2013 IEEE 27th International Symposium on Parallel & Distributed Processing

On Feasibility of Fingerprinting Wireless Sensor Nodes Using Physical Properties

Xiaowei Mei, Donggang Liu
Computer Science and Engineering Department
The University of Texas at Arlington

Kun Sun
Center for Secure Information Systems
George Mason University

Dingbang Xu
Computer Science Department
Governors State University

***Abstract*—Fingerprinting wireless devices using physical properties has been recently suggested as an alternative for device identification and authentication. It has been found that the clock skew caused by the frequency discrepancy of the quartz crystals in different devices can be used as a reliable source for fingerprinting. Researchers have studied the application of the clock skew-based fingerprinting in sensor networks and claimed that it can detect fake identities, wormholes, and node replicas. However, the study in this paper draws a completely opposite conclusion, i.e., the clock skew of sensor nodes can be easily forged by adversaries to evade the detection. This paper then studies the feasibility of using the distribution of signal power in space to fingerprint sensor nodes. The result shows that a sensor node's signal power distribution in space is not only reliable for being used as a source for fingerprinting but also very hard to forge. Finally, the paper discusses the application of using signal power distribution for detecting various attacks as well as the limitations and open problems.**

I. Introduction

In wireless networks, node identification and authentication is always a fundamental security issue. For example, in sensor networks, it is important to ensure that the node you are communicating with is really who he claims to be. Traditionally, cryptographic techniques are used for node authentication, assuming that the two communicating parties share a secret key or know each other's public key. Thus, the problem of authenticating a node using cryptography essentially becomes a key management issue. Many key management techniques have been proposed to provide keys needed for authentication [16], [5], [13], [12].

Recently, fingerprinting using physical properties has been suggested by researchers as an alternative to achieve node identification and authentication [11], [10], [1], [9], [17], [3], [18]. The rationale is that different nodes exhibit slightly different physical properties (e.g., clock) after they are manufactured; these physical properties can be measured to identify and authenticate a wireless node. The benefits are as follows. First, it does not rely on cryptography. Hence, it can work even if we do not have a central trusted server for certifying public keys or pre-loading key materials. Second, it can be combined with traditional cryptography-based authentication to provide the so-called *two-factor* authentication. For example, the attacker may capture a sensor node and create many replicas in the network [15]. These replicas have all the secrets learned from the compromised sensor node and can thus evade all cryptography-based authentication schemes. With fingerprinting as a second factor, it is possible to detect replicas since it is difficult to create nodes with exactly the same physical properties as the compromised node.

In general, the physical property used for fingerprinting should at least satisfy the following two requirements: *reliability* and *unforgibility*. Reliability means that the fingerprint created from the physical property should be stable enough so that we do not get a lot of false positives. In other words, it should be unlikely that the same node produces different fingerprints at different times. Unforgibility means that it is infeasible or very costly for the attacker to forge the physical property (or the signature created from such property) so that we don't get a lot of false negatives.

This paper focuses on fingerprinting nodes in sensor networks. The physical properties studied in the past include the *clock skew* [11], [10], [1], [9], [17] and the radio *time interval error* (TIE) [3], [18]. Measuring TIE requires sophisticated equipment that is not available on sensor nodes; we do not consider this a viable source for fingerprinting in sensor networks. The clock skew is mainly caused by the frequency discrepancy between the quartz crystals in different devices. The result of such discrepancy is that the clock at one node runs a little bit faster (or slower) than the clock at the other node. Such clock speed difference is called the *clock skew*. Several studies have found that the clock skew is quite reliable in differentiating sensor nodes [9], [17], and can therefore be used to authenticate nodes as well as detect the sybil attacks [14], wormhole attacks [8], and node replication attacks [15].

However, these existing studies on clock skew-based fingerprinting overlooked the unforgibility requirement. Although the clock skew is reliable, its measurement at a remote node could be fooled. Basically, a malicious node first measures its clock skew with respect to the target node and then use such clock skew to compensate its local time. In this way, the malicious node can be well time-synchronized with the target node and thus impersonate the target node without being detected. We demonstrate the feasibility and effectiveness of such fingerprint forgery attack through extensive experiments.

We then investigate the feasibility of using the space distribution of signal power as the physical property for fingerprinting sensor nodes. For each sensor node, this physical property mainly captures the power distribution of its radio signal within its radio range when this node is transmitting

1530-2075/13 $26.00 © 2013 IEEE
DOI 10.1109/IPDPS.2013.15

a message. In this paper, for simplicity, we only pick a number of positions in the field and use the characteristics of the received signal strengths (RSS) at these positions to represent such physical property. Our observation is that the RSS distribution (called RSSD) for the same sensor node is often quite stable, while the RSS distributions of different nodes are often quite different from each other. We confirm this observation through experiments. In addition, this paper also discusses the unforgibility of such physical property and shows that it is very hard, if not infeasible, for any adversary to produce the same RSSD as any benign sensor node in the network. In conclusion, we believe that RSSD is a promising candidate for fingerprinting sensor nodes.

Our contributions are as follows. First, in contrast to the common belief that clock skew can be used for fingerprinting sensor nodes, we show in this paper that it can be easily manipulated by adversaries to evade all detection methods. As a result, it cannot be used to detect fake identities, wormholes, or node replicas. Second, we also show through extensive experiments that RSSD is quite promising for fingerprinting sensor nodes in a static sensor network. We also analyze its unforgibility and conclude that it is very hard for adversaries to create the same RSSD fingerprint as any benign node in the network.

The remainder of the paper is organized as follows. The next section reviews the network and adversary models. Section III describes an attack to forge the clock skew and shows the feasibility and effectiveness of this attack through theoretical analysis and extensive experiments. In Section IV, we present our study on the feasibility of using RSSD for fingerprinting. In Section V, we review existing fingerprinting techniques for wireless networks. In Section VI, we conclude this paper and discuss future work.

II. Network and Adversary Models

In this paper, we consider a network of sensor nodes that are deployed in a large area of interest for applications such as border security and monitoring of critical facilities like nuclear power plants. We assume that fingerprinting sensor nodes is required by the application for authentication purpose. However, it is used to complement, rather than as a complete replacement of, other authentication approaches such as cryptography-based authentication. For example, it can be used as a second authentication factor if a cryptography-based authentication scheme fails due to the leakage of cryptographic keys when sensor nodes are compromised.

We assume that the network is free of node compromise during the initial deployment. Some initial network parameters as well as the fingerprinting signatures of sensor nodes can be correctly extracted during this phase. We consider this compromise-free initial deployment phase a reasonable assumption. The reasons are as follows. First, this phase is often short. It is possible for the network owner to keep an eye on the field for signs of malicious activities. Second, a physical contact is often needed in order to capture and compromise a sensor node, and such physical contact will certainly expose the adversary himself in the field. However, we do assume that malicious sensor nodes might be already present in the field during the initial deployment since it is quite possible for the attacker to figure out the possible area of deployment and place his sensor nodes before the actual deployment. Certainly, when the initial deployment is over, we may have some compromised sensor nodes in the field.

We assume that the attacker's investment is limited. In other words, we are not expecting that a fingerprinting method shall establish a piece of evidence that is as strong as a cryptography-based scheme can provide. We say that a fingerprinting method is good enough if it is very hard or costly for a resource-limited adversary to duplicate the physical property of a sensor node in the field. If a fingerprint can be easily forged by one or a few colluding sensor nodes, then we say that this scheme is broken. Therefore, by "costly", we mean that it should take an adversary many (e.g., 10) times more resources than one single sensor node to forge the fingerprint of a given sensor node with a reasonably large probability.

III. Feasibility of Clock Skew-Based Fingerprinting

In this section, we will first review how to measure the clock skew between two communicating parties. We will then describe the attack on the measurement of clock skew and study how effective this attack is through experiments. Finally, we will discuss the implication of our attack on the detection of various attacks in sensor networks.

A. Measuring Clock Skew

Clock skew measures the difference between the clocks in devices. The difference is mainly due to the inherent physical impairments during the manufacturing of the oscillators in devices. There will be offset between the time from the clock of one sensor node and the time from the clock of another. The growing speed of this offset turns out to be a stable value, which is hence defined as clock skew. This inherent physical property of oscillators has been explored by researchers to fingerprint a physical device.

The clock skew between two nodes A and B is estimated as follows. Suppose that node B is trying to measure its clock skew with respect to node A. Node A first send n messages to B, each carrying a timestamp indicating the local time at A when the message is sent. Let T_i^A denote the timestamp included in the i-th message from node A. When B receives the i-th message from A, it records the local time T_i^B when the message is received. Certainly, the timestamp in the message should be the local time when the message hits the channel, and the arrival time at B should be the local time when the message is just received at the physical layer.

In the end, node B will have the following two timing sequences: $\{T_1^A, T_2^A, ..., T_n^A\}$ and $\{T_1^B, T_2^B, ..., T_n^B\}$. Based on these two sequences, Kohno et al. [11] produces n points $\{X_i, Y_i\}_{i=1,2,...,n}$ in a two-dimensional space, where

$$\begin{cases} X_i &= T_i^B - T_1^B \\ Y_i &= T_i^A - T_1^A - (T_i^B - T_1^B) \end{cases} \quad (1)$$

Y_i is also called the *clock offset* of the i-th measurement. We then find a line $L_{B,A}$: $y = p_{B,A} \times x + b_{B,A}$ that best fits this set of n points using either linear programming method (LPM) or Least Square Fitting (LSF). In this paper, we use the Lease Square Fitting (LSF) algorithm. The slope $p_{B,A}$ is denoted as the estimated clock skew, which is the one that many researchers have used for fingerprinting. In addition to the slope $p_{B,A}$, the intercept $b_{B,A}$ is also a possible candidate for fingerprinting. Ideally, the intercept $b_{B,A}$ should be zero. However, in practice, due to some measurement errors, the LSF fitting algorithm often outputs a non-zero intercept. Thus, researchers in [1] have also pointed out that the intercept may be used for fingerprinting.

B. Our Proposed Attack

In this subsection, we will describe an attack to manipulate the clock skew as well as the intercept. Suppose now we have an attacker M who is trying to impersonate a benign node A during its communication with another benign node B. Let $p_{B,A}$ and $b_{B,A}$ denote the actual clock skew and intercept that B has estimated from its past communication with A. The goal of the attacker M is to create a sequence of timestamps for its messages that will result in the same clock skew $p_{B,A}$ and the same intercept $b_{B,A}$ at node B.

1) Forging Clock Skew: We will first explain how to forge the clock skew. Let us consider one of the messages sent from M to B. Suppose this message is sent at its local time t_m. Certainly, if M always includes t_m as the timestamp in the message every time, then node B will detect that someone is impersonating A since the clock skew computed will be different from $p_{B,A}$. As a result, the goal of M is to make sure that the timestamp included in the message is very close to the local time of A when the packet is sent by M. In other words, node M needs to first figure out what is the local time t_a at A when its local time is t_m and then include t_a (instead of t_m) in the message. If M does this for every message, then it can fool node B into believing that the clock skew is $p_{B,A}$.

Hence, the remaining question is the estimation of t_a. The basic idea to estimate t_a given t_m is to take advantage of the clock skew between nodes M and A. Node M can obtain this by talking to node A directly or eavesdropping on the communication between A and B. Suppose M has intercepted the messages from A to B and recorded its local time sequence $\{T_1^M, T_2^M, ..., T_n^M\}$. Node M can then estimate the clock skew $p_{M,A}$ in the same way as B estimates the clock skew between A and B. Then M will have a fitting line $L_{M,A}$: $y = p_{M,A} \times x + b_{M,A}$.

From t_a and t_m, we can produce the following point: $(X' = t_m - T_1^M, Y' = t_a - T_1^A - (t_m - T_1^M))$. This point should be very close to, if not exactly on, the line $L_{M,A}$. Consider the fact that the point $(X', p_{M,A} \times X' + b_{M,A})$ is on line $L_{M,A}$. We know that we have

$$p_{M,A} \times X' + b_{M,A} \approx Y' = t_a - T_1^A - (t_m - T_1^M)).$$

Thus, we have

$$t_a \approx (1 + p_{M,A}) \times (t_m - T_1^M) + T_1^A + b_{M,A} = t'_a. \quad (2)$$

In the above equation, we will notice that $p_{M,A}$, t_m, T_1^M, T_1^A, and $b_{M,A}$ are all known values to node M. In other words, the adversary can estimate t_a at node M. Therefore, node M only needs to use the estimated value t'_a as the timestamp in every message in order to impersonate node A.

2) Forging Intercept: Now let us look at how to forge the intercept. Assume that node B has received a sequence of n forged timestamps from the adversary. B will then derive n points $(X'_i, Y'_i)_{i=1,...,n}$. According to Equation 1, X'_i is purely determined by the local timestamp sequence at B, but Y'_i can be modified easily by the adversary. More specifically, if the adversary reduces the value of the first forged timestamp by a constant ϵ, then except the first point, which is always $(0, 0)$, all other points $(X'_i, Y'_i)_{i=2,...,n}$ will be moved up by ϵ. This means that the intercept will be approximated increased by ϵ. In addition, we can also see that this does not introduce much change to the clock skew since every point except the first one moves up or down by the same distance. Therefore, the adversary can increase or reduce the intercept at a fingerprinter by an arbitrary value without affecting the clock skew.

One may argue that the fingerprinter can always collect N ($N > n$) timestamps $\{T_1, T_2, ..., T_N\}$ and randomly pick a T_i from $\{T_1, T_2, ..., T_{N-n+1}\}$ as the starting timestamp. Thus, the sequence of timestamps used to estimate the clock skew will be $\{T_i, T_{i+1}, ..., T_{i+n-1}\}$. Since the choice of i is unknown to the attacker, it will be very difficult for him to increase or reduce the intercept by an arbitrary value. However, the problem of this idea is that the starting timestamp has a lot more impact on the intercept than other points. For example, if the second timestamp is reduced by a constant value, then only Y'_2 is reduced by a constant value. None of the other $n - 1$ points will change. It is thus unlikely that we will see much change in the intercept. In contrast, if the starting timestamp is changed, every other point will be changed accordingly. Hence, Our intuition tells us that selecting a different starting timestamp will greatly change the intercept and make it unreliable for fingerprinting. In fact, our later experiment confirms this intuition.

In the remainder of this section, we will investigate how successful our proposed attack can be through extensive experiments on TelosB motes.

C. Experiments

The platform we use is the TelosB mote running TinyOS [7]. Each sensor mote has an 8MHz Texas Instruments MSP430 microcontroller with 10KB RAM and 1 MB external flash memory for data logging, and has a high radio data rate of 250 kbps. Data collection and programming are via USB.

The accuracy of the clock skew depends largely on how precisely we can measure the exact moment when a packet hits the medium, and when it is received at the physical layer at the receiver. TinyOS 2.x provides an interface called $PacketTimeStamp < TMicro, uint32\_t >$ for TelosB motes through which we can get a 32-bit timestamp of microsecond resolution, which is accurate enough for our measurement. At the sender side we get the sending time by calling the $timestamp$ command of this interface in the

$sendDone$ event handler. The $timestamp$ command returns the time of actual transmission, which is stored into the timestamp field of the packet.

At the receiver side, the time information is obtained by calling the $timestamp$ command in the $receive$ event handler. One thing that should be noticed is that the local clock at the receiver may overflow or reset after the packet was timestamped at the sender, which will create an inconsistent sequence of timestamps. As a result, we need to call the $isValid$ command first and make sure that it returns TRUE so that the return value of the $timestamp$ command can be trusted.

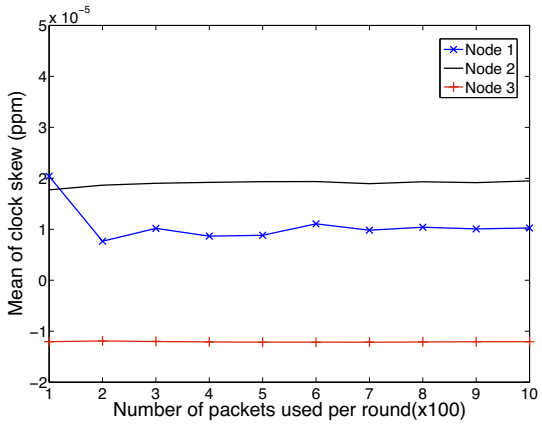
1) Experiment Overview: We consider the following scenario in our experiment. There are three nodes A, B, and M. Node A and B are benign nodes. Node B is connected to a PC for data analysis. Whenever it is asked, A will send n messages to B for fingerprinting, i.e., computing the clock skew $p_{B,A}$ and the intercept $b_{B,A}$. Node M is configured as a malicious sensor node; it will either eavesdrop on the communication between nodes A and B or try to talk to A directly to estimate the clock skew $p_{M,A}$ and the intercept $b_{M,A}$ with respect to node A. In this paper, we simply let node A send n messages to node M whenever it is asked. Whenever M is asked to impersonate node A, node M will send n messages to node B, each carrying a timestamp that is computed from Equation 2. Let p' and b' denote the clock skew and the intercept that node B estimated from the forged timestamp sequence from M, respectively.

In our experiment, we first evaluate how many messages are needed to have a reliable estimation of the clock skew. In other words, we will first try to find a good n for our experiment. We will then study if it is possible to use the intercept for fingerprinting sensor nodes. We will show that the intercept is reliable only when the starting timestamp is always fixed, which is consistent with our intuition in Section III-B2. We then study the distribution of $p_{B,A}$ and p' and show that it is statistically hard to distinguish them under our attack. In the end, we will show that our attack will also work even if we fix the starting timestamp and use the intercept for fingerprinting.

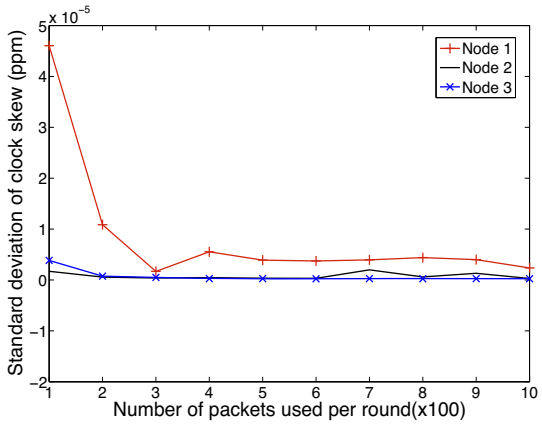
2) Experiment Results: The goal of the first set of experiments is to find a good n for our later experiments. n should be small for cost reason but still lead to a reliable estimation of the clock skew. For each n we pick, we estimate $p_{B,A}$ 20 times and then calculate the mean and standard deviation from these values. Note that in each round of estimation, A will send n messages to node B at a rate of one packet every second. Figures 1(a) and 1(b) shows the mean and standard deviation of $p_{B,A}$ under different values of n. From the figure, we notice that $n = 200$ gives a quite reliable estimation of the clock skew $p_{B,A}$. As a result, unless otherwise specified, we always send messages at a rate of one per second and set $n = 200$ in our later experiments.

The goal of the second set of experiments is to see if the intercept could be used to fingerprint a sensor node. We consider two cases in the experiment. In the first case, we always use a new sequence of n ($n = 200$) timestamps in each round of estimation. Thus, the starting timestamp in each round of estimation is different. We then pick three different nodes and measure their corresponding intercepts $1,000$ times. Figure 2(a) shows the cumulative distribution of the intercepts for different sensor nodes. We can clearly see that the intercept of the same node varies a lot, and the distributions of different nodes are very close. This indicates that statistically speaking, the intercept cannot be used to fingerprint sensor nodes if each estimation uses a new sequence of timestamps. In the second case, we always use the same starting timestamp. Hence, all old timestamps are included during the estimation. In other words, the i-th round of estimation will include all the timestamps used in the previous $i-1$ rounds (a total of $i \times n$ timestamps). We also pick three nodes and conduct 20 rounds of estimation for each of them. The result is given in 2(b). This figure shows that the intercept is very stable. However, our later experiment will show that the intercept can be easily forged by using the same attack mentioned in Section III-B1.

The objective of the third set of experiments is to show that it is statistically hard to distinguish p' and $p_{B,A}$. We first run one round of experiments to estimate $p_{B,A}$ and one round

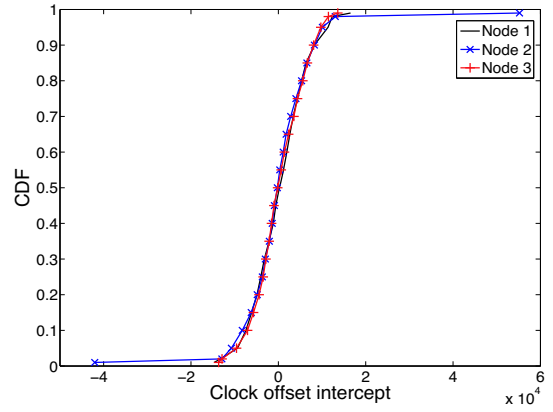


(a) Mean

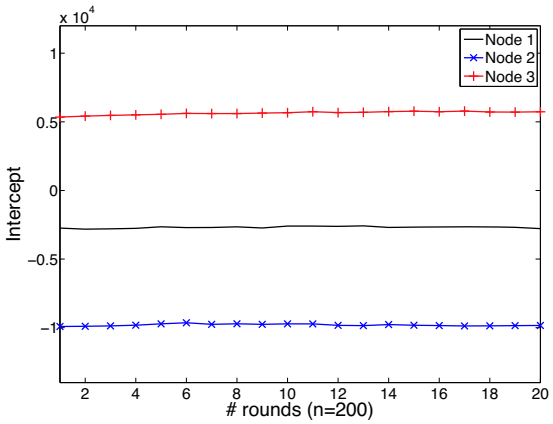


(b) Standard deviation

Fig. 1. Mean and standard deviation v.s. n

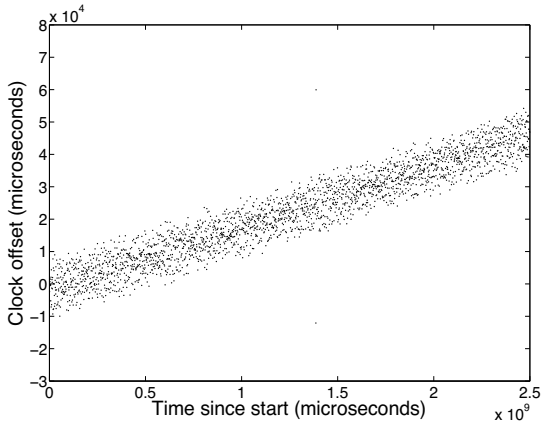


(a) Reset starting timestamp

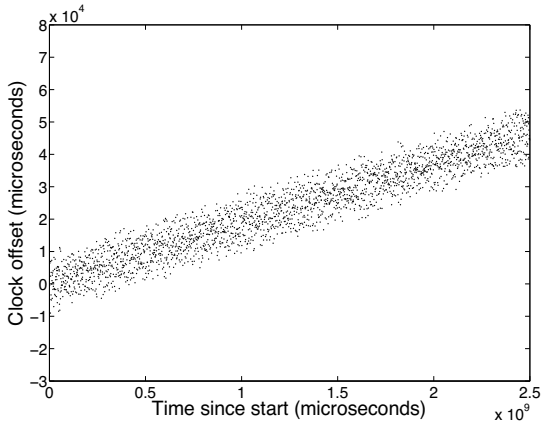


(b) Fixed starting timestamp

Fig. 2. Reliability of the intercept



(a) Real node



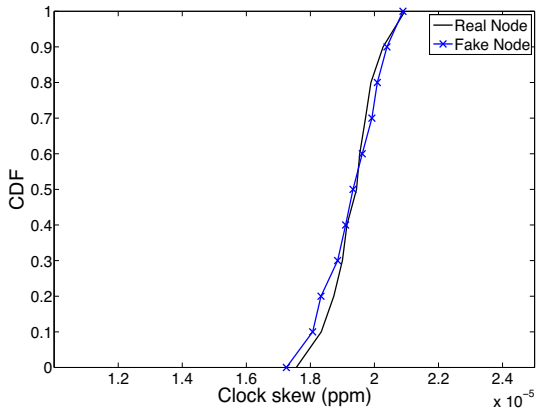
(b) Fake node

Fig. 3. Clock offsets for the real and fake nodes

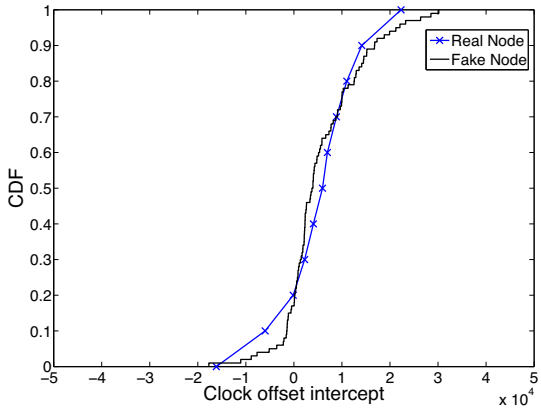
of experiment to estimate p' ($n = 2,500$). We use Equation 1 to derive n points from the n timestamps from A to B and display them in Figure 3(a). We then derive the n points from the n *forged* timestamps from M to B in the same way and display them in Figure 3(b). By comparing these two figures, we can see that it is very hard to distinguish the two sequences of clock offsets. In other words, our proposed attack works effectively in sensor networks. In addition, it also shows that our attack against clock skew-based fingerprinting is more effective than the attack described in [1], where one can clearly see the periodic dips in the offset sequence produced from the timestamp sequence of the fake node.

We then also conduct 100 rounds of estimation for p' and $p_{B,A}$ respectively ($n = 200$). Figure 4(a) displays the cumulative distributions for p' and $p_{B,A}$. We can clearly see that they are indeed very close to each other. This further shows that it is statistically hard to distinguish p' and $p_{B,A}$ when our attack is launched.

In the last set of experiments, we investigate the effectiveness of our attack in forging the intercept when the fingerprinter B always uses the same starting timestamp. In the experiment, we first let node A send a set of n packets to node B. This set of packets is also intercepted by the malicious node M. Node M can then estimate its clock skew and intercept with respect to node A. Then we let node M send n packets to B with timestamps modified according to Equation 2. When node B received the n forged timestamps, it will estimate the clock skew and the intercept using the newly received timestamp sequence and the old sequence received from the real sender A (a total of $2 \times n = 400$ timestamps from a mix of the new and the old timestamp sequences). We then repeat the process for the malicious node 100 times and compute 100 forged intercepts. (Each of these 100 estimations uses a new sequence of n timestamps and the same old timestamp sequence from A to B.) We then repeat the same process with the real node A and get 100 real intercepts. The cumulative distributions for these two sets (real and fake) of intercepts are shown in Figure 4(b). We can see that these two distributions are indeed very similar to each other. In addition, we also used the Kolmogorov-Smirnov test to quantitatively compare the real intercept data set with the forged intercept data set. The result shows that the maximum difference between the two



(a) Clock skews p' and $p_{B,A}$



(b) Intercepts b' and $b_{B,A}$

Fig. 4. Impact of our attack on clock skews and intercepts

cumulative distributions, D, is 0.1818 with a corresponding P-value of 0.985, which indicates that it is indeed very difficult to distinguish the real node from the fake node by inspecting the intercept. Thus, we know that our attack also works very well in forging the intercept.

D. Discussion

In the following, we will explain that our proposed attack allows an adversary to evade clock skew-based detection schemes in a sensor network. First, to detect sybil attacks, researchers have proposed to measure the clock skews corresponding to different identities. If these clock skews are very close to each other, then they must be created by the attacker [9], [17]. However, with our attack, the adversary can create a different clock skew for each of the fake identities using Equation 2. Hence, the detection of sybil attacks using clock skew will be ineffective.

Second, to detect wormhole attacks, researchers have proposed to look at the clock skews related to different identities. The identities that have the same clock skew must belong to the nodes at the other end of the wormhole [9], [17]. However, with our attack, instead of directly relaying every packet, the attacker can modify the timestamp of every packet from the same node, which is at the other end of the wormhole, to create a clock skew that is close to this node. This will evade the detection.

Third, the intuition behind the idea of detecting node replication attacks using clock skews is that different replicas exhibit different clock skews. If we notice two different clock skews for the same node, then we know that this node has been compromised and there are replicas in the network [9], [17]. However, with our attack, the adversary can modify the timestamps of all replicas such that they exhibit the same clock skew as the original compromised node. This will make the clock skew-based replica detection useless.

IV. Feasibility of RSSD-Based Fingerprinting

In the previous section, we have clearly shown that the clock skew is not secure enough for fingerprinting sensor nodes. In this section, we will look into another physical property – the distribution of signal power in space. We assume a static sensor network where a sensor node never changes its location after deployment. This assumption is often reasonable since moving around consumes significant energy.

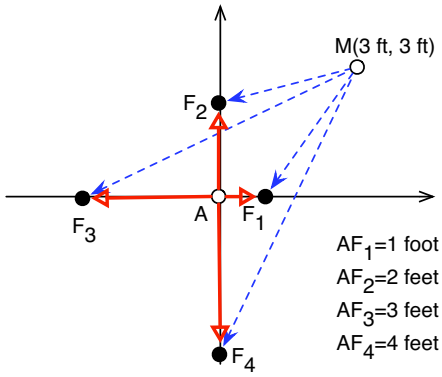


Fig. 5. An example of fingerprinting using RSSD

We will use the example in Figure 5 to explain our basic idea. In this example, the sender A is located at the center. Ideally, when a sensor node emits a radio signal, the signal power at distance d only depends on the transmission power and the distance d. Thus, we can pick four locations $\{F_1, F_2, F_3, F_4\}$ (marked as solid circles) that are within A's radio range to fingerprint this benign node A. The fingerprinting signature for node A will be a four-tuple (S_1, S_2, S_3, S_4) (this is just a signature in the ideal situation, the actual fingerprint signature in practice will include more information), representing the received signal strength measured at the four selected points when node A is transmitting messages. We assume that this signature is produced when there are no attacks.

Now suppose that a malicious node M is impersonating node A. After measuring the received signal strength at each selected point, we will have another four-tuple (S'_1, S'_2, S'_3, S'_4). We can see that this new four-tuple will

be very different from the original signature (S_1, S_2, S_3, S_4) unless M is at the same location as node A and is also transmitting messages at the same power level. This is because it is impossible that $|MF_i| = |AF_i|$ for all $i \in \{1, 2, 3, 4\}$ when M and A are located at different points.

Certainly, the adversary can simply place M at the same location as A such that the signature for M will be the same as that of A. However, this attack has the following constraints. First, since node M is really close to A, A will overhear the communication and immediately notice that someone is trying to impersonate him. Second, due to the environmental factors and the physical difference between sensor nodes (e.g., antenna orientation), the power distribution for different nodes will be quite different even if they are at the same location. We will confirm this in our later experiments.

We must recognize that in practice, the signal power at any point in the field will be influenced by many environmental factors besides the distance d from the sender. It is possible that the signal powers at different points are different even if their distances to the sender are the same. Nevertheless, we still believe that the signal power at the same point in the field should not change dramatically in a short period of time as long as the sender and the receiver do not move. In other words, the RSSD for a static sender should be quite stable even if we are not in an ideal environment. We thus strongly believe that our idea of using RSSD-based fingerprinting works. The remainder of this section will be focusing on the investigation of its feasibility through analysis and experiments in real environments.

A. Reliability of RSSD

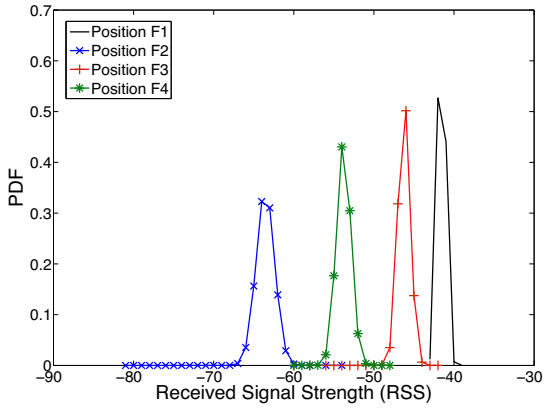


Fig. 6. RSS stability at the same measurement point

The objective of this section is to show that RSSD is reliable enough for fingerprinting. In our experiment, we place the sender at a fixed location and then select four other locations for measuring the signal strength as shown in Figure 5. Every packet from the sender will produce four RSS measurements. We then have the sender transmit 10,000 packets and show the distribution of received signal strength at these four points in Figure 6. According to this figure, we can see that the received signal strength at the same point is indeed stable given a static sender in a fixed environment. This figure also shows that the received signal strength changes noticeably when we are at a different measurement point. This is very critical since the difference between the RSSDs of two different nodes should be large enough for distinguishing them. Overall, we believe that RSSD is stable enough for fingerprinting. (We will discuss the unforgibility requirement later.)

B. RSSD-Based Fingerprinting

This section describes the RSSD-based fingerprinting procedure. Since we need the received signal strength characteristics at several points in the field, the RSSD-based fingerprinting scheme requires assistance from other nodes. For simplicity, we assume that every sensor node has a number of trusted "friends" in its neighborhood. These friends can be identified at the beginning of the deployment when there are no attacks. Of course, some friends may become compromised and untrustful after the sensor network is on operation for a while. However, detecting compromised friends and updating the friend list is a typical Byzantine General problem, which is out of the scope of this paper. We also assume that a trusted friend will always report the RSS characteristics honestly whenever being asked to. We consider the case where an honest node B wants to check whether the other party in its communication is indeed node A as the message indicates. We always ask the node being checked, i.e., the fingerprintee, to transmit the messages at a pre-determined power level during the fingerprinting process. Note that in our method, the fingerprinter does not move; it simply asks its neighboring friends to measure the RSSD values and collect these measurements for building the model and detecting fake nodes.

Building the model: We pre-select $k - 1$ friends in A's communication range for RSSD-based fingerprinting. The signature is a k-tuple $(\langle\mu_1, \sigma_1\rangle, ..., \langle\mu_k, \sigma_k\rangle)$. Each $\langle\mu_i, \sigma_i\rangle$ in the signature represents the actual RSS characteristic at the i-th selected measurement point; μ_i and σ_i are the mean and the standard deviation of the received signal strength measured at this selected measurement point. Specifically, we sample the received signal strength at i-th point m times and calculate the mean μ_i and the standard deviation σ_i from these m samples. We assume that node B already has the RSSD-based fingerprinting signature of the real node A, which was obtained when there are no attacks.

Fingerprinting a node: Suppose a malicious node is impersonating node A. From the packets sent by the malicious node, node B and its friends will sample the signal strength l times and compute the average. Let $(\bar{P}_1, \bar{P}_2, ..., \bar{P}_k)$ be the averages of the samples from these k points. The question now is: given $(\bar{P}_1, \bar{P}_2, ..., \bar{P}_k)$, how likely is the other party really node A? In this paper, we use the following simple rule to make the decision. For each $\bar{P}_i$, we check whether $\mu_i - 2\sigma_i \leq \bar{P}_i \leq \mu_i + 2\sigma_i$. If all answers are YES, then we say that the other party is indeed A.

1) False Positive Rate: For simplicity, we assume that the received signal strength follows the normal distribution. This

is a reasonable assumption based on our experiment result in Figure 6. We also assume that the signature accurately captures the distribution of the received signal strength at each selected point. Therefore, in a benign situation, we know that the average $\bar{P}_i$ will follow the normal distribution $\mathcal{N}(\mu_i, \frac{\sigma_i^2}{l})$. Thus, the probability that $\mu_i - 2\sigma_i \leq \bar{P}_i \leq \mu_i + 2\sigma_i$ can be estimated by $\texttt{erf}(\sqrt{2 \times l})$. Hence, the probability that all answers are YES can be estimated by $\texttt{erf}^k(\sqrt{2 \times l})$. As a result, the false positive rate is $FP = 1 - \texttt{erf}^k(\sqrt{2 \times l})$. For example, suppose $k = 4$ and $l = 4$, then $FP \approx 1 - (0.99994)^4 \approx 0.0024$.

2) Parameter k and l: We first discuss how to set k. Obviously, k should be at least 3 since otherwise, there will be at least one another location that have the same distances to the measurement points. Hence, the attacker only needs to put the fake node at one of these points to impersonate node A. On the other hand, k should be small enough so that it is feasible for B to get the fingerprint in a short period of time and without adding too much overhead. In summary, we believe that it is reasonable to set k to 3, 4 or 5.

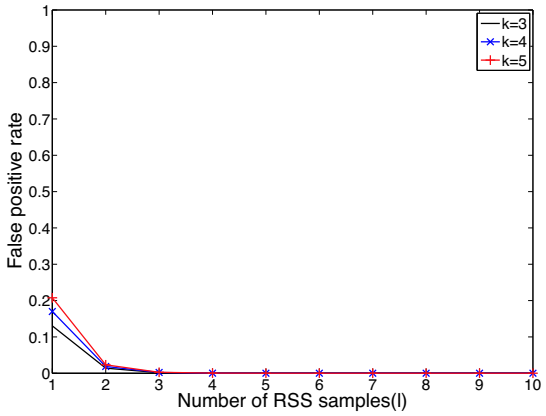


Fig. 7. Fase positive rate v.s. k and l

Once k is set, we configure l based on our expectation about the false positive rate. Suppose that we would like to limit the false positive rate to no more than P. Then we have $1 - \texttt{erf}^k(\sqrt{2 \times l}) < P$, where $\texttt{erf}$ is the Gauss error function. We can then estimate the smallest l to meet our requirement. For example, if we would like to limit the false positive rate to 0.01, then l can be set to 3 if $k = 4$. Figure 7 shows the false positive rate for different combination of k and l. This figure can be used to guide the configuration of k and l.

C. Unforgibility of RSSD

In this subsection, we will explain that it is very difficult for an adversary to forge the RSSD. We are not trying to prove that forging RSSD is hard in the cryptographic sense. In fact, we believe that given sufficient investment, the adversary is able to precisely create the same signature. For example, the adversary may send a team to the deployment field to physically locate all neighbors of a target node and then carefully place special devices to adjust the power to each of these neighbors. Therefore, what we are trying to show is that the cost of successfully forging the same fingerprinting signature as a benign node greatly outweighs the benefit this signature can provide.

We start with a simple scenario where a malicious node is trying to impersonate a benign sensor node in an ideal environment. By ideal environments, we mean that the received signal strength is only affected by the distance and the transmission power at the sender. Since we set a fixed transmission power for the sender during fingerprinting, we can directly estimate the distance to the fingerprintee based on the received signal strength. Since there are k ($k >= 3$) different measurement points, we have k distances to the fingerprintee. Based on multilateration, these k distances determine a unique location for the fingerprintee. Therefore, there is only one position in the field that allows the malicious node to create exactly the same signature in an ideal environment. This means that to evade the detection, the adversary has to locate the original sender and place the malicious node at the same position. In addition, the adversary also needs to guarantee that the packets forged by the malicious node do not reach the original sender since otherwise the original sender will notice that someone else is trying to impersonate him. Achieving all these requirements is a certainly non-trivial task.

Next we want to point out that the real environment makes it even more challenging to successfully forge RSSD signatures. First, in practice, the received signal strength is impacted by not only the environmental factor but also some physical features about the sender such as the height, the antenna orientation, etc. As a result, even if the adversary place the node at the same position, it is very hard to create the same signature. This will be confirmed by our later experiments. Second, it is generally very hard to precisely pinpoint the location of an arbitrary sensor node in the area, not to mention the deployment of the malicious node at the same position as this node. Certainly, it is possible that the adversary is lucky enough to find several sensor nodes in the field if he can physically access the network. However, in this case, it is better to just capture and compromise these sensor nodes. In summary, we believe that it is unrealistic for an adversary to forge RSSD signatures.

D. Experiment Evaluation

We will use the same deployment shown in Figure 5 for our experiment. We pick four measurement points $\{F_1, F_2, F_3, F_4\}$ as marked in the figure for fingerprinting ($k = 4$). We first study the false alarm rate in a real environment. Remember that we need the original signature to perform the detection. This signature is also obtained through fingerprinting as well, which will inevitably introduce some error. We will thus study the impact of the signature quality. Intuitively, the better the quality of the signature, the lower the false positive rate. As mentioned, to obtain the signature, we measure the signal strength at each selected i-th position m times and then calculate the mean and standard deviation (μ_i, σ_i) from these samples, which is then used as the i-th element in the signature. Clearly, the larger the sample size

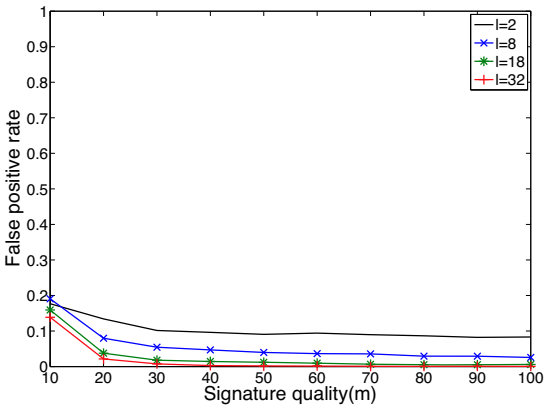


Fig. 8. False alarm rate v.s. signature quality (m) and l

(m), the better the quality of the signature. In other words, we will evaluate the impact of parameter m on the false alarm rate. Figure 8 shows the false positive rate under different signature qualities. We can see that the better the quality of the signatures, the higher the performance, which is consistent with our intuition. The figure also shows that the higher the l (i.e., more samples during fingerprinting), the lower the false positive rate.

Next we will study the detection rate of the proposed fingerprinting scheme when a malicious node is impersonating a benign node. We assume that we have already obtained a signature with good quality ($m = 20$) according to the result in 8. We will study the impact of the parameter l and the distance between the malicious node and the good node B on the detection rate. As we mentioned in Section IV-C, one may think that as long as we place the malicious node close enough to the real node, it would be hard to detect the impersonation attack. This sounds plausible since the radio signal from the malicious node may be influenced in the same way as the signal from the real node. However, our experiment shows otherwise. We select three different positions, $\{(0\ ft, 0\ ft), (2\ ft, 2\ ft), (4\ ft, 4\ ft)\}$ for malicious nodes and vary parameter l from 1 to 10 during the detection. The result shows that we can always detect the impersonation attack. This tells us that the distance does not impact the detection rate. This confirms that the RSSD is not only impacted by the environment but also some physical features of a sensor node, e.g., the direction of antenna, etc. Hence, we strongly believe that RSSD is suitable for fingerprinting in sensor networks.

E. Applications and Limitations

In the following, we will discuss the application of our RSSD-based fingerprinting technique in detecting the sybil attacks [14], wormhole attacks [8], and node replication attacks [15]. We will also discuss the limitations of RSSD-based fingerprinting techniques.

When RSSD-based fingerprinting is employed, a sensor node will always measure the RSSD fingerprint signature before communicating with any other sensor node. If several sensor nodes have the same fingerprint signature, then we know that they are either fake identities created by the adversary or tunneled by a wormhole. In either case, these IDs cannot be trusted. For the replicas of the same compromised sensor node, since they are essentially different sensor motes deployed by adversaries, we will extract different RSSD signatures with respect to the same set of measurement points. As a result, the RSSD-based fingerprinting can detect replica nodes at a very high probability.

So far we haven't talked about when to check and whom to report the result to. The reason is that they are application dependent. What we are trying to provide is a method to fingerprint the sensor node whenever it is needed. As one example, the application may require a node to verify the ID of the next hop first before sending or forwarding a packet. In this case, the result is used by the node who is sending or forwarding a packet. As another example, the application may require nodes to periodically report their RSSD readings about their neighbors back to a base station so that the base station can monitor and identify the nodes that are physically captured. As a result, we leave the decision to the application developer or the network owner.

There are several limitations for RSSD-based fingerprinting. First, it does not handle mobility. If a sensor node moves, its RSSD signature will also change. Second, the fingerprinting signature is also quite sensitive to the environmental changes. If there are some objects (e.g., cars, animals) or another electromagnetic source present in the range, the RSSD signature will be different too. As a result, we do not recommend RSSD fingerprinting methods for highly mobile sensor networks or applications where the environment changes frequently. Nevertheless, the impact of other sources can be reduced in some of the cases. For example, if the source is temporarily, we can still verify the ID once it moves away. If the source is legitimate and stays in the area, we can ask the source to notify all nodes in the area to re-build the model. If nothing indicates that the source is legitimate, then this means that there is something wrong in the field. Finally, deriving RSSD signatures requires collaboration between several nodes and a considerable number of RSS samples need to be collected to have a high-quality signature. This will add some noticeable overhead to the selected fingerprinting nodes as well as latency during the fingerprinting.

V. Related Work

Fingerprinting through clock skew was first introduced by Kohno et al. in [11]. They proposed to fingerprint a computer in wide-area networks by computing the clock skew from the timestamps in TCP or ICMP packets. Jana and Kasera [10] proposed to use the clock skew as a fingerprint to detect unauthorized wireless access point (AP). They observe the IEEE 802.11 Time Synchronization Function (TSF) timestamps in the beacon frames broadcasted by the AP. Since an access point usually broadcasts beacons at a high frequency, its clock skew can be identified very quickly, e.g., in seconds.

Arackaparambil et al. [1] studied Jana and Kasera's work in [10] and proposed a better way to measure the beacon arrival time and thus a more reliable fingerprinting result. A possible attack on clock skew-based fingerprinting is also studied in [1]. In this attack, the attacker builds two virtual APs on one physical AP. One of them observes the timestamps from the real (target) AP and synchronizes its time with it, and the other uses such synchronized time to impersonate the real AP. The effectiveness of this attack is studied and the counter-measures are also proposed.

Clock skew-based fingerprinting was later used as an alternative for node identification and authentication or a second factor (when combined with cryptography-based solutions) in node authentication in sensor networks. Huang et al. [9] proposed to identify a node by measuring its clock skew. They found that the clock skew is reliable enough to identify different nodes. They also discussed the application of using clock skew in detecting sybil attacks. Uddin and Castelluccia [17] also showed that the clock skew is reliable for fingerprinting sensor nodes. They studied how the clock skew changes with temperature and how to use this property to detect and locate malicious or malfunctioning nodes. They also explained how to use clock skew to detect sybil attacks, wormhole attacks, and node replication attacks. In contrast to these studies, our paper found that the clock skew can be easily forged and cannot be used to detect any attack mentioned in early studies.

There are many studies on how to use received signal strength (RSS) for purposes such as detecting Sybil attacks in [4], [6], [2]. These detection methods share the same intuition with our proposed RSSD-based fingerprinting, i.e., measuring signal strength at multiple points can be used as second-factor authentication. In this paper, we point out that RSSD can be used for not only the detection of Sybil attacks but also other attacks that essentially target the authenticity of sensor nodes. Given these existing studies and our findings, we strongly believe that RSSD is a promising candidate for fingerprinting sensor nodes.

VI. Conclusion and Future Work

In this paper, we study the feasibility of fingerprinting sensor nodes using two physical properties, the clock skew and the RSSD. Different from the common belief, we pointed out that clock skew cannot be used for node authentication or detecting attacks in sensor networks. We confirmed this via extensive experiments. We have also found that the RSSD-based fingerprinting is promising for static sensor networks. We demonstrated its reliability and analyzed its unforgibility.

In the future, we would like to investigate the following directions. First, we only tested our clock skew attack against sensor networks. We strongly believe that our attack will also work in local wireless networks where one tries to fingerprint wireless access points. This is because our analysis in Section III-B does not make any assumption about the hardware. Second, there are some other physical properties that could be used for fingerprinting, e.g., link quality indicator (LQI). It is interesting to study these physical properties as well. Finally, our RSSD-based method requires collaboration between sensor nodes. The security of such collaboration needs further study.

Acknowledgments: This material is based upon work supported by the National Science Foundation under Grant No. CNS-0916221 and ARO's Grant No. W911NF-12-1-0060. The authors would like to thank the anonymous reviewers for their valuable comments.

References

[1] C. Arackaparambil, S. Bratus, A. Shubina, and D. Kotz. On the reliability of wireless fingerprinting using clock skews. In *Proceedings of the ACM Conference on Wireless Network Security (WiSec)*, March 2010.

[2] Y. Chen and R. P. Martin. Detecting and localizing wireless spoofing attacks. In *Proceedings of the Annual IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2007.

[3] B. Danev, H. Luecken, S. Capkun, and K. E. Defrawy. Attacks on physical-layer identification. In *Proceedings of the ACM Conference on Wireless Network Security (WiSec)*, March 2010.

[4] M. Demirbas and Y. Song. An rssi-based scheme for sybil attack detection in wireless sensor networks. In *Proceedings of the International Symposium on on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2006.

[5] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proc. ACM Conf. on Computer and Communications Security (CCS)*, November 2002.

[6] D. B. Faria and D. R. Cheriton. Detecting identity?based attacks in wireless networks using signalprints. In *Proceedings of ACM Workshop on Wireless Security (WiSe)*, 2006.

[7] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D.E. Culler, and K. S. J. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[8] Y.C. Hu, A. Perrig, and D.B. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. In *Proceedings of INFOCOM*, April 2003.

[9] D. Huang, W. Teng, C. Wang, H. Huang, and J. M. Hellerstein. Clock skew based node identification in wireless sensor networks. In *GLOBECOM*, pages 1877–1881, 2008.

[10] S. Jana and S. K. Kasera. On Fast and Accurate Detection Of Unauthorized Wireless Access Points Using Clock Skews. In *Proceedings of the ACM Annual Conference on Mobile Computing and Networking (MobiCom)*, pages 104–115, 2008.

[11] T. Kohno, A. Brodio, and K. Claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2):93–108, April 2005.

[12] A. Liu and P. Ning. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, April 2008.

[13] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proc. ACM Conference on Computer and Communications Security (CCS)*, October 2003.

[14] J. Newsome, R. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: Analysis and defenses. In *Proceedings of IEEE International Conference on Information Processing in Sensor Networks (IPSN 2004)*, Apr 2004.

[15] B. Parno, A. Perrig, and V. Gligor. Distributed detection of node replication attacks in sensor networks. In *IEEE Symposium on Security and Privacy*, May 2005.

[16] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar. SPINS: Security protocols for sensor networks. In *Proc. Intl. Conf. on Mobile Computing and Networks (MobiCom)*, July 2001.

[17] Md. B. Uddin and C. Castelluccia. Toward clock skew based wireless sensor node services. In *Proceedings of the IEEE Annual International Conference on Wireless Internet (WICON*, pages 1–9, 2010.

[18] D. Zanetti, B. Danev, and S. Capkun. Physical-layer identification of uhf rfid tags. In *Proceedings of the ACM Annual Conference on Mobile Computing and Networking (MobiCom)*, September 2010.