Mimicry Attacks against Wireless Link Signature and Defense using Time-Synched Link Signature

Yao Liu, Peng Ning
North Carolina State University, Raleigh, NC 27695
{yliu20, pning}@ncsu.edu

***Abstract*—Wireless link signature is a physical layer authentication mechanism, which uses the unique wireless channel characteristics between a transmitter and a receiver to provide authentication of wireless signals. This paper identifies a new attack, called *mimicry attack*, against existing wireless link signature schemes. It is assumed that an attacker cannot "spoof" an arbitrary link signature and that the attacker will not have the same link signature at the receiver unless it is at exactly the same location as the legitimate transmitter. However, this paper shows that an attacker *can* forge an *arbitrary* link signature as long as it roughly knows or can estimate the legitimate signal at the receiver's location, and the attacker does not have to be at exactly the same location as the legitimate transmitter in order to forge its link signature.**

To provide physical layer authentication capability and defend against the mimicry attack, this paper proposes a novel construction for wireless link signature, called *time-synched link signature*, by integrating cryptographic protection and time factor into wireless physical layer features. This paper reports an extensive set of experimental evaluation of the mimicry attacks and the time-synched link signature scheme on the USRP2 platform running GNURadio. The experimental results confirm that the mimicry attacks against the previous link signature scheme are a real threat and demonstrate that the newly proposed time-synched link signature is effective in physical layer authentication.

I. Introduction

Wireless physical layer security is becoming increasingly important as wireless devices are more and more pervasive and adopted in critical applications. For example, implantable medical devices (IMD) such as pacemaker may grant access to an external control device only when it is close enough [31]. Thus, verifying the physical proximity of the external control device is critical to the security of the IMD and the patient. There have been multiple proposals in recent years to provide enhanced wireless security using physical layer characteristics, including fingerprinting wireless devices (e.g., [5], [13], [16], [25]), authenticating and identifying wireless channels (e.g., [30], [42]), and deriving secret keys from wireless channel features only observable to the communicating parties (e.g., [3], [26], [27], [40]).

Among the recent advances in wireless physical layer security is (wireless) link signature. Link signature uses the unique wireless channel characteristics (e.g., the multi-path effect) between a transmitter and a receiver to provide authentication of the wireless channel. Three link signature schemes [22], [30], [42] have been proposed so far. Since its initial introduction, link signature has been recognized as a channel authentication mechanism for applications where wireless channel characteristics is unique for individual nodes (e.g., [5], [20], [23], [26], [41]).

In this paper, we identify a new attack, which is called the *mimicry attack*, against existing link signature schemes [22], [30], [42]. We start our investigation with the link signature scheme in [30]. It is assumed in [30] that an attacker "cannot 'spoof' an arbitrary link signature" and that the attacker "will not have the same link signature at the receiver unless it is at exactly the same location as the legitimate transmitter." However, we show in this paper that (1) an attacker *can* forge an *arbitrary* link signature as long as the attacker can roughly know or estimate the legitimate signal at the receiver's location, and (2) the attacker does not have to be at exactly the same location as the legitimate transmitter in order to forge its link signature. We also extend the mimicry attack to the link signature scheme in [22]. Since the link signature scheme in [42] is essentially an integration of the techniques in [30], [42], all three link signature schemes are vulnerable to the mimicry attack.

To provide wireless channel authentication capability and defend against the threats identified in this paper, we develop a novel construction for link signature, which is called time-synched (i.e., time synchronized) link signature. Time-synched link signature integrates cryptographic protection as well as time factor into the wireless physical layer features, and provides an effective countermeasure against mimicry attacks. We also perform an extensive set of experimental evaluation of the mimicry attacks and the time-synched link signature scheme on the USRP2 platform [24] running GNURadio [1]. Our experiments confirm that the mimicry attacks against the previous link signature scheme are a real threat and demonstrate that the newly proposed time-synched link signatures are effective in mitigating those attacks.

Our contribution in this paper is three-fold. First, we identify the mimicry attack against existing link signature schemes. Second, we develop the time-synched link signature scheme to defend against various mimicry attacks based threats. Finally, we perform an extensive set of experiments to evaluate the impact of mimicry attacks and demonstrate the effectiveness of the proposed time-synched link signature.

The remainder of this paper is organized as follows. Section II gives some background information for link signatures and the related discussion in this paper. Section III discusses the mimicry attacks against existing link signature schemes. Section IV presents our new time-synched link signature.

Section V gives our experimental confirmation of the mimicry attacks against the previous link signature as well as evaluation of the time-synched link signature. Section VI discusses related work. Finally, Section VII concludes this paper and points out some future research directions.

II. Preliminaries

In this section, we give some background information on the link signature scheme in [30], including multi-path effect, channel impulse response, and how these are used for link signatures.

A. Multi-path Effect, Channel Impulse Response, and Link Signature

Wireless signal usually propagates in the air along multiple paths due to reflection, diffraction, and scattering [30]. As a result, a receiver may receive multiple copies of the transmitted signal from different paths, each of which may have a different delay due to the path it traversed on. The received signal is indeed the sum of these time delayed signal copies. Each path imposes a *response* (e.g., distortion and attenuation) on the signal traveling along it [30], and the superposition of all responses between two nodes is referred to as a *channel impulse response* [17].

The multi-path effects between different pairs of nodes are usually different, and so are the channel impulse responses [30]. Due to this reason, a channel impulse response between two nodes is also called a *link signature*, and has been proposed to provide robust location distinction and location-based authentication [30], [41], [42]. Specifically, to determine if a received signal is from the desired location/channel of the transmitter, the receiver can estimate the link signature of the received signal and compare it with *reference link signatures*, which are estimated when the receiver already knows that signals are from the desired location/channel. The received signal is accepted only if the estimated link signature is similar to reference link signatures.

B. Cryptographic Signatures v.s. Link Signatures

Cryptographic and link signatures achieve different authentication purposes. In general, cryptographic authentication enables a receiver to know if messages are generated by the desired transmitter, whereas link signature authentication enables a receiver to know if messages are from the desired location/channel. For example, sensors can be used to monitor if a fire starts in an area. They report temperature measurements to a remote receiver through wireless channel. An arsonist can burn target area without being detected by moving the sensors to a normal place. Cryptographic signatures don't help, but link signatures can alert the receiver (the sensors' link signatures change as their location/channel changes).

C. Estimating Channel Impulse Responses

With the knowledge of channel impulse responses, wireless systems can achieve high quality communication with low bit error rate [36]. A popular method for estimating channel impulse responses is the *training sequence based estimation* [34], which is adopted by many modern wireless systems, including WLAN, CDMA1x, DVB, WiMAX, and ZigBee. In this method, the transmitter sends a training sequence (i.e., a sequence of bits) over the wireless channel. The receiver then uses the training sequence and the corresponding received signal samples to estimate channel impulse responses, where the data value of the training sequence can be pre-shared [34] or reconstructed from the received signal through demodulation [30].

Note that at the physical layer channel estimation can be processed in either frequency domain (e.g. [30], [42]) or time domain (e.g., [34], [36]). Because of the linear relationship between the two domains, frequency and time domain based methods are inter-convertible. In the following, we describe the channel estimation method in the time domain.

Mathematical Formulation: To estimate channel impulse responses, the receiver exploits the known transmitted data content and the corresponding received samples. As discussed earlier, the sender and the receiver share a *training sequence*. For example, a training sequence can be $[0, 1, 0, 1]$. To transmit the training sequence, the transmitter converts it into M physical layer symbols (i.e., complex numbers that are transmission units at the physical layer [18]). This process is called modulation [18]. The transmitter then sends the M symbols to the wireless channel.

Let $\mathbf{x} = [x_1, x_2, ..., x_M]$ denote the transmitted symbols in the training sequence. Assume that there exist L paths. Thus, the receiver can receive L copies of $\mathbf{x}$, each traveling on one path and undergoing a response (i.e., distortion and attenuation) caused by the corresponding path. The vector $\mathbf{y}$ of received symbols is the convolution sum of the L copies of $\mathbf{x}$. Let $\mathbf{h} = [h_1, h_2, ..., h_L]^T$ be the channel impulse response, where h_i is the response of the i-th path. Assuming an additive white Gaussian noise (AWGN) channel, the received symbols $\mathbf{y}$ can be represented by [34]

$$\mathbf{y} = \mathbf{h} * \mathbf{x} + \mathbf{n}, \tag{1}$$

where $\mathbf{n}$ is the white Gaussian channel noise and $*$ is the convolution operator. The matrix form of Equation (1) is

$$\mathbf{y} = \begin{bmatrix} x_1 & 0 & \cdot & 0 \\ x_2 & x_1 & \cdot & \cdot \\ \cdot & x_2 & \cdot & 0 \\ \cdot & \cdot & \cdot & x_1 \\ x_M & \cdot & \cdot & x_2 \\ 0 & x_M & \cdot & \cdot \\ \cdot & 0 & \cdot & \cdot \\ 0 & 0 & \cdot & x_M \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \cdot \\ \cdot \\ h_L \end{bmatrix} + \mathbf{n} \tag{2}$$

Rewriting Equation (2) in a compact matrix form gives us

$$\mathbf{y} = \mathbf{X}\mathbf{h} + \mathbf{n}, \tag{3}$$

where $\mathbf{X}$ is a $(L + M - 1) \times L$ Toeplitz matrix, containing L delayed versions of the transmitted symbols $\mathbf{x}$, and $\mathbf{y}$ is a vector consisting of $(L + M - 1)$ received symbols.

Estimation: Two types of estimators are generally used to estimate $\mathbf{h}$ from Equation (3): least-square (LS) estimator and linear minimum mean squared error (LMMSE) estimator [4]. If the statistical distribution of the channel impulse responses and noise are unknown, the LS estimator is usually used. If the statistical distribution of the channel impulse responses and noise are known, this a priori information can be exploited to decrease the estimation error, and the LMMSE estimator is often used for the estimation [39].

For the LS estimator, the estimation result is given by [35]:

$$\hat{\mathbf{h}}_{LS} = (\mathbf{X}^H\mathbf{X})^{-1}\mathbf{X}^H\mathbf{y}, \tag{4}$$

where $\mathbf{X}^H$ is the conjugate transpose of $\mathbf{X}$ and $()^{-1}$ is the matrix inverse operation. For the LMMSE estimator, the estimation result is [11]:

$$\hat{\mathbf{h}}_{LMMSE} = \mathbf{R}_h(\mathbf{R}_h + \sigma_n^2(\mathbf{X}\mathbf{X}^H)^{-1})^{-1}\hat{\mathbf{h}}_{LS}, \tag{5}$$

where $\mathbf{R}_h$ is the channel correlation matrix (i.e., the statistical expectation of $\mathbf{h}\mathbf{h}^H$) and σ_n^2 is the variance of the channel noise.

III. Mimicry Attack

In this section, we present the mimicry attack against link signature. We focus on the attack against the scheme in [30] and then extend it to the schemes in [22] and [42].

A. Overview

The root cause of mimicry attack is the linear relationship between the transmitted symbols $\mathbf{x}$, the received symbols $\mathbf{y}$, and the link signature $\mathbf{h}$, as indicated in Equation (3).

Let $\mathbf{y}_t$ and $\mathbf{y}_a$ denote the received symbols that are from the transmitter and the attacker, respectively. The attacker's goal in mimicry attack is to make $\mathbf{y}_a$ approximately the same as $\mathbf{y}_t$. Thus, when the receiver attempts to extract the link signature from the attacker's symbols $\mathbf{y}_a$, it will get a link signature that is similar to the one estimated from $\mathbf{y}_t$.

The attacker should meet two requirements to launch a mimicry attack: First, the attacker needs to roughly know the received symbols $\mathbf{y}_t$. Second, the attacker needs to manipulate its own symbols, such that when those manipulated symbols arrive at the receiver, they are similar to $\mathbf{y}_t$ (i.e., $\mathbf{y}_a \approx \mathbf{y}_t$).

B. Learning Symbols $\mathbf{y}_t$

Intuitively, an attacker should be co-located with the receiver in order to know $\mathbf{y}_t$. However, it is possible for an mimicry attacker to avoid satisfying such an extreme condition. The attacker can learn $\mathbf{y}_t$ by placing a sensing device in the proximity of the receiver. For the sake of presentation, we call this device the *symbol sensor*. It records the symbols sent from the transmitter and reports them to the attacker through any available communication channel. Since the symbol sensor is geographically close to the receiver, its received symbols are similar those received by the receiver, and can be used as $\mathbf{y}_t$.

Experimental Observation: We perform experiments to examine the similarity between symbols received by the symbol sensor and those received by the receiver. We use three USRP2s as the transmitter, the receiver, and the symbol sensor. The communication frequency is 2.4GHz, and the distance D between the transmitter and the receiver is 6.5 meters. The transmitter sends a training sequence of 63 symbols for 1000 times. We place the symbol sensor relatively far away from the receiver, i.e., the distance between the receiver and the symbol sensor is as far as the distance D between the receiver and the transmitter, and we record the normalized amplitude of each received symbol. Then, we reduce the the distance between the receiver and the symbol sensor to $0.4D$, and repeat the experiment.

Figures 1 and 2 plot the average amplitude of received symbols. When the distance between the symbol sensor and the receiver is large, symbols received by the symbol sensor are different from those received by the receiver (Figure 1). However, when the symbols sensor is close to the receiver, both received symbols become similar to each other (Figure 2).

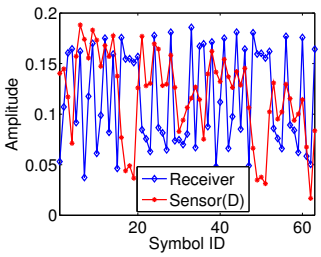

Fig. 1. The sensor is D meters away from the receiver

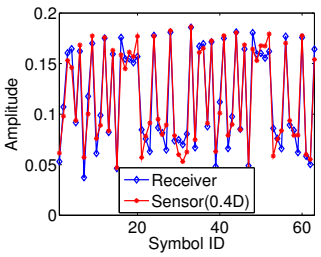

Fig. 2. The sensor is $0.4D$ meters away from the receiver

It should be noted that radio channels correlate within several wavelengths [10]. If the symbol sensor is within several wavelengths away from the receiver, the symbols received by the symbol sensor will be similar to those received by the receiver. Our experiment system uses GHz frequency, and thus has a very short wavelength. However, a couple of radio systems work at MHz frequencies [2]. Those systems have a longer wavelength than our experiment system (i.e., the wavelength of MHz is 10^3 times of that of GHz), and thus they enable a much larger distance between the symbol sensor and the receiver.

Reducing Time Delay: In the above method, the attacker needs to wait for the symbol sensor to report $\mathbf{y}_t$, which causes time delay for the attacker. Indeed, before the symbol sensor receives $\mathbf{y}_t$ or even before the transmitter starts current transmission, the attacker may directly use the mathematical model $\mathbf{y}_t = \mathbf{h}_t * \mathbf{x} + \mathbf{n}$ to estimate $\mathbf{y}_t$, where $\mathbf{x}$ is the training sequence used by the communication system and $\mathbf{h}_t$ is the link signature between the transmitter and the receiver. Again, due to proximity reason, the attacker can use the link signature between the transmitter and the symbol sensor as $\mathbf{h}_t$ to calculate $\mathbf{y}_t$. In this way, the symbol sensor only needs to estimate the link signatures between itself and the transmitter, and report the estimated link signatures to the attacker from time to time, and the attacker can directly calculate $\mathbf{y}_t$ anytime it needs.

C. Manipulating Transmitted Symbols

The symbols $\mathbf{y}_a$ received from the attacker can be represented as $\mathbf{y}_a = \mathbf{h}_a * \mathbf{x}_a + \mathbf{n}_a$, where $\mathbf{x}_a$, $\mathbf{h}_a$, and $\mathbf{n}_a$ are the symbols transmitted by the attacker, the link signature of the attacker, and the channel noise, respectively. To make $\mathbf{y}_a$ equal to $\mathbf{y}_t$, the attacker can treat $\mathbf{x}_a$ as a unknown variable, and solve it from the following equation

$$\mathbf{h}_a * \mathbf{x}_a + \mathbf{n}_a = \mathbf{y}_t, \tag{6}$$

where link signature $\mathbf{h}_a$ of the attacker can be obtained from the symbol sensor as well. The solution to this equation enables $\mathbf{y}_a$ to be similar to or the same as the transmitter's symbols $\mathbf{y}_t$. As a result, the link signatures that are estimated from $\mathbf{y}_a$ will also be close to those estimated from $\mathbf{y}_t$. In Theorem 1, we give a way to solve Equation (6) and calculate $\mathbf{x}_a$ from $\mathbf{y}_t$.

Theorem 1: Let $\mathbf{y}_t$ and $\mathbf{y}_a$ denote the received symbols that are sent by the transmitter and the attacker, respectively. Further let $\mathbf{H}_a$ be the Toeplitz matrix of the attacker's link signature. If $\mathbf{x}_a = (\mathbf{H}_a^H \mathbf{H}_a)^{-1} \mathbf{H}_a^H \mathbf{y}_t$, then $\mathbf{y}_a = \mathbf{y}_t$.

Proof: Let $\mathbf{x}_a = [x_{a1}, x_{a2}, ..., x_{aM}]^T$ denote the symbols transmitted by the attacker, and $\mathbf{h}_a = [h_{a1}, h_{a2}, ..., h_{aL}]^T$ denote the link signature of the attacker. We have

$$\begin{aligned}
\mathbf{y}_t &= \mathbf{h}_a * \mathbf{x}_a + \mathbf{n}_a = \mathbf{X}_a \mathbf{h}_a + \mathbf{n}_a \\
&= \begin{bmatrix} x_{a1} & 0 & \cdot & 0 \\ x_{a2} & x_{a1} & \cdot & \cdot \\ \cdot & x_{a2} & \cdot & 0 \\ \cdot & \cdot & \cdot & x_{a1} \\ x_{aM} & \cdot & \cdot & x_{a2} \\ 0 & x_{aM} & \cdot & \cdot \\ \cdot & 0 & \cdot & \cdot \\ 0 & 0 & \cdot & x_{aM} \end{bmatrix} \begin{bmatrix} h_{a1} \\ h_{a2} \\ \cdot \\ \cdot \\ h_{aL} \end{bmatrix} + \mathbf{n}_a \\
&= \begin{bmatrix} h_{a1} & 0 & \cdot & 0 \\ h_{a2} & h_{a1} & \cdot & \cdot \\ \cdot & h_{a2} & \cdot & 0 \\ \cdot & \cdot & \cdot & h_{a1} \\ h_{aL} & \cdot & \cdot & h_{a2} \\ 0 & h_{aL} & \cdot & \cdot \\ \cdot & 0 & \cdot & \cdot \\ 0 & 0 & \cdot & h_{aL} \end{bmatrix} \begin{bmatrix} x_{a1} \\ x_{a2} \\ \cdot \\ \cdot \\ x_{aM} \end{bmatrix} + \mathbf{n}_a \\
&= \mathbf{H}_a \mathbf{x}_a + \mathbf{n}_a.
\end{aligned}$$

Therefore, $\mathbf{y}_t = \mathbf{h}_a * \mathbf{x}_a + \mathbf{n}_a \Leftrightarrow \mathbf{y}_t = \mathbf{H}_a \mathbf{x}_a + \mathbf{n}_a$. We can solve $\mathbf{x}_a$ from $\mathbf{y}_t = \mathbf{H}_a \mathbf{x}_a + \mathbf{n}_a$. Since $\mathbf{n}_a$ is unknown, we use the standard least square approach [35] to solve $\mathbf{x}_a$. Specifically, we minimize $||\mathbf{y}_t - \mathbf{H_a}\hat{\mathbf{x}}_\mathbf{a}||^2$, where $\hat{\mathbf{x}}_a$ is the approximate solution of $\mathbf{x}_a$. The minimization yields

$$\hat{\mathbf{x}}_a = (\mathbf{H}_a^H \mathbf{H}_a)^{-1} \mathbf{H}_a^H \mathbf{y}_t. \tag{7}$$

■

Elements in $\mathbf{x}_a$ are already physical layer symbols, and thus they can be transmitted directly. The attacker does not need to modulate them again for transmission.

D. Initial Validation via CRAWDAD Data Set

As an initial validation, we simulate mimicry attacks using CRAWDAD data set [37], which contains over 9,300 link signatures measured in an indoor environment with obstacles (e.g., cubicle offices and furniture) and scatters (e.g., windows and doors). Our simulation is done in MATLAB 7.7.0.

Simulation Process: We randomly pick two link signatures from the CRAWDAD data set, and use them as the transmitter's link signature $\mathbf{h}_t$ and the attacker's link signature $\mathbf{h}_a$, respectively. We generate a training sequence of 256 bits using a pseudorandom number generator, and compute $\mathbf{y}_t$ according to Equation (3) (i.e., $\mathbf{y}_t = \mathbf{X}\mathbf{h}_t + \mathbf{n}_t$, where $\mathbf{X}$ is the Toeplitz matrix of the training sequence and $\mathbf{n}_t$ is a randomly generated Gaussian noise). To launch mimicry attacks, the attacker needs to calculate transmitted symbols $\mathbf{x}_a$. Therefore, assuming that the attacker knows $\mathbf{y}_t$, we calculate $\mathbf{x}_a$ based on Theorem 1 (i.e., $\mathbf{x}_a = (\mathbf{H}_a^H \mathbf{H}_a)^{-1} \mathbf{H}_a^H \mathbf{y}_t$, where $\mathbf{H}_a$ is the Toeplitz matrix of $\mathbf{h}_a$). The attacker transmits $\mathbf{x}_a$ to the receiver, and the corresponding received symbols $\mathbf{y}_a$ can also be computed by Equation (3) (i.e., $\mathbf{y}_a = \mathbf{X}_a \mathbf{h}_a + \mathbf{n}_a$, where $\mathbf{X}_a$ is the Toeplitz matrix of $\mathbf{x}_a$). Finally, the receiver estimates link signatures from $\mathbf{y}_a$.

Simulation result: Figure 3 shows that the transmitter's link signature $\mathbf{h}_t$, the attacker's link signature $\mathbf{h}_a$, and the attacker's forged link signature estimated from $\mathbf{y}_a$. (All link signatures are normalized by amplitude and each contains 50 elements.) Though the attacker's original link signature is different from the transmitter's, after the mimicry attack, the forged link signature is very close to the transmitter's link signature.

The CRAWDAD data set has 5 link signatures for each link. We randomly pick one as the comparison base $\mathbf{h}_t$. The Euclidean distance between the other four link signatures and $\mathbf{h}_t$ ranges between 1.1432e-004 and 2.2558e-004. The Euclidean distance between attacker's true link signature and $\mathbf{h}_t$ is 1.2e-003, which is out of the above range. However, the Euclidean distance between attacker's forged link signature and $\mathbf{h}_t$ is 2.0584e-004, which falls into the normal range of variation of the link signature. This implies that when the forged link signature from the attacker is used, it will be considered valid.

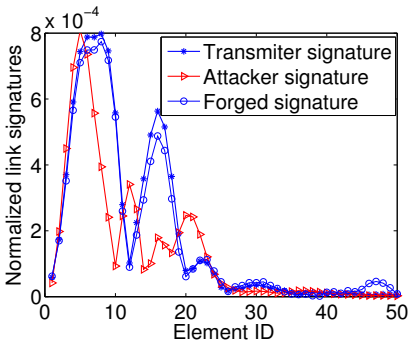

Fig. 3. Mimicry attack using CRAWDAD data

This initial validation demonstrates the theoritical correctness of mimicry attacks. In Section V, we will further inves-

tigate the practical impact of mimicry attacks with real world experiments.

E. Extending Attack to Multiple Tone Probing based Link Signature

As discussed earlier, there are two other link signature schemes [22], [42] besides the one we just attacked [30]. The scheme in [22], referred to as the *multiple tone probing based link signature*, uses complex gains at different frequencies to build a link signature, and the scheme in [42] is an integration of the techniques in [30] and [22]. In the following, we show that we can extend the mimicry attack to also compromise the multiple tone probing based link signature, thus making all existing link signature schemes vulnerable.

In the multiple tone probing link signature, K carrier waves are simultaneously transmitted to the receiver. The transmitted signal is $s(t) = \sum_{\kappa=1}^{K} e^{j2\pi f_\kappa t}$ [22], [42], where f_κ is the frequency of the κ-th carrier. Each carrier wave undergoes an attenuation at its center frequency [42]. Thus, the received signal is $r(t) = \sum_{\kappa=1}^{K} H_\kappa e^{j2\pi f_\kappa t}$, where H_κ is the complex channel gain reflecting the amount of attenuation on the κ-th carrier. The vector $\mathbf{h} = [H_1, H_2, ..., H_K]$ of the complex channel gain is used as the link signature [22], [42].

The mimicry attack identified in this paper can also be adapted to attack the multiple tone probing based link signature. Let $\mathbf{h}_a = [H_{a_1}, H_{a_2}, ..., H_{a_K}]$ denote the multiple tone link signature between the attacker and the receiver, and $\mathbf{h}_t = [H_{t_1}, H_{t_2}, ..., H_{t_K}]$ denote the one between the transmitter and the receiver. With the knowledge of $\mathbf{h}_t$, the attacker can generate a signal in the following form,

$$s_a(t) = \sum_{\kappa=1}^{K} \frac{H_{t_\kappa}}{H_{a_\kappa}} e^{j2\pi f_\kappa t} = \sum_{\kappa=1}^{K} \frac{\|H_{t_\kappa}\|}{\|H_{a_\kappa}\|} e^{j(2\pi f_\kappa t + \theta_{a_\kappa} - \theta_{t_\kappa})},$$

where $\|.\|$ denote the magnitude of a complex number, θ_{a_κ} and θ_{t_κ} are the phases of H_{a_κ} and H_{t_κ}, respectively. After channel attenuation, the corresponding received signal is

$$r_a(t) = \sum_{\kappa=1}^{K} \frac{H_{t_\kappa}}{H_{a_\kappa}} H_{a_\kappa} e^{j2\pi f_\kappa t} = \sum_{\kappa=1}^{K} H_{t_\kappa} e^{j2\pi f_\kappa t},$$

which equals to the signal $r_t(t)$ received from the transmitter. As a result, the multiple tone link signature estimated from $r_a(t)$ is the same as that estimated from $r_t(t)$.

Since the link signature scheme in [42] is essentially an integration of the scheme in [30] and [22], the above result also makes the scheme in [42] vulnerable to mimicry attacks.

IV. Time-synched Link Signature

In this section, we develop a novel time-synched link signature to defend against the mimicry attacks. A key feature of this new mechanism is the integration of cryptographic protection and time factor into wireless link signatures.

We first clarify our assumptions and the threat model.

A. Assumptions and Threat Model

We assume that there are a *Transmitter* and a *Verifier*, who share a secret key K that is only known to them. The Transmitter sends packets, or more precisely, physical layer *frames*, to the Verifier, who then verifies if these frames are directly transmitted by the Transmitter. We assume that the attacker can eavesdrop, overhear, and jam wireless communications. However, we assume that the attacker cannot compromise the Transmitter or the Verifier, and thus does not know their secret. The attacker's goal is to generate or forward frames to the Verifier and convince the verifier that the frames were transmitted by the Transmitter.

Given that a cryptographic authentication mechanism (e.g., digital signature, Message Integrity Code (MIC)) can be added to a message to detect forged messages, the main threat is from the frames that are originally generated by the Transmitter but forwarded by the attacker.

Two cases are of particular interest: (1) when the attacker can jam and replay the Transmitter's frames (i.e., the jam-and-replay attack [15]), and (2) when the Transmitter and the Verifier are out of communication range, but the jammer forwards frames from the Transmitter to the Verifier, attempting to impersonate the Transmitter and mislead the Verifier to believe that the Transmitter is still nearby. In the other cases where the Verifier can receive the original transmission by the Transmitter, a duplication detection mechanism (e.g., sequence number) along with authentication can properly detect the frames forwarded by the attacker.

We assume that the attacker can launch *frame repeater attacks*. That is, the attacker is able to receive a frame transmitted by the Transmitter and then forward the frame to the Verifier. Such frame repeaters are widely available commercially (e.g., various brands of 802.11 repeaters).

We also assume that the attacker can launch physical layer *symbol repeater attacks*. That is, the attacker can observe the transmission of each physical layer symbol, which may represent one or multiple bits in the frame, and then forward the symbol to the Verifier directly. Such repeaters can be developed using noise canceling techniques and proper positioning of antennas, as described in [7]. Compared with frame repeater attacks, symbol repeater attacks are much harder to defend against.

Link signatures are specific to wireless communication channels, and usually require a training phase for two nodes to learn the actual value. The attacker may target at either the *training phase* to mislead the Transmitter and the Verifier about their link signature, or the *operational phase* (as described in Section III) when the link signature is used for wireless channel authentication. Thus, a secure link signature has to protect both the training and the operational phases.

B. Design Strategy

The fundamental reason for the mimicry attack is that the attacker can establish a set of equations based on two pieces of information: (1) the knowledge of the training sequence and (2) the Transmitter's signal (i.e., physical layer symbols) at

the Verifier's location. These allow the attacker to manipulate the transmitted physical layer symbols so that a frame sent by the attacker has a valid link signature.

Initial Idea: To defend against this attack, our strategy is to deprive the attacker at least one of these two pieces of information. It is in general very difficult to prevent a passive attacker from receiving signals (and then extracting valid link signatures). However, it is possible to prevent the attacker from knowing the training sequences.

Thus, our initial idea is to use *unpredictable*, *dynamic*, and *authenticated* training sequences for extracting link signatures from wireless packets (frames).

Detecting Frames Forwarded by Attackers: It is not hard to realize that simply using unpredictable, dynamic, and authenticated training sequences is still insufficient. The attacker can receive and analyze the Transmitter's signal to learn the training sequence. If the Verifier cannot receive the original transmission (e.g., due to jam-and-replay attack or being outside of the Transmitter's signal range), the attacker can still forge link signatures by manipulating and forwarding a frame received from the Transmitter.

To handle this threat, we propose to bring "time" into the scheme. We assume the Transmitter and the Verifier have synchronized clocks. (As we will show in the proposed scheme, in the training phase the Transmitter and the Verifier will synchronize their clocks to meet this assumption.) The Transmitter may include a timestamp in the transmitted frame, which indicates the time when a particular bit or byte called the *anchor* (e.g., the Start of Frame Delimiter (SFD) field in an IEEE 802.11 or 802.15.4 frame [19]) is transmitted over the air. We assume that the Transmitter can use authenticated timestamping techniques (e.g., [38]) to ensure that the timestamp precisely represents the point in time when the anchor is transmitted in air. Upon receiving a frame, the Verifier can use the timestamp included in the frame and the time when it receives the frame, which should also be obtained through PHY or Medium Access Control (MAC) layer timestamping [38], to estimate the frame traverse time. An overly long time indicates that the frame has been forwarded by an intermediate attacker.

Using MAC layer timestamping can defend against the frame repeater attack fairly well. For example, in an 802.11g wireless network, which supports 54 Mbps bandwidth, the transmission of a 100-byte frame takes about $14.8\mu s$. To maximize the chance to detect forwarded frames, we may force certain critical frames (e.g., those used to extract physical layer properties such as Received Signal Strength (RSS)) to have a large frame size. In the case of 802.11g, the maximum frame size is 2,346 octets (bytes), which will take about $347.6\mu s$ to transmit. A frame repeater will have to double the transmission time, giving the Verifier a good chance to detect the extra delay and thus detect the attack.

Defending against Physical Layer Symbol Repeater Attacks: A physical layer symbol repeater attack is much harder to detect than frame repeater attacks. If the attacker knows where the training sequence is located in the frame, she can start repeating the physical layer symbols right after she finishes receiving all symbols corresponding to the training sequence. This reduces the delay that the physical layer symbol repeater has to tolerate to the transmission time of only the training sequence, which could be much shorter than the transmission time of the entire frame.

To defend against such physical layer symbol repeater attacks, we propose to integrate a third idea into the scheme, that is, to make the location of the training sequence *unpredictable until the end of the frame transmission*. Specifically, we propose to insert the training sequence at a *randomly* selected location in the payload, and place this location, which can be represented as the offset from the start of the frame header, at the end of the frame. In order for a physical layer symbol repeater to mimic the link signature of the Transmitter, she has to manipulate the physical layer symbols corresponding to the training sequence in a frame. If the location of the training sequence is not revealed until the end of the frame, the attacker will have to wait until the end of the transmission to learn it. This forces a physical layer symbol repeater attack to degenerate into a frame repeater attack, which can be handled as discussed earlier.

Minimum Frame Length: If a frame payload is too short, the Verifier may have difficulty seeing the extra delay caused by a frame repeater. One solution is to pad extra bits into the frame payload if the frame length is less than a minimum frame length.

The minimum frame length can be determined based on the errors of the time synchronization and time measurement. Assume that the maximum errors in clock discrepancy and transmission time are e_δ and e_τ, respectively. Further assume that the maximum time measurement errors in the Transmitter and the Verifier are e_T and e_V, respectively. Thus, the maximum error that the Verifier has to tolerate is $e_{all} = e_\delta + e_\tau + e_T + e_V$. Assume that the data rate of the wireless communication is R. It is easy to see that when the frame length is greater than the minimum frame length $L_{min} = R \cdot e_{all}$, the Verifier is guaranteed to detect frames forwarded by frame repeaters.

It has been demonstrated in an implementation of Radio Frequency (RF) distance bounding protocol [32] that nanosecond processing delay is feasible to achieve. The time-synched link signature requires much less precision in time synchronization between the Transmitter and the Verifier. For example, even assuming e_{all} is between $1\mu s$ and $10\mu s$, in a 54 Mbps 802.11g wireless network, L_{min} will range between 7 bytes and 68 bytes.

Overall Design: Figure 4 illustrates how these ideas can be integrated into a physical layer protocol. The upper portion of Figure 4 shows the layout of a typical physical layer frame, which consists of a series of preamble symbols, the frame header, and the payload. To detect frames forwarded by attackers, we include in each frame a timestamp t_s, which indicates the transmission time of the frame. To defend against physical layer repeater attacks, we include the randomly generated offset P of the training sequence in each frame at

the end of the frame (to force the attacker to wait until the end of frame transmission).

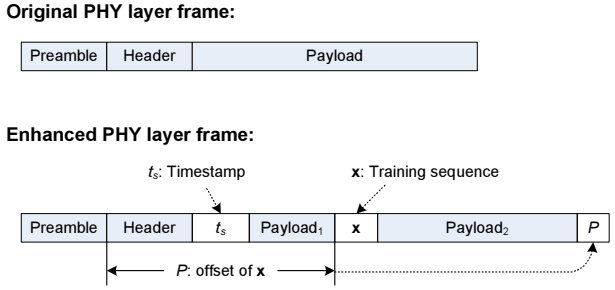

Fig. 4. PHY layer frame format: Integrating dynamic training sequence with random offset

Assume the Transmitter and the Verifier share a secret key K. Given the shared secret key, there are may ways to generate an unpredictable, dynamic, and authenticated training sequence. One simple method is to piggyback the authentication of the frame with the generation of the training sequence, that is, to use the MIC of the entire frame as the training sequence $\mathbf{x}$. In situations where there is a mismatch between the MIC and the training sequence (e.g., when a longer training sequence is needed), we can simply generate the training sequence as $\mathbf{x} = F(K, t_s)$, where F is a pseudo-random generator, and compute the frame MIC separately. The use of K and t_s makes $\mathbf{x}$ dynamic and unpredictable, and the frame MIC allows $\mathbf{x}$ to be authenticated.

In the following, we present the details of the training and the operational phase in time-synched link signature.

C. Training Phase

The training phase is intended for the Verifier to collect enough information from the Transmitter so that the Verifier can verify the link signatures of the future frames from the Transmitter. The Verifier should obtain the valid link signature from the Transmitter whenever the link signature between them may change. This can be accomplished by executing the training phase protocol periodically or whenever one of them moves.

In the training phase, the Verifier needs to synchronize its clock with the Transmitter, and obtain the link signature for the current communication channel. At the same time, the Verifier needs to confirm that there is no successful attack during the training phase.

We use the classic time synchronization technique to estimate the clock discrepancy between the Transmitter and the Verifier as well as the frame traverse time. This approach has been used in the past for secure time synchronization (e.g., [15], [38]). For the sake of presentation, we refer to the point in time when the anchor (e.g., the SFD field) in a frame is transmitted or received as the *transmission time* or the *receiving time* of this frame. Specifically, the Verifier sends a *request frame* to the Transmitter, and at the same time records the frame transmission time t_1 in the Verifier's local clock. When the Transmitter receives the request frame, it records the receiving time t_2 of this frame, and then sends a *reply frame* to the Verifier, in which t_2 and the transmission time t_3 of the reply frame, which are both measured in the Transmitter's clock, are included. Finally, the Verifier receives the reply frame and records the receiving time t_4 in its clock. The clock discrepancy δ between the Verifier and the Transmitter and the one-way frame traverse time τ can then be estimated as follows (e.g., [15], [28], [38]):

$$\delta = \frac{(t_2 - t_1) - (t_4 - t3)}{2}; \quad \tau = \frac{(t_2 - t_1) + (t_4 - t_3)}{2}. \quad (8)$$

The Transmitter and the Verifier face a subtle difficulty in time synchronization due to the need of authentication: The timestamps t_1 and t_3 should be the actual transmission time of the request and reply frames; however, the MIC computation requires the timestamp value before the actual transmission. Fortunately, a solution has been previously developed for this problem [38]. It is observed that in the physical layer protocol component, all computation is deterministic if the wireless channel is available for transmission. Thus, we can estimate how much time the deterministic processing will take before (the anchor of) the frame is transmitted and thus determine the transmission time before computing the frame MIC. If the frame transmission does not happen due to channel unavailability, the estimation, the computation of the MIC, and the transmission can be repeated.

To defend against potential frame repeater and physical layer symbol repeater attacks, we use the design given in Section IV-B. That is, the Transmitter pads the reply frame payload so that after all necessary components of the frame are included, the frame length is at least the minimum frame length L_{min}. The Transmitter uses the MIC of the entire frame as the link signature training sequence, and places it at a random offset in the frame payload. Finally, the Transmitter places the random offset at the end of the frame.

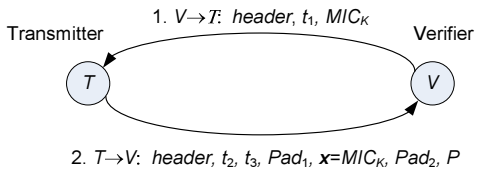

Fig. 5. Training phase protocol

Figure 5 shows the training phase protocol between the Transmitter and the Verifier.

Training Request: The Verifier sends the first training request frame to the Transmitter, which includes the frame header, the transmission time t_1 of this frame, and the frame MIC that covers the entire frame (excluding the preambles). Upon receiving of the request frame, the Transmitter immediately records the receiving time t_2 of the frame, and authenticates the request frame by verifying the MIC.

We can also filter some bogus requests before verifying the MIC. Though a clock discrepancy between the Transmitter and the Verifier is expected, there is usually a maximum clock

discrepancy δ_{max}. If $(t_2 + \delta_{max} - t_1)$ is too large, it is likely that the request frame is a replay of a previous request frame, and should be discarded without verification.

Training Reply: Upon verifying an incoming training request frame, the Transmitter should send back a training reply frame. The Transmitter should include time t_2 and the actual transmission time t_3 of the reply frame in the frame. The Transmitter also pads the frame payload to at least the minimum frame length L_{min} and randomly selects an offset P to place the training sequence as discussed earlier. The Transmitter then leaves a placeholder (e.g., all 0's) in place of the training sequence and computes the frame MIC using the shared key K. Finally, the Transmitter places the frame MIC as the training sequence $\mathbf{x}$ in the reply frame and sends it over the air.

Once the Verifier receives the training reply frame, the Verifier first computes the clock discrepancy δ and the one-way transmission time τ according to Equation (8). If τ is greater than a threshold τ_{max}, which is the maximum possible direct transmission time, the Verifier should consider the reply frame as possibly forwarded by the attacker and discard it. Otherwise, the Verifier locates the frame MIC by following the offset P at the end of the frame, authenticates the frame MIC using the shared key K, and uses the frame MIC, which is also the training sequence $\mathbf{x}$, to extract the link signature. The Verifier may run the training phase protocol several times to get a better quality link signature. As a result, the Verifier obtains the valid link signature to perform physical layer authentication of future frames from the Transmitter.

D. Operational Phase

Once the Verifier obtains the clock discrepancy and the valid link signature from the Transmitter, the two nodes can go into the operational phase, during which the Verifier can use this link signature to verify frames that require physical layer authentication.

Transmitter: To defend against the threats discussed in Section IV-A, the Transmitter follows the design shown in Figure 4. Specifically, the Transmitter randomly selects an offset in the frame payload to include the field for the training sequence[1]. The Transmitter also includes the transmission time t_s, places the offset P at the end of the frame, and computes the frame MIC using the shared secrete key K, with a placeholder (e.g., all 0's) for the training sequence. The Transmitter then uses the frame MIC as the training sequence $\mathbf{x}$, puts it in the frame, and sends the frame over the air. Similar to the training phase, the Transmitter estimates the frame transmission t_s based on the current time and the estimated duration for the deterministic MIC computation.

Verifier: When the Verifier receives the frame, it immediately records the receiving time t_r. The Verifier then retrieves the frame transmission time t_s from the received frame and estimates the frame traverse time $\tau = t_s - t_r - \delta$, where δ is the clock discrepancy between the Verifier and the Transmitter learned in the training phase. If τ is greater than the threshold τ_{max}, the maximum possible direct transmission time, the Verifier should consider the frame possibly forwarded by the attacker and discard it. Otherwise, the Verifier locates the frame MIC by using the offset P at the end of the frame, verifies the frame MIC using the shared key K, and then uses the frame MIC as the training sequence to extract the link signature. Finally, the Verifier compares this link signature with the one derived during the training phase. The frame is accepted if this link signature does not deviate from the valid one learned in the training phase. Otherwise, the frame is considered forged and discarded.

[1]Note that the *training* sequence is necessary for the Verifier to extract the link signature. It is used in the operational phase even though its name has "training" in it.

E. Security Analysis

Now let us examine the ability of the time-synched link signature to defend against the malicious threats.

First of all, the time-synched link signature uses a training sequence authenticated with a shared secret key only known to the Transmitter and the Verifier, and the training sequence changes from frame to frame due to the involvement of the timestamp in the computation of the training sequence. Thus, the training sequence is authenticated, dynamic, and unpredictable. This effectively prevents the attacker from forging frames with training sequences of its choice. The only choice left for the attacker is to reuse and manipulate valid frames from the Transmitter.

The use of random offset for the training sequence in the frame payload forces the attacker to wait for the end of the frame transmission to understand where the training sequence is located in the frame. As a result, the attacker cannot launch physical layer symbol repeater attacks and at the same time manipulate the training sequence correctly to bypass link signature verification. The attacker may still perform the frame repeater attack. However, due to the enforcement of the minimum frame length, a frame forwarded by a frame repeater will introduce at least the amount of delay caused by the receiving of the frame, which is detectable by the Verifier.

The attacker may launch a probabilistic mimicry attack by randomly guessing the location of the training sequence and forging the frame symbols accordingly. Indeed, the attacker may also try to overestimate the length of the training sequence and perform the forgery. If the assumed training sequence $\mathbf{y}'_t$ is a superset of the actual one $\mathbf{y}_t$ (i.e., $\mathbf{y}_t$ is a subsequence of $\mathbf{y}'_t$), due to the linear property of Equation (7), the forged symbols $\hat{\mathbf{x}}'_a$ will also include $\hat{\mathbf{x}}_a$ as a subsequence. This will allow the attacker's symbols to be accepted by the receiver. However, the attacker cannot delay the transmission of a frame for L_{min} or more; otherwise, its interference will be detected. This means that the probability for the attacker to succeed is at most $p = \frac{L_{min}-|x|+1}{F-|x|+1}$ when L_{min} is greater than or equal $|x|$, where $|x|$ and F are the lengths of the training sequence and the frame payload, respectively. When L_{min} is less than $|x|$, the probability of a successful mimicry attack degrades to 0. For example, in a 54Mbps 802.11g wireless network, if we can achieve $2.96\mu s$ precision in the time

synchronization and measurement error (i.e., $e_{all} = 2.96\mu s$ and $L_{min} = 159.84$ bits) and use HMAC-SHA1 to generate the training sequence (i.e., $|x|$=160 bits), the probabilistic mimicry attack is guaranteed to fail.

Nevertheless, the probabilistic mimicry attack does increase the requirement for time synchronization. In other words, the Transmitter and the Verifier need to obtain fine-grained time synchronization so that the success probability of a probabilistic mimicry attack becomes negligible.

V. Experimental Evaluation

We have implemented the link signature scheme in [30], the mimicry attack, and the newly proposed time-synched link signature. We have also implemented the frame repeater attack, which can be used along with the mimicry attack. Our prototype uses the second generation of USRPs (i.e., USRP2) [24], which are equipped with AD and DA converters as the RF front ends, and XCVR2400 daughter boards operating in the 2.4 GHZ range as transceivers. The software implementation is based on GNURadio [1].

USRP2s are capable of processing signals up to 100MHz wide. Such a high bandwidth enables the use of them for capturing multipath effects and measuring link signatures. GNURadio configuration requires setting the values of interpolation (decimation) rate at the transmitter (receiver) and the number of samples per symbol. If the values of those parameters are too high, the actual bandwidth will be significantly reduced. To guarantee the capture of multipath effect, we set those parameters the minimum values allowed by GNURadio (i.e., 5 for the interpolation and decimation rate, and 2 for the number of samples per symbol).

We have performed experiments using this prototype system to evaluate the impact of mimicry attacks and the performance of time-synched link signature.

A. Evaluation Methodology

Evaluation Scenarios: Our prototype system consists of a transmitter, a receiver (i.e., the verifier in case of time-synched link signature), and an attacker. Each node is a USRP2 connected to a commodity PC. The receiver estimates the received link signatures and compares them with the transmitter's link signature.

We consider three scenarios in our evaluation: (1) *normal scenario*, (2) *forgery scenario*, and (3) *defense scenario*. In a normal scenario, the attacker simply sends original symbols to the receiver. In both the forgery and the defense scenarios, the receiver functions as the symbol sensor for the attacker. It estimates the link signatures for the attacker and provides this link signature and the received symbols from the transmitter to the attacker. Upon obtaining this information, the attacker launches the mimicry attack, during which it transmits manipulated symbols to the receiver. However, the forgery scenario uses the previous link signature scheme in [30], while the defense scenario uses the newly proposed time-synched link signature.

Evaluation Metrics: Intuitively, the attacker wants to reduce the difference between its own link signature and the transmitter's link signature, whereas the defense method aims to increase this difference to alert the receiver. Thus, the link difference between both the attacker's and the transmitter's link signatures can visually reveal the impact of mimicry attacks and the effectiveness of the defense method.

The receiver measures N link signatures of the transmitter, where we set N to 50 in our evaluation. Let $\mathcal{H}$ denote the set formed by the N link signatures. We collect 500 link signatures from the attacker, and calculate the link difference $d_{a,\mathcal{H}}$ between $\mathcal{H}$ and them. For the purpose of comparison, we also let the receiver collect 500 link signatures from the transmitter, and calculate the link difference $d_{t,\mathcal{H}}$ between $\mathcal{H}$ and those newly collected link signatures.

According to [30], the above link difference (i.e., $d_{a,\mathcal{H}}$ and $d_{t,\mathcal{H}}$) is calculated using $\frac{1}{\sigma}\min_{\mathbf{g}\in\mathcal{H}}\|\mathbf{g}-\mathbf{h}\|$, where $\mathbf{h}$ is a link signature of the attacker or the transmitter, and σ is the *historical average difference* between link signatures in $\mathcal{H}$ [30], which is given by $\sigma = \frac{1}{N(N-1)}\sum_{\mathbf{g}\in\mathcal{H}}\sum_{\mathbf{q}\in\mathcal{H}-\mathbf{g}}\|\mathbf{q}-\mathbf{g}\|$.

Link signature based authentication serves as a detector that decides whether or not a received signal is from the desired source. Thus, besides link difference, we also use detection rate P_D (i.e., the rate that an attacker's link signature is successfully detected by the receiver) and false alarm rate P_{FA} (i.e., a transmitter's link signature is incorrectly identified as the attacker's link signature) as two additional evaluation metrics. Finally, we measure the time delay introduced by the transmitter and the attacker to assess how well the frame repeaters can be detected.

B. Evaluation Results

We now show how mimicry attacks affect the link difference, false alarm rate, detection rate, and the tradeoff between the detection and the false alarm rates in the normal, forgery, and defense scenarios.

1) Link Difference: Figures 6, 7, and 8 show the link difference for the attacker $d_{a,\mathcal{H}}$ and that for the transmitter $d_{t,\mathcal{H}}$ in the normal, forgery, and defense scenarios, respectively. In the normal scenario, we see in Figure 6 that $d_{a,\mathcal{H}}$ is generally larger than $d_{t,\mathcal{H}}$. The histograms $d_{a,\mathcal{H}}$ and $d_{t,\mathcal{H}}$ are shown in Figure 9. Most of the transmitter's link difference is less than 0.6, whereas most of the attacker's link difference is larger than 0.6. Thus, based on the value of link difference, the receiver can achieve a high accuracy in distinguishing between the transmitter and the attacker.

In the forgery scenario, the attacker launches mimicry attacks to make its own link signatures similar to the transmitter's link signatures. We see in Figure 7 that $d_{a,\mathcal{H}}$ decreases to the same level as $d_{t,\mathcal{H}}$, and $d_{a,\mathcal{H}}$ and $d_{t,\mathcal{H}}$ substantially overlap with each other. The histogram of $d_{a,\mathcal{H}}$ (i.e., the top graph in Figure 10) shows that the link difference distribution of the attacker gets very close to that of the transmitter. The mimicry attack reduces the difference between the attacker's

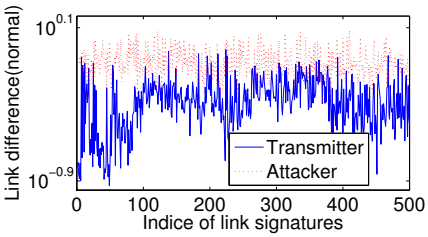

Fig. 6. Link difference for the transmitter's and the attacker's link signatures in normal scenario

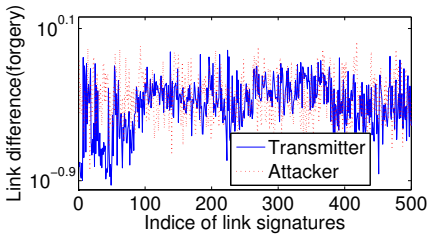

Fig. 7. Link difference for the transmitter's and the attacker's link signatures in forgery scenario

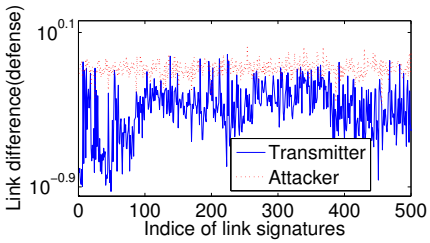

Fig. 8. Link difference for the attacker's link signatures in forgery and defense scenarios

link signatures and the transmitter's link signatures, leading to high false negative rate at the receiver.

In the defense scenario, as indicated in Figure 8, the use of time-synched link signature increases $d_{a,\mathcal{H}}$ of forged link signatures. In particular, the mean value of $d_{a,\mathcal{H}}$ under defense and forgery scenarios are 0.7368 and 0.4648, respectively. The histogram of $d_{a,\mathcal{H}}$ in the defense scenario (i.e., the bottom graph in Figure 10) shows that the link difference computed from a majority of forged signatures is larger than 0.6. Thus, the receiver can again distinguish between the transmitter and the attacker with low error rate.

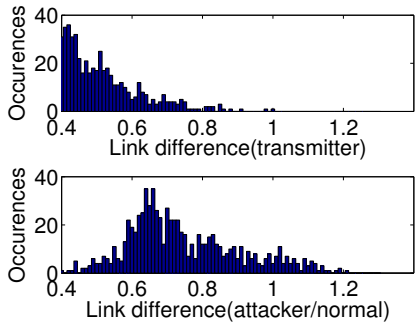

Fig. 9. Histograms of link difference for the transmitter's and the attacker's link signatures in normal scenario

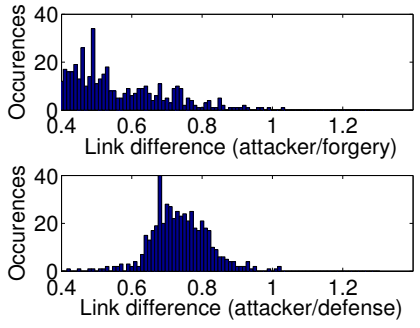

Fig. 10. Histograms of link difference for the attacker's link signatures in forgery and defense scenarios

2) Detection and False Alarm Rates: As mentioned earlier, a history of N transmitter's link signatures is measured and stored at the receiver, and the receiver computes the link difference d between a newly measured link signature and history link signatures. In our experiment, we follow the same detection rule as in [30]: If d is smaller than a certain threshold r, the receiver concludes that this link signature is from the transmitter; otherwise, the receiver assumes it is from the attacker.

Let N_{FA} denote the number of link signatures that are actually from the transmitter but incorrectly identified as from the attacker, and N_D denote the number of link signatures that are from the attacker and detected by the receiver. The false alarm rate P_{FA} is calculated as the ratio of N_{FA} to the total number of the transmitter's link signatures, and the detection rate P_D is computed as the ratio of N_D to the total number of the attacker's link signatures.

Figure 11 shows P_{FA} and P_D as a function of the threshold r. A large threshold can reduce false alarm rate P_{FA}, whereas a small threshold can increase detection rate P_D. A common decision is to pick the *operational threshold* as the point where the distance between P_{FA} and P_D is the largest (i.e., $P_D - P_{FA}$ is the largest).

The operational thresholds of the normal, defense, and forgery scenarios are 0.5811, 0.6329, and 0.4182, respectively. For the normal scenario, the corresponding P_{FA} and P_D achieved by the operational threshold are $P_{FA} = 0.0936$ and $P_D = 0.9064$. The defense scenario slightly outperforms the normal scenario in terms of reducing P_{FA} and increasing P_D with the operational threshold, leading to $P_{FA} = 0.0635$ and $P_D = 0.9365$.

The forgery scenario has the worst performance. With the operational threshold, $P_{FA} = 0.4045$ and $P_D = 0.5935$. Note that, in our experiment, a link signature is either from the transmitter or from the attacker, and thus the probability that a blind guess hits the true source of this link signature is 0.5. The false alarm rate P_{FA} and detection rate P_D in the forgery scenario are just slightly better than a blind guess.

Figure 12 shows the receiver operating characteristic (ROC) curves for the normal, forgery, and defense scenarios, in which the P_{FA} and P_D are the x-axis and y-axis, respectively. The curve representing the defense scenario is on the top-left corner of the figure, indicating good performance of the time-synched link signature.

3) Frame Time Delay: The proposed time-synched link signature uses estimated frame traverse time to filter out frames forwarded by the attacker. In this experiment, we measure the time delay of frames from the transmitter and the attacker, respectively, to examine this approach. In our experiment, the frame length is 190 bits and the transmission rate is set to

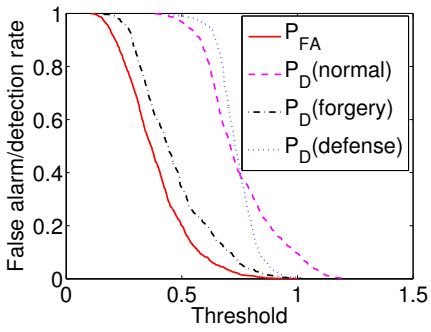

Fig. 11. False alarm rate P_{FA} and detection rate P_D as a function of threshold

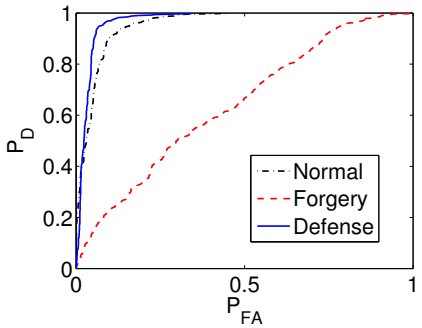

Fig. 12. Tradeoff between false alarm and detection rate for normal, forge, and defense scenarios

500Kbps. The transmitter sends 130 frames, and the attacker forwards all of them.

Figure 13 shows that the time delays of frames forwarded by the attacker significantly exceed those of the frames directly sent by the transmitter. Our further analysis indicates that the ratio of the attacker's delay to the transmitter's delay ranges between 2.2 and 2.6, indicating that the forwarding by the attacker approximately doubles the delay.

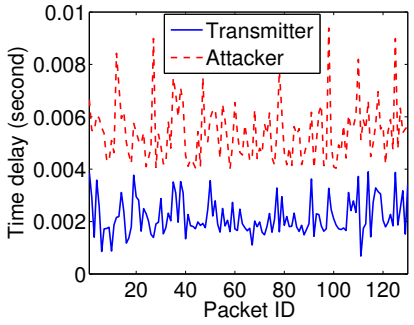

Fig. 13. Delay of original and forwarded frames

We would like to caution the reader that due to the limitation of USRP2, our implementation does not perform physical layer timestamping. Thus, the time delays measured in our experiments include the processing time on the PC and the USRP2 boards. In a real deployment, physical layer timestamping is necessary to increase the precision of time synchronization and time measurement.

VI. Related Work

Wireless Transmitter Authentication: Existing non-cryptographic techniques for authenticating wireless transmitters can be divided into three categories [5]: software fingerprinting (e.g., [14], [21], [29]), location distinction (e.g., [6], [22], [30], [42]), and radiometric identification (e.g., [5], [9], [33])

In software fingerprinting approaches, discrepancies in software configuration are used as fingerprints to distinguish between wireless nodes [5]. For example, Franklin et al. [14] proposed to use the implementation dependent differences among device drivers to identify 802.11 nodes. Kohno et al. [21] proposed to use clock skews in TCP and ICMP timestamps to fingerprint networked devices.

In location distinction based authentication, a signal is authenticated by verifying whether it originates from the expected location of the transmitter. RSS (e.g., [6]) and link signatures have been used to enable such location distinction [30]. The RSS based methods directly estimate the location of a signal origin using the RSS values. However, such methods can be defeated with an array antenna, which can fake arbitrary source locations [30]. The link signature based approaches authenticate the channel characteristics between the transmitter and the receiver [22], [30], [42]. In this paper, we showed that all these link signature schemes are vulnerable to mimicry attacks. Our newly proposed time-synched link signature is developed to fill this gap.

In radiometric identification approaches, the distinctive physical layer characteristics exhibited by wireless devices are utilized to distinguish between them. Transient based techniques (e.g., [9], [33]) identify a wireless device by looking at the unique features "during the transient phase when the radio is turned on" [8]. Modulation based techniques (e.g., [5]) measure differentiating artifacts of individual wireless frames in the modulation domain to identify the device.

Attacks on Radiometric Identification: Recently, it was demonstrated in [8] and [12] that radiometric identification techniques were vulnerable to impersonation attacks. The results in [8] revealed that both transient and modulation based techniques are vulnerable to impersonation attacks, though transient-based techniques are harder to reproduce. Edman et al. [12] showed that an attacker can significantly reduce the accuracy of such techniques by simply using a commodity RF hardware platform. These works are complementary to ours in this paper.

VII. Conclusion

In this paper, we identified the mimicry attack against existing wireless link signature schemes [22], [30], [42]. Our results indicated that some assumptions that form the foundation of the previous link signature construction are not true. In particular, we demonstrated that an attacker *can* forge an *arbitrary* link signature as long as it knows the legitimate signal at the receiver's location, and the attacker does not have to be at exactly the same location as the legitimate transmitter to forge its link signature.

To provide physical layer authentication capability and defend against the mimicry attack, we proposed the novel time-synched link signature scheme by integrating cryptographic protection and time factor into wireless physical layer features. This new technique provides an effective and practical solution for authenticating physical layer wireless signals. We also performed an extensive set of experiments, and demonstrated both the feasibility of mimicry attacks and the effectiveness of time-synched link signature on the USRP2 platform running GNURadio.

REFERENCES

[1] GNU Radio - The GNU Software Radio. http://www.gnu.org/software/gnuradio/.

[2] U.s. frequency allocation chart. http://www.ntia.doc.gov/osmhome/allochrt.pdf.

[3] B. Azimi-Sadjadi, A. Kiayias, A. Mercado, and B. Yener. Robust key generation from signal envelopes in wireless networks. In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*, pages 401–410, 2007.

[4] M. Biguesh and A. B. Gershman. Training-based mimo channel estimation: A study of estimator tradeoffs and optimal training signals. *IEEE Transaction on Signal Processing*, 54(3):884–893, March 2006.

[5] V. Brik, S. Banerjee, M. Gruteser, and S. Oh. Wireless device identification with radiometric signatures. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking (MobiCom '08)*, pages 116–127, 2008.

[6] R. Chen, J. Park, and J. H. Reed. Defense against primary user emulation attacks in cognitive radio networks. *IEEE Journal on Selected Areas in Communications*, 26(1):25–37, 2008.

[7] J. Choi, M. Jain, K. Srinivasan, P. Levis, and S. Katti. Achieving single channel, full duplex wireless communication. In *Proceedings of the 16th ACM Mobicom (Mobicom '10)*, September 2010.

[8] B. Danev, H. Luecken, S. Čapkun, and K. El Defrawy. Attacks on physical-layer identification. In *In Proceedings of the 3rd ACM Conference on Wireless Network Security (WiSec '10)*, pages 89–98, March 2010.

[9] B. Danev and S. Čapkun. Transient-based identification of wireless sensor nodes. In *In Proceedings of ACM/IEEE Conference on Information Processing in Sensor Networks (IPSN'09)*, 2009.

[10] G. D. Durgin. *Space-Time Wireless Channels*. Prentice Hall PTR, 2002.

[11] O. Edfors, M. Sandell, J. J. van de Beek, S. K. Wilson, and P. O. Börjesson. Ofdm channel estimation by singular value decomposition. *IEEE Transaction on Communications*, 46(7):931–938, July 1998.

[12] M. Edman and B. Yener. Active attacks against modulation-based radiometric identification. *TR 09-02, Rensselaer Institute of Technology,*, 2009.

[13] D. Faria and D. Cheriton. Detecting identity-based attacks in wireless networks using signalprints. In *Proceedings of ACM Workshop on Wireless Security (WiSe 2006)*, pages 43–52, 2006.

[14] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. V. Randwyk, and D. Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. In *Usenix Security Symposium*, 2006.

[15] S. Ganeriwal, S. Capkun, C. Han, and M. B. Srivastava. Secure time synchronization service for sensor networks. In *Proceedings of 2005 ACM Workshop on Wireless Security (WiSe 2005)*, pages 97–106, September 2005.

[16] R. Gerdes, T. Daniels, M. Mina, and S. Russell. Device identification via analog signal fingerprinting: A matched filter approach. In *Proceedings of the 13th Annual Sympium on Network and Distributted System Security (NDSS '06)*, 2006.

[17] A. Goldsmith. *Wireless Communications*. Cambridge University Press, New York, NY, USA, 2005.

[18] A. Goldsmith. *Wireless Communications*. Cambridge University Press, August 2005.

[19] IEEE Std 802.15.4-2003. IEEE standard for information technology – telecommunications and information exchange between systems – local and metropolitan area networks – specific requirements – part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs).

[20] A. Kalamandeen, A. Scannell, E. de Lara, A. Sheth, and A. LaMarca. Ensemble: cooperative proximity-based authentication. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys '10)*, pages 331–344, 2010.

[21] T. Kohno, A. Broido, and K. C. Claffy. Remote physical device fingerprinting. *Dependable and Secure Computing*, 2(2):93–108, 2005.

[22] Z. Li, W. Xu, R. Miller, and W. Trappe. Securing wireless systems via lower layer enforcements. In *In Proceedings of ACM Workshop on Wireless Security (WiSe'06)*, 2006.

[23] Y. Liu, P. Ning, and H. Dai. Authenticating primary users' signals in cognitive radio networks via integrated cryptographic and wireless link signatures. In *Proceedings of 2010 IEEE Symposium on Security and Privacy (S&P '10)*, pages 286–301, May 2010.

[24] E. R. LLC. The USRP product family products and daughter boards. http://www.ettus.com/products. Accessed in April 2011.

[25] D. Loh, C. Cho, C. Tan, and R. Lee. Identifying unique devices through wireless fingerprinting. In *Proceedings of the first ACM Conference on Wireless Network Security (WiSec '08)*, pages 46–55, 2008.

[26] S. Mathur, W. Trappe, N. Mandayam, C. Ye, and A. Reznik. Radio-telepathy: extracting a secret key from an unauthenticated wireless channel. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking (MobiCom '08)*, pages 128–139, 2008.

[27] U. Maurer. Secret key agreement by public discussion from common information. *IEEE Transactions on Information Theory*, 39(4):733–742, 1993.

[28] D. Mills. Internet time synchronization: The network time protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, 1991.

[29] J. Pang, B. Greenstein, R. Gummadi, S. Seshan, and D. Wetherall. 802.11 user fingerprinting. In *In Proceedings of the 13th annual ACM international conference on Mobile Computing and Networking (MobiCom'07)*, 2007.

[30] N. Patwari and S. K. Kasera. Robust location distinction using temporal link signatures. In *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 111–122, New York, NY, USA, 2007. ACM.

[31] K. Rasmussen, C. Castelluccia, T. Heydt-Benjamin, and S. Čapkun. Proximity-based access control for implantable medical devices. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09)*, 2009.

[32] K. Rasmussen and S. Čapkun. Realization of rf distance bounding. In *Proceedings of the USENIX Security Symposium*, 2010.

[33] K. B. Rasmussen and S. Čapkun. Implications of radio fingerprinting on the security of sensor networks. In *In Proceedings of International ICST Conference on Security and Privacy in Communication Networks (SecureComm'07)*, 2009.

[34] R. Safaya. A multipath channel estimation algorithm using a kalman filter. http://www.ittc.ku.edu/research/thesis/documents/rupul_safaya_thesis.pdf.

[35] K. S. Shanmugan and A. M. Breipohl. *Random signals: detection, estimation, and data analysis*. Wiley, May 1988.

[36] C. Shen. Efficient utilization of channel state information in modern wireless communication systems. *Ph.D Dissertation, UCLA*, 2009.

[37] SPAN. Measured channel impulse response data set. http://span.ece.utah.edu/pmwiki/pmwiki.php?n=Main.MeasuredCIRDataSet.

[38] K. Sun, P. Ning, C. Wang, A. Liu, and Y. Zhou. TinySeRSync: Secure and resilient time synchronization in wireless sensor networks. In *Proceedings of 13th ACM Conference on Computer and Communications Security (CCS '06)*, pages 264–277, October/November 2006.

[39] Wikipedia. Channel state information, 2011. [Online; accessed 14-April-2011].

[40] C. Ye, A. Reznik, and Y. Shah. Extracting secrecy from jointly gaussian random variables. In *Proceedings of IEEE International Symposium on Information Theory (ISIT '06)*, pages 2593–2597, July 2006.

[41] K. Zeng, K. Govindan, and P. Mohapatra. Non-cryptographic authentication and identification in wireless networks. *Wireless Communications*, 17(5):56–62, October 2010.

[42] J. Zhang, M. H. Firooz, N. Patwari, and S. K. Kasera. Advancing wireless link signatures for location distinction. In *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking*, New York, NY, USA, 2008. ACM.