# Proximity Based IoT Device Authentication

Jiansong Zhang[1†], Zeyu Wang[2†], Zhice Yang[2], and Qian Zhang[2]
Microsoft Research[1] The Hong Kong University of Science and Technology[2]
jiazhang@microsoft.com[1], {zwangas, zyangab, qianzh}@cse.ust.hk[2]
Co-primary Authors[†]

***Abstract*—Internet of Things (IoT) devices are largely embedded devices which lack a sophisticated user interface, *e.g.*, touch screen, keyboard, *etc.* As a consequence, traditional Pre-Shared Key (PSK) based authentication for mobile devices becomes difficult to apply. For example, according to our study on home automation devices which leverage smartphone for PSK input, the current process does not protect against active impersonating attack and also leaks the Wi-Fi password to eavesdroppers, *i.e.*, currently these IoT devices can be exploited to enter into critical infrastructures, *e.g.*, home networks. Motivated by this real-world security vulnerability, in this paper we propose a novel proximity-based mechanism for IoT device authentication, called *Move2Auth*, for the purpose of enhancing IoT device security. In Move2Auth, we require user to hold smartphone and perform one of two hand-gestures (moving towards and away, and rotating) in front of IoT device. By combining (1) large RSS-variation and (2) matching between RSS-trace and smartphone sensor-trace, Move2Auth can reliably detect proximity and authenticate IoT device accordingly. Based on our implementation on Samsung Galaxy smartphone and commodity Wi-Fi adapter, we prove Move2Auth can protect against powerful active attack, *i.e.*, the false-positive rate is consistently lower than $0.5\%$.**

## I. Introduction

The Internet of Things (IoT) has quickly moved from hype to reality. Gartner estimates that the number of deployed IoT devices will reach 20.8 Billion in 2020 [1]. Like other disruptive technologies, such as smartphones and cloud computing, IoT holds the potential for societal scale impact by transforming many industries as well as our daily lives.

However, IoT also brings security challenges due to its large scale and embedded device nature [2]. In this paper we discuss security of a basic IoT device function, *i.e.*, associating to Internet gateway (*e.g.*, Wi-Fi access point). In particular, we found authenticating an IoT device is non-trivial, and existing design actually leads to security vulnerability in practice. For example, according to our experimental study on a popular home automation brand, we can obtain the secrets that are sufficient for stealing home Wi-Fi password from all (million of) the devices based on our attack on one device. From further discussion on this real world example, we show the need for a carefully designed IoT device authentication mechanism. We will elaborate the experiments and discussion in section II-A.

In Figure 1, we take home automation scenario as an example to describe IoT device authentication. Home Wi-Fi router needs to authenticate home automation devices (*e.g.*, smart power switch) before allowing them to connect. On the mean time, a nearby attacker (*e.g.*, deployed attacking device around home) can perform (1) passive attack by sniffing all

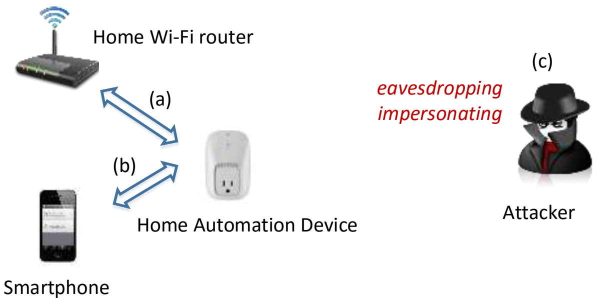


Fig. 1. We take home automation device as an example to illustrate the IoT device authentication problem. (a) Home Wi-Fi router needs to authenticate the device before connecting. (b) Smartphone is leveraged to input Wi-Fi password. (c) An attacker can eavesdrop by sniffering Wi-Fi channel, or impersonate the IoT device to connect to router/smartphone.

message exchanges on Wi-Fi channel, or (2) active attack by impersonating the home automation device and connecting to home router. Therefore, a successful attack may obtain sensitive information (*e.g.*, home Wi-Fi password as we observed in section II-A), or get the access to home network which enables further attack.

From Wi-Fi router point of view, an IoT device is all the same as a mobile device (*e.g.*, smartphone or tablet), on which *Pre-Shared Key* (PSK) is widely used for device authenticate. Specifically, 802.11 standards incorporate a Diffie-Hellman key exchange based mechanism, called *Simultaneously Authentication of Equals* (SAE) [3], for mutual authentication between router and device. SAE plus a limited number of retries provides solution against the attacks shown in Figure 1. However, from device point of view, IoT brings new challenge because the devices usually lack means for PSK (*e.g.*, Wi-Fi password) input, as they are mostly embedded devices. Specifically, in this paper we assume the IoT device (1) does not contain sophisticated user interface like screen or keyboard, (2) does not equip sensors like camera, accelerometer, gyroscope, NFC, microphone, *etc.* (3) is not easy to move (*e.g.*, power switch plugged on walls).

Many IoT device vendors leverage smartphone to input PSK. As shown in Figure 1(b), they connect IoT device to smartphone first. As long as the connection between smartphone and IoT device is secure, IoT device can obtain PSK from smartphone and perform PSK based authentication with router as mentioned above. In this way, the problem is reduced from router-IoT authentication to smartphone-IoT authentication. In this paper, we also take this solution and focus our study on IoT-to-smartphone authentication.

In literatures, there are a set of mobile/wearable device-

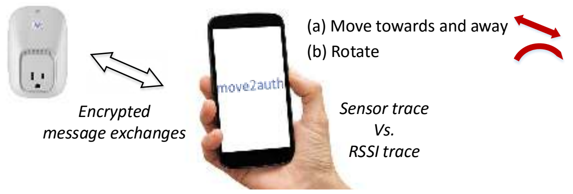

(a) Move towards and away
(b) Rotate
move2auth
Encrypted message exchanges
Sensor trace Vs. RSSI trace

Fig. 2. Proximity based IoT device authentication. User holds smartphone and performs small gestures (a. move towards and away, b. rotate) in front of IoT device while IoT device is transmitting. Smartphone compares sensor trace with rssi trace to determine whether the device is in proximity.

pairing solutions [4]–[9]. We notice that those solutions are not prefered in our problem because of IoT devices' constraints mentioned above (three assumptions). For example, moving both devices together [4] or along predefined path [5] will not apply because IoT device does not equip accelerometer/gyroscope and does not move. We will discuss related authentication problems and solutions in details in section II-B to further explain why our problem calls for new design.

In this paper, we propose a proximity based mechanism for smartphone to authenticate IoT devices, called Move2Auth. As shown in Figure 2, we require user to hold smartphone and perform one of two hand gestures (randomly picked by smartphone) in front of the IoT device, while on the mean time the IoT device is keep sending packets. The two gestures, *i.e.*, moving smartphone towards and away from IoT device, and rotating smartphone, both lead to significant (around $15dB$) variation in Received Signal Strength (RSS) because of fast-changing attenuation and antenna polarization, respectively. In Move2Auth, we combine (1) large RSS-variation detection, and (2) matching between RSS-trace and smartphone's sensor-trace, to perform reliable proximity detection, where (1) can effectively differentiate devices in-proximity and far-away, and (2) can protect against powerful active attacker who can arbitrarily tune transmission power.

We implement Move2Auth on Samsung Galaxy smartphone and commodity Wi-Fi adapter. We invite 5 users to test our prototype and conduct experiments on a test-bed containing 12 IoT device locations. The evaluation results show that (1) Move2Auth is reliable in differentiating sender-in-proximity and sender-far-away. (2) Move2Auth is reliable against active attacker who can tune transmission power or even has user's historical gesture traces. The false-positive rate in proximity detection is consistently lower than $0.5\%$.

In this paper, we make two major contributions.

- We take experiments to study the security of IoT device association and find (unreported) vulnerability on popular home automation brand. This study motivates the design of IoT-specific device authentication mechanism.
- We design, implement and evaluate Move2Auth which provides a reliable IoT device authentication mechanism.

## II. Motivation and Related Work

In this section, we present our experimental study on IoT device association, during which we observe unreported security vulnerability from a popular (millions of devices) home automation brand. We analyze this case and argue that an IoT-specific device-authentication mechanism is needed.

Based on detailed discussion of related authentication problems and solutions, we motivate the design of a new mechanism which we propose in this paper.

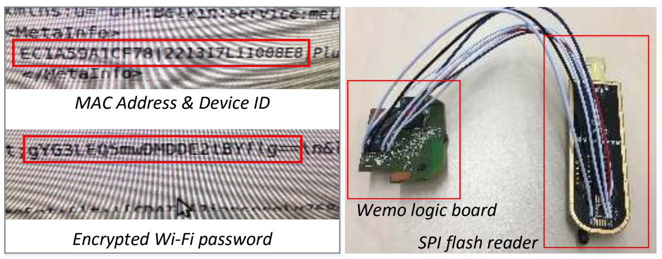
### *A. Vulnerability in IoT Device Association*

Recently, IoT has raised many security concerns and also attracted a lot of research interests [2], [10]. In this paper, our study on device association is motivated by several recent reports that home automation devices may leak Wi-Fi password while associating to home Wi-Fi router [11], [12]. Specifically, during the process shown in Figure 1, password leakage happens while smartphone is sending password to the device, because the transmission is all in plain text.

While the case appears like an implementation issue, we further take our own experiments and find that the issue is actually non-trivial to fix. We tried a popular home automation brand Belkin Wemo which provides a rich set of products, such as smart power switch, motion detector, camera, lighting LED, *etc.* [13]. Wemo's association process is also the same as in Figure 1. When we capture the message exchanges between smartphone and Wemo devices, we found Wemo did add protection for the password leakage issue by encrypting the password. Figure 3(a) shows an example of encrypted Wi-Fi password which was captured by wireshark. Deriving from the format, the password is encoded using Base64 [14] (after some encryption algorithm).

However, the encryption is not sufficient to protect password. Since Wemo devices can decrypt the password, Wemo devices must have carried all the secrets for decryption (*e.g.*, key, initial vector, salt, encryption algorithm, *etc.*). Therefore, we read out Wemo firmware and took binary analysis to retrieve the secrets. Figure 3(b) show the flywires we made between the SPI flash chip on Wemo logic board (Various Wemo devices contain the identical logic board) and a USB flash reader ($10). After obtaining Wemo firmware from the flash memory, we perform binary analysis as follows. First, with simple string analysis, we can learn that wemo firmware is built on top of the popular embedded operating system OpenWRT. Then, we use firmware analysis tool (*e.g.*, binwalk [15]) to recover the entire file system as well as individual program binaries. After manually locating the related programs (mostly by program names), we finally leverage disassembler tool (*e.g.*, IDA [16]) to convert program binaries to assembly codes and retrieve the secrets from the codes. As we expected, the encryption algorithm calls *openssl* libaries [17] and takes a combination of device ID and MAC address as key, initial vector and salt.

More importantly, we have actually obtained the secrets for decrypting all (millions of) the Wemo devices! Because Wemo do not carry any device specific secret (*e.g.*, a secret code in the firmware of each Wemo device) except device ID and MAC address, and device ID and MAC address must be sent from Wemo device to smartphone before password encryption, as we have captured in Figure 3(a).



(a) Captured message exchanges (b) Read out firmware binary

Fig. 3. Our experiment retrieved the secrets which can be used to decrypt home Wi-Fi password from all (millions of) Wemo devices. (a) Captured message exchanges between smartphone and Wemo device. First, Wemo device sends device ID and MAC address to smartphone (up message). Then, smartphone sends encrypted Wi-Fi password to Wemo device (bottom message). (b) We made flywires between the SPI flash chip on Wemo logic board and a USB flash reader ($10). We then use this tool to obtain Wemo firmware binary and perform binary analysis to retrieve the secrets.

To conclude our experiments, the protection implemented by device vendor (*i.e.*, Wemo designer) is indeed not sufficient to fix the password leakage issue. In the following, we discuss this defeated solution as well as other two possible solutions. We will show that carefully-designed device authentication mechanism is a must for secure device association.

*1) Defeated Vendor Solution:* Encrypting Wi-Fi password actually provides a low cost solution for device authentication, *i.e.*, even an attacker is connected to smartphone, it will not be able to retrieve the Wi-Fi password if the secrets are unknown. Unfortunately, the secrets are identical to all the devices, therefore we can defeat the entire solution by attacking only one device.

*2) Unique Secrets for Every Device:* Security will be enhanced if unique secrets are allocated for every device, as attacking one device will not help in cracking other devices. However, we would argue that the cost of unique secrets can be too high to afford, because IoT devices come in large scale. Specifically, every device can be assigned a unique key during manufacturing. The key can be (1) printed on device, or (2) recorded in a database and indexed by device ID or MAC address. When the device needs to be authenticated, the printed key can be directly read by user and inputted in the other party (*e.g.*, smartphone), or the stored key can be queried from the database. In case (1), the problem is that the same key should be stored simultaneously at two places, *i.e.*, hardcoded inside firmware and printed on device. While manufacturing in large scale, maintaining a sufficiently low mismatch rate will be a big challenge to device vendors. In case (2), for symmetric key, we need additional means for determining which user can query the key of a device ID. Otherwise, the key will be leaked to attackers. Private/public key pair might mitigate the problem, for which each vendor can build an infrastructure similar to the Public Key Infrastructure (PKI) [18]. Again, the maintenance cost will be a big challenge when devices come in large scale.

*3) Encrypting the Channel between Smartphone and IoT device:* Encrypting the channel can prevent eavesdroppers from capturing the message exchanges. However, while encrypting the channel is not difficult, for example, generating symmetric key using Diffie-Hellman key exchange or providing private/public key pair from either side, encryption does not prevent active attackers. In the Wemo case (as well as many other home automation devices we tried), device sets itself as Wi-Fi access point for smartphone to connect. As in Figure 1, an active attacker can impersonate the Wemo device by broadcasting the same SSID and using the same MAC address. If smartphone is connected to the attacker, home Wi-Fi password will be sent to the attacker directly.

To conclude the discussions, we argue that *IoT device authentication is practically needed and non-trivial to fulfill, IoT-specific mechanism is desired.*

## B. Related Authentication Problems and Solutions

In this subsection, we discuss related authentication problems and mechanisms in literatures and in practice. We use the discussion to motivate the new design proposed in this paper.

The most related problem is authenticating some other embedded devices, such as Wi-Fi router, wireless display adapter, embedded wearable (bluetooth earphone), *etc.* Different from IoT devices which lack user interface, Wi-Fi router and wireless display adapter actually contain display. For example, user can assign arbitrary password to Wi-Fi router by connecting it to a laptop through Ethernet port. Wireless display adapter is by default connected to a display, therefore the same device key pre-assigned in firmware can be shown on display (instead of printed on device). Similar to IoT devices, Bluetooth earphone does not contain display, therefore the same security problem also exists. In particular, convenient mechanism like pushing button simultaneously on both sides does not prevent active impersonating attack [19]. Therefore, in practice PSK is used. But for the sake of convenience, usually an identical pin is used for all device, such as "0000", which completely defeats the purpose of PSK. However, we would notice that the same security problem on embedded wearables does not raise as much as the concerns on IoT devices, because active attack on wearable device is more difficult to play as device association can happen anywhere (together with the user), while IoT devices are usually operating in fixed locations (*e.g.*, home appliances).

Mobile device pairing (*e.g.*, two smartphones) is also similar to our problem, except that IoT devices are usually difficult to move and do not equip popular sensors. Therefore, mechanisms using sensors on both side do not apply, such as biometric [20], accelerometer and gyroscope [4], [5], microphone [9], *etc.* Moreover, mechanisms using explicit out-of-band channel also do not apply, such as infrared [6], touch [7], visible light [8], *etc.*

Sensor network devices are also a type of embedded devices therefore are related. However, we would argue that, unlike IoT devices which will be connected to unknown access point, sensor network devices are usually designed for specific applications, *e.g.*, ocean or wildlife monitoring, and manufactured in batch [21]. Therefore, identical PSK can be distributed into a set of sensor network devices during manufacturing.

Proximity based mechanisms using radio interface [4], [22]–[25] are closely related to our proposed solution in this paper.

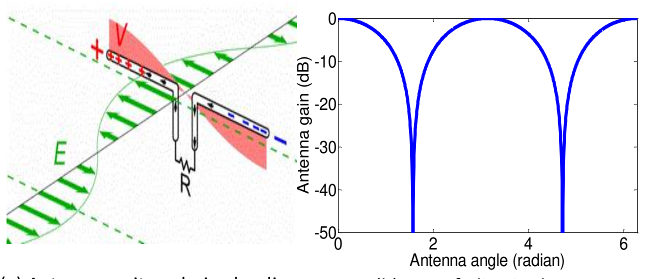

V
E
R
0
-10
-20
-30
-40
-50
Antenna gain (dB)
0
2
4
6
Antenna angle (radian)
(a) Antenna emits polarized radio wave
(b) Deep fading in theory

Fig. 4. Antenna polarization primer. (a) The physics behind antenna polarization is radio wave polarization. Radio antenna (*e.g.*, dipole) essentially emits polarized radio wave. (b) Deep fading exists in certain antenna angle accroding to the Malus's law.

We divided them into two types: passive and active. In passive solutions [22], [23], two nearby devices determine the proximity by common ambient radio environment. However, they require rich variation of ambient signal to provide sufficient information for reliable proximity detection, which is nontrivial to fulfil in practice. To enrich the information, they may require shaking device together, or equipping customized radio to sense additional signal like FM and TV [23], which are not prefered in our problem. Moreover, they may require a sufficiently short distance between two devices, *e.g.*, few centermeters on Wi-Fi frequency. On the other hand, in active solutions, Castelluccia [4] requires two devices to shake together to confuse eavesdroppers, therefore does not apply to our problem. Cai [24] and Pierson [25] rely on multiple antennas to detect the proximity by the large signal-strength difference. They actually partially inspired our solution. However, our solution differs in various ways. (1) Instead of multiple-antenna, we propose hand-gesture to create signal-strength difference, therefore our solution can be applied to smartphones which usually contain single antenna. (2) In our design, we not only use the event of large signal-strength difference, but also take advantage of the correlation between device movement and signal-strength variation, therefore can reliably protect against powerful active attacker. (3) In addition to distance, we also explore device angle (because of antenna polarization) which also provides large signal-strength difference in proximity.

## III. Proximity Detection Based on RSS Variation

In this paper, we propose a *Received Signal Strength* (RSS) based scheme for proximity detection on single antenna devices. The basic idea is that when two devices are in proximity, small device movement can cause significant RSS-variation. Specifically, we explored two types of movement which both lead to around $15dB$ RSS-variation, *i.e.*, moving towards and away from each other, and rotating, as shown in Figure 2.

The large RSS-variation upon moving towards and away are caused by the fast-changing channel attenuation when two devices are in proximity. While it has been introduced before [24], [25], in the following, we focus on introducing *antenna polarization* which leads to large RSS-variation upon (relative) rotating when two devices are in proximity.

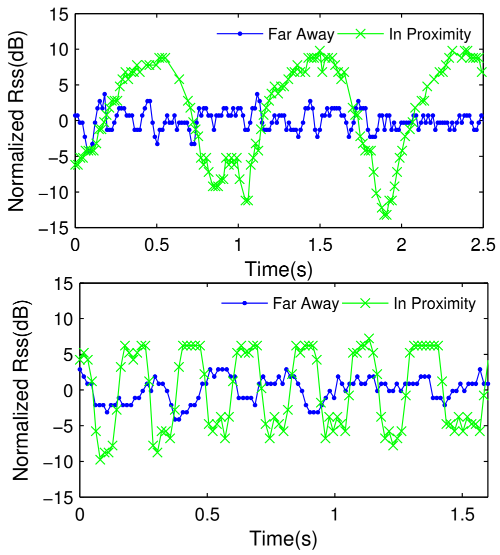


Fig. 5. Smartphone movement caused RSS-variation under sender-in-proximity vs. sender-far-awary. Sender-in-proximity can lead to more than $15dB$ RSS-variation while sender-far-away only leads to $5dB$ RSS-variation due to small-scale fading. Upper: moving towards and away. Bottom: Rotating.

### A. Antenna Polarization

The key idea of Move2Auth is leveraging antenna polarization to generate deep fading events in near field, which is also undetectable in far field. In this section, we provide a brief introduction for antenna polarization.

The physics behind antenna polarization is radio wave polarization [26]. Radio antenna essentially emits polarized radio wave. Figure 4(a) shows an illustrative example using dipole antenna which is the most popular antenna type in today's radio devices. As shown in the figure, the electric field always oscillates along antenna direction because it is the only possible direction for the electrons inside antenna to move.

Antenna polarization causes a phenomenon that RSS changes with the angle between transmitting (TX) antenna and receiving (RX) antenna. The relationship is described by the Malus' law [27] as follows.

$$G_\theta = G_0 \cos^2 \theta \tag{1}$$

where $\theta$ is the angle between TX antenna and RX antenna, $G_\theta$ is the antenna gain with angle $\theta$, and $G_0$ is the maximum antenna gain achieved when TX and RX antennas are parallel. Figure 4(b) shows simulation results with various $\theta$. When TX and RX antenna are perpendicular to each other, antenna gain will be zero which we call it *deep fading*. Therefore, when we rotate one of the devices, we will observe large RSS variation.

In practice, equation 1 only holds when two devices are in proximity. The reason is the rich-multipath wireless channel. A deep fading angle on one path will be overwhelmed by received signal on other paths, because polarization direction of radio wave changes upon reflection. Moreover, scattering and diffraction, which is also rich in microwave bands (*e.g.*, Wi-Fi, cellular, bluetooth, Zigbee, *etc.*) because of the comparable wavelength with many objects [28], further make deep fading unobservable in longer distance.

### B. RSS-Variation in Practice

Figure 5 shows examples of RSS-trace when sender is in-proximity or far-away, where the far-away sender is placed in a neighbor room of the receiver (smartphone). The upper sub-figure shows RSS-traces while user moves smartphone towards and away from sender. When sender is in-proximity, the fast-changing attenuation leads to more than $15dB$ RSS-variation. The bottom sub-figure shows RSS-traces when user rotates smartphone. When sender is in-proximity, antenna polarization leads to around $15dB$ RSS-variation. We notice that the RSS-trace appears different from Figure 4 because smartphone is rotated only $90°$ around the deep fading angle. Finally, RSS-variation from far-away sender is usually much smaller (around $5dB$) and mainly caused by small-scale fading [28].

## IV. Move2auth Design

In this section, we elaborate the design of Move2Auth which provides a novel mechanism for IoT device authentication. While the design can be easily extended to other radio technologies, we introduce in the context of Wi-Fi.

### A. Goal and Threat Model

Our goal is to build a device-authentication mechanism for the purpose of facilitating IoT device to securely associate to Wi-Fi router. In particular, we leverage smartphone in the way that connecting IoT device to smartphone first, and input the password of Wi-Fi router on smartphone, as we discussioned in the Introduction and Figure 1. As a result, the whole process can be considered secure as long as the IoT-smartphone connection is secure.

We consider attacker who can receive the packets from IoT device and smartphone, but is not physically close to IoT device, *e.g.*, outside of the home as in home automation scenario. We consider powerful attackers. For example, the attacker can sniff all the Wi-Fi channels and capture all the packets; he may have arbitrarily high-sensitivity receiver; he can actively connect to smartphone by impersonating the IoT device; he may have arbitrarily high transmission power and can adjust the transmission power arbitrarily; he may have full knowledge of our scheme; he may have exact copy of the IoT device; he may know the exact location of the IoT device.

In the following, we focus on one-way authentication, *i.e.*, smartphone authenticates IoT device. The other way, *i.e.*, IoT device authenticates smartphone, is not necessary in our problem, as we will discuss in section IV-I.

### B. Basic Scheme

We assume IoT device is not moveable. When an IoT device is in pairing mode, it keeps sending encrypted packets (section IV-G). On the mean time, we require user to hold smartphone in front of (*e.g.*, $20cm$ distance) the IoT device and perform small gesture for a while (*e.g.*, three seconds). User will be asked to perform one of two gestures, *i.e.*, moving towards and away from IoT device and rotating, as shown in Figure 2. The gesture is randomly picked by smartphone. While the gesture is performed, smartphone receives a series of packets with significantly-varying RSS, as the reasons shown in section III.

Smartphone determines whether the packets are sent from a nearby device based on two criteria, *i.e.*, (1) RSS-variation exceeds a threshold, (2) RSS-trace matches with smartphone sensor trace. In our design, we set $10dB$ as the RSS-variation threshold for both gestures.

Matching between RSS-trace and sensor-trace is an important building block of Move2Auth. The idea behind trace-matching is that, both traces can precisely describe smartphone movement when two devices are in proximity, but when two devices are far-apart, RSS-trace will not reflect the movement well. In our design, we not only consider shape of traces, but also involve *timing* for trace-matching. Timing information creates big-barrier for attacker who can fake large RSS-variation (*e.g.*, by tuning its transmission power). Even if the faked RSS-variation reflects the pace of smartphone-movement well, the faked RSS trace will not exactly match smartphone-movement because of their different start time. In our design, both sensor-trace and RSS-trace are recorded on smartphone so that we can easily synchronize them using smartphone clock. In section IV-C and IV-D, we will discuss how we transform one of the traces to perform trace-matching.

### C. Trace Transformation - Moving Towards and Away

We require user to move smartphone for around $20cm$, and the shortest distance to IoT device is around $20cm$. This smartphone movement causes around $15dB$ RSS-variation.

In our design, moving smartphone towards and away from IoT device is captured by accelerometer. For the sake of simplicity, we assume smartphone moves strictly on a line (towards IoT device). Therefore, the accelerometer-reading can be reduced from 3-dimension to 1-dimension.

Since converting acceleration into distance (by integration) will introduce accumulative error, we choose to transform RSS-trace into accelerometer-trace. Specifically, we first convert RSS-trace into distance-trace as described in [24], then we convert distance into acceleration by performing difference-operation for two times, *i.e.*, from distance to speed, then to acceleration. In order to avoid parameter-setting, *e.g.*, absolute distance to IoT device, we normalize both converted trace and real accelerometer trace into the same scale, *e.g.*, [0,1].

Finally, we synchronize two traces and calculate correlation. We determine whether two traces match with each other using a threshold on correlation result. We will discuss threshold setting in section VI.

### D. Trace Transformation - Rotating

We require user to rotate smartphone for around $180°$, so that RSS deep fading caused by antenna polarization will be reliably captured which provides around $12dB$ RSS-variation.

In our design, rotation is captured by gyroscope. As discussed in section III and in Figure 4, rotation causes deep-fading in RSS-trace because of antenna polarization. However, on the rest of RSS-trace, RSS is relatively flat. As a consequence, we will not be able to capture the turns (*i.e.*, from clockwise to counter-clockwise, and vice versa) from

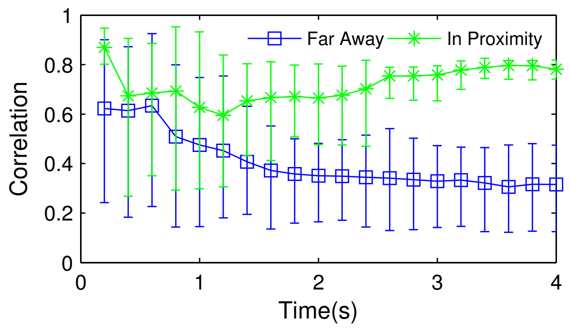


Fig. 6. Correlation results for trace-matching based on smartphone rotation. Three-second gesture clearly differentiates sender-in-proximity and sender-far-away. We notice that RSS-variation will further enhancing differentiation.

RSS-trace. Therefore, we choose to transform gyroscope-trace into RSS-trace, *i.e.*, we first reduce gyroscope-reading from 3-dimension to 1-dimension, then we derive RSS-trace using Equation 1.

Moreover, in order to provide better usability, we do not make assumption on IoT device's antenna angle, *e.g.*, vertical, horizontal, *etc.* Instead, from gyroscope-trace we try all possible angles and derive RSS-trace accordingly. We calculate correlation between all the derived RSS-traces and the real RSS-trace (after normalization), and choose the maximum as correlation result. According to our experiments, our design choice still provides sufficiently-low false-positive rate, as we will discuss in section VI.

### *E. Duration of User Gesture*

In our current design, we require user to push a (virtual) button on smartphone to start the user gesture.

We derive proper duration of user gesture from experiment results. In Figure 6 we show the correlation results of trace-matching based on smartphone rotation. We calculate correlation from part of the trace (# of seconds from starting point). Error bar shows the deviation on $50$ traces. When the traces are longer than three seconds, correlation result can clearly differentiate sender-in-proximity and sender-far-away. We also notice that RSS-variation will further enhance the differentiation. And rotating already represents high correlation results for sender-far-away, as we have discussed in section IV-D.

As a result, in Move2Auth we require user to perform a gesture for at least three seconds. Every sensor trace is also cut to three-second before trace-matching.

### *F. Dealing with RSS Inaccuracy*

A practical problem for our scheme is *RSS Saturation* that RSS will saturate to a value (*e.g.*, $-10dBm$ on our platform) when transmitter is very close (*e.g.*, $20cm$). Therefore, we require the IoT device to transmit in a lower power-level. According to our experience, $20dB$ lower transmission power is sufficient to avoid RSS saturation, which is easy to realize on commodity Wi-Fi chipsets.

Another problem is that RSS reading will be inconsistent with different data rates (more specifically, different packet preambles [24]). Therefore, we require IoT device to use the basic rate (*i.e.*, $6Mbps$) during authentication.

### *G. Encryption*

Besides the authentication scheme we described above, we also need to encrypt the communication channel between smartphone and IoT device to protect against eavesdropping. [29], [30] provide approaches to derive a shared key from characteristics of the wireless channel.

Alternatively, we could use cryptographic techniques to derive a shared secret. We notice that key generation does not affect the authentication scheme, because authentication only measures RSS from the preamble of each packet while key generation uses the payload of the packet.

We propose a straightforward key generation protocol, where IoT device sends a public key to smartphone at the beginning. Then IoT device sends a series of identical packets encrypted by the corresponding private key (to facilitate authentication). The packet content contains smartphone's information, *e.g.*, MAC address. Upon successful authentication, smartphone verify the received public key by decrypting the packets. Finally, smartphone generates a random shared key and sends it to the authenticated IoT device in a packet encrypted by the verified public key.

### *H. Move2Auth Protocol*

Our final protocol integrates both device authentication and key generation, as described below.

1) User triggers pairing mode by pressing a button on the IoT device for three seconds. IoT device clears its states in pairing mode, and sets itself as access point for smartphone to connect.
2) User finds IoT device's Wi-Fi network by SSID, and connect smartphone to the IoT device.
3) Upon receiving the connection from smartphone, IoT device sends a random public key and a series of identical packets encrypted by the corresponding private key.
4) Smartphone determines whether the IoT device is in proximity by both checking the strength of RSS-variation and matching its sensor trace with RSS trace, as described in section IV-C and IV-D.
5) Upon successful authentication by determining the IoT device is in proximity, smartphone verifies the public key by decrypting the packets and checking packet contents.
6) Upon successful decryption and content checking, smartphone encrypts a random shared session key using the verified public key and sends to IoT device.
7) Finally, a secure communication channel is established with the shared session key. User can perform IoT-to-router association on smartphone with this secure communication channel.
8) If one the above steps fails, user should re-trigger pairing mode, as we will explain in section IV-I.

### *I. Security Analysis*

In this subsection, we briefly analyze the security of Move2Auth by considering various attacks.

*1) Eavesdropping:* As long as the public/private key cryptography is not defeated, an attacker will not obtain any information from sniffing the packet transmissions.

*2) Impersonating IoT device:* During smartphone-IoT pairing, an attacker may impersonate the IoT device by broadcasting the same SSID/MAC-address and using higher transmission power. Smartphone could be connected to the attacker instead of IoT device because of the higher signal strength. In Move2Auth, we protect against this attack by checking whether the connected device is in proximity. Specifically, we consider an attacker who follows the protocol in section IV-H to send a series of packets. We discuss two cases. First, if the attacker does not tune its transmission power. RSS-variation (caused by smartphone movement) will be small and mismatch with sensor trace. The sender will be determined not in proximity and authentication will fail. Second, if the attacker tunes its transmission power to create high RSS-variation, we will mainly rely on trace-matching to detect proximity. Our experiments show that the trace-matching leads extremely-low false-positive rate because we consider both the shape of traces and their exact timing. We will present details in section VI.

*3) Denial-of-Service (DoS) attack:* DoS attacks can be performed in various ways. For example, jamming the wireless channel to breach the communications. In Move2Auth, we do not explicitly protect against DoS attack. However, we believe DoS attack will unlikely cause serious troubles which are difficult to deal. For example, we will be able to locate the attacker with the help of pinpointing tool and remove it.

*4) The other direction - Should IoT device authenticate smartphone?:* We explain why authentication in the other direction is not necessary in our problem. Firstly, an already-connected IoT device (with either router or smartphone) will not enter pairing mode unless user triggers. Secondly, on IoT-router link, a rogue AP may send dis-associate packet to break the link and cheat IoT device to connect to it (by impersonating the router). We can protect against this attack by requiring IoT device to always perform PSK-based secure association, because IoT device will also authenticate router in this way. Thirdly, during smartphone-IoT pairing, an attacker may successfully connect to the IoT device. In order to avoid any harm on this temporary illegitimate connection, we require IoT device to only accept one connection in pairing mode. In this way, this attacker-IoT connection will be easily detected if smartphone fails to connect to IoT device. And user should re-trigger pairing mode by pressing the IoT device button. Moreover, in pairing mode, IoT device should clear all its states and data. Therefore, nothing will be leaked to attacker on the temporary attacker-IoT connection. Finally, given the short attacker-IoT connection time, we assume attacker will not be able to completely compromise the IoT device and turn it into a "zombie" (*e.g.*, by pushing a fake firmware update).

### *J. Discussion*

*1) Usability:* Since the two gestures in Move2Auth are both straightforward, in our current design, we require user to perform the gesture strictly, *e.g.*, moving on the line towards IoT device, or rotating in a plane which is perpendicular to the line between smartphone and IoT device. In theory, we can perform sophisticated gesture recognition and remove the requirements. We take this as our future work.

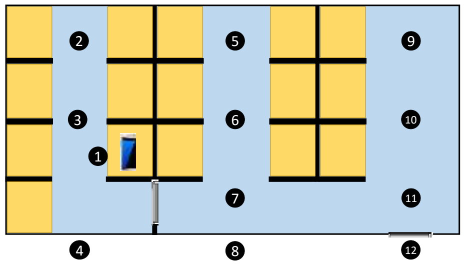


Fig. 7. We tested 12 locations, where we assume user and smartphone are at location 1 as shown in the figure. Most locations are within the same room but only contain non-line-of-sign paths to smartphone.

*2) Requirement on Computation:* Taking Wemo as an example, it equips a $320MHz$ MIPS processor which is capable to call *openssl* libraries and run in real-time. We believe the computation of public/private key cryptography also can be fulfilled on many other today's IoT devices. On smartphone side, similar computation is easy to fulfill as smartphone has become very powerful today.

*3) Antenna types:* Most antennas today are dipole which generates linearly-polarized radio wave as shown in Figure 4. Many other antenna types also emit polarized radio wave such as patch antenna. According to our experience on smartphones and IoT devices, *e.g.*, iPhone, Samsung Galaxy, Wemo devices, *etc.*, antenna polarization holds on all these devices.

*4) Other IoT security issues:* We notice that there are other security problems reported for IoT devices [2]. For example, a home automation device could be accessed and controlled remotely. We believe these problems call for additional security mechanisms. However, these are out of scope of this paper.

## V. Implementation

We implemented a prototype of Move2Auth on Android smartphone and commodity Wi-Fi adapter. We tried Atheros and Intel Wi-Fi adapters on Linux PC to act as IoT device. We successfully decreased transmission power and fixed data rate on both Wi-Fi chipsets to fulfill Move2Auth. In order to obtain RSS on smartphone, we installed a customized Wi-Fi driver which can turn smartphone Wi-Fi into monitor mode. Currently, the driver only works on Samsung Galaxy 2 and 3. In Move2Auth, we record accelerometer and gyroscope output upon a (virtual) button press, as discussed in section IV-E. We added timestamp to RSS trace and sensor trace to faciliate trace-matching. To implement public/private key cryptogrophy, we called the *openssl* library. In our current implementation of Move2Auth, we programmed IoT device (Linux PC) to send packets every $1ms$, *i.e.*, $1000Hz$ RSS sampling rate.

## VI. Evaluation

In this section, we present our evaluation for Move2Auth. We focus on reliability of proximity detection and consider two cases which may cause false-positive detection, *i.e.*, another far-away IoT device which is also in pairing mode and an active attacker who can arbitrarily tune transmission power. We present two cases in section VI-B and VI-C, respectively.

We set up a testbed as shown in Figure 7, where we tested 12 IoT device locations inside and outside an office room

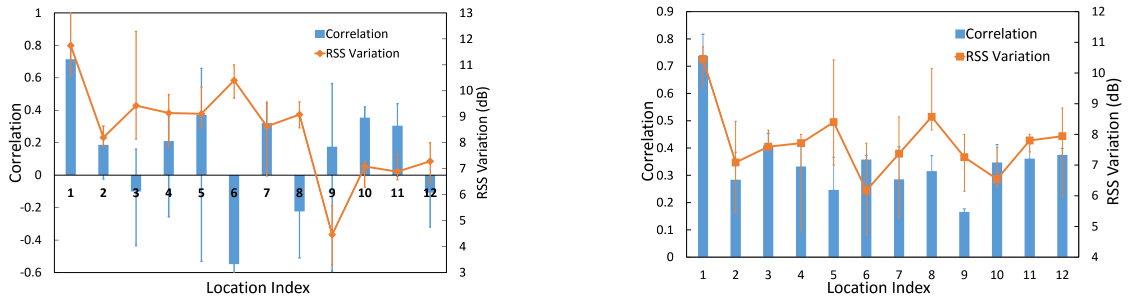


Fig. 8. Correlation and RSS-variation at 12 locations. We present average and error bar (maximum and minimum) from all the tests conducted at a location. User and smartphone is fixed to location 1, so location 1 represents in-proximity case. Left: gesture moving towards and away; Right: gesture rotating.

($10m \times 8m$). We fixed user's location (and smartphone) to locaion 1, as shown in the figure. We moved the IoT device to each of 12 locations, then perform both gestures to collect RSS-trace and sensor-trace. We also invited five users for the tests. Among them, two users performed tests for all the IoT device locations, rest of the users (three) only performed tests at location 1 and 12. We tested two gestures separately and repeat all the tests with both gestures. For each test, we repeat 10 times and present statistics. Collectively, we get 600 sensor-traces and RSS-traces in total.

In all the experiment, the threshold for RSS-variation is $10dB$, and threshold for correlation (trace-matching) is $0.6$.

### A. Detection Rate In Proximity

We first study successful-detection rate when smartphone and IoT device are in proximity. Figure 8 shows the statistic for all the tests on a location, where we also mix tests from different users. Two sub-figures represent two gestures. At location 1, both correlation and RSS-variation are clearly higher than other locations. Since all the correlation and RSS-variation at location 1 (50 for each gesture) exceed thresold, proximity detection was always successful in the tests *i.e.*, false-negative rate is zero.

### B. Reliability against Another Far-Away IoT Device

Then, we consider the case that another far-away IoT device is also in pairing mode. In this way, we study the false detection rate that smartphone actually authenticate a far-away IoT device. Location $2 \sim 12$ represents far-away IoT device locations. As shown in Figure 8, at location $2 \ \sim 12$, false-positive rate is zero because correlation and RSS-variation never exceeds the thresholds simultaneously. Actually, usually neither of them exceeds threshold.

Comparing two gestures, we found rotating leads to higher correlation results. This is expected result as we have explained in section IV-D. However, rotating also leads relatively lower RSS-variation. Therefore, both gestures can perform well in the case of far-away IoT device.

### C. Reliability against Active Attacker

Then, we consider an active attacker who can arbitrarily tune transmission power. In this case, RSS-variation threshold will be eaily defeated. Therefore, we only consider correlation (trace-mapping). Although we conducted the tests for both gestures, due to page limit, we only present the results from rotating gesture. As we have mentioned above, rotating represents the more challenging case because the correlation tends to be high (explained in section IV-D).

We consider four types of RSS wave that attacker can create.

The first one is sine wave where we assume attacker has no knowledge of user's gesture. Specifically, attacker may try different frequency of sine wave. In the tests, we tried 16 frequencies uniformly distributed from $0.5Hz$ to $4Hz$. We assume attacker transmit the RSS wave continuously. The tests was conducted by setting different starting time for a real gesture (sensor trace), and we performed trace-matching between sensor-trace and sine RSS wave (*i.e.*, calculating correlation). We notice that we actually assume the RSS-variation caused by gesture is much smaller than the RSS wave created by attacker. We plot the cumulative distribution function (CDF) of correlation results on left-top of Figure 9. With correlation threshold of $0.6$, the false-positive is $0.27\%$.

The second one is RSS wave derived from gesture of another user, *e.g.*, attacker himself. Specifically, we collect the RSS-traces from all five users when they are at location 1 (in-proximity). We calculate correlation between all combination of different users. Also, we include different trace start time. Right-top of Figure 9 shows the CDF of the correlation result for this case. The false-positive rate is $0.31\%$. It is interesting to see that gesture information from different user actually does not help in the attack, because different user usually perform gestures in different pace, speed and extent, even the gesture is as simple as rotating.

The third one is historical RSS wave recorded from the same user. Specifically, we collect all the RSS traces from the same user at location 1 (in proximity). We calculate correlation between all combination of different traces from the same user. We also include different trace start time. Left-bottom of Figure 9 shows the CDF of the correlation result in this case. The false-positive rate is $0.28\%$. It is also interesting to see that the historical information from the same user actually does not help a lot, because the details differ when a user perform a gesture again. We believe this difference is more significant with simple gestures like rotating, but for complicated gestures like writing a signature would be more consistent.

Finally, we consider a non-real case that attacker knows user's gesture in advance. Specifically, we calculate the exactly same trace but consider different trace starting time. As shown in right-bottom of Figure 9, the false-positive rate increases, but is still as small as $8.2\%$. We notice that the timing information plays important role in trace-matching, which can

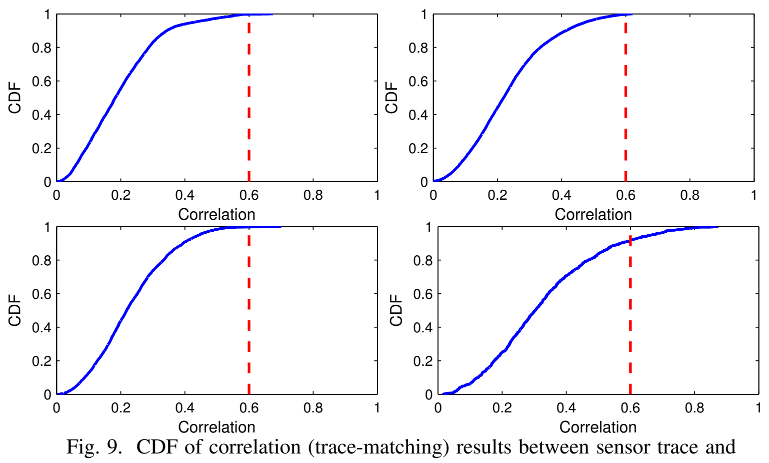


Fig. 9. CDF of correlation (trace-matching) results between sensor trace and RSS-variation created by attacker. Left top: attacker sending sine RSS wave with different frequencies from $0.5Hz$ to $4Hz$, false-positive rate is $0.27\%$. Right top: attacker sending RSS wave recorded from other user's gesture (*e.g.*, attacker's), false-positive rate is $0.31\%$. Left bottom: attacker obtained RSS wave recorded from user's previous gesture, false-positive rate is $0.28\%$. Right bottom: attacker knows what gesture the user will perform, practically infeasible, false-positive rate is $8.2\%$.

prevent even the (non-real) most strongest attacker. However, this attack itself will not happen in practice.

To conclude our evaluation, Move2Auth provides reliable proximity detection with zero false-negative rate in our tests. Move2Auth effectively prevents smartphone to authenticate a far-away IoT device, the false-positive rate is also zero in our tests. Move2Auth also effectively protect against strong attacker that can arbitrarily tune transmission power, the false-positive rate is consistently lower than $0.5\%$ in practical cases.

## VII. Conclusion

Motivated by our observation of IoT security vulnerability in real world, we propose a novel proximity based authentication mechanism for IoT devices called Move2Auth. Move2Auth detects proximity by checking (1) large RSS-variation and (2) matching between RSS-trace and smartphone sensor-trace during two user gestures, *i.e.*, moving smartphone toward or away from IoT device, and rotating smartphone. We implement Move2Auth on Samsung smartphone and prove its reliability against powerful active attacker.

We believe our study will help in building secure infrastructure for the coming IoT era.

## Acknowledgement

The research was supported in part by grants from 973 project 2013CB329006, RGC under the contracts CERG M-HKUST609/13, 622613, 16212714, and 16203215,ITS/143/14FP-A, as well as the grant from Huawei-HKUST joint lab, IoT WiFi key technologies.

## References

[1] "Gartner says 6.4 billion connected "things" will be in use in 2016, up 30 percent from 2015," *http://www.gartner.com/newsroom/id/3165317*.

[2] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu, "Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things," in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*. ACM, 2015, p. 5.

[3] D. Harkins, "Simultaneous authentication of equals: A secure, password-based key exchange for mesh networks," in *Proceedings of the 2008 Second International Conference on Sensor Technologies and Applications*, ser. SENSORCOMM '08, 2008, pp. 839–844.

[4] R. Mayrhofer and H. Gellersen, "Shake well before use: Intuitive and secure pairing of mobile devices," *Mobile Computing, IEEE Transactions on*, vol. 8, no. 6, pp. 792–806, 2009.

[5] I. Ahmed, Y. Ye, S. Bhattacharya, N. Asokan, G. Jacucci, P. Nurmi, and S. Tarkoma, "Checksum gestures: continuous gestures as an out-of-band channel for secure pairing," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2015, pp. 391–401.

[6] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong, "Talking to strangers: Authentication in ad-hoc wireless networks." in *NDSS*, 2002.

[7] D. G. Park, J. K. Kim, J. B. Sung, J. H. Hwang, C. H. Hyung, and S. W. Kang, "Tap: touch-and-play," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*.

[8] N. Saxena, J.-E. Ekberg, K. Kostiainen, and N. Asokan, "Secure device pairing based on a visual channel," in *Security and Privacy, 2006 IEEE Symposium on*. IEEE, 2006, pp. 6–pp.

[9] D. Schurmann and S. Sigg, "Secure communication based on ambient audio," *Mobile Computing, IEEE Transactions on*, vol. 12, no. 2.

[10] E. Fernandes, J. Jung, and A. Prakash, "Security Analysis of Emerging Smart Home Applications," in *Proceedings of the 37th IEEE Symposium on Security and Privacy*, May 2016.

[11] "Owners of heatmiser wifi thermostats warned of password leaks and other vulnerabilities," *https://www.grahamcluley.com/2014/09/heatmiser-wifi-thermostats-password-leak/*.

[12] "Kettles are leaking wifi passwords (and other failures of the internet of things)," *http://www.newstatesman.com/science-tech/future-proof/2015/10/kettles-are-leaking-wifi-passwords-and-other-failures-internet*.

[13] "Belkin wemo home automation," *http://www.belkin.com/us/Products/home-automation/c/wemo-home-automation/*.

[14] "Base64," *https://en.wikipedia.org/wiki/Base64*.

[15] "Binwalk firmware analysis tool," *http://binwalk.org/*.

[16] "Ida disassembler and debugger," *https://www.hex-rays.com/*.

[17] "Openssl libraries," *https://www.openssl.org/*.

[18] "Public key infrastructure," *https://en.wikipedia.org/wiki/Public key infrastructure*.

[19] "How does wi-fi protected setup work?" *http://www.wi-fi.org/discover-wi-fi/wi-fi-protected-setup*.

[20] I. Buhan, B. Boom, J. Doumen, P. H. Hartel, and R. N. Veldhuis, "Secure pairing with biometrics," *International Journal of Security and Networks*, vol. 4, no. 1-2, pp. 27–42, 2009.

[21] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks," *Communications of the ACM*, vol. 47, no. 6.

[22] A. Varshavsky, A. Scannell, A. LaMarca, and E. De Lara, *Amigo: Proximity-based authentication of mobile devices*. Springer, 2007.

[23] S. Mathur, R. Miller, A. Varshavsky, W. Trappe, and N. Mandayam, "Proximate: proximity-based secure pairing using ambient wireless signals," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM, 2011, pp. 211–224.

[24] L. Cai, K. Zeng, H. Chen, and P. Mohapatra, "Good neighbor: Secure pairing of nearby wireless devices by multiple antennas," in *Proceedings of the 18th Annual Network and Distributed System Security Symposium*, 2011.

[25] T. J. Pierson, X. Liang, R. Peterson, and D. Kotz, "Wanda: securely introducing mobile devices." InfoCom, 2016.

[26] "Polarization (waves)," $https : //en.wikipedia.org/wiki/Polarization(waves)$.

[27] Z. Yang, Z. Wang, J. Zhang, C. Huang, and Q. Zhang, "Wearables can afford: Light-weight indoor positioning with visible light," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '15, 2015.

[28] P. V. David Tse, "Fundamentals of wireless communications," 2004.

[29] S. Jana, S. N. Premnath, M. Clark, S. K. Kasera, N. Patwari, and S. V. Krishnamurthy, "On the effectiveness of secret key extraction from wireless signal strength in real environments," in *Proceedings of the 15th annual international conference on Mobile computing and networking*. ACM, 2009, pp. 321–332.

[30] 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*. ACM, 2008, pp. 128–139.