Active User-side Evil Twin Access Point Detection Using Statistical Techniques

Chao Yang, Yimin Song, and Guofei Gu, *Member, IEEE*

***Abstract*—In this paper, we consider the problem of "evil twin" attacks in wireless local area networks (WLANs). An evil twin is essentially a rogue (phishing) Wi-Fi access point (AP) that looks like a legitimate one (with the same SSID). It is set up by an adversary, who can eavesdrop on wireless communications of users' Internet access. Existing evil twin detection solutions are mostly for wireless network administrators to verify whether a given AP is in an authorized list or not, instead of for a wireless client to detect whether a given AP is authentic or evil. Such administrator-side solutions are limited, expensive, and not available for many scenarios. Thus, a lightweight and effective solution for these users is highly desired. In this work, we propose a novel user-side evil twin detection technique that outperforms traditional administrator-side detection methods in several aspects. Unlike previous approaches, our technique does not need a known authorized AP/host list, thus it is suitable for users to identify and avoid evil twins. Our technique does not strictly rely on training data of target wireless networks, nor depend on the types of wireless networks. We propose to exploit fundamental communication structures and properties of such evil twin attacks in wireless networks and to design new active, statistical and anomaly detection algorithms. Our preliminary evaluation in real-world widely deployed 802.11b and 802.11g wireless networks shows very promising results. We can identify evil twins with a very high detection rate while maintaining a very low false positive rate.**

***Index Terms*—Wireless Security, Rogue AP Detection, Evil Twin Attack.**

I. Introduction

WIRELESS networks are becoming extremely popular with the rapid advance of wireless LAN techniques and the wide deployment of Wi-Fi equipments. While users (especially smartphone users) can access Wi-Fi wireless internet "hotspot" connections in public more easily, they become to be more vulnerable to fraud and identity theft, referred to as Evil Twin attacks. Evil twin is a term for a rogue Wi-Fi access point that appears to be a legitimate one offered on the premises, but actually has been set up by a hacker to eavesdrop on wireless communications among Internet surfers [1]. Evil twin attacks, essentially a real-world wireless version of phishing scams, have been reported and studied by many security researchers [2], [3], [4], [5], [6]. Particularly, in 2011, security experts from Guardian launched two evil twin attacks conducted with volunteers, in which they successfully gather users' usernames, passwords, messages and even credit card information [7]. The fact that such attack can work is mainly because many public Wi-Fi hotspots have no (or weak) form of identification except their Wi-Fi names (SSID), which can be easily impersonated.

An evil twin attack is easy to launch. As illustrated in Fig. 1, by using specific readily-available software [8], an attacker can simply configure a laptop to be a rogue access point (AP) to mimic the legitimate access point used in a free public Wi-Fi area. Such areas could be restaurants (e.g., MacDonald's), cafes (e.g., Starbucks), airport lounges, student community areas or hotel lobbies. Then, by physically setting the evil twin AP near-by the target victims, the rogue AP can attract the victims' wireless connections, either through passively waiting or actively sending de-associate frames to force victims to change connections. Then, by simply relaying victims' network packets between the attacker's rogue AP and the legitimated AP, the attacker can both provide Internet access to victims and steal victims' personal information, without the need of creating additional network connection channels to the Internet such as wired channels or 3G. In this way, the rogue AP essentially works as an "evil twin" AP, (networking-topologically not necessarily physically) between victims and the legitimate AP.

Fig. 1. Illustration of launching an evil twin attack.

An evil twin attack is also easy to be successful. Since the "evil twin" AP is usually (physically) set closer to victims than the legitimate AP, the evil twin AP usually shows stronger wireless signal than the legitimate AP within the range of victims. Many end-users' laptops or smartphones may automatically connect to the "evil twin" AP with the highest signal strength among multiple APs associated with the same SSID. This is mainly because when the wireless card senses local available wireless networks, most operating systems of laptops or smartphones will choose to connect to the AP with the highest Received Signal Strength Indication (RSSI) [9] for each unique SSID, as these operating systems believe different APs with the same SSID belong to the same organization. In addition, such an attack can also be set to hard to trace, since the attacker can shut off the attacks suddenly or randomly after achieving the malicious goals, leading to a very short time to detect. Through successfully launching such evil twin attacks, the attacker can intercept sensitive data such as passwords or credit card information by snooping at the communication

Chao Yang, Yimin Song and Guofei Gu are with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX, 77840. E-mail: {yangchao,songym,guofei}@cse.tamu.edu
A preliminary (short) version of this paper appeared in DSN'10. This material is based upon work supported in part by the Texas Higher Education Coordinating Board under NHARP Grant no. 01909.

links, or launching man-in-the-middle attacks as shown in the real-world experiments [7]. The attacker can also manipulate DNS servers/communications, control the routing, and launch more severe phishing or other attacks. In short, evil twin attacks compose a serious threat to wireless LAN security.

Most existing evil twin detection solutions can be classified into two categories. The first kind of approaches [10], [11], [12], [13], [14], [15], [16], [17], [18], [19] monitor Radio Frequency (RF) airwaves and/or additional information gathered at routers/switches and then compare with a known authorized list. The second kind of approaches [20], [21], [22], [23], [24], [25], [26] monitor traffic at the wired side (a traffic aggregation point such as a gateway), and determine whether a machine uses wired or wireless connections. Such information is further compared with an authorization list to detect if the associated AP is a rogue one. These approaches are limited because they all require the knowledge of an authorization list of APs and/or users/hosts. We consider these solutions to be network administrator oriented, as opposed to user oriented. That is, they are designed for a wireless network administrator to perform authorization and access control policies for wireless APs/users. However, for a client user, it is of particular importance to be able to identify evil twins. For example, traveling users who use wireless networks at airports, hotels, or cafes need to protect themselves from evil twin attacks (instead of relying on those wireless network providers, which typically may not provide strong security monitoring/management service). In addition, to protect wireless security, IEEE 802.1x protocol is also designed to provide a secured authentication mechanism for the wireless devices to connect to a LAN or WLAN. However, 802.1x needs an trustable authentication server to authorize the wireless devices, which may not be practical or convenient for the huge amount of traveling users to detect evil twin attacks by themselves in the most of current public areas. Thus, a lightweight, effective and user-side solution for these client users is highly desired but is currently missing.

In this paper, we propose a novel user-side evil twin detection technique which has the following advantages compared to a traditional administrator-side solution: (i) Our technique does not require a known authorized AP/user list; (ii) An end user can be warned of an evil twin attack immediately to prevent being exposed to the attacker, even when the attack may last for a short time and a typical administrator-side solution may not help that much; (iii) From the user side, the parameters in a detection system can be customized according to local environment which may lead to a more accurate result; (iv) The user-side detection is resource-saving. The system can be activated only once when the users are trying to connect a new wireless AP. In addition, there is no need to modify the network architecture or any client- or server-side applications.

Our technique exploits the fundamental communication structure and properties of an evil twin attack: an evil twin typically still requires the good twin for Internet access. That is, an evil twin sits in the middle of the victim host and the good twin to relay communications. Thus, the wireless hops for a user to access Internet are actually increased (from one to two). In contrast, although legitimate wireless providers may use wireless bridges to extend the coverage, they do not change the single hop physical layer wireless channel to the users. According to the 802.11 protocol, we also observe that as long as the attacker obeys the TCP protocol and the IEEE 802.11 standard, the increased delays introduced by one additional wireless hop cannot be neglected. Based on these observations, we design new, active, statistical and anomaly detection algorithms to detect evil twins by differentiating the wireless hops (one or two hops). In addition, we consider the effect of throughput variance due to wireless network saturation and different RSSI ranges. We propose two algorithms: one is named Trained Mean Matching (TMM), requiring training knowledge of one-hop and two-hop wireless channels; and the other one is named Hop Differentiating Technique (HDT), which does not rely on any training information or knowledge. We apply these algorithms in the form of Sequential Probability Ratio Test (SPRT) [27].

In short, our paper makes the following contributions:

- We propose a new lightweight user-side evil twin detection solution. Our technique does not rely on "fingerprint" checking of suspect devices nor require a known authorized AP/host list. Thus, this solution is particularly attractive to traveling users.
- We propose to exploit the intrinsic communication structure and property of evil twin attacks. Furthermore, we propose two statistical anomaly detection algorithms for evil twin detection, TMM and HDT. In particular, our HDT improves TMM by removing the training requirement. HDT is resistant to the environment change such as network saturation and RSSI fluctuation.
- We implement our techniques in a prototype system, ETSniffer (Evil Twin sniffer). We have extensively evaluated ETsniffer in several real-world wireless networks, including both 802.11b and 802.11g. Our evaluation results show that ETSniffer can detect an evil twin quickly and with high accuracy (a high detection rate and a low false positive rate).

II. Related Work

Existing rogue AP detection solutions can be mainly classified into two categories. The first approach monitors RF airwaves and/or additional information gathered at routers/switches and then compares with a known authorized list. For example, AirDefense [10], similar to several other studies [11], [12], [13], [14], [15], [16], [17], [18], [19], scans RF from the Intranet APs to locate suspicious ones, and then compares specific "fingerprints" of the RF with an authorized list to verify. Specifically, for the scanning part, some studies [16], [17], [18] rely on sensors instead of sniffers to scan the RF, and some studies such as [19] propose a method to turn existing desktop computers into wireless sniffers to improve the efficiency. For verification, these studies verify MAC addresses, SSID, and/or location information of the AP by using an authorized list. However, these studies still have the risk of falsely claiming a normal neighbor AP as a rogue AP with a high probability. To solve this problem, they need to further verify whether such a rogue AP is indeed in the internal network. For example, Beyah's work [28] uses a verifier to send packets to the wireless side. If

such packets are received by the internal sensor, the associated AP is internal and thus an Evil Twin.

The second approach of rogue AP detection, proposed in [20], [21], [22], [23], [24], [25], [26], [29], detects evil twins by differentiating whether clients come from wireless networks or wired networks, relying on the differences in diverse network protocols. If a client comes from a wireless network while it is not authorized to do so (comparing with an authorized list), the AP attached to this host is considered as a rogue AP. [24], [23], [25] use some statistical features of the traffic time ([25] relies on the entropy, [24] relies on the median and the entropy, and [23] relies on the mean) to make decisions. [30] detect rogue AP by analyzing the TCP-ACK pairs in their mathematical model. [26] treats different ranges of a TCP connection separately. [31] relies on the RTT sent to hosts to distinguish WLAN, and it takes some traffic factors into consideration to increase the precision. [32], [33] rely on the frequent rate adaptation in the wireless network to distinguish it with wired networks. However, this line of work should solve the problem of falsely claiming an authorized wireless user who connects to Intranet with wireless networks. Thus, they may still need to further verify a wireless device is an authentic AP or not with some "fingerprint" from authorized lists. Two hybrid studies provide the fingerprint comparisons in the integrated systems [34], [35].

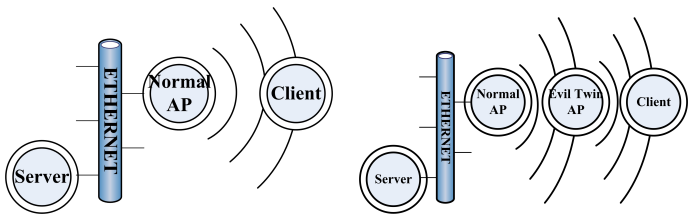
Recently, Jana et. al. focus on the similar threat models as the ones depicted in our work [2]. This work utilizes the fact that different APs usually have different clock skew to detect unauthorized wireless access points. However, this work still utilizes the "fingerprint" technique, which needs a whitelist of authorized access points. Han et. al. [3], [29] utilize time interval information to detect rouge APs. Specifically, it calculates the round trip time (RTT) between the user and the DNS server to independently determine whether an AP is legitimate or not without assistance from the WLAN operator. Since this work mainly utilizes the training detection technique and uses a relatively static threshold to differentiate normal and malicious scenarios, it needs to pre-collect the information of the target wireless network. Thus, such learning-based approaches highly depend on the knowledge of the target wireless network. They could not be effectively applied to those traveling users at the client side, since once the traveling users are in different areas, the network situation may have significantly changes.

Our work, ETSniffer, is very different from existing approaches. Firstly, unlike those administrator-oriented approaches highly relying on a whitelist of authorized APs/hosts, our approach is a *client-side* one, which does not require the knowledge of an authorized AP/host list. Secondly, unlike [3], [29] merely based on the learning knowledge, our work designs two different algorithms (*a learning-free algorithm and a learning-free algorithm*) to detect evil twin attacks. For the learning-based algorithm, we theoretically obtain the threshold from the intrinsic WLAN properties rather than using relatively static and empirical values, through exploiting fundamental communication structures and properties in the evil twin scenario. In addition, we also utilize SPRT technique to tolerate reasonable noise. Thirdly, our work is somehow motivated by Wei et. al. [20], which mainly aims at detecting rogue APs by using statistical methods (SPRT) to check whether the network packets go through a wireless channel or not. However, our work designs two algorithms to detect evil twin attacks, based on *two different wireless network statistics* and our analyses of *intrinsic wireless network properties*, with the considerations of dynamic changes of *real-world wireless network parameters*. Also, unlike [20] designed as a server-side approach, our work is a client-side approach suitable for traveling users. Thus, our work is a good supplement to such lines of approaches.

III. Problem Statement

The goal of our work is to detect evil twin attacks in real time under real wireless network environments. In our targeted evil twin attacks, the evil twin AP pretends to be a legitimate one to allure victims to connect and utilizes the legitimate AP to relay users' network packets to the Internet. As described in Section I, this kind of evil twin attacks are very practical and easy to launch in many public free Wi-Fi hotspots [36], [4], [7] and have caught security researchers' eyes [3]. Under such evil twin attacks, attackers can easily steal victims' information without the need of creating additional network connection channels to the Internet. Also, we clearly acknowledge the limitation of our work on detecting other types of man-in-the-middle attacks, which will be discussed in Section VII.

Next, we briefly describe the topological difference between the normal AP scenario and evil twin AP scenario under our target evil twin attacks, which provides the intuition of our detection approach. As illustrated in Fig. 2(a), under the normal AP scenario, a user communicates with the remote (DNS/Web) server through the normal AP (a one-hop wireless channel); on the other hand, as illustrated in Fig. 2(b), under the evil twin AP scenario, the victim client communicates with the remote server through an evil twin AP *and* a normal AP (a two-hop wireless channel). Thus, compared with the normal AP scenario, the evil twin AP scenario has one more wireless hop. This observation gives us the intuition to detect evil twin attacks by differentiating one-hop and two-hop wireless channels from the user-side to the remote server.



(a) Normal AP scenario (b) Evil twin AP scenario

Fig. 2. Illustration of normal scenarios and evil twin scenarios.

To achieve our research goal, we have to answer the following four questions: (1) What statistics can be used to effectively distinguish one-hop and two-hop wireless channels on the user side? (2) Are there any dynamic factors in a real network environment that can affect such statistics? (3) How to design robust and efficient detection algorithms with the considerations of these influencing factors? (4) What kinds of remote servers can be utilized to measure those statistics? Next, we provide a high-level description about our solutions to these questions (Details refer to Section IV and V).

For the first question, we choose Inter-packet Arrival Time (IAT) as the detection statistic. IAT is the time interval between two consecutive data packets sent from the same device (the remote server or the connected AP) to the client. In order to compute IAT more effectively and accurately, we adopt a new ACK-packet sending policy – Immediate-ACK policy[1] in our detection software ETSniffer, i.e., it immediately acknowledges every data packet received and the server sends the next data packet only when receiving an acknowledge for the previous one. We achieve this through setting a specified value of Maximum Segment Size (MSS) in the TCP header in the initial packets sent from ETSniffer, as explained in Section VI-A. Note that ETSniffer does not need to change any existing network protocols, software, or hardware on either client or server side. This policy does not affect normal network communications under default TCP protocol. Instead, it only affects the specific probing sessions initialized by ETSniffer for the detection, lasting for a very tiny time period.

Then, under the immediate-ACK policy, if ETSniffer receives two consecutive data packets P_1 and P_2, and sends corresponding TCP layer ACK packets A_1 and A_2. Then, on the client side the packet sequence is in an order of $P_1A_1P_2A_2$. If we let T_{P_1} and T_{P_2} be the time when the client receives P_1 and P_2, respectively, then the IAT of this pair of inter-packets can be computed as $T_{P_2} - T_{P_1}$.

For the second question, in a real wireless network environment, two main factors will affect IAT: Received Signal Strength Indication (RSSI) [9] and wireless network saturation. In wireless networks, RSSI fluctuates due to the multi-path and fading effects of the radio signal propagation. Since most wireless network cards have a transmission rate adaptation mechanism to adjust to different RSSI levels, the fluctuation of RSSI directly influences the actual available wireless bandwidth, causing the fluctuation of IAT. In addition, wireless network saturation is another influencing factor. When multiple devices synchronously attempt to send packets to the same AP, the medium access collisions emerge and spur the phenomenon of network saturation. This phenomenon stochastically increases the time for transmitting packets from a client to the AP. Specifically, according to CSMA/CD mechanism, the collisions set the exponential back-off time and account for an additional distributed inter-frame spacing (DIFS) [39] time and a short inter-frame spacing (SIFS) [39] time. Previous work such as [40], [41] shows that the throughput decreases with the increased number of the wireless clients, leading to larger IAT.

For the third question, we develop two new algorithms: Trained Mean Matching (TMM) and Hop Differentiating Technique (HDT). Both algorithms utilize the wireless IAT network statistic, consider the influencing factors of RSSI and saturation, and employ Sequential Probability Ratio Test (SPRT) technique to make the final detection.

For the last question, since we need a remote server to calculate IATs, it is important to figure out how to select those servers. Typically, we recommend the following three types of remote servers. First, the remote servers can be DNS servers as used in existing work [29]. It can be applied in many network environments, because we can easily find DNS servers in most networks due to the performance reasons to resolve domain names [42]. Also, DNS servers are usually set by administrators and trustable. Also, the time spent on the wired network to the local DNS server is usually small, which will decrease the effect of the wired link to our approach. Second, the remote servers can be web servers hosting local portal websites. Since at most public free Wi-Fi hotspots (e.g, hotels), the trusted AP will have (or connect to a local sever that has) a portal webpage to show some introductions for customs, this kind of servers near to the trusted APs will usually be maintained by administrators to provide HTTP service and trustable. Third, the remote servers can also be public trustworthy servers such as "google.com". These public servers are relatively stable and usually have long lifespan. More discussions refer to Section VII.

[1]It is different from delayed-ACK policy in wireless networks, in which a receiver sends a TCP layer ACK packet after receiving two continuous packets or after the delayed-ACK timer is triggered [37], [38].

IV. Server IAT Analysis

A. Theoretic Analysis of Server IAT

In this section, we show our theoretic analysis of Server IAT (IAT computed by the data packets sent from the server) and further demonstrate that Server IAT can be used to differentiate one-hop wireless channels and two-hop wireless channels, and thus it can be used to detect evil twin attacks.

Based on IEEE 802.11 standard [30], we denote the mathematic variances in our detection model, as summarized in TABLE I.

TABLE I
VARIABLES AND SETTINGS IN OUR MODEL

Protocol	802.11b	802.11g
B_W	11MBps	54MBps
B_E	100MBps	100MBps
W_0	32	16
T_{DIFS}	$50\mu s$	$28\mu s$
T_{SIFS}	$10\mu s$	$10\mu s$
$L_{ACK(MAC)}$	278Bytes	278Bytes
$L_{ACK(TCP)}$	338Bytes	338Bytes
L_P	402Bytes	402Bytes
$L_{\tilde{P}}$	375Bytes	375Bytes

As we consider both influencing factors (RSSI and network saturation), to better describe our model, we define the special wireless network environment with a perfect signal strength (RSSI = 100%) and no wireless collisions as "an ideal network environment". Let ΔA_S and $\widetilde{\Delta} A_S$ be one Server IAT under the "real network environment" and the "ideal network environment", respectively. B_W and B_E denote the bandwidth of the wireless network and Ethernet, respectively. Let ρ denote the bandwidth occupancy of Ethernet. W_0 is the initial contention window size. T_{DIFS} is one DIFS time and T_{SIFS} is one SIFS time. T_{BF} denotes the back-off time which follows a uniform distribution in terms of the contention window size. $L_{ACK(MAC)}$ and $L_{ACK(TCP)}$ are the size of an ACK-packet in the MAC layer and in the TCP layer, respectively. L_P denotes the size of one data packet that the client receives

and $L_{\widetilde{P}}$ is the average packet size on the Internet, which is usually between 300 and 400 bytes [43].

Then, based on IEEE 802.11 standard and our settings, we can show that the mean of ΔA_S is theoretically differentiable between the normal AP scenario and the evil twin scenario.

Theorem 1. *If we denote $E(\widetilde{\Delta}A_S)_{one-hop}$ as the mean of Server IAT $\widetilde{\Delta}A_S$ in a one-hop wireless channel, then in the normal AP scenario, we can obtain*

$$E(\widetilde{\Delta}A_S)_{one-hop} = 2T_{DIFS} + T_{SIFS} + 2E(T_{BF}) + \frac{L_{ACK(MAC)} + L_{ACK(TCP)} + L_P}{B_W} + E(T_{MAX})$$

where, $T_{MAX} = max(T_{SIFS} + \frac{L_{ACK(MAC)}}{B_W}, \frac{L_{ACK(TCP)}+L_P}{B_E} + T_{ex})$ *and* $E(T_{ex}) = \frac{\rho}{2(1-\rho)} \cdot \frac{L_{\widetilde{P}}}{B_E}$

Proof: In the normal AP scenario, considering the procedure that the client receives two consecutive data packets P_1 and P_2 from the remote server and it sends ACK packets A_1 and A_2 correspondingly, we show the analysis in Fig 3.

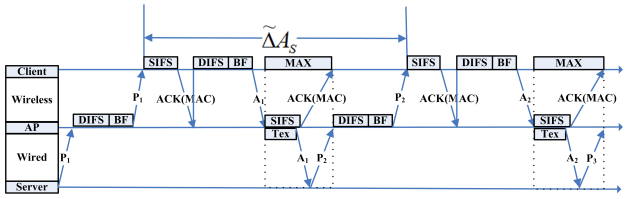


Fig. 3. Illustration of Server IAT for the normal AP scenario (one-hop wireless channel) in an ideal network environment.

When A_1 arrives at the AP, the AP will wait for one T_{SIFS} time and then send an ACK-packet in the MAC layer back to the client. Since in the Ethernet part, packets from other traffic may occupy the wired link, the AP will have to use some extra time to finish transmitting A_1, compared with that under the maximal bandwidth. We denote this extra time as T_{ex}. Similar to [20], we also model the incoming data packets to the server as M/D/1 queues. Based on the M/D/1 queue theory, we can obtain $E(T_{ex}) = \frac{\rho}{2(1-\rho)} * \frac{L_{\widetilde{P}}}{B_E}$.

After receiving A_1, the server will send P_2 to the AP. If the AP has not finished sending the ACK packet in the MAC layer to the client, the AP could not begin to send P_2 to the client. Thus, after A_1 arrives at the AP from the client, the AP will have to use T_{MAX} time to begin to prepare to send P_2 to the client, where $T_{MAX} = max(T_{SIFS} + \frac{L_{ACK(MAC)}}{B_W}, \frac{L_{ACK(TCP)}+L_P}{B_E} + T_{ex})$.

Thus, from Fig. 3, we can obtain that

$$\begin{aligned}\widetilde{\Delta}A_S &= T_{P_2} - T_{P_1} \\ &= T_{SIFS} + \frac{L_{ACK(MAC)}}{B_W} + T_{DIFS} + T_{BF} \\ &\quad + \frac{L_{ACK(TCP)}}{B_W} + T_{MAX} + T_{DIFS} + T_{BF} + \frac{L_P}{B_W} \\ &= 2T_{DIFS} + T_{SIFS} + 2T_{BF} \\ &\quad + \frac{L_{ACK(MAC)} + L_{ACK(TCP)} + L_P}{B_W} + T_{MAX}\end{aligned}$$

Thus,

$$E(\widetilde{\Delta}A_S)_{one-hop} = 2T_{DIFS} + T_{SIFS} + 2E(T_{BF}) + \frac{L_{ACK(MAC)} + L_{ACK(TCP)} + L_P}{B_W} + E(T_{MAX})$$

■

Theorem 2. *If we denote $E(\tilde{\Delta}A_S)_{two-hop}$ as the mean of Server IAT $\tilde{\Delta}A_S$ in a two-hop wireless channel, then in the evil twin AP scenario, we can obtain*

$$E(\widetilde{\Delta}A_S)_{two-hop} = 4T_{DIFS} + T_{SIFS} + 4E(T_{BF}) + \frac{L_{ACK(MAC)} + 2L_{ACK(TCP)} + 2L_P}{B_W} + E(T_{MAX})$$

where, $T_{MAX} = max(T_{SIFS} + \frac{L_{ACK(MAC)}}{B_W}, \frac{L_{ACK(TCP)}+L_P}{B_E} + T_{ex})$ *and* $E(T_{ex}) = \frac{\rho}{2(1-\rho)} \cdot \frac{L_{\widetilde{P}}}{B_E}$

Proof: Similar to the proof of Theorem 1, based on the Protocol 802.11, we show the analysis of the evil twin AP scenario in Fig. 4.

Thus, from Fig. 4, we can obtain that

$$\begin{aligned}\widetilde{\Delta}A_S &= T_{P_2} - T_{P_1} \\ &= T_{SIFS} + \frac{L_{ACK(MAC)}}{B_W} + T_{DIFS} + T_{BF} + \frac{L_P}{B_W} \\ &\quad + T_{DIFS} + T_{BF} + \frac{L_P}{B_W} + T_{MAX} + T_{DIFS} \\ &\quad + T_{BF} + \frac{L_{ACK(TCP)}}{B_W} + T_{DIFS} \\ &\quad + T_{BF} + \frac{L_{ACK(TCP)}}{B_W} \\ &= 4T_{DIFS} + T_{SIFS} + 4T_{BF} \\ &\quad + \frac{L_{ACK(MAC)} + 2L_{ACK(TCP)} + 2L_P}{B_W} + T_{MAX}\end{aligned}$$

Thus,

$$E(\widetilde{\Delta}A_S)_{two-hop} = 4T_{DIFS} + T_{SIFS} + 4E(T_{BF}) + \frac{L_{ACK(MAC)} + 2L_{ACK(TCP)} + 2L_P}{B_W} + E(T_{MAX})$$

■

From Theorem 1 and 2, if we let $E(\tilde{\Delta}_S)$ be the difference of $E(\tilde{\Delta}A_S)_{one-hop}$ and $E(\tilde{\Delta}A_S)_{two-hop}$, then

$$\begin{aligned}E(\tilde{\Delta}_S) &= E(\tilde{\Delta}A_S)_{two-hop} - E(\tilde{\Delta}A_S)_{one-hop} \\ &= 2T_{DIFS} + 2E(T_{BF}) + \frac{L_{ACK(TCP)} + L_P}{B_W}\end{aligned}$$

Under the real network environment, either the decrease of RSSI or the increment of wireless collisions can increase Server IAT, causing the distribution of Server IAT to become unstable compared to that under the ideal network environment. However, in the evil twin AP Scenario, there are two wireless hops leading to a larger probability of increasing Server IAT than that of the normal AP scenario. Therefore, if we let $E(\Delta A_S)_{one-hop}$ and $E(\Delta A_S)_{two-hop}$ as two means

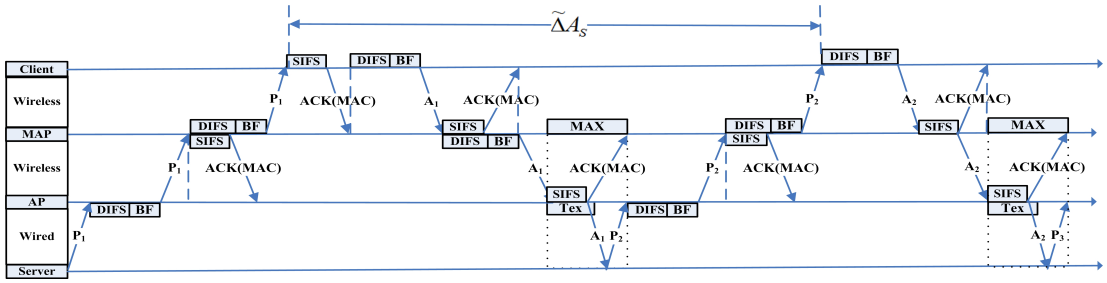


Fig. 4. Server IAT illustration in the evil twin AP scenario (two-hop wireless channel) in an ideal network environment.

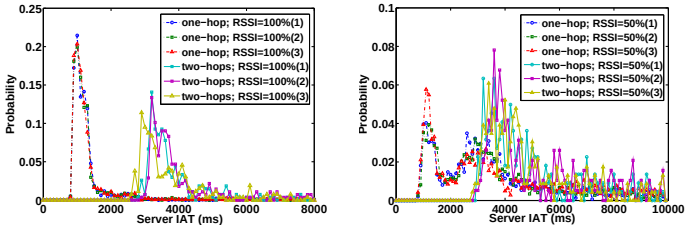
of ΔA_S in one-hop and two-hop wireless channels, under the real network environment, we can obtain,

$$E(\Delta_S) = E(\Delta A_S)_{two-hop} - E(\Delta A_S)_{one-hop} \approx E(\tilde{\Delta}_S)$$
$$= 2T_{DIFS} + 2E(T_{BF}) + \frac{L_{ACK(TCP)} + L_P}{B_W}$$

From the above equation, according to 802.11 protocol, we can find that even the attacker can maintain the evil twin AP to have a high bandwidth (e.g., 54MBps) with the victim and with the legitimate AP to decrease the time spent on transmitting data packets, as long as the legitimate AP provides relatively normal bandwidth, the difference of the mean of ΔA_S between the evil twin AP scenario and the normal AP scenario will be around $2T_{DIFS} + 2E(T_{BF})$. That is mainly because such a difference is generated due to the time spent on waiting for transmitting the packets for the additional wireless hop according to the 802.11 protocol, rather than merely due to the time spent on transmitting the packets for the additional wireless hop. Thus, this observation can be used to detect evil twin attacks.

B. Practical Validation of Server IAT

We next show our experimental results to validate whether Server IAT is indeed a suitable and effective statistic to differentiate between one-hop and two-hop wireless channels.



(a) RSSI = 100% (b) RSSI=50%

Fig. 5. IAT distribution in one-hop and two-hop wireless channels.

To minimize data bias, for both one-hop and two-hop wireless situations, we build our datasets under real network environments at three different times. We compute Server IAT in one-hop and two-hop wireless channels by collecting the packets under the conditions of RSSI[2] = 100 and 50. The results are shown in Fig. 5. We can see that the distribution of Server IAT remains stable when RSSI is 100%. The two means of IAT in one-hop and two-hop wireless channels are about 1,300ms and 3,300ms, respectively. The gap of these two means is obvious. Even though a weak signal strength (e.g., RSSI at 50%) will lead a less stable distribution of Server IATs, this gap can still be observed.

[2]Since the users can only obtain the RSSI level for the first hop, the RSSI here refers to the first hop. In the two-hop scenario, we set the value of the RSSI in the second hop within the range from 80% to 100%.

V. Detection Algorithm

Based on our theoretical analysis and practical validation in the previous section, we present two algorithms to detect evil twin attacks: Trained Mean Matching (TMM) and Hop Differentiating Technique (HDT). Both algorithms utilize the Sequential Probability Ratio Test (SPRT) technique [27]. TMM algorithm requires knowing the distribution of Server IAT as a priori (trained) knowledge. However, the HDT algorithm does not have such a requirement. Instead, it is directly based on theoretical analysis. Thus, it is more suitable for scenarios where the distribution of IAT is either unknown, instable, or unable to be (perfectly) trained.

A. Trained Mean Matching Algorithm

1) TMM Algorithm Description: We have demonstrated that the distributions of Server IAT in one-hop and two-hop wireless channels differ significantly. According to this observation, in this section, we develop a detection algorithm named Trained Mean Matching (TMM). Specifically, given a sequence of observed Server IATs, if the mean of these Server IATs has a higher likelihood of matching the trained mean of two-hop wireless channels, we conclude that the client uses two wireless network hops to communicate with the remote server indicating a likely evil twin attack, and vice versa.

In the training phase, we adopt a quadratic-mean technique to train a detection threshold. First, we collect Server IAT in both one-hop and two-hop wireless channels. Then, we compute the mean and the standard deviation of Server IAT collected in the one-hop (normal AP) scenario, denoted as $\mu_{1,NAP}$ and $\sigma_{1,NAP}$, respectively. Then, we filter out the Server IATs beyond the range $[\mu_{1,NAP} - \sigma_{1,NAP}, \mu_{1,NAP} + \sigma_{1,NAP}]$. Next, we derive the second mean using the residual Server IAT, denoted as $\mu_{2,NAP}$. Similarly, we can obtain the second mean of Server IAT in the two-hop (evil twin AP) scenario, denoted as $\mu_{2,EAP}$. We compute the average of $\mu_{2,NAP}$ and $\mu_{2,EAP}$, as T_θ, set as the boundary to differ one-hop and two-hop Server IAT. In addition, in order to use the SPRT technique, we obtain two probabilities of one Server IAT in these two scenarios exceeding the trained threshold, denoted as P_1 and P_2, by

computing the percentage of collected Server IATs deviating from T_θ in the normal and evil twin AP scenario, respectively.

In the detection phase, given a sequence of Server IAT observations, represented by $\{\delta\}_{i=1}^{n}$, we use a binary random variable γ_i to denote whether the ith observed Server IAT belongs to evil twin AP scenario or not. Specifically, if $\delta_i \geq T_\theta$, then $\gamma_i = 1$, indicating an estimated evil twin AP scenario; otherwise, $\gamma_i = 0$, indicating an estimated normal AP scenario. Thus, we get a sequence of $\{\gamma\}_{i=1}^{n}$. Let H_1 be the hypothesis that it belongs to an evil twin AP scenario and H_0 be the hypothesis that it belongs to a normal AP scenario. We denote $P(\gamma_i = 1|H_1) = \theta_1$ and $P(\gamma_i = 1|H_0) = \theta_0$. According to the training data, we can set $\theta_0 = P_1$ and $\theta_1 = P_2$. We can compute the log-likelihood ratio Λ_n with the assumption that the Server IAT observations are i.i.d. (independent and identically-distributed) using the following formula:

$$\Lambda_n = \ln \frac{Pr(\gamma_1 \dots \gamma_n|H_1)}{Pr(\gamma_1 \dots \gamma_n|H_0)} = \ln \frac{\prod_{i=1}^{n} Pr(\gamma_i|H_1)}{\prod_{i=1}^{n} Pr(\gamma_i|H_0)} = \sum_{i=1}^{n} \ln \frac{Pr(\gamma_i|H_1)}{Pr(\gamma_i|H_0)}$$

According to SPRT [27], we perform a threshold random walk to calculate the log-likelihood ratio. The walk starts from zero. If $\gamma_i = 1$, then it goes up with a length of $\ln(\theta_1) - \ln(\theta_0)$; if $\gamma_i = 0$, then it goes down with a length of $\ln(1-\theta_1) - \ln(1-\theta_0)$. We define every random walk as one decision round. Let us denote f_p and f_n as the user-chosen false positive rate and false negative rate, respectively. If the random walk reaches the upper boundary $B = \ln(1-f_n) - \ln f_p$, we report evil twin AP scenario; if it reaches the lower boundary $A = \ln f_n - \ln(1-f_p)$, we report normal AP scenario; otherwise, it is pending and we watch for the next decision round. The details of TMM algorithm can be seen in Appendix A.

2) Discussions of TMM Algorithm: Based on the training technique, the TMM algorithm affords an effective approach to detect evil twin attacks. However, in some cases, it is too time-consuming or impractical for a normal user to acquire a priori knowledge, particularly the training data for two-hop wireless channels. In addition, the trained knowledge in one wireless network can be hardly directly applicable to another network. These limitations motivate us to design an effective and practical non-training-based algorithm to detect evil twin attacks – Hop Differentiating Technique (HDT).

B. Hop Differentiating Technique

In HDT algorithm, instead of using IAT, we adopt another metric – the ratio of a Server IAT to an AP IAT. We define it as SAIR (Server-to-AP IAT Ratio). Next, we theoretically prove that it can be used to robustly detect evil twin attacks.

1) Theoretic Analysis of SAIR: Before illustrating our theoretical analysis of SAIR, we first make three reasonable assumptions: (1) The wireless network environment does not change dramatically, which implies a relatively steady RSSI and collision number at least during the period when we collect one pair of Server IAT and AP IAT to compute a SAIR; (2) In the evil twin AP scenario, the RSSI and the level of network saturation between the victim client and the evil twin AP are not worse than that between the victim and the normal AP. (3) The Ethernet network is not severely congested.

For the first assumption, since the time cost during collecting one pair of Server IAT and AP IAT is in seconds, it is reasonable to assume the wireless network environment does not change dramatically during such a short time interval. For the second one, since the attacker wants to successfully allure victim clients to connect with the evil twin AP, it is more likely for the attacker to provide a better RSSI and a smaller wireless collision probability. For the last one, if there is a severe network congestion in the Ethernet, few people would choose the normal AP to surf the Internet.

Next, we introduce some variables to better describe our model. Let ΔA_A be the AP IAT and α be the SAIR, under the real network environment. Let $\tilde{\Delta} A_A$ be the AP IAT and $\tilde{\alpha}$ be the SAIR under the ideal network environment. Then, we can obtain

$$\alpha = \frac{\Delta A_S}{\Delta A_A} \; and \; \tilde{\alpha} = \frac{\tilde{\Delta} A_S}{\tilde{\Delta} A_A}$$

Then, based on the IEEE 802.11 standard and our settings, we next prove that the mean of α is theoretically differentiable between the normal AP scenario and the evil twin AP scenario, and thus can be used to effectively detect the evil twin attacks. Similar to Theorem 1, we can obtain the mean of AP IAT as illustrated in Fig. 6.

$$E(\widetilde{\Delta} A_A)_{one-hop} = 2T_{DIFS} + 2T_{SIFS} + 2E(T_{BF}) + \frac{2L_{ACK(MAC)} + L_{ACK(TCP)} + L_P}{B_W}$$

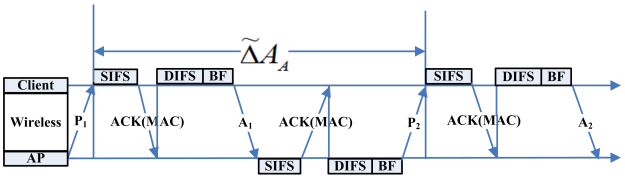


Fig. 6. AP IAT illustration in an ideal network environment.

We have the following two theorems that give us theoretic evidence on the effectiveness of this detection statistic.

Theorem 3. *If we denote $E(\alpha_{one-hop})$ and $E(\tilde{\alpha}_{one-hop})$ as the mean of α and $\tilde{\alpha}$ in one-hop wireless channels, then we can obtain: for 802.11b, $E(\alpha_{one-hop}) \leq E(\tilde{\alpha}_{one-hop}) = 1.00$; for 802.11g, $E(\alpha_{one-hop}) \leq E(\tilde{\alpha}_{one-hop}) = 1.11$.*

Proof: Based on Theorem 1 and the equation for the mean of AP IAT, we can obtain

$$E(\tilde{\alpha}_{one-hop}) = E(\frac{\tilde{\Delta} A_S}{\tilde{\Delta} A_A})_{one-hop} = \frac{2T_{DIFS} + T_{SIFS} + 2E(T_{BF})}{2T_{DIFS} + 2T_{SIFS} + E(T_{MAX}) + \frac{L_{ACK(MAC)} + L_{ACK(TCP)} + L_P}{B_W}} + 2E(T_{BF}) + \frac{2L_{ACK(MAC)} + L_{ACK(TCP)} + L_P}{B_W}$$

Since $E(T_{MAX}) \geq T_{SIFS} + \frac{L_{ACK(MAC)}}{B_W}$ with typical settings, we know $E(\tilde{\alpha}_{one-hop}) \geq 1$. Based on our settings about the values of these variances in TABLE I, we can get $E(\tilde{\alpha}_{one-hop}) = 1.00$ for WLAN 802.11b and $E(\tilde{\alpha}_{one-hop}) = 1.11$ for WLAN 802.11g. Since the period when we collect one pair of Server IAT and AP IAT to compute one SAIR is so short, we can reasonably assume that the network situation during this time will not change dramatically. Thus, we can assume that $E(\Delta_{r_c}) = E(\Delta A_A - \tilde{\Delta} A_A) = E(\Delta A_S - \tilde{\Delta} A_S) \geq 0$ so we can obtain that

$$E(\alpha_{one-hop}) = E(\frac{\Delta A_S}{\Delta A_A})_{one-hop} = \frac{E(\tilde{\Delta} A_S) + E(\Delta_{r_c})}{E(\tilde{\Delta} A_A) + E(\Delta_{r_c})} \leq \frac{E(\tilde{\Delta} A_S)}{E(\tilde{\Delta} A_A)} = E(\tilde{\alpha}_{one-hop})$$

For 802.11b, $E(\alpha_{one-hop}) \leq E(\tilde{\alpha}_{one-hop}) = 1.00$; for 802.11g, $E(\alpha_{one-hop}) \leq E(\tilde{\alpha}_{one-hop}) = 1.11$. ■

Theorem 4. *Based on the IEEE 802.11 standard and our settings, if we denote $E(\alpha_{two-hop})$ and $E(\tilde{\alpha}_{two-hop})$ as the mean of α and $\tilde{\alpha}$ in two-hop wireless channels, then we can obtain: for 802.11b, $E(\alpha_{two-hop}) \geq E(\tilde{\alpha}_{two-hop}) = 1.74$; for 802.11g, $E(\alpha_{two-hop}) \geq E(\tilde{\alpha}_{two-hop}) = 1.94$.*

Proof: Based on Theorem 2 and the equation for the mean of AP IAT, we can obtain

$$E(\tilde{\alpha}_{two-hop}) = E(\frac{\tilde{\Delta} A_S}{\tilde{\Delta} A_A})_{two-hop} = \frac{4T_{DIFS} + T_{SIFS} + 4E(T_{BF})}{2T_{DIFS} + 2T_{SIFS} + E(T_{MAX}) + \frac{L_{ACK(MAC)} + 2L_{ACK(TCP)} + L_P}{B_W}} \Big/ \frac{}{+2E(T_{BF}) + \frac{2L_{ACK(MAC)} + L_{ACK(TCP)} + L_P}{B_W}}$$

Based on our settings about the values of these variances, for Protocol 802.11b, we can obtain $E(\tilde{\alpha}_{two-hop}) = 1.74$; for Protocol 802.11g, we can obtain $E(\tilde{\alpha}_{two-hop}) = 1.94$. Similar to the Theorem 3, we denote $E(\Delta A_A) = E(\tilde{\Delta} A_A) + E(\Delta_{T_{C1}})$ and $E(\Delta A_S) = E(\tilde{\Delta} A_S) + E(\Delta_{T_{C1}}) + E(\Delta_{T_{C2}})$ where, $E(\Delta_{T_{C1}})$ and $E(\Delta_{T_{C2}})$ are the means of the time increment for the first hop and the second hop due to the decreased RSSI and increased collision numbers, respectively. So, we can obtain

$$\Delta = E(\alpha)_{two-hop} - E(\tilde{\alpha})_{two-hop} = \frac{E(\tilde{\Delta} A_S) + E(\Delta_{T_{C1}}) + E(\Delta_{T_{C2}})}{E(\tilde{\Delta} A_A) + E(\Delta_{T_{C1}})} - \frac{E(\tilde{\Delta} A_S)}{E(\tilde{\Delta} A_A)} = \frac{E(\tilde{\Delta}_{T_{C2}}) - (E(\tilde{\alpha})_{(two-hop)} - 1) \cdot E(\tilde{\Delta}_{T_{C1}})}{E(\tilde{\Delta} A_A) + E(\Delta_{T_{C1}})}$$

Since it is reasonable to assume that the link between the client and the evil twin AP has higher (or equal) RSSI level and smaller (or equal) collision probability than that of the link between the client and the normal AP, we can obtain that $E(\Delta_{T_{C1}}) \leq E(\Delta_{T_{C2}})$. Then, since under both protocols, $E(\tilde{\alpha})_{two-hop} \geq 1$ (specifically, for Protocol 802.11b, $E(\tilde{\alpha})_{two-hop} = 1.74$; for Protocol 802.11g, $E(\tilde{\alpha})_{two-hop} = 1.94$), so under both protocols, we can obtain $\Delta = E(\alpha)_{two-hop} - E(\tilde{\alpha})_{two-hop} \geq 0$. Thus, for WLAN 802.11b, $E(\alpha_{two-hop}) \geq E(\tilde{\alpha}_{two-hop}) = 1.74$; for WLAN 802.11g, $E(\alpha_{two-hop}) \geq E(\tilde{\alpha}_{two-hop}) = 1.94$. ■

From Theorem 3 and 4, we can see that the theoretical mean of α in evil twin AP scenario is significantly larger than that in the normal AP scenario, thus it can be used to detect evil twin attacks.

2) HDT Algorithm Description: In the previous section, we have proved that SAIRs in one-hop and two-hop wireless channels differ significantly. Even under the real network environment, we can still compute a theoretical SAIR bound to distinguish these two scenarios. According to this observation, in this section, we develop a non-training-based detection algorithm named Hop Differentiating Technique (HDT) algorithm.

We now describe the HDT algorithm in detail. Different from the TMM algorithm, in the HDT algorithm, we use a theoretical value for the threshold rather than a trained threshold to detect evil twin attacks. In the theoretical computation phase, we compute a threshold α_θ as the SAIR boundary to differentiate one-hop SAIR and two-hop SAIR. In order to use the SPRT technique, we also compute the upper bound for the probability of the SAIR exceeding the threshold α_θ in the normal AP scenario, and the lower bound for the probability of the SAIR exceeding the threshold α_θ in the evil twin AP scenario. The specific explanations of the computation of these three parameters will be discussed shortly. The details of HDT algorithm can be seen in Appendix A.

We acknowledge that once an attacker knows about HDT algorithm, he may attempt to evade it by making the server IAT similar to AP IAT under the evil twin scenario (e.g, maintaining different bandwidth for the first wireless hop and second wireless hop). We will discuss the issue, implication, and possible solution in Section VII.

3) Threshold Setting: In this section, we develop a discrete numerical algorithm to compute the theoretical SAIR threshold α_θ for HDT algorithm, with a goal of minimizing the probability of making a wrong decision. According to Theorem 3 and 4, the threshold α_θ should be between 1 and 2. So, if we denote $P_1 = P(\alpha_{one-hop} \geq \alpha_\theta)$ and $P_2 = P(\alpha_{two-hop} \geq \alpha_\theta)$, the problem can be transformed to compute $\hat{\alpha}_\theta$, s.t., $\hat{\alpha}_\theta = \arg min_{1 \leq \alpha_\theta \leq 2}(P_1 + 1 - P_2)$.

If we denote $\tilde{P}_1 = P(\tilde{\alpha}_{one-hop} \geq \tilde{\alpha}_\theta)$ and $\tilde{P}_2 = P(\tilde{\alpha}_{two-hop} \geq \tilde{\alpha}_\theta)$, then, similar to the proofs in Theorem 3 and Theorem 4, we can derive that $P_1 \leq \tilde{P}_1$ and $P_2 \geq \tilde{P}_2$. This implies that we can compute the threshold $\hat{\alpha}_\theta$ by using the variances of $\tilde{P}_1$ and $\tilde{P}_2$, which are the probabilities under the ideal network environment. So the problem is transformed to compute $\hat{\alpha}_\theta$, s.t., $\hat{\alpha}_\theta = \arg min_{1 \leq \alpha_\theta \leq 2}(\tilde{P}_1 + 1 - \tilde{P}_2)$.

In the process of our computation, we let α_θ increase from 1 to 2 in fine-grained steps. In every step, we increase α_θ by 0.01 and compute $\tilde{P}_1 + 1 - \tilde{P}_2$. Once α_θ reaches 2, we can find the value of $\hat{\alpha}_\theta$ leading to the minimal $\tilde{P}_1 + 1 - \tilde{P}_2$. In this way, according to the IEEE 802.11 standard and our settings, we can derive the following results:

- If we consider the packets without any collisions, then,
  - for Protocol 802.11b, $\hat{\alpha}_\theta = 1.31$, $P_1 \leq \tilde{P}_1 = 21.8\%$, $P_2 \geq \tilde{P}_2 = 76.9\%$;

  - for Protocol 802.11g, $\hat{\alpha}_\theta = 1.48$, $P_1 \leq \tilde{P}_1 = 27.3\%$, $P_2 \geq \tilde{P}_2 = 71.5\%$;
- If we consider the packets whose collision numbers are under three, then,
  - for Protocol 802.11b, $\hat{\alpha}_\theta = 1.34$, $P_1 \leq \tilde{P}_1 = 21.2\%$, $P_2 \geq \tilde{P}_2 = 74.9\%$;
  - for Protocol 802.11g, $\hat{\alpha}_\theta = 1.48$, $P_1 \leq \tilde{P}_1 = 27.3\%$, $P_2 \geq \tilde{P}_2 = 71.2\%$;

C. Improvement by Data Preprocessing

In this section, we describe two data preprocessing techniques to improve the results: data filtering and data smoothing. For the first technique, we filter noisy data (according to the theoretical Server IAT) with a large number of network collisions. For the second technique, we use the mean of multiple input data, rather than only one collected data, to smooth the input.

1) Data Filtering: With the considerations of some unexpected or unpredictable factors in the dynamic wireless networks, we also adopt similar data filtering policy as used in [22], [29] to filter out those packets that may contain some errors. Specifically, in order to filter noisy data, we only consider the packets whose collision numbers may be at most three. (According to [39], when the number of users is under 20, the probability that a packet has at most 3 collisions is over 85%). In this way, we can both filter the noisy data and keep sufficient data to implement the detection. Thus, according to the IEEE 802.11 standard and our filter policy, we filter out the packets whose AP IATs exceed 21,000μs or Server IAT exceed 39,800μs.

2) Data Smoothing: To further improve the result, we also use the mean of multiple input data rather than only one input data in one decision round. Specifically, we use the mean of multiple Server IATs or the mean of multiple SAIRs instead of only one Server IAT or one SAIR in one decision round to perform the threshold random walk. We name TMM algorithm and HDT algorithm using multiple Server IATs and multi SAIRs as multi-TMM algorithm and multi-HDT algorithm.

VI. EVALUATION

We evaluate the results and the performance of our evil twin attack detection algorithms by implementing a detection prototype system named ETSniffer (Evil Twin Sniffer). In this section, we describe our evaluation methodology, including the experimental setup, datasets, effectiveness, efficiency, cross-validation and comparison with ICMP.

A. Implementation and Experimental Setup

We implement ETSniffer using Windows raw socket, since we need packet level control (including TCP parameters). Specifically, to guarantee the efficiency and accuracy of the computation of IAT, we use an immediate-ACK policy, in which a TCP server should wait to receive the ACK packet for the previous data packet before sending out the next data packet. This policy is achieved by setting the TCP Maximum Segment Size (MSS) in the TCP header equal to the TCP Window Size without the need of changing default TCP protocol. Note, since our immediate-ACK policy is only applicable to the specific probing connections initialized and controlled by ETSniffer, this policy will not devour network bandwidth. In addition, we use a fixed and small number for MSS setting in every connection to guarantee sufficient data packets received to detect evil twin attacks. By initiating a TCP connection with customized TCP options and settings to make the server respond in the way we desire (e.g., sending packets with small size), ETSniffer can collect enough packets needed for detection even from a small-sized web page (which may only result in one or two packets in the normal setting).

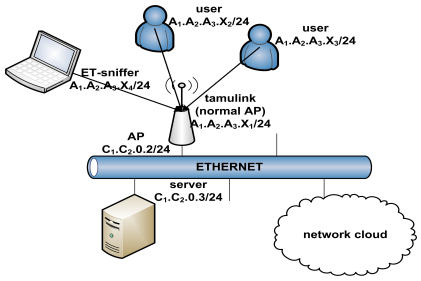


Fig. 7. Experimental environment setting for the normal AP scenario.

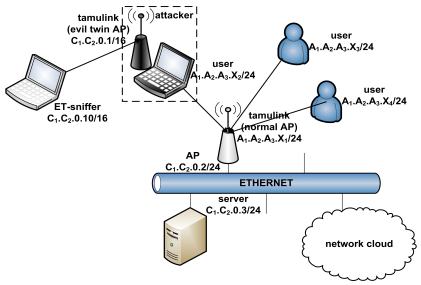


Fig. 8. Experimental environment setting for the evil twin AP scenario.

We set up our ETSniffer under Texas A&M University campus network environment. To achieve user-side detection, we install ETSniffer in a laptop with a wireless network card. The ETSniffer can capture the packets, along with the current timestamp, to compute IAT and SAIR. To simulate a normal AP scenario, we use a laptop installed with ETSniffer as a user/detection client to communicate with a campus server through TAMULink (an official Texas A&M's campus wireless network Access Point). To simulate an evil twin AP scenario, we deploy another laptop as a wireless access point with the same SSID as the TAMULink to act as an evil twin AP near to the detection client. And the evil twin AP connects to the server through the campus TAMULink AP, with the RSSI level between 80% and 100%. Thus, in this scenario, the detection client communicates with the server through a two-hop wireless channel. The experimental environment setting can be illustrated in Fig. 7 and 8.

B. Datasets

We build our datasets in real network environments at different times and with different RSSI levels (The RSSI here refers to the first hop, which can be obtained in the user side.). To better evaluate our results, in our experiments, we denote different RSSI levels into 6 ranges: A, B+, B-, C+, C-, and D, as illustrated in TABLE II.

TABLE II
RSSI RANGES AND CORRESPONDING LEVELS

Range	A	B+	B-	C+	C-	D
Upper	100%	80%	70%	60%	50%	40%
Lower	80%	70%	60%	50%	40%	20%

As described in Section V-C1, we filter the packets whose collision numbers may exceed three. The percentages of filtered packets can be seen in TABLE III.

TABLE III
THE PERCENTAGE OF FILTERED PACKETS

Tech	Protocol	A	B+	B-	C+	C-	D
TMM	802.11g	0.62%	0.68%	2.59%	2.66%	3.30%	6.02%
	802.11b	0.99%	1.04%	3.33%	4.82%	7.44%	8.29%
HDT	802.11g	0.80%	0.86%	3.91%	3.72%	4.69%	7.09%
	802.11b	1.38%	1.44%	5.61%	6.17%	9.42%	10.36%

C. Effectiveness

We evaluate the effectiveness of our algorithms based on different RSSI ranges under 802.11b and 802.11g network. In the normal AP scenario, the RSSI refers to the link between the user and the normal AP; in the evil twin AP scenario, the RSSI refers to the link between the user and the evil twin AP. As shown in TABLE IV under most cases, both HDT and TMM under two protocols can achieve a high detection rate (over 90%). Especially, when the RSSI level is high (A and B+), the detection rates of two algorithms can be maintained higher than 95%.

TABLE IV
DETECTION RATE FOR HDT AND TMM

Tech	Protocol	A	B+	B-	C+	C-	D
TMM	802.11g	99.39%	99.97%	99.49%	99.5%	98.32%	94.36%
	802.11b	99.81%	95.43%	94.81%	96.09%	91.94%	85.71%
HDT	802.11g	99.08%	98.72%	93.53%	94.31%	87.29%	81.39%
	802.11b	99.92%	99.99%	99.96%	99.95%	96.05%	94.64%

As shown in TABLE V, under most of cases, our approach can achieve a relatively low false positive rate. In addition, we can also find that the results obtained in the 802.11g are usually similar to or better than those obtained in 802.11b. This is caused by the low bandwidth and larger initial window size in the 802.11b protocol, leading to a larger variance of IAT distribution.

TABLE V
FALSE POSITIVE RATE FOR HDT AND TMM

Tech	Protocol	A	B+	B-	C+	C-	D
TMM	802.11g	1.08%	1.76%	1.97%	1.48%	1.75%	1.73%
	802.11b	0.78%	1.00%	1.07%	1.27%	6.65%	7.01%
HDT	802.11g	2.19%	1.41%	2.06%	1.93%	2.48%	6.52%
	802.11b	8.39%	8.76%	5.39%	6.96%	5.27%	5.15%

As described in Section V-C2, we use multi-TMM and multi-HDT to improve the results. Fig. 9 shows that detection rate increases by using more input data in one decision round. However, once the number attains to some bound (in our experiment, it is 70), the detection rate becomes relatively steady, which is nearly to be 100%.

As seen in TABLE VI, when we increase the the number of input data in one decision round to 50, the detection rates

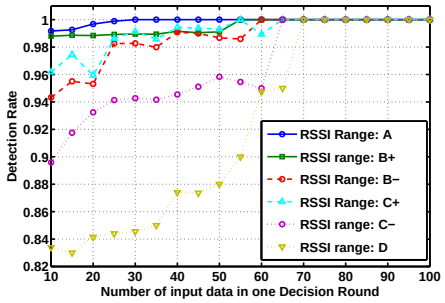


Fig. 9. Detection rate for multi-HDT using different numbers of input data in one decision round.

of two algorithms are higher than that of using one input data. Also, under most of cases, the detection rate can be achieved around 100%.

TABLE VI
DETECTION RATE FOR MULTI-TMM AND MULTI-HDT, WHEN THE NUMBER OF INPUT DATA IN ONE DECISION ROUND IS 50

Tech	Protocol	A	B+	B-
Multi-TMM	802.11g	99.62%	100%	100%
	802.11b	100%	100%	100%
Multi-HDT	802.11g	100%	99.11%	98.73%
	802.11b	100%	100%	100%
Tech	**Protocol**	**C+**	**C-**	**D**
Multi-TMM	802.11g	99.95%	100%	100%
	802.11b	100%	100%	100%
Multi-HDT	802.11g	99.88%	95.83%	88%
	802.11b	100%	100%	100%

As seen in Table VII, compared with using one input data in one decision round, when we use 50 input data, the false positive rate can be obviously decreased on both TMM and HDT algorithm. Under most cases, the false positive rates can be maintained lower than 1%.

TABLE VII
FALSE POSITIVE RATE FOR MULTI-TMM AND MULTI-HDT, WHEN THE NUMBER OF INPUT DATA IN ONE DECISION ROUND IS 50

Tech	Protocol	A	B+	B-
Multi-TMM	802.11g	0%	0.77%	0%
	802.11b	0%	0.03%	0.02%
Multi-HDT	802.11g	0%	0.96%	0.16%
	802.11b	0%	1.07%	1.16%
Tech	**Protocol**	**C+**	**C-**	**D**
Multi-TMM	802.11g	0%	0%	0%
	802.11b	0.11%	0.73%	0.1%
Multi-HDT	802.11g	0.13%	0.55%	0.96%
	802.11b	1.02%	1.36%	1.41%

D. Time Efficiency

We also evaluate time efficiency of our algorithms. Specifically, we use the average number of decision rounds to output a correct decision as the evaluation metric. The average consuming time (data collection and detection) for each decision round is 4.68ms, 5.93ms, 6.39ms, 7.41ms, 8.80ms, 9.21ms, when the RSSI range is equal to A, B+, B-, C+, C-, and D, respectively. We use cumulative probability to express the process of the log-likelihood ratio to reach the bounds. As seen in Fig. 10, under most values of RSSI, our HDT algorithm can output a correct decision within 25 rounds, which takes less than 0.25s. Even though a low RSSI (e.g. RSSI range is D) will require more decision rounds, our algorithm can obtain a

correct result within 45 decision rounds, which consumes less than 0.5s. TMM has a very similar performance.

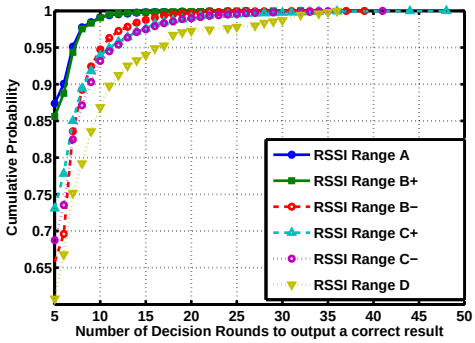


Fig. 10. Cumulative probability of the number of decision rounds for HDT to output a correct result.

E. Cross-Validation

From Section VI-C, we can find that both the TMM algorithm and the HDT algorithm demonstrate high efficiency and effectiveness. Especially, the TMM algorithm, based on the trained knowledge, performs a little bit better than the HDT algorithm. However, as described in Section V-A2, in many practical cases, the prior knowledge is difficult to obtain. In addition, the TMM algorithm does not accommodate well to the changes of the wireless network environment. Thus, to evaluate such limitations of the TMM algorithm, in this section, we design cross-validation experiments under different levels of RSSI and different locations/networks.

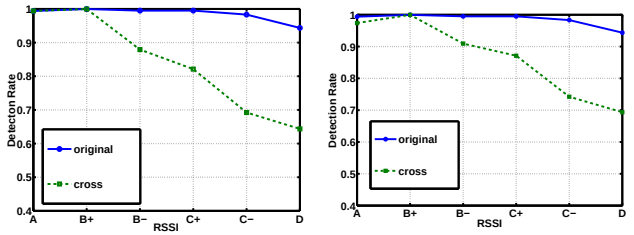
1) Cross-validation under different RSSI: In this section, we implement the cross-validation experiments under different RSSI ranges. Specifically, we evaluate TMM by training the threshold under one specific RSSI level and detecting the evil twin attacks under all RSSI ranges. From Fig. 11, we can see that the detection rate drops dramatically, revealing TMM algorithm's tight dependency on the (perfect) training data.

2) Cross-validation under different locations/networks: To validate the performance of both algorithms in different network environments, we conduct a cross-validation under different locations. For TMM algorithm, we train the Server IAT threshold using the data collected in one wireless environment, and execute the detection in another location.

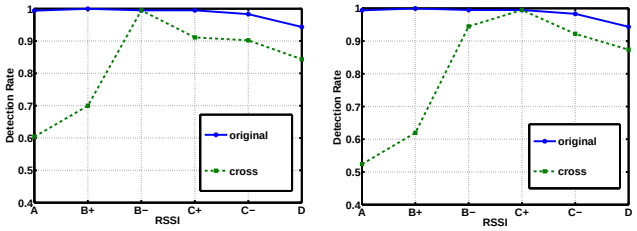
From Fig. 12 and 13, we can see that, if we train and test in different environments, the performance of the TMM algorithm decreases significantly. However, the performance of the HDT algorithm remains steady.

3) Comparison With ICMP: In this section, in order to show the advantages of using TCP packets to compute our detection statistics – IAT, we make a comparison of the detection performance of collecting TCP packets and ICMP packets to detect evil twin attacks. For using ICMP packets, similar to using TCP packets to compute IATs, we collect ICMP packets and compute one IAT as the time interval of each pair of ICMP reply packets that arrive at the client side.

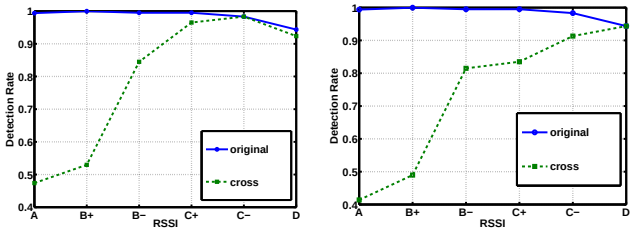
From Fig. 14 and 15, we can find that using TCP packets to compute IATs can obtain a higher detection rate and a lower false positive rate. That is probably because TCP protocol provides reliable transmission service. In this way, the computation of one IAT and SAIR by using TCP packets is more stable and yields to better results as demonstrated in our experiments.



(a) Training RSSI range is A (b) Training RSSI range is B+

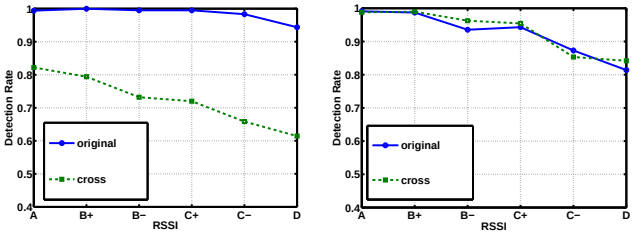


(c) Training RSSI range is B- (d) Training RSSI range is C+



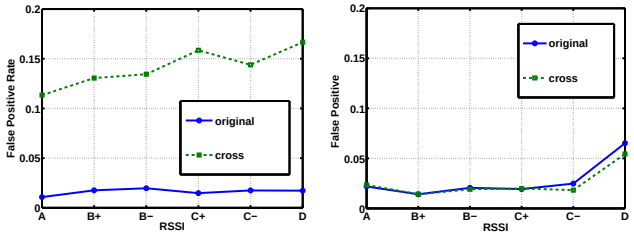
(e) Training RSSI range is C- (f) Training RSSI range is D

Fig. 11. The detection rate for TMM algorithm under different training RSSI ranges.



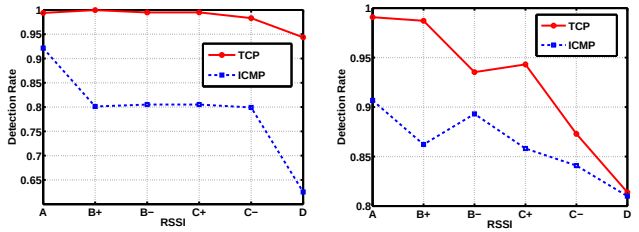
(a) TMM algorithm (b) HDT algorithm

Fig. 12. Detection rate under different 802.11g networks.

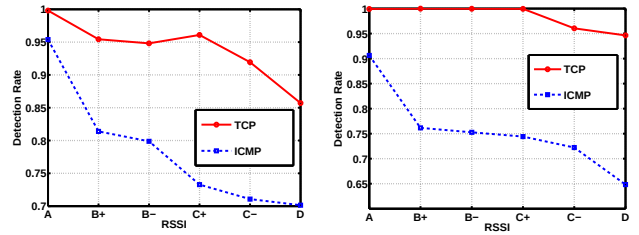


(a) TMM algorithm (b) HDT algorithm

Fig. 13. False positive rate under different networks.

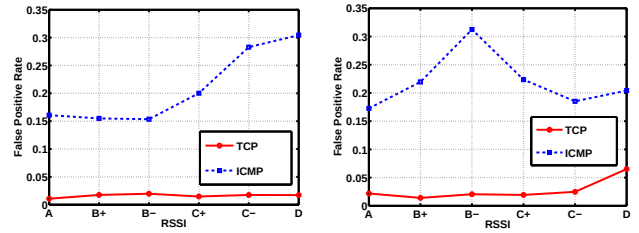


(a) TMM algorithm under 802.11g (b) HDT algorithm under 802.11g

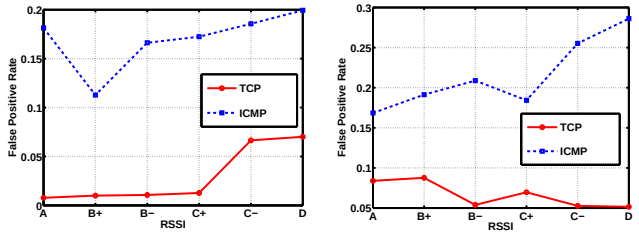


(c) TMM algorithm under 802.11b (d) HDT algorithm under 802.11b

Fig. 14. The comparison of detection rate of both algorithms under 802.11g and 802.11b networks using TCP packets and ICMP packets.



(a) TMM algorithm under 802.11g (b) HDT algorithm under 802.11g



(c) TMM algorithm under 802.11b (d) HDT algorithm under 802.11b

Fig. 15. The comparison of false positive rate of both algorithms under 802.11g and 802.11b networks using TCP packets and ICMP packets.

4) Comparison With Unbalanced RSSI: In this experiment, we evaluate our algorithm by setting different unbalanced RSSI levels in the first-hop wireless and second-hop wireless channel. Specifically, for each RSSI level in the first-hop wireless channel, we test detection rates of HDT algorithm under 802.11g network by varying the RSSI levels in the second-hop channel, as shown in TABLE VIII (B and C denote RSSI ranging from 60% to 80%, and from 40% to 60%, respectively.). In addition, while calculating the detection rate in this experiment, we also evaluate the false positive rate, as shown in TABLE IX.

From TABLE VIII and IX, we can find that, even under unbalanced RSSI levels for the first and second hop, our algorithm can still obtain high detection rate and maintain

TABLE VIII
DETECTION RATE BY SETTING UNBALANCED RSSI LEVELS FOR HDT ALGORITHM UNDER 802.11G NETWORKS

First-hop \ Second-hop	A	B	C
A	99.33%	100%	100%
B+	98.18%	97.69%	98.33%
B-	94.17%	95.16%	96.67%
C+	94.02%	96.67%	96%
C-	88.46%	86.67%	89.29%

TABLE IX
FALSE POSITIVE RATE BY SETTING UNBALANCED RSSI LEVELS IN THE FIRST AND SECOND WIRELESS HOP FOR HDT ALGORITHM UNDER 802.11G

Range	**A**	**B+**	**B-**	**C+**	**C-**
False Positive Rate	2.0%	1.25%	2.5%	2.61%	2.13%

reasonable false positive rate, especially when the RSSI in the first hop is high. In addition, while keeping the same RSSI level for the first hop, we can find that the detection rate of under the RSSI level as "C" in the second hop is even higher than that of under "A" in the second hop. That is mainly because a much lower RSSI level in the second hop can lead to a longer time for the attacker to transmit the packets making it more obvious to be detected.

VII. LIMITATION AND FUTURE WORK

We clearly admit that our designed methods can *not* detect all kinds of man-in-the-middle attacks in the WLAN. For example, our ETSniffer is not applicable under the scenario that attackers use other methods such as wired links, 3G or WiMax, rather than the legitimate AP to relay the traffic. In our preliminary work, our targeted problem is evil twin AP detection, where the evil twin AP utilizes the normal AP to connect to Internet. In fact, this problem is indeed a very realistic threat faced by public WLANs provided at airports, hotels, libraries, or cafes, etc, because it is easy for an attacker in the public area to get free Internet access from public free Wi-Fi to launch such kinds of attacks. In our future work, we plan to extend our evil twin AP detection to more *general malicious* AP detection, where a malicious AP may not require the normal AP to relay traffic, or not need to impersonate a normal AP. We also plan to study the problem in wireless infrastructures (e.g., 3G or WiMax) that have multi-hop wireless channels.

We acknowledge that once the remote servers are not available or compromised, our approach may not work correctly. As described in Section III, we clearly describe three types of servers that can be used as remote servers. Also, we can enhance our ETsniffer to combine (or selectively use) these three types of servers, if any type of servers are not available or trustable. Also, it is true that more wired hops between the remote server to the legitimate AP may involve likely more "noise". Thus, in reality, we recommend to set the remote server within small hops from the legitimate AP. In our experiment, the remote server is four wired hops away from the AP. To decrease possible effect of the wired channel, we also utilize SPRT technique to tolerate reasonable noise, if we tradeoff for more decision rounds. In addition, if the user has to use a server with many hops, we could consider using techniques similar to "traceroute" to calculate the (wired)

transfer time and then exclude/subtract them to minimize the effect at wired side.

Note that our adopted immediate-ACK policy may affect the packets transmitting speed in the specific connections initialized by ETSniffer software. However, since this policy is achieved by setting the Window Size as the TCP Maximum Segment Size, our ETsniffer still strictly follows default TCP protocol and thus no changes of any network software or hardware are required on both the client and server side. In addition, our immediate-ACK policy is not a global policy, instead it is only applicable to the specific probing connections initialized by ETSniffer (such connections are typically for a very tiny period, and controlled by ETSniffer). All other network communications on users' machines are not affected.

It is possible that attackers may attempt to evade our detection scheme, because attackers, between the victims and normal AP, can manipulate the traffic to affect IAT. For example, attackers can attempt to evade our HDT algorithm by making the server IAT similar to AP IAT under the evil twin scenario. Attackers should intentionally increase AP IATs to make them less differentiable from Server IATs. They can achieve this by either maintaining low bandwidth (e.g., 1MBps) between the evil twin AP and the victim and high bandwidth (e.g, 54MBps) between evil twin AP and the legitimate AP, or simply delaying transmitting packets to the victims. However, by doing this, attackers need to exactly know how HDT work. Also, a low practical bandwidth between the attackers to victims may decrease attackers' attractions to victims. In addition, our designed TMM algorithm can be combined with HDT and be used to detect such anomaly.

We also acknowledge that if the workload of legitimate AP is extremely heavy, the time difference between one-hop server IAT under the normal AP scenario and two-hop server IAT under the evil twin AP scenario becomes less distinguishable. Thus, the accuracy of our TMM algorithm may be decreased. However, in this way, our HDT algorithm can perform better. Particularly, HDT does not rely on any training data and relies on the server IAT to AP IAT ratio. Specifically, if it is under the evil twin AP scenario and the legitimate AP is busy, then the ratio of two-hop server IAT (between the client and the server) to one-hop AP IAT (between the client and the evil twin AP) will become even larger. In addition, as shown in our evaluation results, as long as the legitimate AP is not so heavy that most network packets can be successfully sent from the legitimate AP to the receiver within three wireless collisions, our TMM algorithms can still achieve a high accuracy. (Those packets, which have more than three wireless collisions, have been filtered in our evaluation.)

Finally, we acknowledge that our timing-based detection techniques may not perform well once attackers pretend to be the users to get the next data packet and send it back to the users, which is also a challenge to most of current timing-based evil twin detection approaches. However, in order to achieve this, attackers need to make much more efforts on deeply understanding our detection algorithm and successfully cheating users that the next data packet is the right one. More further studies are needed in this area.

VIII. Conclusion

In this paper, we propose a novel lightweight user-side evil twin attack detection technique. We present two algorithms, TMM and HDT. We implement our prototype system and evaluate it in several real-world wireless networks, and our evaluation results proved its effectiveness and efficiency.

References

[1] "Evil twin in Wikipedia," http://en.wikipedia.org/wiki/Evil_twin_(wireless_networks).

[2] S. Jana and S. Kasera, "On Fast and Accurate Detection of Unauthorized Wireless Access Points Using Clock Skews," in *IEEE Transactions on Mobile Computing, g, vol. 9, no. 3, pp. 449-462*, Mar. 2010.

[3] H. Han, B. Sheng, C. Tan, Q. Li, and S. Lu, "A Measurement Based Rogue AP Detection Scheme," in *IEEE International Conference on Computer Communications (Infocom'09)*, 2009.

[4] "Smartphones and public wi-fi Evil Twin attacks," http://blog.netsafe.org.nz/2011/04/28/smartphones-and-public-wi-fi-evil-twin-attacks/).

[5] "Top Ten Ways to Avoid an Evil Twin Attack," http://www.esecurityplanet.com/views/article.php/3908596/Top-Ten-Ways-to-Avoid-an-Evil-Twin-Attack.htm).

[6] "Evil Twin attacks: scamming wireless network users," http://scamsinc.com/2012/02/13/evil-twin-attacks-scamming-wireless-network-users/).

[7] "Wi-Fi security flaw for smartphones," http://www.guardian.co.uk/technology/2011/apr/25/wifi-security-flaw-smartphones-risk).

[8] "Soft AP solutions," White paper, http://www.marvell.com/products/wireless/softap.jsp.

[9] "Received Signal Strength Indication," http://en.wikipedia.org/wiki/Received_signal_strength_indication.

[10] AirDefense, "TIRED OF ROGUES? Solutions for Detecting and Eliminating Rogue Wireless Networks," White paper, http://wirelessnetworkchannel-asia.motorola.com/pdf/.

[11] "The Airmagnet project," http://www.airmagnet.com/.

[12] "The Netstumbler project," http://www.netstumbler.com.

[13] "WiSentry - Wireless Access Point Detection System," http://www.wimetrics.com/Products/WAPD.htm.

[14] "The Inssider software," http://www.metageek.net/products/inssider.

[15] "The Airwave project," http://www.airwave.com.

[16] "Wavelink," http://www.wavelink.com.

[17] "Cisco Wireless LAN Solution Engine (WLSE)," White paper, http://www.cisco.com/en/US/products/sw/cscowork/ps3915/.

[18] "Rogue access point detection: Automatically detect and manage wireless threats to your network," White paper, http://www.proxim.com.

[19] P. Bahl, R. Chandra, J. Padhye, L. Ravindranath, M. Singh, A. Wolman, and B. Zill, "Enhancing the security of corporate Wi-Fi networks using DAIR," in *Proc. MobiSys'06*, 2006.

[20] W. Wei, K. Suh, B. Wang, Y. Gu, J. Kurose, and D. Towsley, "Passive online rogue access point detection using sequential hypothesis testing with TCP ACK-pairs," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (IMC'07)*, 2007.

[21] H. Yin, G. Chen, and J. Wang, "Detecting protected layer-3 rogue APs," in *Proceedings of the Fourth IEEE International Conference on Broadband Communications, Networks, and Systems (BROADNETS'07)*, 2007.

[22] W. Wei, B. Wang, C. Zhang, J. Kurose, and D. Towsley, "Classification of access network types: Ethernet, wireless LAN, ADSL, cable modem or dialup?" *Computer Networks*, vol. 52, no. 17, pp. 3205–3217, 2008.

[23] S. Shetty, M. Song, and L. Ma, "Rogue access point detection by analyzing network traffic characteristics," in *IEEE Military Communications Conference (MILCOM'07)*, 2007.

[24] W. Wei, S. Jaiswal, J. Kurose, and D. Towsley, "Identifying 802.11 traffic from passive measurements using iterative Bayesian inference," in *Proc. IEEE INFOCOM'06*, 2006.

[25] V. Baiamonte, K. Papagiannaki, G. Iannaccone, and P. D. Torino, "Detecting 802.11 wireless hosts from remote passive observations," in *Proc. IFIP/TC6 Networking*, 2007.

[26] L. Watkins, R. Beyah, and C. Corbett, "A passive approach to rogue access point detection," in *Proc. IEEE Globecom'07*, 2007.

[27] A. Wald, *Sequential Analysis*. Dover Publications, 2004.

[28] R. Beyah, S. Kangude, G. Yu, B. Strickland, and J. Copeland, "Rogue access point detection using temporal traffic characteristics," in *IEEE Global Telecommunications Conference (GLOBECOM'04)*, 2004.

[29] H. Han, B. Sheng, C. Tan, Q. Li, and S. Lu, "A Timing Based Scheme for Rogue AP Detection," in *IEEE Transactions on Parallel and Distributed Systems*, 2010.

[30] S. Garg and M. Kappes, "An experimental study of throughput for UDP and VoIP traffic in IEEE 802.11 b networks," in *2003 IEEE Wireless Communications and Networking (WCNC'03)*, 2003.
[31] C. Mano, A. Blaich, Q. Liao, Y. Jiang, D. Cieslak, D. Salyers, and A. Striegel, "RIPPS: Rogue identifying packet payload slicer detecting unauthorized wireless hosts through network traffic conditioning," *ACM Transactions on Information and System Security (TISSEC)*, vol. 11, no. 2, pp. 1–23, 2008.
[32] C. Corbett, R. Beyah, and J. Copeland, "A passive approach to wireless NIC identification," in *IEEE International Conference on Communications (ICC'06)*, 2006.
[33] A. Venkataraman and R. Beyah, "Rogue Access Point Detection Using Innate Characteristics of the 802.11 MAC," in *International Conference on Security and Privacy in Communication Networks (SecureComm'09)*, 2009.
[34] L. Ma, A. Teymorian, and X. Cheng, "A hybrid rogue access point protection framework for commodity Wi-Fi networks," in *Proc. IEEE Infocom'08*, 2008.
[35] S. Srilasak, K. Wongthavarawat, and A. Phonphoem, "Integrated Wireless Rogue Access Point Detection and Counterattack System," in *International Conference on Information Security and Assurance*, 2008, pp. 326–331.
[36] "Evil twin attack - also known as wifi phishing," http://www.firewalls.com/blog/post/view/identifier/wifi_phishing_evil_twin_attack/.
[37] D. MacDonald and W. Barkley, "Microsoft Windows 2000 TCP/IP Implementation Details," White paper, http://technet.microsoft.com/en-us/library/bb726981.aspx.
[38] P. Sarolahti and A. Kuznetsov, "Congestion control in linux tcp," in *Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference*, 2002.
[39] H. Vu and T. Sakurai, "Collision probability in saturated IEEE 802.11 networks," in *Australian Telecommunication Networks and Applications Conference*, 2006.
[40] C. Foh and J. Tantra, "Comments on IEEE 802.11 saturation throughput analysis with freezing of backoff counters," *IEEE Communications Letters*, vol. 9, no. 2, pp. 130–132, 2005.
[41] G. Bianchi and D. e Inf, "IEEE 802.11-saturation throughput analysis," *IEEE communications letters*, vol. 2, no. 12, pp. 318–320, 1998.
[42] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "DNS performance and the effectiveness of caching."
[43] K. Thompson, G. Miller, and R. Wilder, "Wide-area Internet traffic patterns and characteristics," *IEEE network*, vol. 11, no. 6, pp. 10–23, 1997.

APPENDIX

A. TMM and HDT Algorithm

Algorithm 1 Trained Mean Matching

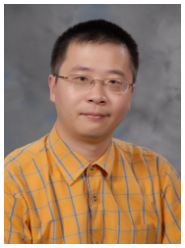
/* Training Phase: */
1. Compute $\mu_{1,NAP}$ and $\sigma_{1,NAP}$
2. Filter one-hop server IATs beyond the range
3. Compute $\mu_{2,NAP}$
4. Compute $\mu_{1,EAP}$ and $\sigma_{1,EAP}$
5. Filter two-hop server IATs beyond the range
6. Compute $\mu_{2,EAP}$
7. $T_\theta = \frac{1}{2}(\mu_{2,NAP} + \mu_{2,EAP})$
8. Compute P_1 and P_2
/* Detection Phase: */
$\Lambda = 0$, $\theta_0 = P_1$, $\theta_1 = P_2$
for $i = 0$ **do**
  Compute δ_i
  if $\delta_i \geq T_\theta$ **then**
    $\Lambda = \Lambda + \ln\theta_1 - \ln\theta_0$
  else
    $\Lambda = \Lambda - \ln(1-\theta_1) - \ln(1-\theta_0)$
  end if
  if $\Lambda \geq B$ **then**
    return evil twin AP scenario
  else if $\Lambda \leq A$ **then**
    return normal AP scenario
  end if
end for

Algorithm 2 Hop Differentiating Technique

$\Lambda = 0$, $\theta_0 = P_1$, $\theta_1 = P_2$
for $i = 0$ **do**
  Compute α_i
  if $\alpha_i \geq \alpha_\theta$ **then**
    $\Lambda = \Lambda + \ln\theta_1 - \ln\theta_0$
  else
    $\Lambda = \Lambda - \ln(1-\theta_1) - \ln(1-\theta_0)$
  end if
  if $\Lambda \geq B$ **then**
    return evil twin AP scenario
  else if $\Lambda \leq A$ **then**
    return normal AP scenario
  end if
end for

Chao Yang is a Ph.D. candidate in the Department of Computer Science and Engineering at Texas A&M University. He received his B.S. degree in Mathematics and M.S. degree in Computer Science from Harbin Institute of Technology in China. His research interests include network security, web security (especially social networking website security) and smartphone security.

Yimin Song is a M.S. graduate from Department of Computer Science and Engineering at Texas A&M University (TAMU). During his Master's career, he was working on protecting the AP users from hacking with Dr. Guofei Gu in SUCCESS Lab. Upon his graduation, he started working for Juniper Networks.



Guofei Gu is an assistant professor in the Department of Computer Science & Engineering at Texas A&M University (TAMU). Before coming to Texas A&M, he received his Ph.D. degree in Computer Science from the College of Computing, Georgia Institute of Technology. His research interests are in network and system security, such as malware analysis/detection/defense, intrusion/anomaly detection, web and social networking security. Dr. Gu is a recipient of 2010 NSF CAREER award and a co-recipient of 2010 IEEE Symposium on Security and Privacy (Oakland'10) best student paper award. He is currently directing the SUCCESS (Secure Communication and Computer Systems) Lab at TAMU.