2021 IEEE Wireless Communications and Networking Conference (WCNC) | 978-1-7281-9505-6/20/$31.00 ©2021 IEEE | DOI: 10.1109/WCNC49053.2021.9417287

Push yoUr Password: Secure and Fast WiFi Connection for IoT Devices

Junyoung Choi, Jaewon Hur, and Saewoong Bahk
Department of ECE and INMC, Seoul National University
Email: jychoi@netlab.snu.ac.kr, hurjaewon@snu.ac.kr, sbahk@snu.ac.kr

***Abstract*—Internet of things (IoT) is an indispensable paradigm in today's industrial change. IoT interconnects appliances around us through the Internet, and user's private information is deeply placed inside the network. Nonetheless, security vulnerabilities in IoT have not been addressed appropriately so far due to various reasons. Even in the initial WiFi connection procedures of IoT devices, WiFi credentials can be leaked, making the WiFi access point (AP) a hacking path. We propose a secure connection scheme, termed `PUP`, that aims to securely and quickly connect an IoT device to the AP in proximity while improving the user experience. `PUP` shows perfect security performance, enabling connection time within 11 s.**

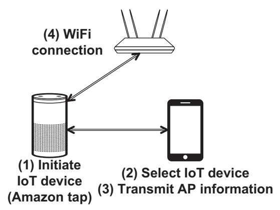
I. Introduction

A variety of home IoT devices have become widespread with the tendency to remotely connect and control home appliances. The number of IoT devices is expected to increase to 125 billion by 2030 [1]. IoT devices are connected to the Internet and have WiFi built-in [2]. Therefore, IoT devices should be initially connected to a WiFi access point (AP) to have remote access.

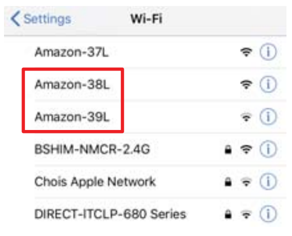
To do this, the user needs to find an SSID and enter its password to start a WiFi connection (e.g, authentication and association) for an IoT device. The problem here is that most IoT devices do not have a suitable user interface like a touch screen. To tackle this problem, vendors have introduced the *WiFi connection* procedures for IoT devices, using a user's smartphone [3, 4]. The user sends AP information to the IoT device of interest via the smartphone.

Fig. 1(a) shows the detailed WiFi connection procedures of Amazon tap as an example of an IoT device connection: (1) The user presses the WiFi connection button on Amazon tap, and configures its own WiFi network. (2) The user finds the network that seems to be Amazon tap, and selects the SSID for WiFi association using the smartphone. (3) After the smartphone is connected to the Amazon tap's network, the user chooses the target AP for Amazon tap. After that, the smartphone transfers the security information of the target AP to Amazon tap. (4) The process ends when Amazon tap successfully connects to the target AP. To complete the above process, the user spends about 3 minutes.

As explained earlier, the WiFi connection procedures are cumbersome and time-consuming for the user. In addition,


(4) WiFi connection
(1) Initiate IoT device (Amazon tap)
(2) Select IoT device
(3) Transmit AP information

(a) WiFi connection procedures of Amazon tap



(b) Impersonating device examples (Amazon-38L and Amazon-39L)

Fig. 1. WiFi connection procedures and impersonating devices.

if the smartphone is already connected to AP, the user should disconnect the original WiFi connection to conduct the WiFi connection procedures. These procedures can create security problems. The security is critical to IoT devices, but manufacturers have not considered this seriously. 25% of cyber attacks are expected to target IoT devices in 2020 [5] For example, hackers can access sensitive information for IoT devices through a compromised WiFi AP [2].

Due to lack of security, the WiFi connection procedures may leak AP's password in two typical ways [6]. First, passive hacker devices with high antenna sensitivity may eavesdrop AP's password. Even if the AP's password is encrypted by Diffie-Hellman algorithm or RSA encryption,hackers still can sniff authentication information. Second, hacker devices impersonate real IoT devices, thereby not being identified by the user. Fig. 1(b) shows three SSIDs, one of which is Amazon tap and the other two are impersonating devices (i.e., Amazon-38L and Amazon-39L). A hacker device can deceive the user to connect to itself, and obtain the AP's password through the WiFi connection. There has been a proposal to attach a sticker that contains product's SSID, but there are limitations on manufacturing conditions [6].

To solve the above problem, the secure communication channel between the smartphone and the IoT device is required. At the same time, the user should be able to distinguish the genuine IoT device. We assume that there exist both eavesdropping and impersonating hacker devices, but they can not be located in the user's private area, such as inside of home or office. The genuine IoT device is located in close proximity to the user. Following these conditions, the problem is authenticating the device in proximity, which has been recently studied [6–12]. However, none of them

Saewoong Bahk is the corresponding author of this paper.

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2020R1A2C2101815).

focused on both ensuring security and improving latency in the WiFi connection process.

In this paper, we propose **PUP** to achieve the following two goals: (1) transferring WiFi AP's configuration securely to the genuine IoT device, and (2) reducing the latency for the WiFi connection process with minimal user intervention. We distinguish between near IoT devices and distant hacker devices based on the fact that signals received by antennas at close locations go through similar shadow and multipath fading. **PUP** requires simple back and forth hand gestures to make the shadowing effect as shown in Fig. 2(a). It collects received signal strength indicator (RSSI) traces, and authenticates the device of interest using the RSSI traces.

To minimize user intervention and reduce latency, we leverage Bluetooth Low Energy (BLE) advertising packets. Thus, **PUP** does not include any WiFi connection procedures, simply enabling the user to choose the appropriate IoT device. **PUP** can authenticate multiple devices at once, maintaining low protocol latency owing to its connection-less nature.

The main contributions of this work are summarized as follows:

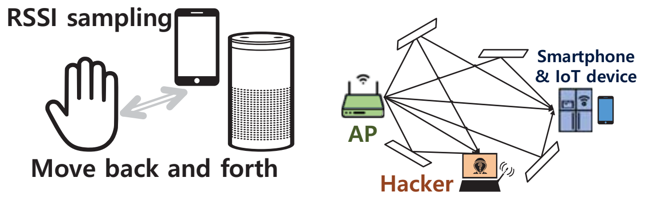
- We propose **PUP** that aims to autonomously authenticate multiple devices, and to securely transfer AP's password to the chosen IoT device using BLE advertising packets. **PUP** significantly reduces user intervention and the time needed for authentication by simplifying the connection process.
- We leverage RSSI samples to authenticate devices in proximity. Our algorithm does not rely on packet reception to get RSSI samples, but it is simply enabled by any RF signals, such as WiFi and Bluetooth. To the best of our knowledge, it is the first trial that uses raw RSSI samples in proximity authentication.
- We implement a prototype of **PUP** on Ubertooth platform [13], and verify its high security and low latency performance. The evaluation results show that **PUP** successfully block hackers with the false positive rate of 0%, and maintains the latency lower than 11 s.

II. Background and Related Work

A. Characteristics of Wireless Channel

As in Fig. 2(b), wireless signals propagate through multiple paths to reach the receiver. The received signal power is fluctuating when the receiver receives multiple signals with different phases. This is called multipath fading. Multiple paths make this effect stronger with the number of paths. The signal is attenuated as it passes through obstacles that cause shadowing effects.

As a result of this effect, spatial differences causes different fluctuating patterns of RSSI. In addition, the wireless channel changes much more due to the movement around. Even simple gestures by humans cause fluctuations in RSSI and significantly increase unpredictability. However, receivers within one wavelength are likely to hear signals that are experiencing similar radio channels. Thus, it is almost impossible to predict the received signals of receivers far from each other.



(a) Hand gestures (b) Multipath wireless channels

Fig. 2. Example of shadowing and multipath fading.

For the 2.4GHz ISM band signal which has the wavelength of 12.5 cm, antennas within tens of centimeters may obtain different RSSI samples. Focusing on this fact, we authenticate devices in proximity based on the similarity of RSSI values.

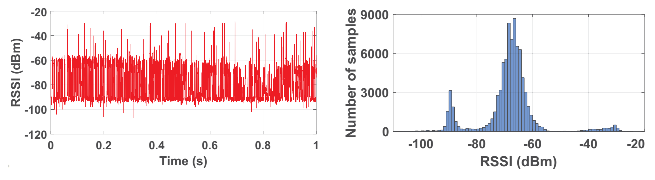
B. BLE Advertisement

BLE is introduced for low energy short range communications used by low power sensors and IoT devices [14]. Smartphones on Android 4.3 platform released in 2012 also supports BLE [15]. The BLE module uses one of 40 channels for communications on the 2.4 GHz unlicensed spectrum. Especially, BLE defines channels 37, 38, and 39 as advertising channels that are used only for transmitting BLE advertising packets. A BLE advertising packet contains preamble, access address, protocol data unit (PDU), and cyclic redundancy check (CRC). Eddystone and iBeacon use the PDU format to carry reserved fields for data transmission [16]. The size of the PDU is limited to 31 Bytes by the specification.

C. Related Work

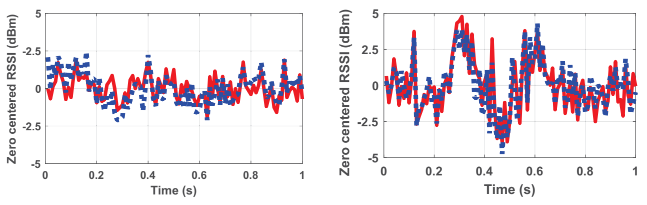
Using ambient wireless signals. Amigo [7] proposes using RSSIs of ambient WiFi packets to authenticate devices in proximity. Ensemble [12] suggests authenticating a device with the help of multiple authenticated devices. Proximate [11] converts wireless channel measurements to construct shared bit sequences between nearby devices. It leverages continuous TV broadcasting signals or FM radio signals, and needs special hardware. There have been several researches to construct shared bit sequences using ambient ISM band signals [17–20]. However, they need RSSI samples of received packets, not raw RSSI samples. On the contrary, we propose an authentication scheme based on raw RSSI samples, including ambient signal and noise samples.

Using channel information between devices. Another approaches use channel information between adjacent devices. Small spatial changes make a large difference to RSSI values between nearby devices. Good Neighbor [8] introduces multiple antennas to detect large RSSI differences and authenticates a nearby device. Wanda [10] devises a stick with multiple antennas to authenticate devices in proximity and securely transfer data. vBox [21] demands the user to shake the pairing device to create an authenticated channel. PCASA [9] continuously authenticates nearby devices, but needs acoustic signals. Move2Auth [6] requires the user to move the smartphone in a specific way to authenticate stationary IoT devices



(a) Raw RSSI samples (b) Sample distribution

Fig. 3. RSSI sampling results in the ISM band.



(a) Without hand gestures (b) With hand gestures

Fig. 4. Filtered and zero centered RSSI traces of two close antennas.

in line of sight. Above schemes just focus on proximity-based authentication, not the quality of user experience in the WiFi connection process.

III. Overview

A. Motivation

PUP is designed with two conflicting motivations: low latency and high level of security. Security is an important issue in communications. However, achieving a high level of security requires additional overhead, which increases latency and complexity. This is the same in the WiFi connection process of IoT devices.

In general, the initial WiFi connection process requires user intervention and takes a few minutes. The possibility of a WiFi password leak creates a significant security problem. Most users connect IoT devices to the Internet through APs. Since the AP is the first gateway on the network, it can be a source of hacking if breached. Impersonating hacker devices consistently try to sniff passwords during the WiFi connection process. Even worse, stationary IoT devices, such as smart outlet or smart air conditioner, are susceptible to attacks because the connection process takes place in a fixed position. Therefore, WiFi password transmission should be with proximity-based authentication for secure data transmission.

However, the authentication process increases protocol latency. Especially, authentication based on existing connection-oriented procedures incurs linearly increased protocol latency with the number of devices to be authenticated [6]. We solve the latency problem by using BLE.

B. Problem Definition and Threat Model

The goal of **PUP** is to safely connect a genuine IoT device to the AP with low latency and minimal user intervention. We suppose that the IoT device is equipped with WiFi/Bluetooth module but lacks a user interface, enabling to send the AP information directly. In addition, the user does not have prior knowledge about the IoT device.

In the threat model, hacker devices can sniff all packets going between the smartphone and the IoT device. They already know all the **PUP** process and can impersonate the genuine IoT device. However, the hacker devices can never be within a few centimeters from the user. The hacker's intention is to sniff the AP password. In this scenario, we believe the association and authentication procedures of the AP and IoT device are secure, and focus on transmitting the WiFi password in a secure way.

C. Design Features

The two key features of **PUP** are: (1) to use BLE advertising packets in the entire process, and (2) to authenticate IoT devices using raw RSSI samples from surrounding channels instead of packet reception. For data transmission, we leverage BLE advertising packets and RSSI sampling synchronization to remove connection overhead. We don't use WiFi beacons because broadcasting WiFi beacons requires the smartphone to disconnect the existing connection with the AP.

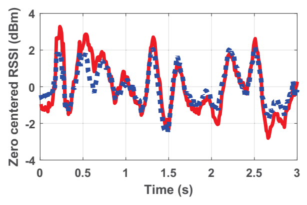
Authentication utilizing raw RSSI samples also has the benefit that it works without packet reception. In addition, using raw RSSI samples enables the device to be authenticated more quickly owing to its faster sampling rate, compared to the methods that use packet reception based RSSI sampling. Ensemble [12] takes at least 30 seconds for authentication because it uses packet reception based sampling.

IV. Proposed Scheme

A. Preliminaries for Proximity-based Authentication

We use the trace of raw RSSI samples for proximity-based authentication. The sampling interval is set to 0.1 ms considering the minimum duration of WiFi beacon frame and sampling overhead [22]. Before RSSI sampling starts, **PUP** triggers the remote server to send UDP packets to the smartphone at the highest data rate through the associated AP. This makes the wireless channel saturated by the AP for the short RSSI sampling period. Fig. 3(a) shows raw RSSI samples of one second, and Fig. 3(b) is the distribution of samples, which clearly shows two clusters. The cluster composed of higher RSSI values represents signals from the associated AP. We use K-means clustering to extract RSSI samples from the AP of interest. To determine the number of clusters, we applied the elbow method to the clustering method [23]. We average samples every 10 ms, and obtain a 100 Bytes RSSI trace per second using multiple samples.

PUP differentiates distant hacker devices from nearby IoT devices based on the correlation of RSSI samples between two antennas. This is because two close antennas share a similar wireless channel for the AP. One antenna is on the IoT device, and the other is on the smartphone. Fig. 4(a) shows zero centered traces of RSSI samples from the two close antennas. The given traces show a similar variation, but a less fluctuating channel may give hackers a higher chance to get a similar RSSI trace for the IoT device. Therefore, we added simple back and forth hand gestures of the user to create a shadow


(a) Shadow fading effect

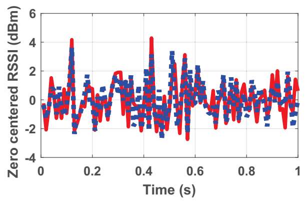
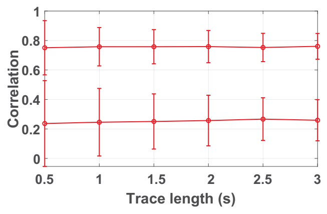

(b) Multipath fading effect

Fig. 5. Low-pass and high-pass filtered traces.


(a) Multipath fading correlation

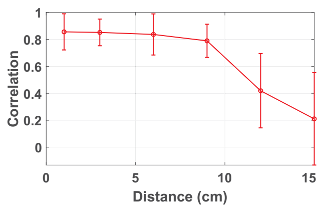

(b) Antenna distance

Fig. 6. Impacts of the trace length and antenna distance on correlation.

fading effect, which cause large channel fluctuations. Fig. 4(b) presents similar RSSI traces with higher fluctuation made by hand gestures, compared to Fig. 4(a).

We separate the RSSI trace into two filtered components to observe multipath and shadow fading effects. The low-pass filtered trace shows shadow fading, and the high-pass filtered trace shows multipath fading. Fig. 5 shows two filtered traces of Fig. 4(b). Based on the given traces, we calculate the correlation of each trace, and differentiate genuine IoT devices from hacking devices if the correlation exceeds a specific threshold.

To determine the threshold, we verify the impact of the trace length and the distance between antennas on correlation. The close antenna (on the IoT device) is located within 10 cm from the smartphone, and the distant (hacking) antenna is located 1 m apart from the smartphone. The AP antenna is also 1 m away from the IoT device. First, we collect RSSI samples and calculate the correlation with various trace lengths and distances between two close and distant antennas as shown in Fig. 6(a). The red and black lines represent the correlation of the close and distant antennas. The results show that longer trace steadily results in lower deviation of the correlation. After 3 s, two traces are clearly separated with the correlation gap of 0.1, centered at 0.5. Therefore, **PUP** recognizes the device only when the both correlations of the traces (i.e., shadow and multipath fading) are over 0.5 for 3 s.

Second, we calculate the correlation according to the distance between the smartphone and the genuine IoT antenna. We conduct the experiment for 100 s at each distance, and the correlation abruptly decreases when the distance exceeds 9 cm, which is the wavelength of the 2.4GHz radio signal, as shown in Fig. 6(b). If the smartphone and IoT device are located within 9 cm, we can successfully identify the IoT device.

B. Secure and Fast WiFi Password Transfer

PUP uses RSSI traces of the smartphone and IoT device to authenticate proximity, and each RSSI trace has the size of 300 Bytes in total. However, transmitting the trace of RSSI samples in plain text can give hackers a chance to eavesdrop. Therefore, the transmission of the RSSI trace from the IoT device to the smartphone must be encrypted as well as the password transmission. Since participants in **PUP** process initially do not share any prior information, we first utilize the RSA encryption algorithm that works asymmetrically. The RSA encryption algorithm uses one pair of encryption keys, private and public. The data encrypted by the public key only can be decrypted by the host which has the correct private key [24]. Therefore, the RSA algorithm is primarily used in the initial access.

C. Overall Procedures

The smartphone starts the **PUP** process and broadcasts a randomly generated public key, keeping the private key safely. The IoT device that has successfully received the public key sends RSSI samples encrypted with the public key after RSSI sampling to the smartphone. Following the authentication of the IoT device, the smartphone transfers the encrypted AP password which can be decrypted by the IoT device only. We leverage the AES encryption algorithm [25] that runs in a symmetric way. The smartphone and IoT device in proximity share the same data (e.g., trace of RSSI samples) that is used to generate the shared AES encryption key. The smartphone transmits the AP password to the authenticated IoT device, which is encrypted with the AES encryption key generated from the shared RSSI trace.

When **PUP** calculates the correlation using the RSSI traces from multiple devices, one challenge is that the smartphone and IoT device do not share the synchronized time and frequency channel for RSSI sampling. This can be easily tackled by using BLE advertising packets. We design *BLE trigger frame* to contain the synchronization flag and frequency channel information. Right before the smartphone starts RSSI sampling, it sends a *BLE trigger frame*. Participating IoT devices constantly scan the BLE advertising channels, and start RSSI sampling on their target channel. The sampling period is set to three seconds as defined in Section IV-A, so the devices can obtain the frequency, synchronized time, and RSSI traces with the same length.

We use BLE advertising packets in all the procedures. Since the size of BLE advertising packet is 31 Bytes, which is smaller than the RSSI trace size of 300 Bytes, we fragment a RSSI trace into small advertising packets for delivery. We generate a large BLE advertising packet by filling the vacant field with the fragmentation header that uses the Eddystone frame format [16]. Each BLE advertising packet includes fragmentation size, fragmentation number, identifier (ID), and trace. The IoT device identifies the BLE advertising packet of **PUP** process using the ID. For reliable data transmission, we make the entities broadcast the data multiple times, and alternately send and receive in the process.

The overall protocol procedures are illustrated in Fig. 7.

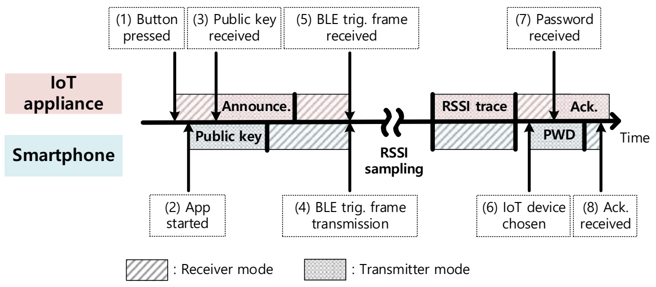


Fig. 7. Overall procedures of **PUP**.

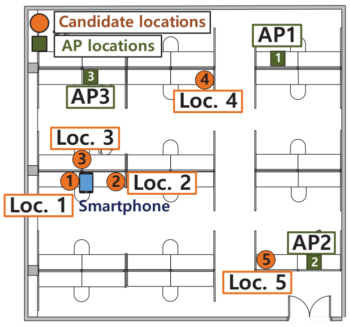


Fig. 8. Experimental topology.

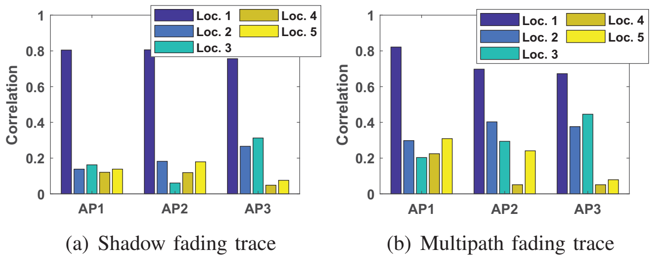


Fig. 9. Correlation of RSSI traces according to AP and location pair.

- The IoT device starts listening to BLE advertising packets when the trigger button is pressed. It constantly scans BLE advertising packets that have the ID of **PUP**.
- After starting **PUP**, the smartphone starts broadcasting the public key and the AP's MAC address. The broadcasting lasts for the predefined period. Following this period, the smartphone immediately starts BLE channel scanning for the same period. The smartphone can find the IoT device from the scanning.
- If the IoT device successfully receives the public key and the AP's MAC address, it announces its presence by broadcasting a BLE advertising packet that contains its own MAC address and the AP's MAC address. The period of announcement is the same as above. If the announcement period of the smartphone is longer than that of IoT device, the smartphone may not discover the IoT device.
- At the end of the announcement listening period, the smartphone sends the *BLE trigger frame* and performs RSSI sampling for 3 s.
- The IoT device receiving *BLE trigger frame* also performs RSSI sampling on the same frequency channel. After 3 s, the IoT device broadcasts the trace of RSSI samples encrypted by the public key, and the smartphone receives the trace. They stop RSSI sampling at the same time, and the IoT device starts BLE scanning to receive the AP password. If the received trace passes the correlation condition, the smartphone displays the authenticated IoT device, and asks the user to choose the device that the user wants to connect to.
- After the user chooses the authenticated IoT device, the smartphone sends the IoT device the AP password encrypted by the RSSI samples of the IoT device.
- The chosen IoT device receives the AP password, and transmits the acknowledgement packet by using BLE. After receiving the acknowledgement packet, the smartphone ends the **PUP** process.

V. Performance Evaluation

A. Implementation and Measurement Setup

Implementation: We consider an AP connected to Ubertooth one through USB port [13] as an IoT device. Ubertooth one is an open source Bluetooth platform equipped with CC2400 transceiver. In the raw RSSI sampling period, Ubertooth one samples raw RSSI every 0.1 ms, and aggregates samples to generate a 3 s trace. We empirically set the public key, RSSI sample, and password transmission period as 1.5, 3 and 0.5 s, respectively. We do not implement **PUP** using raw RSSI samples on the smartphone because Android does not provide API to report raw RSSI values of ambient signals. Therefore, we design **PUP** to operate based on the RSSI sampling through packet reception in the case of the smartphone.

Measurement setup: We verify the proximity-based authentication capability of **PUP** with the topology of an indoor office environment with multiple APs, as shown in Fig. 8. Dark green squares represent candidate locations of APs, which are used to create the saturated channel. Orange colored circles represent locations of participating devices. The authenticator device is located at Location 1 (Loc. 1) and the smartphone is right next to it. We perform RSSI sampling for 300 s on each pair of AP and device's candidate location.

B. Authentication Performance

Fig. 9 shows the RSSI trace correlation between the smartphone and the device on each location. We show the results of multipath fading (i.e., high-pass filtered) and shadow fading (i.e., low-pass filtered) traces. The IoT device on Loc. 1 obtains highly correlated trace for all the three APs. The device acceptance rate by location for each AP is summarized in Table I. The IoT device on Loc. 1 achieves almost 95% acceptance rate. Although the IoT devices on Loc. 2 and 3 also show some acceptance rate, it is allowable because they are located almost within 1 m from the authenticator device.

TABLE I
DEVICE ACCEPTANCE RATE BY LOCATION FOR EACH AP IN [%]

	Loc. 1	Loc. 2	Loc. 3	Loc. 4	Loc. 5
AP 1	95.7	5.4	1.1	0	0
AP 2	94.6	9.7	2.1	0	0
AP 3	96.8	9.7	1.7	0	0

It is noteworthy that none of IoT devices located on Loc. 4 and 5 (belonging to different partitions) pass the test.

In the extreme case, it is possible that a hacker can mimic user movement (i.e., hand gestures). We also evaluate the performance when the hacker is close to the IoT device on Loc. 2. During the RSSI sampling period, the hacker performs the same action as the user. Fig. 10 shows the empirical CDF of RSSI correlations. The hacker's action can not generate the similar RSSI difference because the characteristics of the wireless channel, such as shadow and multipath fading, are different even within a short distance of 1 m.

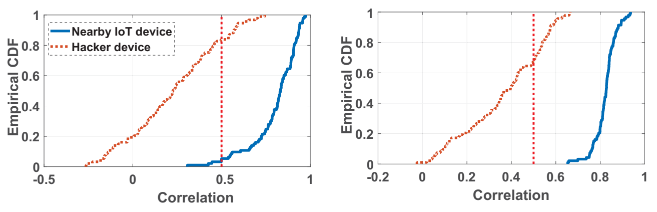
Finally, we conduct the same experiment with the smartphone to show the feasibility and practicality. We use Nexus 5 as the authenticator device. We modified the sampling method to get RSSIs from received BLE packets because Android does not have API to report raw RSSI values. Fig. 11(a) shows the effect of trace length on the correlation. The trace length is much longer than 3 s because the RSSI sampling rate through packet reception is much lower than that of ambient signals. Therefore, we chose the trace length of 30 s for evaluation. We conducted the evaluation 300 times, and obtain the empirical CDF graph as shown in Fig. 11(b). We can successfully identify IoT devices near the smartphone. Therefore, it is possible to use ambient radio signals in proximity authentication.

C. Latency Performance

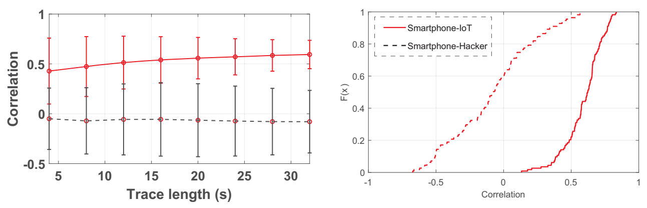
PUP aims to minimize latency for secure authentication. The generation periods for the public key, RSSI trace, and password in **PUP** are set to 1.5, 3, and 0.5 s, respectively. Since the BLE beacon can send packets up to 100 times per second, the chosen period is sufficiently large to run the procedures. We achieve the connection success rate of 95% within 11 s. It is possible to rerun the protocol, and the total connection latency is about 12 s even in the case of the initial connection failure, which is much shorter than 3 minutes of connection time on the Amazon tap. More importantly, **PUP** is very convenient because it does not require any user intervention except for the first button press.

VI. CONCLUSION

In this paper, we focused on security issues and user experience for IoT device authentication. We proposed PUP that aims to successfully authenticate IoT devices and quickly transmit the WiFi password. PUP uses ambient radio signals to authenticate devices in proximity, minimizing latency and improving user experience. PUP shows significantly high security performance and very low connection time of 11 s, compared to the existing scheme.



(a) Shadow fading trace (b) Multipath fading trace
Fig. 10. Empirical CDF of correlation in the mimicking hacker case.



(a) Trace length vs. correlation (b) Empirical CDF of correlations
Fig. 11. Feasibility of using ambient radio signals on Nexus 5.

REFERENCES

[1] Number of Connected IoT Devices Will Surge to 125 Billion by 2030. [Online]. Available: https://technology.ihs.com/596542
[2] M. B. Barcena and C. Wueest, "Insecurity in the Internet of Things," *Security Response*, 2015.
[3] Amazon Tap Website. https://www.amazon.com/dp/B01BH83OOM.
[4] Samsung SmartHome Website. https://smarthome.samsungsds.com.
[5] IoT Device Security is Being Seriously Neglected. [Online]. Available: https://www.aberdeen.com/techpro-essentials/iot-device-security-seriously-neglected/
[6] J. Zhang, Z. Wang, Z. Yang, and Q. Zhang, "Proximity based IoT Device Authentication," in *Proc. IEEE SECON*, 2017.
[7] A. Varshavsky, A. Scannell, A. LaMarca, and E. De Lara, "Amigo: Proximity-based Authentication of Mobile Devices," in *Proc. ACM UbiComp*, 2007.
[8] L. Cai, K. Zeng, H. Chen, and P. Mohapatra, "Good Neighbor: Ad Hoc Pairing of Nearby Wireless Devices by Multiple Antennas." in *Proc. NDSS*, 2011.
[9] P. Hu, P. H. Pathak, Y. Shen, H. Jin, and P. Mohapatra, "PCASA: Proximity based Continuous and Secure Authentication of Personal Devices," in *Proc. IEEE SECON*, 2017.
[10] T. J. Pierson, X. Liang, R. Peterson, and D. Kotz, "Wanda: Securely Introducing Mobile Devices," in *Proc. IEEE INFOCOM*, 2016.
[11] S. Mathur, R. Miller, A. Varshavsky, W. Trappe, and N. Mandayam, "Proximate: Proximity-based Secure Pairing using Ambient Wireless Signals," in *Proc. ACM MobiSys*, 2011.
[12] A. Kalamandeen, A. Scannell, E. de Lara, A. Sheth, and A. LaMarca, "Ensemble: Cooperative Proximity-based Authentication," in *Proc. ACM MobiSys*, 2010.
[13] Ubertooth-one Website. https://ubertooth.sourceforge.net.
[14] S. Bluetooth, "Specification of the Bluetooth System-Covered Core Package Version: 4.0," 2010.
[15] J. Choi, G. Lee, Y. Shin, J. Koo, M. Jang, and S. Choi, "BLEND: BLE Beacon-aided Fast WiFi Handoff for Smartphones," in *Proc. IEEE SECON*, 2018.
[16] Google Beacon Platform Website. https://developers.google.com/beacons.
[17] 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 *Proc. ACM MobiCom*, 2009.
[18] S. Mathur, W. Trappe, N. Mandayam, C. Ye, and A. Reznik, "Radio-telepathy: Extracting a Secret Key from an Unauthenticated Wireless Channel," in *Proc. ACM MobiCom*, 2008.
[19] D. Schürmann and S. Sigg, "Secure Communication based on Ambient Audio," *IEEE Transactions on Mobile Computing*, vol. 12, no. 2, pp. 358–370, 2013.
[20] S. N. Premnath, P. L. Gowda, S. K. Kasera, N. Patwari, and R. Ricci, "Secret Key Extraction using Bluetooth Wireless Signal Strength Measurements," in *Proc. IEEE SECON*, 2014.
[21] W. Wang, J. Lin, Z. Wang, Z. Wang, and L. Xia, "vBox: Proactively Establishing Secure Channels between Wireless Devices without Prior Knowledge," in *Proc. ESORIC*, 2015.
[22] J. Yi, W. Sun, J. Koo, S. Byeon, J. Choi, and S. Choi, "BlueScan: Boosting Wi-Fi Scanning Efficiency Using Bluetooth Radio," in *Proc. IEEE SECON*, 2018.
[23] A. Ng, "Clustering with the K-means Algorithm," *Machine Learning*, 2012.
[24] B. Kaliski and J. Staddon, "PKCS #1: RSA Cryptography Specifications Version 2.0," The Internet Society, Tech. Rep., 1998.
[25] J. Daemen and V. Rijmen, *The Design of Rijndael: AES-The Advanced Encryption Standard.* Springer Science & Business Media, 2013.