(\tau)$ at X . Then, if B starts communicating, it will increase the amount of traffic that X witnesses coming from that network identity. As a result, it should be expected that the interarrival times will decrease on average. One might expect that an anomalous $f_E(\tau)$ will be shifted to the left compared to $f_A(\tau)$. As we shall see later in the experimental section, we do witness an average decrease in the interarrival times. We have depicted this in Fig. 4. However, the characteristics of an anomalous $f_E(\tau)$ can exhibit complicated phenomena compared to $f_A(\tau)$. As in Fig. 4, the combination of two CBR sources can create a bimodal distribution.

The traffic arrival consistency check can be a very powerful method for detecting anomalous traffic. It should be emphasized, however, that this powerful detection method is possible because the device A follows a fixed traffic pattern. This necessitates that A performs traffic shaping and controls the time at which packets are transmitted. When higher-layer services supply packets to the network stack, it is necessary that the network stack perform traffic shaping at the various buffers internal to the network software in order to control the transmission rate of packets. Ultimately, this detection strategy is appropriate when delay-tolerant services are running on the wireless network.

B. Joint Traffic Load and Interarrival Time Detector

Merely using the traffic arrival statistics alone in deciding whether spoofing is present does not yield a reliable detector as there are other factors besides spoofing that can affect the traffic pattern. In particular, for wireless networks that employ carrier sensing as a basis for MAC, the presence of background traffic from other communicators can affect the observed traffic arrival pattern. Although we may attempt to use judicious buffering strategies to control the release of packets to conform to desired interarrival distribution, for carrier-sensing-based wireless networks, we are nonetheless forced to delay transmission until an existing transmission completes. This introduces extra delay to the observed interarrival times, and this amount of delay increases as the background traffic levels increase. Additionally, packet losses will also cause the traffic interarrival increase.

Therefore, using just the interarrival distribution alone is not suitable for detecting spoofing when there are large amounts of background traffic. Rather, we should take into account the level of background traffic and how this might affect the interarrival times. For this more general scenario, we instead propose jointly examining the average interarrival time and the background traffic load. The key observation here is that in the presence of no background traffic, we expect a specific average interarrival time, which can be directly determined from the original traffic distribution that A transmits, while this average interarrival time will increase as the background traffic levels increase. Suppose we define $\bar{\tau}$ to be the observed average interarrival time, and Λ to denote the observed traffic load. Following the methods of anomaly detection, we may partition $(\Lambda, \bar{\tau})$ space into two distinct regions. The first region, Region I, corresponds to normal system behavior (when the system is not under attack). Region I may be defined either by using formally derived specifications for normal system behavior (as is done in specification-based anomaly detection [26]) or empirically through a training phase (such as is done in network intrusion detection [27], [28]). In our work, we shall focus on empirical methods in which, during a training phase, $(\Lambda, \bar{\tau})$ data are collected from nonadversarial traffic scenarios. The region of normal system behavior may then be defined by binning the data according to Λ values and calculating a prediction interval for a specified percentile level for observed $\bar{\tau}$ data. For example, we may calculate the 99% prediction interval for the observed data. Region I is then defined as the region falling above the horizon connecting the percentile levels from different Λ bins. Region II is defined as the complement of Region I, and is the region that falls below Region I. Region II corresponds to non-normal system behavior and, thus, represents anomalous scenarios.

Later, in Section VI, we will revisit this classifier by presenting case studies where we illustrate how these regions may be defined through an example using experimental data.

V. ENHANCED DETECTION USING MULTILEVEL CLASSIFICATION

The previous two sections described two practical methods for detecting anomalous traffic. These methods were binary classifiers. In practice, however, it is desirable to attach a measurement of the severity of a threat with an assessment. For example, if the detector is to send a message to the network administrator, it is highly desirable (from the administrator's point

of view) that the message should measure the severity of the potential security breach, thereby allowing the administrator to prioritize and gauge his or her response. In this section, we will show how to take an RRCC and associate it with an additional measurement quantifying the severity of the violation. Due to space limitations, we present our multilevel classifier for only the \mathcal{R}_{seq} case and note that the extension to both of the traffic analysis methods of Section IV is straightforward.

To begin, we note that for \mathcal{R}_{seq} described earlier, using shorter time windows makes it harder to draw a conclusion about a long-term threat pattern (perhaps a single abnormal gap value is not considered to be a severe threat). Further, in the discussion of \mathcal{R}_{seq} , there was no discrimination based on the size of a sequence number gap: a gap of 10 is weighted the same as a gap of 1000, though clearly a gap of 1000 is a more anomalous event.

When assessing the severity of a threat, it is desirable to utilize the distribution of the observed gap statistics over a sufficiently long time window, and to weight larger gaps more heavily. In order to meet these two criteria, we propose the use of a weighted severity metric. In describing the weighted severity metric, suppose that the monitor has knowledge of the normal (single source for a single identity) distribution f_j for $j \in [1, 4096]$, corresponding to the sequence number gaps of devices in the ad-hoc network (e.g., this could be measured *a priori* empirically). During operation of the network, the monitor will observe the sequence number gaps for windows of L packets (here, L should be large enough to characterize the current traffic pattern). From this window, the monitor obtains the frequency for each possible gap value (i.e., q_j for $j \in [1, 4096]$). The weighted severity metric is defined as

$$D(f, q) = \sum_{j=1}^{4096} w_j |f_j - q_j| \quad (2)$$

where w_j is the weight for the j th sequence number gap. There are various ways to define w_j , and the definition should satisfy the following properties: 1) all w_j should be non-negative and 2) w_j should be monotonic with j , as a large sequence number gap should contribute more to the severity metric than a small sequence number gap. In this work, we define $w_j = -\log(p^{j-1}) = -(j-1)\log p$, where p is the average network packet-loss rate. Hence, a sequence number of gap 1, which indicates single source traffic, does not contribute to the severity metric, as $w_1 = 0$. It is clear that the severity metric not only describes the degree of the deviation of the gaps, but also the frequency of large deviations. A few occurrences of small sequence number gaps will not produce a large value, while either frequent small gaps or a few large gaps will yield larger severity values.

Using the severity metric, the network administrator may choose to define two or more classification regions. We now show how it is possible to define a multilevel classifier that classifies traffic into three different categories: benign, low threat, and severe threat. Our multilevel classifier uses an L -packet time window, and the severity $D = D(f, q)$ for that window. Under the normal (single-source) scenario, the average and standard deviation of sequence number gaps is small. For example, for a packet-loss rate of 50%, the average and standard

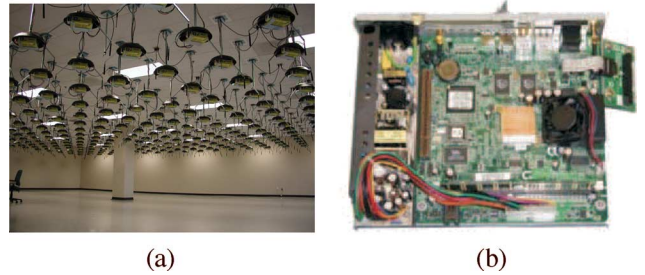


Fig. 5. (a) ORBIT wireless testbed. (b) ORBIT node.

deviation are 2 and 1.41, respectively. We thus can map out a benign region (Region I), corresponding to small D values. For this region, the detector concludes that no spoofing is present. For the dual-source case, the average and standard deviation of sequence number gaps were 2048.5 and 1182 and, hence, larger w_j values will factor into the calculation of D . We may thus define a severe-threat region, Region III, which corresponds to large D values. A D measurement falling in Region III would indicate significant amounts of spoofing activity and, hence, trigger a severe-threat alarm. Region II corresponds to an intermediate case (e.g., the adversary was not persistent and only spoofed for a short period of time) and falls in between Region I and Region III. In this case, a warning message would state that a low-threat violation occurred. The specification of the boundaries corresponds to selecting the threshold levels d_1 and d_2 , and depends on the link quality and the definition of w_j . These values may be adjusted according to the network administrator's definition of benign, low-threat, and high-threat events, and is a matter of the administration security policy. Later, in Section VI, we will explore a more detailed example of this scheme.

VI. EXPERIMENTAL VALIDATION ON THE ORBIT WIRELESS TESTBED

Although we have provided a theoretical framework for detecting anomalous traffic associated with device spoofing, we believe that it is necessary to evaluate wireless protocols in actual systems implementations. In particular, it is recognized that theoretical and simulation methods often fall short of capturing actual system behavior, especially for wireless networks [29], [30]. Notably, theory and simulations simplify assumptions about the physical layer and the complex interlayer interactions associated with wireless networking.

In this section, we evaluate our proposed methods using the ORBIT wireless testbed. The ORBIT testbed is an open-access research facility intended to provide a flexible platform for the evaluation of future wireless networking protocols, middleware, and applications [31]. ORBIT consists of a grid of wireless nodes, with the nodes separated by 1 m, as depicted in Fig. 5(a). Each node, depicted in Fig. 5(b), has a 1-GHz VIA C3 processor with 512-MB RAM, a 20-GB local hard disk, and two 802.11a/b/g interfaces. Data are collected via a library of measurement tools [32] and backhauled via Ethernet for databasing and storage. Currently, both Atheros and Intel 802.11a/b/g cards are available on ORBIT. In our experiments, it was not necessary to use the full testbed, and we selected a subset of the Atheros nodes in our tests.

In the discussions that follow, we have chosen not to include results on \mathcal{R}_{TIF} as the use of a cryptographic field in the \mathcal{R}_{TIF} method makes the challenge of detecting spoofing entirely dependent on an adversary's ability to invert the one-way function chain, and not dependent on any properties of the wireless network. Hence, the performance of \mathcal{R}_{TIF} can be completely evaluated using the pseudorandom properties of the underlying one-way function.

A. Validation of Detection Using Sequence Numbers

From the discussion in Section III, the average sequence number gap for consecutively received packets depends on the link packet-loss rate. Further, the average value and standard deviation are very small for packets from the same source, while there is large variation when more than one source uses the same identity.

We performed several experiments to validate the utility of sequence numbers for detecting more than one entity using a device. Our experiments consisted of two sources A (which acted as the legitimate source) and B (which acted as a nonadaptive spoofing source) that shared the same MAC address, and a monitoring device X . A was located at (3,2) on ORBIT while B was at (4,5) and X was at (5,4). In order to capture the effect of different link-quality conditions on ORBIT, we selected several of the other ORBIT nodes to act as background traffic sources. The background traffic sources used their own MAC addresses and only served to adjust packet loss. The first experiment we conducted was to quantify the mean difference between consecutive packets from the same MAC address when there was a single source A , and when there was spoofing present (i.e., B using the same MAC address as A). Our sources operated as CBR sources, emitting packets with an interarrival rate of 10 ms and a payload length of 1000 B. We varied the packet loss in the links by introducing additional background sources, ranging from 0 background sources to 15 background sources. Our background sources were also CBR sources with a period of 50 ms and payload of 1000 B. We collected the packet delivery statistics and the sequence number gap statistics and report the results in Table I. From this table, we see that, as we increase the level of background traffic, the packet-loss rate increases for both the single-source and dual-source experiments. When there is only a single source, the mean sequence number gap τ is small (less than 3) and the standard deviation σ of the sequence gap is also small (on the order of 0.5). This trend exists for the single-source case regardless of the background traffic level. In contrast, the dual-source case exhibits significantly higher τ and σ across all background traffic levels.

In particular, examining Table I and referring to the discussion in Section III-A shows that the general trend for theoretical and experimental values for $E[\tau]$ and σ_τ are comparable in terms of magnitude. For example, in Table I(a), when we increased the number of background sources to 4 to obtain a packet-loss rate of $p = 0.011$, we observed an experimental $E[\tau] = 1.1$, while theoretical results would yield $E[\tau] = 1.011$. For the standard deviation, we observed experimental values of $\sigma_\tau = 0.3$ while theoretical values would yield 0.106. Similarly, for the dual-source (adversarial) case, Table I(b) shows the dramatic increase in sequence number gap $E[\tau]$ and σ_τ when compared to the single-source case. This change was observed in

TABLE I
SEQUENCE NUMBER GAP STATISTICS FOR 802.11 SOURCES WITH VARYING LEVELS OF BACKGROUND TRAFFIC. (a) SINGLE-SOURCE CASE. (b) DUAL-SOURCE CASE

# background sources	0	1	4	7	15
Packetloss Rate	0	0.001	0.011	0.062	0.440
$E[\tau]$	1.0	1.1	1.1	1.2	2.0
σ_τ	0	0.2	0.3	0.5	0.2

(a)

# background sources	0	1	4	7	15
Packetloss Rate	0.011	0.030	0.060	0.080	0.498
$E[\tau]$	1955	1985	1916	1964	515
σ_τ	2037	393	1531	599	1083

(b)

TABLE II
SEQUENCE NUMBER GAP STATISTICS FOR THE DUAL-SOURCE CASE WITH TEN BACKGROUND SOURCES VERSUS THE ADVERSARY'S SENDING RATE

Time between packets sent by B	∞ msec	10msec	50msec	100msec	200msec
Packetloss Rate	0.018	0.048	0.037	0.036	0.002
$E[\tau]$	1.4	676	1878	1419	1007
σ_τ	0.5	1175	1140	1346	1340

the earlier theoretical results. We note that although the general trends between theory and experimental results are consistent, the mismatch between the precise values is expected. This mismatch arises as a result that the theoretical calculations are based on several independence assumptions: packet loss is independent across packets and further that the adversary's packets are sent independently of each other. Neither assumption can be controlled in a realistic experimental scenario and, hence, the comparison between theory and experimental values should not proceed past observing general trends.

The second experiment examined the effect of an increased adversarial rate of transmission on the sequence number gap. In this experiment A was a 10-ms CBR source, while we varied the period at which B sent packets. In the experiment, B varied from sending a 200-ms CBR source to a 10-ms CBR source. The average gap along with the standard deviation error bars are reported in Table II. With the exception of the ∞ millisecond case, the gap values are very large. It should be noted that the ∞ millisecond case corresponds to B being absent (single source only).

Taken together, these two experiments conclusively indicate that there is a large deviation in behavior between the behavior of the sequence number gap when there is a single source and when there are two devices sharing the same identity. This suggests that the sequence number field is very promising for detecting spoofing.

We built an anomalous traffic detection algorithm using the window of sequence numbers method described in Section III.

TABLE III
 P_{MD} AND P_{FA} FOR \mathcal{R}_{seq} RRCC WITH DIFFERENT THRESHOLDS γ WHEN A IS POISSON
 WITH INTERARRIVAL TIME 40 ms, B IS POISSON WITH INTERARRIVAL TIME λ , AND 15 BACKGROUND
 POISSON SOURCES OF AVERAGE INTERARRIVAL RATE 40 ms

λ for B	$L = 2$ packets				$L = 10$ packets			
	$\gamma = 4$		$\gamma = 6$		$\gamma = 4$		$\gamma = 6$	
	P_{MD}	P_{FA}	P_{MD}	P_{FA}	P_{MD}	P_{FA}	P_{MD}	P_{FA}
∞	0	0.001	0	0.0004	0	0.008	0	0.001
200	0.228	0.001	0.228	0.0008	0.018	0.001	0.019	0.0007
100	0.248	0.0001	0.249	0	0.013	0.0006	0.015	0
20	0.230	0	0.230	0	0.016	0	0.018	0
10	0.180	0.0001	0.180	0	0.032	0	0.032	0

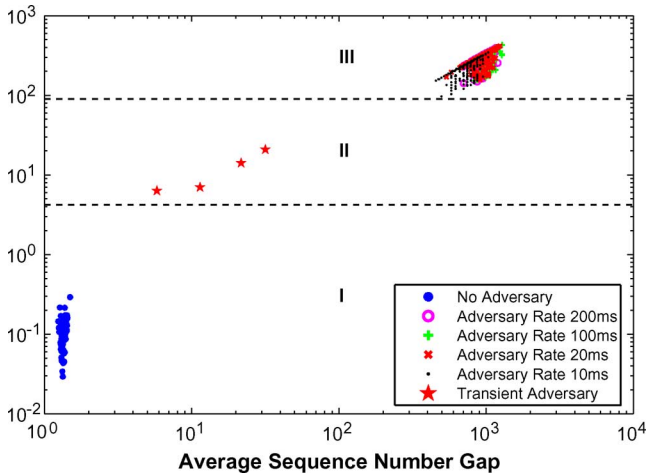


Fig. 6. Severity metric versus the average sequence number gap with a window size of $L = 100$ packets, when A is Poisson with an interarrival time of 40 ms. Several cases for the adversary B are presented: poisson with interarrival time ∞ (no adversary), 200, 100, 20, and 10 ms; and a transient adversary that interjects only a few packets. For all cases, there are 15 background Poisson sources of an average interarrival rate of 40 ms.

In this experiment, we looked at two different window sizes of $L = 2$ and $L = 10$ packets. In both cases, the source A was a Poisson source with an average interarrival time 40 ms. We introduced an adversary that acted as a Poisson source with an average interarrival time λ , and varied λ from ∞ (no adversary) to 10 ms to 200 ms. Additionally, we introduced 15 background sources on ORBIT to control packet loss. These background sources transmitted packets as Poisson sources, each with an interarrival rate of 40 ms. In all cases, the packets transmitted were 1000 B. We calculated P_D and P_{FA} for a threshold of $\gamma = 4$ and $\gamma = 6$, and present the results in Table III. The use of a longer window significantly reduces the probability of missed detection. In some cases, when a longer window is used, there is a slight increase in the likelihood of false alarm.

We next examined the multilevel classifier described in Section V. We processed the same set of data used in Table III with a window size of $L = 100$ packets. The packet loss p was estimated from the collected data for the single-source case and used to define w_j 's. In order to assist in the visualization of our data, we plot the severity versus the average sequence number gap for different adversarial rates in Fig. 6. In this figure, there

are three visible “clusters,” which will correspond to the three regions of Section V.

When there is no adversary, the severity metric is small, yielding values below 0.5 (and the average sequence number gap is 1.6). When two persistent sources exist, despite the adversary's rate, the severity metric significantly increases to 300 (here, the average gap is roughly 900), as shown in the upper right corner. Since our objective was to study a multilevel classifier, we next looked at the case of a “transient” adversary, one that only spoofed a small amount of packets in the time window. Specifically, we had an adversary B that would imitate A by randomly sending between 1 to 3 packets during each 100-packet window, with gap values from 450 to 1000 for these packets. For the transient adversary, the D values were between the values observed for the benign and full adversarial cases, as shown in the figure. We note that adjusting the amount of spoofed packets in a 100-packet window or the gap values would lead to larger or smaller D values that would continuously fall between the two extremes of benign and full adversarial cases.

In practice, the administrator should define the boundaries between these regions. To accomplish this, the regions may be defined heuristically, or based upon an organizationally defined security policy. A quantitative approach to defining these regions would involve a training phase, where benign and full adversarial cases are run, and the data are used to define Region I and Region III directly (Region II would follow directly as the region in between these two regions). For example, in the data presented in Fig. 6, we used the D -values to find the (upper) 99.9% prediction interval thresholds for no adversary case, and the (lower) 99.9% prediction interval thresholds for the full adversary cases. The resulting thresholds were $d_1 = 4.20$ and $d_2 = 90.13$ (calculated as the minimum of the 99.9% prediction intervals from each adversarial data set). We have indicated these thresholds using the dashed line in Fig. 6. We note that, for these thresholds, the transient adversary would result in a “low-threat” warning.

B. Validation of Detection Using Traffic Statistics

In order to validate the \mathcal{R}_{tr} method, we conducted an experiment on ORBIT where we controlled the rate at which data packets were released to the card at the sources, and witnessed

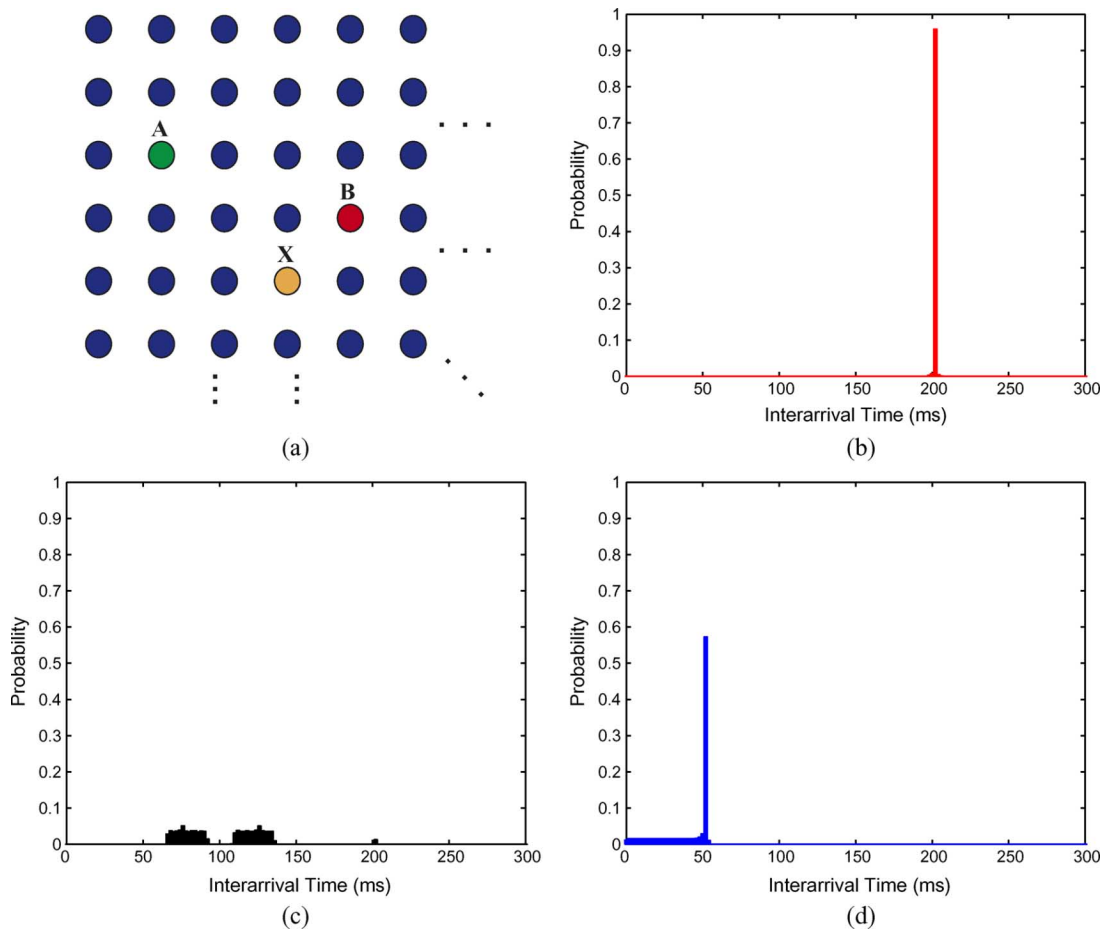


Fig. 7. (a) Experimental setup on the ORBIT grid, node separation is 1 m. Node A is (3,2) and B is (4,5). The monitor X is (5,4). (b) Interarrival time distribution when A communicates as a CBR with a packet rate of 200 ms to X . (c) Interarrival time distribution when both A and B communicate as a CBR with a packet rate of 200 ms. (d) Interarrival time distribution when A is a 200-ms constant rate source, and B is a CBR of 50 ms. All experiments had 1 background CBR source with a period of 2 ms.

the corresponding distributions at the monitoring node. The experimental layout of the traffic statistics tests was slightly different from ORBIT experiments used for \mathcal{R}_{seq} and is presented in Fig. 7(a). In this setup, we had two different source nodes, and one monitoring node. Node A was located at (3,2), node B at (4,5), while the monitor X was at (5,4). For these experiments, B is the adversary and spoofs A by using A 's MAC address.

In these experiments, we used A as the original source and had A release packets at a constant rate of 200 ms. In addition to our source, we had one background traffic source that transmitted as a CBR source with a period of 2 ms between packets. The background traffic did not use the same MAC address as A or the adversarial node B . The purpose of this background source was to provide a channel where occasional medium-access contention would occur. Our monitor X recorded the time at which packets were received and passed up the network stack. The time stamping was conducted at the MAC layer. These readings were used to calculate the interarrival times between packets, and the resulting histogram for the interarrival times when A transmitted with the background traffic is presented in Fig. 7(b).

We then looked at two dual-source cases. The first dual-source case involved node A communicating as a 200-ms source, while B also communicated as a 200-ms constant rate

source. The resulting histogram is presented in Fig. 7(c). The second dual-source case that we performed an experiment for involved A communicating as a 200-ms source, while B communicated as a 50-ms constant rate source. The histogram for the second dual-source case is presented in Fig. 7(d). By examining these histograms, it is clear that neither of the dual source cases looks similar to the single-source case. In the first dual-source case in Fig. 7(c), we see that the resulting distribution is bimodal. In order to see why this occurs, we refer the reader back to Fig. 4, where the time between an A and B packet is less than the time between a B and A packet. In Fig. 7(d), we do not see a bimodal distribution, but instead see a dominant mode at roughly a 50-ms interarrival time, and the rest of the probability spread out between 0 and 50 ms. This is due to the fact that the 50-ms rate is so much faster than the 200-ms rate, which causes only a few occurrences of an interarrival time of less than 50 ms.

Although it is clear that these two dual-source distributions do not look anything like the single-source distribution, we nonetheless performed the chi-squared test. We broke the data from the two dual-source cases into data windows of $n = 250$ data readings, and calculated the χ^2 statistic for each data window for each dual-source data set. We then calculated the mean and variance of the χ^2 statistics. For the first dual-source

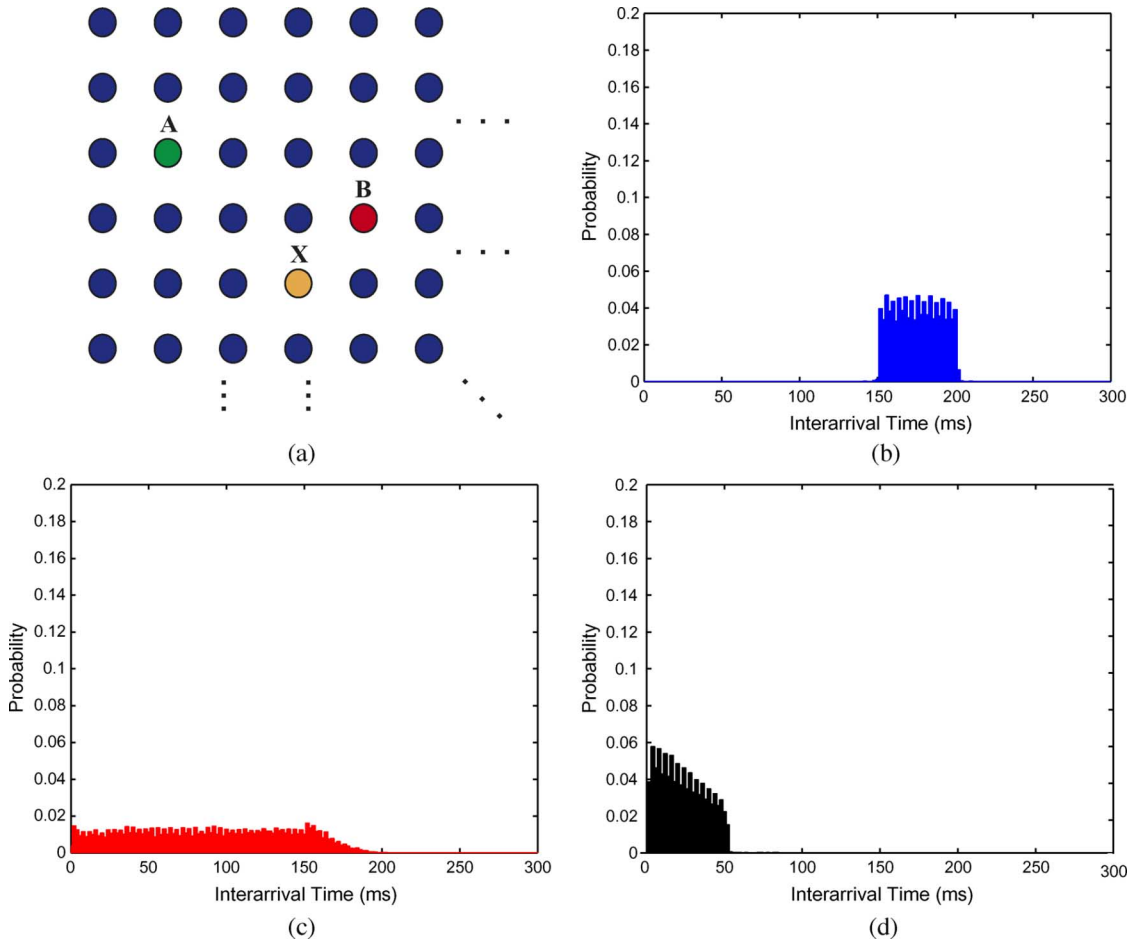


Fig. 8. (a) Experimental setup for uniform source A . (b) Interarrival time distribution observed at X when A is a $\text{Unif}(150, 200)$ msec source and 1 background CBR source with an interarrival time of 2 ms. (c) Interarrival time distribution observed at X when A is a $\text{Unif}(150, 200)$ -ms source, B is $\text{Unif}(150, 200)$ ms, and one background CBR source with an interarrival time of 2 ms. (d) Interarrival time distribution observed at X when A is a $\text{Unif}(150, 200)$ -ms source, B is $\text{Unif}(0, 50)$ ms, and one background CBR source with interarrival time 2 ms.

case, the average χ^2 statistic was $\bar{\chi}^2 = 139386$, and had a standard deviation of $\sigma_{\chi^2} = 2073$. The second dual-source case had an average χ^2 value of $\bar{\chi}^2 = 143073$, with a standard deviation of $\sigma_{\chi^2} = 6 \times 10^{-6}$. The small value for σ_{χ^2} for the second case is due to the fact that the dual-source distribution did not overlap the single-source distribution. For these tests, the 1% critical value was 9.21. Since the χ^2 statistics were significantly larger than the 1% critical value in both cases, we may numerically conclude that the detector would discriminate between single- and dual-source cases.

The second set of traffic experiments we conducted involved A acting as a source that transmitted packets with an interarrival time that was uniformly drawn from 150 to 200 ms. During these experiments, we had a single background traffic source operating as a CBR source with an interarrival time of 2 ms. The resulting histogram of interarrival times witnessed at X is presented in Fig. 8(b). We then performed two dual-source cases. The first involved an adversary B that transmitted packets with an interarrival time of $\text{Unif}(150, 200)$ ms, while the second case involved an adversary B that transmitted packets at an interarrival rate of $\text{Unif}(0, 50)$ msec. The resulting interarrival distribution observed at X is presented in Fig. 8(c) and (d). From these figures, it is again clear that there is significant deviation between the single-source and dual-source cases. To corroborate

this claim, we report the chi-squared test when we broke the data gathered into data windows of $n = 250$ records. The average χ^2 statistic for the first dual-source case was $\bar{\chi}^2 = 7535$, and had a standard deviation of $\sigma_{\chi^2} = 749$. The average χ^2 statistic for the second dual-source case was $\bar{\chi}^2 = 9973$, and had a standard deviation of $\sigma_{\chi^2} = 5 \times 10^{-12}$. The low standard deviation of the second dual-source experiment is due to the lack of overlap between the $A \rightarrow X$ and $A + B \rightarrow X$ distributions for this case. For both dual-source cases, the 1% critical value for the chi-squared test was 37.57 and, thus, conclusively show that the traffic discrimination test is quite powerful at differentiating between single- and dual-source cases under more arbitrary source distributions.

C. Validation of the Joint Traffic Arrival and Traffic Load Detector

The background traffic loads used in the previous set of experiments are relatively light, and we were interested in how traffic arrival statistics would change as we increased the levels of the background traffic. In order to validate our detector of Section IV-B, which jointly used a measurement of the local channel traffic load and the average interarrival times, we conducted a series of experiments on ORBIT where we controlled the rate at which the packets are released from

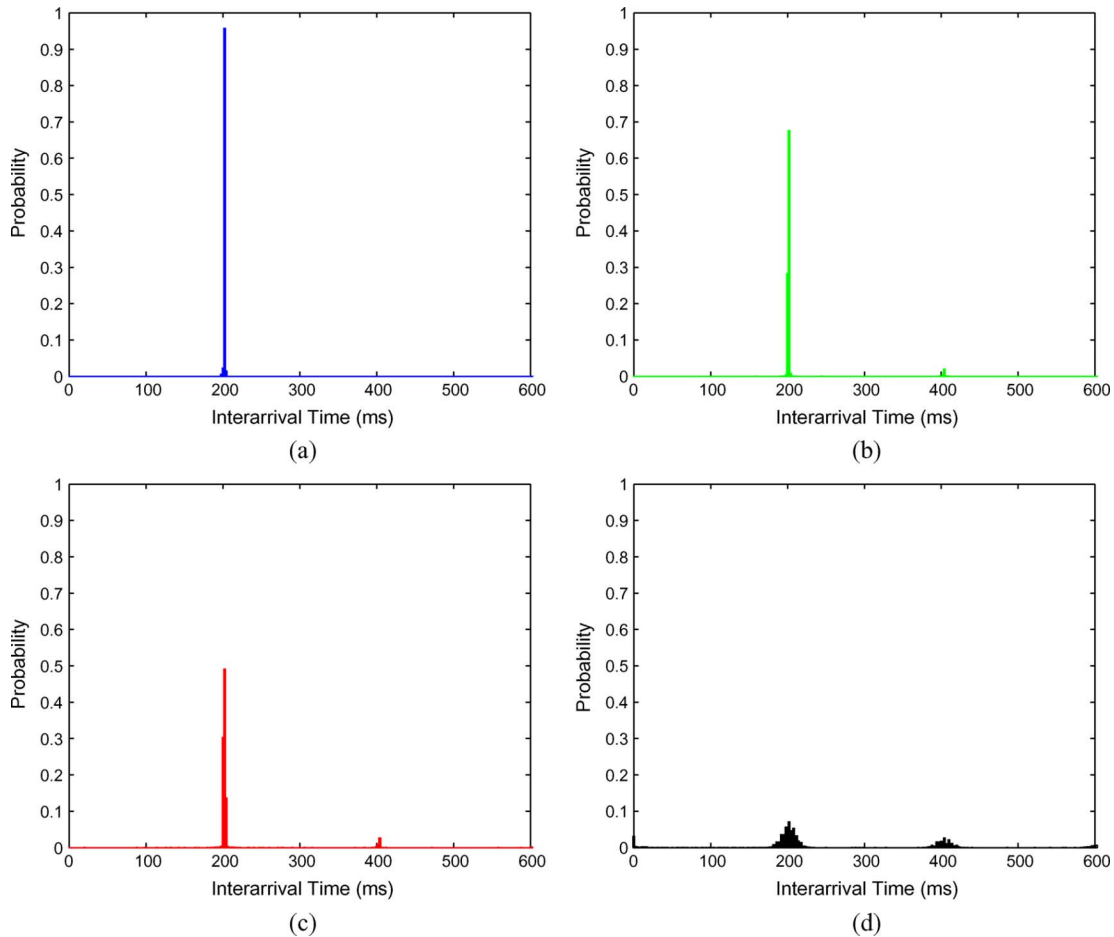


Fig. 9. (a) Interarrival time distribution when A communicates as a CBR with a packet rate of 200 ms to X . (b) Interarrival time distribution when A communicates as a CBR with packet rate 200 ms to X , with 1 background 2-ms CBR traffic source. (c) Interarrival time distribution when A communicates as a CBR with packet rate 200 ms to X , with two background 2-ms CBR traffic sources. (d) Interarrival time distribution when A communicates as a CBR with packet rate 200 ms to X , with four background 2-ms CBR traffic sources.

the sources. Our experimental layout was the same as that in Fig. 7(a), where node A was located at (3,2), node B at (4,5) (which claims the same MAC address with node A), and the monitor X was (5,4). Further, we varied the amount of background traffic sources up to a maximum of four sources (each background source used its own MAC address). It should be noted that the background sources used in this experiment more heavily occupy the channel than the background used in the experiments of Section IV-B. Typically, with most WLAN deployments, all nodes were within the radio range of every other node.

The first set of experiments was chosen to illustrate how the traffic interarrival distribution changes with the total amount of traffic load in the networks. In the first experiment, we used A as a source that released packets at a constant rate of 200 ms. The traffic interarrival distribution is shown in Fig. 9(a). In addition to the source, we added up to four background constant rate traffic sources, each transmitting packet has a constant rate of 2 ms into the network. The traffic interarrival distributions for packets from A with 1, 2, and 4 background sources are shown in Fig. 9(b)–(d), respectively. The network becomes more congested as more background traffic sources are put into the network. The packet-loss rates for 1, 2, and 4 background sources are 3%, 6%, and 45%, respectively. It is clear that the traffic in-

terarrival distribution of congested networks does not follow the distribution corresponding to no background traffic.

We conducted the same experiments for a uniform distribution source, where node A released packets as a uniform distribution of $\text{Unif}(150, 200)$ ms. Similarly, we collected the traffic interarrival distribution for different degrees of background traffic, as shown in Fig. 10. We drew the same conclusion that the traffic interarrival distribution for congested networks does not follow the distribution without any background sources even when no adversary is involved.

In order to validate the use of $(\Lambda, \bar{\tau})$ to differentiate between anomaly and congestion, we conducted a second series of experiments on the ORBIT wireless testbed. We used the same network topology as the first series of the experiments. We collected the total amount of traffic load and interarrival time of packets from node A 's MAC address. As before, we look at cases where the source is a CBR source with a rate of 200 ms, and is a uniform source with rate $\text{Unif}(150, 200)$ ms. In both experiments, we collected data for different levels of background traffic, ranging from lightweight background with a packet-loss rate of 0%, to a severely congested network scenario with a packet-loss rate of around 45%.

The total network traffic load in units of kilobits per second versus the average interarrival time in a window size of 40 s

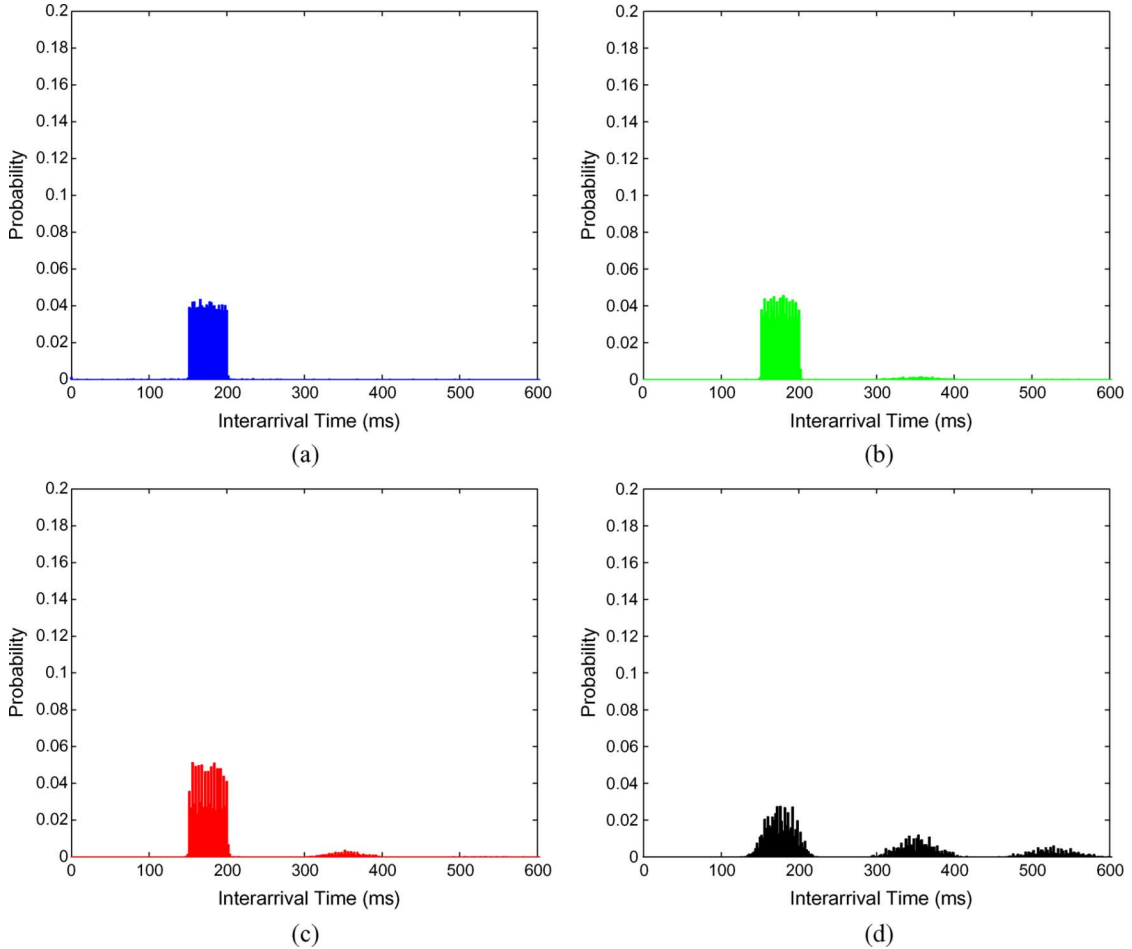


Fig. 10. (a) Interarrival time distribution when A communicates as a uniform with packet rate $\text{Unif}(150, 200)$ msec to X . (b) Interarrival time distribution when A communicates as a uniform with packet rate $\text{Unif}(150, 200)$ msec to X , with one background 2-ms CBR traffic source. (c) Interarrival time distribution when A communicates as a uniform with packet rate $\text{Unif}(150, 200)$ msec to X , with two background 2-ms CBR traffic sources. (d) Interarrival time distribution when A communicates as a uniform with packet rate $\text{Unif}(150, 200)$ msec to X , with four background 2-ms CBR traffic sources.

for the constant rate source is shown in Fig. 11. The comparison of the effect of the same amount of background traffic and anomalous traffic is shown in Fig. 11(a). The average interarrival time remains constant only when lightweight background traffic is present for nonspoofed scenarios. When the adversary sends anomalous traffic at the same rate as the background traffic sources, we observed roughly the same amount of total traffic load, but a smaller average interarrival time. When the network becomes increasingly congested, the total amount of traffic and average interarrival time increases for both spoofed and non-spoofed scenarios, as shown in Fig. 11(b). In all cases, the average interarrival time involving spoofing is always smaller than those of nonspoofed under the same traffic load. In these figures, we have used the 99.5% prediction interval for the observed benign data in order to define the boundary between Region I and Region II. We have depicted the depiction intervals in the figure for each cluster of benign data, and have connected the lower frontier of these regions using a dashed line. Region II is defined as the region that falls below this dashed line.

We also collected the total amount of network traffic load versus average interarrival time for the uniform source. The data are collected with a window size of 40 s and are shown in Fig. 12. Similar to the experiments for the CBR source, we

present the same $(\Lambda, \bar{\tau})$ diagrams in Fig. 12(a) and (b), where we have varied the traffic from light in Fig. 12(a) to heavier in Fig. 12(b). As before, we have divided the two-dimensional space into two regions using the 99.5% prediction intervals from the benign data. From both sets of experiments, we can conclude the validity of using $(\Lambda, \bar{\tau})$ to differentiate between anomalous (spoofed) traffic scenarios and congested traffic scenarios.

VII. CONCLUSION

In this paper, we have presented an alternative to traditional identity-oriented authentication methods for detecting device spoofing on a wireless network. Our strategy uses relationships that exist within a stream of packets coming from an individual network identity. Whenever an adversary spoofs a particular identity, the existence of multiple sources causes these relationships to be difficult for an adversary to forge. As a result, it becomes likely that the adversary will reveal its presence. We proposed two different families of relationships that are suitable for wireless networks. The first family involves introducing additional fields in transmitted packets, while the second family involves the implicit properties associated with the transmission and reception of packets. Specifically, we proposed the use of the monotonicity of the sequence number field and the use of

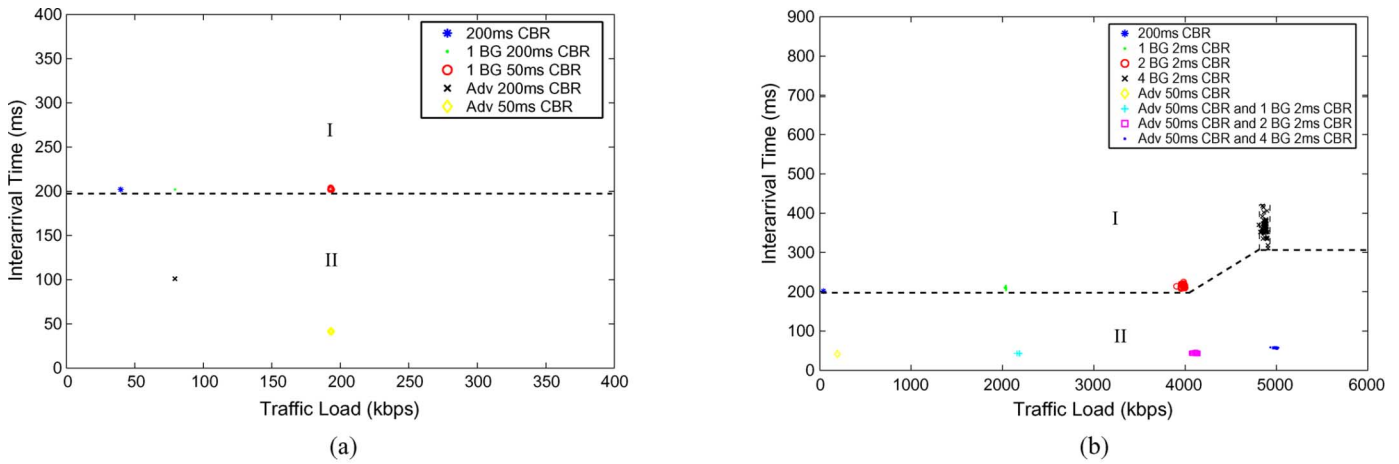


Fig. 11. (a) Total network traffic load in kilobits per second versus the average interarrival time for the constant rate source under light background traffic. (b) Total network traffic load in kilobits per second versus the average interarrival time for the constant rate source under medium to heavy background traffic. In both figures, we have defined the boundary between Regions I and II using the 99.5% prediction interval estimates.

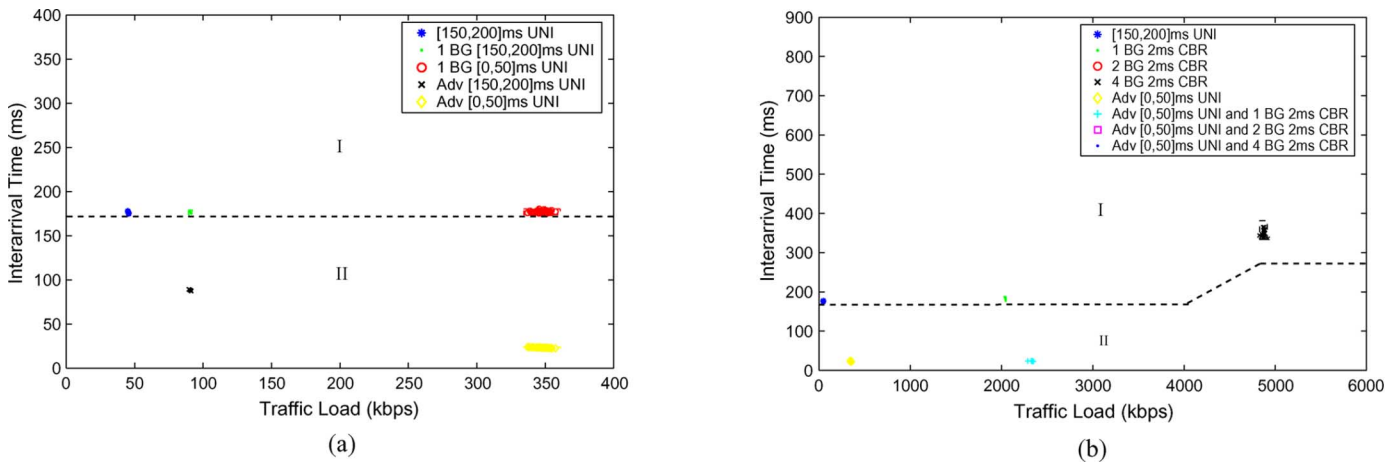


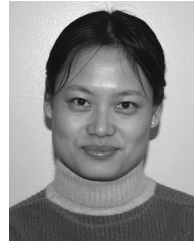
Fig. 12. (a) Total network traffic load in kilobits per second versus the average interarrival time for the uniform source under light background traffic. (b) Total network traffic load in kilobits per second versus the average interarrival time for the uniform source under medium to heavy background traffic. In both figures, we have defined the boundary between Regions I and II using the 99.5% prediction interval estimates.

temporary identifier fields that evolve according to a one-way function chain. Further, for relationships based upon implicit packet-related properties, we propose that traffic interarrival statistics may be used for detecting anomalous traffic scenarios. We illustrated how these RRCCs can be augmented with a measurement of the threat severity in order to facilitate multilevel classification. In all cases, we use these relationships to build forge-resistant consistency checks (RRCCs) to detect anomalous behavior. Our RRCC does not require the explicit use or establishment of cryptographic keys and, thus, RRCCs are suitable for application scenarios where the maintenance of keying material is not practical. We supported the validity of our proposed methods through experiments conducted on the ORBIT 802.11 wireless network testbed.

REFERENCES

- [1] J. Bellardo and S. Savage, "802.11 denial-of-service attacks: Real vulnerabilities and practical solutions," in *Proc. USENIX Security Symp.*, 2003, pp. 15–28.
- [2] A. Mishra and W. A. Arbaugh, An Initial Security Analysis of the IEEE 802.1x Standard Univ. Maryland, College Park, MD, Tech. Rep. CS-TR-4328, 2002.
- [3] A. Mishra, M. ho Shin, and W. A. Arbaugh, "Your 802.11 network has no clothes," *IEEE Commun. Mag.*, vol. 9, no. 6, pp. 44–51, Dec. 2002.
- [4] B. Wu, J. Wu, E. Fernandez, and S. Magliveras, "Secure and efficient key management in mobile ad hoc networks," presented at the 19th IEEE Int. Parallel Distributed Processing Symp., Denver, CO, 2005.
- [5] L. Zhou and Z. Haas, "Securing ad hoc networks," *IEEE Netw.*, vol. 13, no. 6, pp. 24–30, Nov./Dec. 1999.
- [6] A. Khalili and J. Katz, "Toward secure key distribution in truly ad-hoc networks," presented at the IEEE Workshop Security Assurance in Ad-Hoc Networks, 2003.
- [7] A. Wool, "Lightweight key management for IEEE 802.11 wireless lans with key refresh and host revocation," *ACM/Springer Wireless Networks*, vol. 11, no. 6, pp. 677–686, 2005.
- [8] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "LHAP: A lightweight hop-by-hop authentication protocol for ad-hoc networks," in *Proc. IEEE Int. Workshop on Mobile and Wireless Network*, May 2003, pp. 749–755.
- [9] S. Garg, N. Singh, and T. Tsai, "Schemes for enhancing the denial of service tolerance of SRTP," presented at the Int. Conf. Security and Privacy for Emerging Areas in Communication Networks, Athens, Greece, 2005.
- [10] M. Bohge and W. Trappe, "An authentication framework for hierarchical ad hoc sensor networks," in *Proc. ACM Workshop on Wireless Security*, 2003, pp. 79–87.

- [11] Tuomas Aura, Cryptographically generated addresses (CGA) RFC 3972, IETF, Mar. 2005.
- [12] J. Kempf, B. Sommerfeld, B. Zill, J. Arkko, and P. Nikander, Eds., Secure neighbor discovery (SEND) RFC 3971, IETF, 2005.
- [13] G. Montenegro and C. Castelluccia, Statistically unique and cryptographically verifiable 2002.
- [14] H. V. Poor, *An Introduction to Signal Detection and Estimation*, 2nd ed. New York: Springer Verlag, 1994.
- [15] Z. Li, W. Xu, R. Miller, and W. Trappe, "Securing wireless systems via lower layer enforcements," in *Proc. 5th ACM Workshop on Wireless Security*, 2006, pp. 33–42.
- [16] F. Guo and T. Chiueh, "Sequence number-based mac address spoof detection," presented at the 8th Int. Symp. Recent Advances in Intrusion Detection, Seattle, WA, 2005.
- [17] J. Wright, "Detecting wireless LAN MAC address spoofing," [Online]. Available: <http://www.home.jwu.edu/jwright/papers/wlan-mac-spoof.pdf>.
- [18] Q. Li and W. Trappe, "Relationship-based detection of spoofing-related anomalous traffic in ad hoc networks," presented at the IEEE Communications Society Conf. Sensor, Mesh Ad Hoc Communications Networks, 2006.
- [19] J. Fifield, T. Weingart, D. C. Sicker, C. Doerr, M. Neufeld, and D. Grunwald, "Multimac—An adaptive mac framework for dynamic radio networking," presented at the IEEE Int. Symp. New Frontiers in Dynamic Spectrum Access Networks, Baltimore, MD, Nov. 8–11, 2005.
- [20] A. Rao and I. Stoica, "An overlay mac layer for 802.11 networks," in *Proc. 3rd Int. Conf. Mobile Systems, Applications, Services*, 2005, pp. 135–148.
- [21] W. Trappe and L. C. Washington, *Introduction to Cryptography With Coding Theory*. Upper Saddle River, NJ: Prentice-Hall, 2002.
- [22] A. Perrig, R. Canetti, D. Song, J. D. Tygar, and B. Briscoe, TESLA: Multicast source authentication transform introduction IETF working draft, draft-ietf-msec-tesla-intro-01.txt.
- [23] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The TESLA broadcast authentication protocol," *RSA Cryptobytes*, vol. 5, no. 2, pp. 2–13, 2002.
- [24] Y. C. Hu, M. Jakobsson, and A. Perrig, "Efficient constructions for one-way hash chains," in *Appl. Cryptogr. Netw. Security*, 2005.
- [25] A. Law and W. D. Kelton, *Simulation Modeling and Analysis*, 2nd ed. New York: McGraw-Hill, 1991.
- [26] C. Ko, M. Ruschitzka, and K. Levitt, "Execution monitoring of security-critical programs in distributed systems: A specification-based approach," presented at the IEEE Symp. Security Privacy, 1997.
- [27] C. Taylor and J. Alves-Foss, "Nate-network analysis of anomalous traffic events, a low-cost approach," presented at the New Security Paradigms Workshop, Cloudfroft, NM, 2001.
- [28] W. Lee and S. Stolfo, "Data mining approaches for intrusion detection," presented at the USENIX Security Symp., New Orleans, LA, 1998.
- [29] K. Pawlikowski, H. D. J. Jeong, and J. S. R. Lee, "On credibility of simulation studies of telecommunication networks," *IEEE Commun. Mag.*, vol. 40, no. 1, pp. 132–139, Jan. 2002.
- [30] "NSF workshop on network research testbeds," [Online]. Available: http://www.net.cs.umass.edu/testbed_workshop. Chicago, IL, Oct. 2002.
- [31] "Orbit: Open access research testbed for next-generation wireless networks," [Online]. Available: <http://www.orbit-lab.org>.
- [32] M. Singh, M. Ott, I. Seskar, and P. Kamat, "ORBIT measurements framework and library (OML): Motivations, design, implementation and features," presented at the 1st Int. Conf. Testbeds and Research Infrastructures for the Development of Networks and Communities, Trento, Italy, Feb. 2005.



Qing Li received the B.S. degree in physics from Zhongshan University, Guangzhou, China, and the Ph.D. degree from the Wireless Information Network Laboratory (WINLAB) and the Electrical and Computer Engineering Department at Rutgers University, Piscataway, NJ.

Her research interests include wireless security, wireless networking, and network security.



Wade Trappe received the B.A. degree in mathematics from The University of Texas at Austin, Austin, TX, in 1994, and the Ph.D. degree in applied mathematics and scientific computing from the University of Maryland, College Park, in 2002.

Currently, he is an Associate Professor in the Electrical and Computer Engineering Department at Rutgers University, and is Associate Director of the Wireless Information Network Laboratory (WINLAB), Piscataway, NJ. His research interests include wireless security, wireless networking, multimedia security, and network security. He has developed several cross-layer security mechanisms for wireless networks, including jamming detection and jamming defense mechanisms for wireless networks, privacy-enhancing routing methods, cross-layer anomaly detection, and physical-layer security methods. He is a co-author of the textbook *Introduction to Cryptography with Coding Theory* (Prentice-Hall, 2001).