Easy Peasy: A New Handy Method for Pairing Multiple COTS IoT Devices

Heng Ye, Qiang Zeng, Jiqiang Liu, *Senior Member, IEEE*, Xiaojiang Du, *Fellow, IEEE*, and Wei Wang, *Member, IEEE*

Abstract—Context-based paring is a promising direction for pairing IoT devices constrained in user interfaces (UIs). However, it takes a proximate distance or a long time for IoT devices to sense highly correlated context with enough entropy. In this work, we present a fast and secure approach, named MPAIRING, to pairing multiple commercial off-the-shelf (COTS) IoT devices. This approach is based on the key idea that devices co-located within a *physically-secure boundary* can perceive qualified context under the help of human-in-the-loop (HITL). Specifically, we leverage received-signal-strength (RSS) trajectory data with manually-generated interference in a short period as the shared secret to achieve fast and secure pairing. Subsequently, the real-time RSS trajectory data is utilized to generate random numbers in lieu of pre-shared key (PSK), which makes our scheme more resistant to background attacks. We theoretically prove the security of our pairing scheme and implement it in real-world environments. Our experimental results demonstrate that our scheme can effectively defend against malicious devices by imposing a threshold on the similarity of RSS trajectory data. The experimental results also show that, compared with the traditional context-based pairing that takes up to 24 hours, in our scheme it takes only 10 seconds on average for a legitimate device to pass the similarity checking, which is efficient and robust.

Index Terms—IoT pairing, multiple devices, human-in-the-loop

1 Introduction

IoT devices cannot be used for data sharing until they are successfully paired with other counterparts or hubs. Traditional approaches to a secure pairing protocol are usually based on PSKs, which are normally provided by a device's vendor. However, the issue with these contemporary methodologies is their vulnerability to various attacks [1], [2], [3]. For example, with sufficient background knowledge of a legitimate device, a malicious third party may be able to deduce a target device's PSK [1]. Therefore, instead of relying on PSKs, a more secure solution is to use Public Key Infrastructure (PKI), which is still difficult to widely deploy, especially in an IoT scenario. Moreover, a digital certificate in an IoT device can only be used to verify the validity of the IoT device, but cannot verify the counterpart (e.g., a smartphone) being paired with it.

An industrial solution for pairing IoT devices is to rely on user participation to manually passing the secret of a pairing-ready device to another (e.g., inputting a verification code). Unfortunately, in IoT scenarios, most devices are constrained in UIs, meaning that they may not support this method due to the lack of necessary UIs (e.g., screen, keyboard).

In contrast, a context-based pairing approach that relies on the context commonly sensed by devices has been proposed [4], [5], [6], [7], [8]. With this approach, devices could use their surrounding environment (e.g., location, ambient sound, luminance, and even humidity) to generate a shared secret instead of using PSKs, to overcome the problem of UI shortage. With a certain threshold or fuzzy policies, the information extracted from proximate surrounding context must be similar. However, we cannot always expect that all the devices will share a common sensing modality, so it is hard for all COTS devices to be paired. In addition, in modern scenarios devices are usually deployed in a fixed place before they are connected. Sometimes, there is not a close enough distance to extract highly relevant information from the context [9]. Moreover, contextual information such as ambient sounds differs slowly over time, requiring a longer time for devices to extract information with enough entropy to prevent the pairing protocol from brute-force attacks. Last but not least, as aforementioned, the context-based pairing method cannot support multiple devices placed in different locations at the same time, because the context will vary greatly with distance. Effectively, this would mean that in order to pair with multiple COTS IoT devices, the pairing protocol would need to be executed many times, which is inefficient.

1.1 Motivation, Challenges and Main Idea

We aim to find a secure and usable way for context-based multi-COTS IoT device pairing. Not only should our pairing scheme be robust against malicious third parties, but it

• Heng Ye, Jiqiang Liu, and Wei Wang are with the Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing 100044, China.
E-mail: {heng.ye, jqliu, wangwei1}@bjtu.edu.cn.
• Qiang Zeng is with the Department of Computer Science, George Mason University, Fairfax, VA 22030 USA. E-mail: qzeng2@gmu.edu.
• Xiaojiang Du is with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030 USA.
E-mail: xdu16@stevens.edu.

Manuscript received 31 December 2021; revised 23 May 2022; accepted 19 July 2022. Date of publication 17 August 2022; date of current version 11 July 2023.
This work was supported in part by the National Key R&D Program of China under Grant 2020YFB2103802, in part by the National Natural Science Foundation of China under Grant U21A20463, and in part by the Fundamental Research Funds for the Central Universities of China under Grant KKJB320001536.
(Corresponding authors: Jiqiang Liu, Wei Wang, and Qiang Zeng.)
Digital Object Identifier no. 10.1109/TDSC.2022.3199383

should also be easy to implement, fast, and simple enough for a non-expert user to pair all of his devices together.

There are two challenges. (1) Since context, such as sounds and luminance, cannot always be sensed by every device, how to make all devices have the ability to obtain context? (2) Given that the sensed context between multiple devices may vary greatly, how to make all the sensed context have a high degree of similarity?

To tackle these challenges, we design a fast and secure approach, named MPairing ('M' stands for *multiple*), which supports the pairing of multiple COTS IoT devices. It uses received-signal-strength (RSS) trajectory data randomized by manually-generated interference during a short period. The reasons we chose RSS trajectory with manually-generated interference as the context are as follows. First, we can convert commonly-sensed RSS data with manually-generated interference into a random seed, without relying on any PSKs. Second, IoT devices usually have the ability to communicate with a hub and thus can record RSS data with a simple software update, meaning our scheme can be easily deployed in COTS IoT devices without any hardware modifications. Third, as illustrated in Section 6.1, the manually-generated interference can be strongly sensed by nearby COTS IoT devices with high similarity. The context is then considered as shared randomness and allows for multiple devices to be paired simultaneously, which is considered as a critical advantage of our scheme. Our experiment results further show that such interference cannot be precisely captured by an attacker outside a physically secure boundary, such as walls. Therefore, by adding manually-generated interference (e.g., using tin foil to repetitively cover and uncover a router or hub), all IoT devices can be paired efficiently and securely through one pairing attempt.

1.2 Contributions

We make the following contributions:

- We design a fast and secure device pairing method by using only a short period of RSS trajectory data randomized by manual interference. It does not need any specific UIs or hardware modifications.
- With proof-of-concept implementation and extensive experiments, we demonstrate that our pairing method is both fast and secure.
- Our method is able to pair multiple devices during one pairing process, which is an advantage over many traditional methods.

1.3 Organization

The remainder of this paper is organized as follows. We discuss the background and related work in Section 2. In Section 3, the architecture of our scheme and our threat model are introduced. We present the problem statement and describe our schemes in detail in Section 4. In Section 5, we analyze the security of the proposed scheme. The implementation details and the experimental evaluation are presented in Section 6. Finally, we conclude this work in Section 7.

2 Background and Related Work

We will review the literature of IoT pairing and RSS recognition in this section.

IoT Pairing. For IoT device pairing, there are plenty of methods that do not rely on any prior knowledges (e.g., secret code, context or biological characteristics). Usually, these methods require certain special involvement from active users to achieve key agreement. For example, a user could simply compare two authentication strings showed on devices or just use a keyboard to input one device's authentication message to another. However, due to a lack of screen/keyboard on IoT devices, these methods are not always applicable. Therefore, researchers try to use either visual or auditory channel to transfer devices' authentication strings [10], [11]. These are all good attempts but still not a universal solution for all devices.

There is another way of device authentication based on automatic pairing. Instead of secrets generated by HITL, devices rely on the information derived from the particular surrounding environment as the shared secret to execute the pairing protocol. As long as the information stays highly correlated, the success of pairing can be expected. However, recording the surrounding information still demands some common and properly-calibrated sensing capabilities across all devices [4]. Additionally, even with the required capability, the time and storage space consumption for processing the sensed surrounding information is still heavy for IoT devices [5], [6], [7], [8].

Lately, the focus on devices pairing turns into asking an active user to perform some special movements for devices to accumulate a highly-matched message, which could be sensed by the pairing devices simultaneously; the special movements include but are not limited to shaking devices together [12], swiping one through another [13], [14], or tapping in a particular pattern [15], [16]. Ignoring the special requirement of sensors, these solutions raise some problems if, say, devices have fixed positions, i.e., for matched message accumulation, they are too far from one another or cannot be moved as necessary. To address this limitation, some papers suggest using another handy device (a bridge, for instance) to help target devices [17], [18], [19].

In addition, due to attacks such as man-in-the-middle (MITM) [20], [21] or protocol manipulation [2], message sensed by devices should have the characteristics of enough entropy to protect the pairing protocol from brute-force attack [22]. One could use biological features such as touch [23] or heartbeat [24] patterns as the commonly sensed message. The trajectory extracted from RSS [9], Channel State Information (CSI) [25], [26], the gyroscope trend [12] or the related position of device [27] is also effective. Since most IoT devices have the ability to communicate with others via wireless networks (e.g., WiFi, ZigBee, ZWave), RSS is a good choice for device pairing.

RSS-Based Recognition. RSS is also referred to as a received-signal-strength-indicator (RSSI), a measurement of the power present in a received radio signal. Nodes used by an Accuware WiFi Location Monitor and Bluetooth Beacon Tracker are capable of measuring the RSS of nearby WiFi and BLE devices. RSS is usually affected by three factors: path attenuation, shielding, and multi-path effect. RSS values

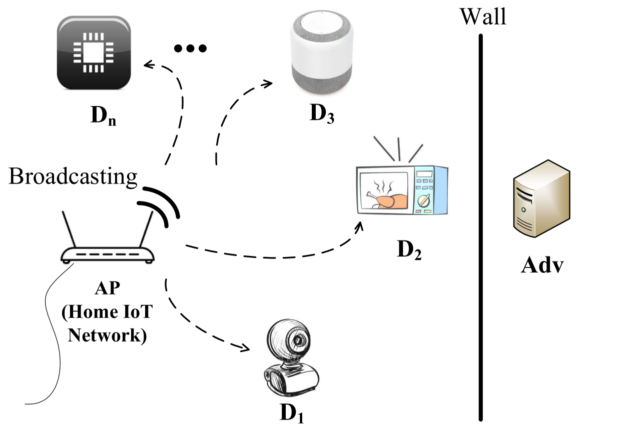


Fig. 1. The system model of our proposed multi-device pairing scheme.

are measured in dBm and typically have negative values, ranging between -110 dBm (extremely poor signal) to 0 dBm (excellent signal).

Various works based on RSS have been proposed in the past few years. At first, the recognition scenario is simple where a pair of devices (transmitter/receiver) is settled. Scholars could use the fluctuation of RSS trajectory from these devices to detect running [28], and even which direction a man is walking [29]. Then, more pairs of devices are planted in different directions to get more detailed reference data. The more data we collect, the higher the RSS recognition accuracy will be [30], [31], [32]. However, RSS itself is a simple measurement and suffers from a precision bottleneck. In present research areas, CSI has slowly taken the place of RSS, since it could provide better fine-grained data for recognition tasks [33]. Nevertheless, RSS can still be used for some applications, such as gesture recognition [34] and breath detection [35]. In comparison to CSI, using RSS does not need special hardware support and takes less time on collecting/calculating. Therefore, it is considered more practical. Furthermore, there are plenty of methods we could apply for our recognition task, such as traditional machine learning methods [36], deep learning methods [37], an ensemble of classifiers [38], and hybrid analysis of features [39].

3 Architecture and Threat Model

3.1 System Model

The system model, shows in Fig. 1, comprises two types of entities: an *Access Point (AP)* and a number of IoT *Devices*.

AP connects to a local area network and provides wireless interfaces for other devices to join the network. For example, in a typical smart home scenario, an AP could be an IoT hub, mobile phone, a smart router, a laptop, etc.

Devices are equipment with the ability and desire to join an IoT network with a legitimate identity. Each device may have different types of functional components such as sensors, actuators, interfaces, etc. We assume that devices do not share any secrets before pairing. However, there could be some devices that have already joined the network.

3.2 Threat Model and Security Requirements

In this paper, we consider the goal of an adversary, *Adv*, is to gain access to users' data. Thus, *Adv* would make efforts to prevent a new device from pairing with an AP and trick it into joining an illegitimate network instead, to obtain privileged access to the device. An *Adv* would additionally disguise himself as a new legitimate device to join an AP's network and get more sensitive data. The *Adv* could achieve this by launching (1) a *deducing attack*, or (2) an *impersonating attack*.

One definition of a deducing attack is when an *Adv* uses his recorded RSS trajectory with manually-generated interferences (outside) to deduce RSS trajectory data with high precision. Then, the *Adv* could use this data to pass similarity checking and join an AP's network illegally. Here, *Adv* may be able to use an enhanced receiver and computing power to generate his RSS trajectory data.

A second definition of a deducing attack is where an *Adv* is familiar with a user's manually-generated interferences pattern and could mimic these interferences around his own device to generate highly-related RSS trajectory data. An *Adv* could use this data to pass similarity checking and join an AP's network illegally.

Also, we define an impersonating attack as one where an *Adv* launches an MITM attack between an AP and a legitimate device, in order to get their identities and disguise himself as one of them. An *Adv* would be able to access the sensitive data *after* the success of the MITM attack.

Similar to [6], we assume that physical boundaries (e.g., walls) draw a natural trust boundary between legitimate devices and an *Adv*'s device present outside a wall. *Adv* is considered to have the ability to eavesdrop, intercept, replay, and modify communications among devices and APs, but he cannot compromise the devices inside. This paper aims to meet the following security requirements:

Data Privacy. The private data (e.g, active time of users and energy consumption) collected by legitimate devices needs to be protected. Legitimate devices should not pair with malicious devices controlled by *Adv*. Recalling the definition we gave for a secure pairing protocol, it usually goes with a two-party mutual authentication and an efficient key agreement process. Therefore, two more security requirements emerge.

Identity Security. Legitimate devices should never accept pairing with a malicious device or reject pairing with the desired one.

Key Security. Data privacy or confidentiality depends on the security of the keys used in transmission. Therefore, the materials for generating keys must have enough entropy.

4 Fast and Secure Multi-Device Pairing Scheme

Before the presentation of our ideas, we investigated and summarized the pairing method for resource-constrained devices, such as wireless headsets, smart speakers, smart door locks, and smart sensors.

4.1 An Industrial Scheme for Resource-Constrained Device Pairing

As shown in Fig. 2, the following steps are taken for the device to join *U*'s home IoT.

- *Setup*. The device waiting for pairing creates an Ad-Hoc network *N* for secret sharing.

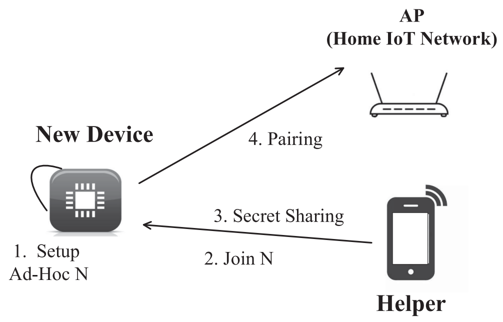


Fig. 2. General scheme for resource-constrained device pairing.

- *Join*. U uses his 'Helper' device (usually a mobile phone) to join the network N.
- *Secret Sharing*. U transmits his secret (e.g., the SSID and password of the home IoT network) to the new device.
- *Pairing*. The device uses the received secret to complete a pairing protocol for joining the home IoT network.

Generally speaking, the industrial realization of *Join* is through the following methods: (1) 'Helper' scans a QR code provided by the pairing-ready device; (2) 'Helper' uses a password marked on the pairing-ready device (3) The pairing-ready device sets N as a public network thus 'Helper' can freely connect to the network. However, preset QR code or password is vulnerable to the background knowledge attack and the use of public network lacks mutual authentication, which can be exploited by adversaries to launch a MITM attack.

4.2 A Typical Context-Based Device Pairing Scheme

A context-based device pairing scheme can be used to cope with the MITM attack. Its procedures are summarized below:

- *Setup*. The pairing devices establish a secure channel, providing confidentiality, integrity, and freshness except authentication.
- *Authentication*. In the authentication phase, the two devices commit to their respective context readings, α and β. Then, devices utilize the secure channel to exchange their commits, before decommitment. After receiving the commit from the other, devices open their commits and compare the context reading. If the context readings, α and β, reach a certain similarity, the device authenticate each other.

According to our experiment showed in Section 6, different locations lead to different contexts. It is still an open problem to make different devices installed in different locations share a similar context.

4.3 Details of Fast and Secure Multi-Device Pairing Scheme

Our fast and secure multi-device pairing scheme is specified in Fig. 3.

First, we give some definitions used in our scheme. Suppose we got n devices in our scenario, just starting to join AP's network, denoted as D_1, D_2, $\cdots$, D_n. An AP connected to a home IoT network is settled in our scenario. We also assume that there is a helper device, denoted as 'Helper' or 'D_H' for the rest of this paper, which has joined the AP's network already in a secure way.

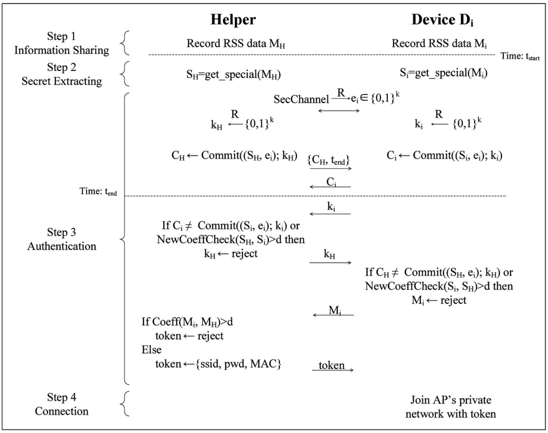

Helper
Device D_i
Step 1
Information Sharing
Record RSS data M_H
Record RSS data M_i
Time: t_{start}
Step 2
Secret Extracting
S_H=get_special(M_H)
S_i=get_special(M_i)
SecChannel $\xrightarrow{R} e_i \in \{0,1\}^k$
$k_H \xleftarrow{R} \{0,1\}^k$
$k_i \xleftarrow{R} \{0,1\}^k$
$C_H \leftarrow$ Commit((S_H, e_i); k_H)
$\{C_H, t_{end}\}$
$C_i \leftarrow$ Commit((S_i, e_i); k_i)
C_i
Time: t_{end}
k_i
Step 3
Authentication
If $C_i \neq$ Commit((S_i, e_i); k_i) or NewCoeffCheck(S_H, S_i)>d then $k_H \leftarrow$ reject
k_H
If $C_H \neq$ Commit((S_H, e_i); k_H) or NewCoeffCheck(S_i, S_H)>d then $M_i \leftarrow$ reject
M_i
If Coeff(M_i, M_H)>d
token ← reject
Else
token ←{ssid, pwd, MAC}
token
Step 4
Connection
Join AP's private network with token

Fig. 3. Our multi-device pairing scheme MPairing.

Initially, we treat all the devices as legitimate devices, each of which has an identity. The pairing protocol runs as follows.

* *Information Sharing*. The pairing devices dive into pairing mode and gather information from AP to extract secrets.
  - 1. Broadcasting (t_i). After getting a 'start' instruction from the user, an AP will continuously broadcast its current system time (in millisecond format), t_i. During the broadcasting period, some manually-generated interferences are applied around AP's transmitting component. The message sent by AP is denoted as $c = (t_1, t_2, \cdots, t_m)$, where t_i denotes the time that the ith packet is sent. Since devices do not share any secret in advance, the broadcasting message is sent in plaintext.
  - 2. Recording (T_{AP}). While AP is broadcasting its message, device D_i will simultaneously monitors all packets around itself as it is on pairing mode (unpaired devices can change their pairing state by receiving wireless broadcast commands). During this process, the RSS of each packet will be recorded. The dataset M_i written by device D_i is defined as $M_i = \{RSS, T_{AP}, Source\} = \{(r_{it_1}, t_1, source), \cdots, (r_{it_j}, t_j, source)\}$, where r_{it_j}, t_j, and $source$ denote the received signal strength at time t_j, the sending time of received message, and the sending source of received message, respectively. Also, Helper will be woken up by AP to record its own dataset, M_H. When AP stopped broadcasting its system time t_i, the next coming second after t_i (in second format) is identified as t_{start} to all the pairing devices and 'Helper'.
* *Secret Extracting*. After gathering enough information, devices will extract the demanding secrets from the context information for secure pairing.
  - 3. Processing (M_i). In this step, a less sensitive but effective message S_i is generated by calling the function $\texttt{get\_special}(M_i)$, $i \in [1, 2, \cdots, n, H]$. The function details can be seen in Algorithm 1.
* *Authentication*. Then pairing devices try to get authenticated by AP.
  - 4. 'Helper' Committing (S_H). In order to let legitimate devices join AP's network N, the password,

pwd, of N should be sent to those devices identified as legitimate, by 'Helper'. First, a secure channel (e.g., a unique session key is established using RSA and a public key announced by the Helper) between pairing-ready device D_i and 'Helper' with credential e_i will be established to ensure the confidentiality, integrity, freshness but not authenticity of their communication [4], as the public key may be announced by an attacker outside the physical boundary impersonating the Helper. All the communication below is over this secure communication channel. Then, D_H will randomly generate its commit key k_H and make its commitment C_H=`Commit`$((S_H, e_i); k_H)$ during time [t_{start}, t_{end}], where t_{start} is the starting time and $t_{end} = t_{start} + Xseconds$ is the ending time of accepting device D_i's commitment. Commitment C_H will be sent to device D_i before t_{end}.

– 5. Devices Committing (S_i). After hearing from D_H, device D_i will return its own commitment C_i=`Commit`$((S_i, e_i); k_i)$ to D_H before t_{end}, where k_i and S_i are obtained by D_i in a manner similar to D_H.
– 6. Devices Reveal (k_i). After t_{end}, device should reveal its commitment by sending its k_i to D_H. It is critical to note that we use t_{end} as a deadline for sending the devices' commitments: (malicious) devices that send commitments after the deadline will be ignored, and thus 'Helper' will not reveal its commitment to them.
– 7. Preliminary Screening (C_i, k_i). 'Helper' opens every commitment and checks the special point from each device. If the correlation coefficient `NewCoeffCheck`(S_H, S_i) reaches a certain threshold, D_H asks device D_i for its complete RSS trajectory data by sending k_H. The function NewCoeffCheck details can be seen in Section 4.5.
– 8. Matching (C_H, k_H). D_i will check the legitimacy of 'Helper' by comparing the similarity between S_H and S_i, where S_H is extracted through `Open` (C_H, k_H). The entire RSS data of D_i then will be sent to the 'Helper' via the secure channel they established earlier.
– 9. Sending (*token*). 'Helper' will send AP's network *token* to device D_i who finally passed the whole data similarity checking, `coeff` (M_H, M_i), where *token* includes network N's *ssid*, *pwd*, *MAC*, etc.

* *Connection*. Devices connected to AP and get ready for users.
– 10. Join. Hearing from D_H, device D_i will join network N with password *pwd*. Then, device D_i and AP step into the key agreement procedure to get their paired key key_i.

Due to the fluctuation of signal emission power and the noise among signal transmission tunnel, the RSS variation tendencies recorded in a stable environment from two devices, which are not close to each other, usually do not reach a high similarity. Therefore, we add some manually-generated interferences around AP to enlarge the fluctuation, for a better RSS similarity. Obviously, the variation tendency while AP is broadcasting is considered as the shared secrets. Hence, the choice of how to apply manually-generated interferences is significant: the better method we used, the more likely that devices are getting RSS datasets with high similarity.

Algorithm 1. `get_special`

```
Input: Collecting dataset, $M_i = \{R_i, T, source\}$.
Output: Special points, $S_i$.
1: $M_i$=removeoutlier($M_i$);
2: $M_{idwt}$=dwt($M_i$), decompose RSS numbers using dwt;
3: $M_i$=waverec(remove_noise($M_{idwt}$)), get the new RSS
   while remove all the signals not in the scope of 5hz-40hz;
4: $S_i$=$\{t_1, t_2, \cdots, t_{2l-1}, t_{2l}\}$=find_anomaly($M_i, \Delta T$), get the
   rough result of some pairs of special time points;
5: $S_i$=check($S_i$), check all the detected points and discard
   some pairs (for too short, too long or mis-classified before);
6: return $S_i$.
```

4.4 Different Types of Manually-Generated Interferences

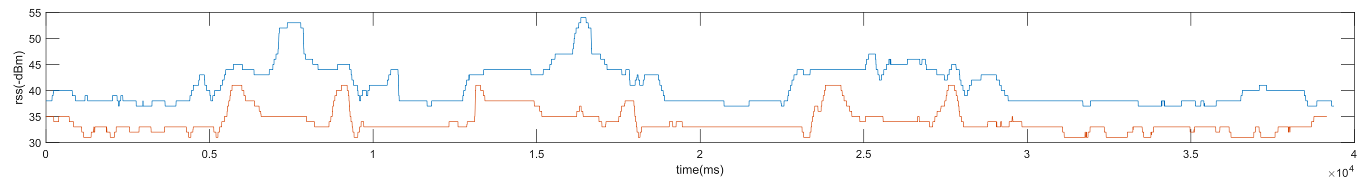
We now present the RSS result of different manually-generated interferences. Generally speaking, using different kinds of manually-generated interferences would lead to different pairing rates. Intuitively, we choose *'open/close door'* in the *line of sight* between AP and devices, as the method of our human intervention. It turns out that we could draw a proximate line to tell which state those devices are in, as shown in Fig. 4a, the red line is the RSS result recorded by 'Helper' and the blue line is recorded by pairing device. However, it takes about 5s to perform a door operation (open and close), while our scheme demands more than 3 times (15s more) of door operations to achieve a better pairing accuracy. Still, we cannot always expect a door to be available in the *line of sight* between AP and devices. So, we start to find another way to manually generate interferences, instead of performing door operations.

Inspired by the poor signal caused by metal material, we try to use a Tin foil to cover/release AP's transmitting component (see Fig. 4b for more details). We use a 20cm long and 10cm wide Tin foil and fold it in half to the shape of 10cm long and 10cm wide. Then, we move the tin foil in a direction perpendicular to the line between AP's transmitting component and the device. As expected, we could draw a line to tell the state of the environment around AP. Even better, the total time used for a successful and accurate pairing could decrease to 10s.

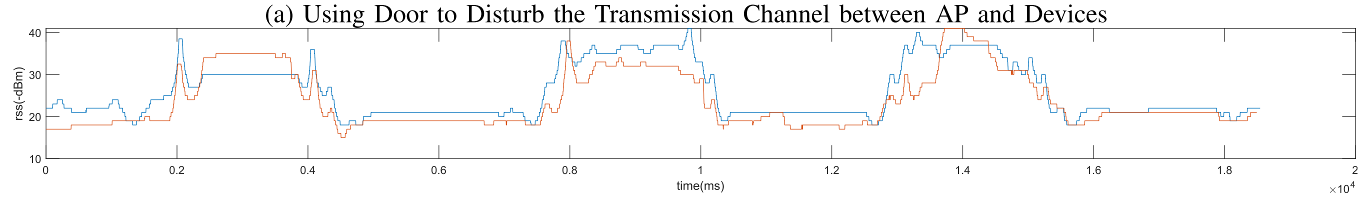
Although, nowadays, we could easily find Tin foil in ordinary homes, we try to find another handy method which could be faster and does not need extra materials. Later, we found waving hands, which could be done in a shorter time, as a good substitution (see Fig. 4c). Although the similarity of RSS results generated by waving hands is not as high as using Tin Foil, with careful design and processing, we can still obtain a pleasant similarity.

4.5 Calculate the Correlation Between Collected RSS Dataset

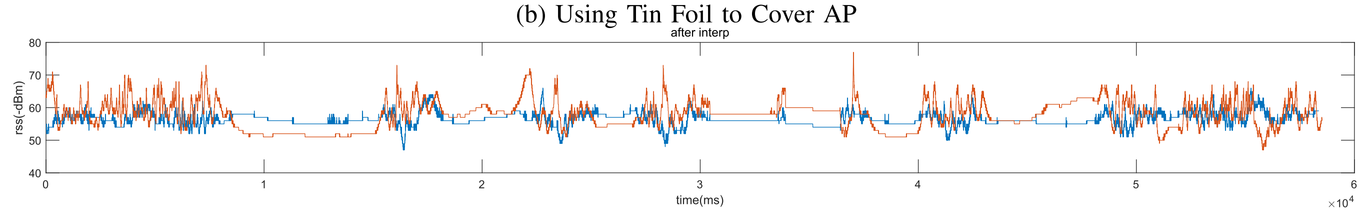
Another question is that how could we efficiently determine the relationship between two RSS datasets. In statistics, the Pearson correlation coefficient is a measure of the linear correlation between two variables X and Y, especially when they are temporal sequences. The coefficient ranges between -1



(a) Using Door to Disturb the Transmission Channel between AP and Devices



(b) Using Tin Foil to Cover AP



(c) Waving Hands around AP

Fig. 4. Comparison of different interventions.

and 1, where 1 indicates a totally positive linear correlation, 0 indicates no linear correlation, and -1 indicates a totally negative linear correlation. Given a pair of random variables *(X,Y)*, the Pearson correlation coefficient can be calculated as:

Definition 1. Pearson correlation.

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{1}$$

where cov is the covariance, σ_X *is the standard deviation of* X, *and* σ_Y *is the standard deviation of* Y.

A key mathematical property of the Pearson correlation coefficient is that it is invariant under separate changes in location and scale, which makes it the basic selection to deal with different RSS trajectory sampled at various locations. However, the asynchronous fluctuation on RSS may still get a high correlation result. To overcome this problem and measure how well the fluctuations are matched between two sampled RSS datasets from different devices, we introduce the concept of another coefficient named *editing distance of multiple pairs of points*.

For simplicity, we give an example on how to get the coefficient of editing distance between two extracted time sequences, in Fig. 5. Obviously, it takes a distance of 3 for

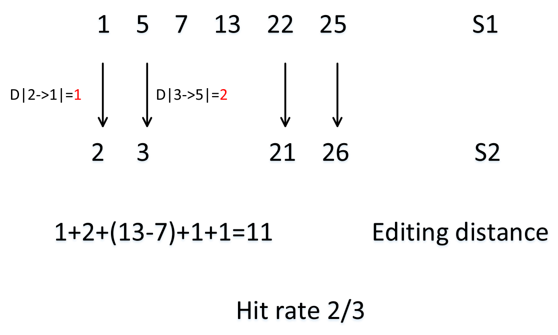


Fig. 5. An example for how to count editing distance.

the first pair of nodes in S1 (1, 5) changes to nodes (2, 3) in S2. Then, it takes a distance of 2 for nodes (22, 25) becomes nodes (21, 26). Since we cannot find another pattern in S2 similar to (7, 13) in S1. In order to eliminate the differences caused by this (7, 13), we calculate the distance of deleting (7, 13), which could be counted as $13-7=6$. So the final editing distance between S1 and S2 is 3+6+2=11. About $\frac{11}{5-1+13-7+25-22} = \frac{11}{13} \approx 85\%$ of the total fluctuation time is not perfectly matched; besides, only $\frac{2}{3} \approx 67\%$ of the total fluctuations are sensed at both devices. Then we calculate the correlation coefficient of these two time sequences as $(1-\frac{11}{13}) \cdot \frac{2}{3} \approx 0.10 < 0.8$, therefore, we consider their similarity is low.

While we give our new method for correlation coefficient calculation, getting the original RSS dataset and its corresponding special points, which are used in similarity checking, remains an unsolved question.

As shown in Algorithm 1, we use Discrete Wavelet Transformation (DWT) to eliminate those noises that are not caused by the user's special movement (the moving frequency of human is considered among 5-40hz [40]). Then, we use a time window ΔT to detect the possible special pair of points (a pair of points denote the period that user performed the required movement) among dataset M_i.

For each target record, we calculate the average value of its neighbors (records collected among the time window). If the value of the target record is beyond the average value, we consider it as an anomaly. According to Fig. 6, for best accuracy, we set the window size to 1 second.

After the process of RSS dataset M_i, we could get S_i for correlation coefficient calculation. Algorithm 2 shows the procedures on counting coefficient in detail.

4.6 The Key-Agreement Protocol With AP's 'Imperfection'

Usually, a key agreement protocol requires the involvement of a secure pseudo-random number generator (PRNG) for

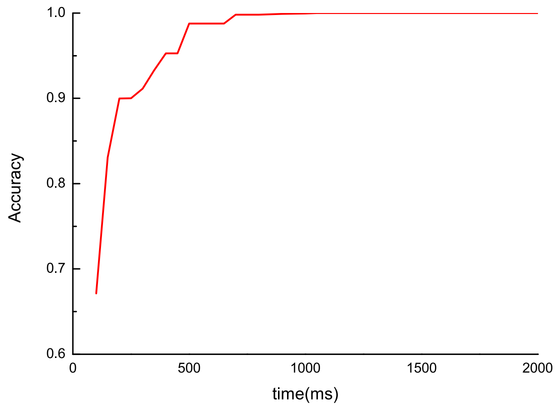


Fig. 6. Different window size and its accuracy for anomaly detection.

key generation and nonce selection. Using PSKS of devices as the random seed for PRNG to generate random numbers will increase the possibility of a background attack. Securely generating required random numbers becomes a problem.

Algorithm 2. NewCoeffCheck

```
Input: Special points correspond to user's activities, $S_i, S_j$.
Output: Correlation coefficient C
 1: $N_i = zeros(S_i(end), 1)$;
 2: for each $s_{2k-1}, s_{2k} \in S_i$ do
 3:     $N_i(s_{2k-1} : s_{2k}) = 1$;
 4: end for
 5: $N_j = zeros(S_j(end), 1)$;
 6: for each $s_{2k-1}, s_{2k} \in S_j$ do
 7:     $N_j(S_{2k-1} : S_{2k}) = 1$;
 8: end for
 9: $c_p = Pearson(N_i, N_j)$;
10: $c_d = \frac{Editing\_Distance(S_i,S_j)}{Total\_Distance(S_i,S_j)}$;
11: $C = c_p * (1 - c_d)$;
12: return $C$.
```

Recall that we let AP continuously broadcast its current system time t_i to build up the final sensing dataset, $T_{AP} = (t_1, t_2, \cdots, t_m)$. Next, from dataset T_{AP}, we are able to get the sending time of each packet in detail. If we set the broadcasting frequency to kHz (usually 1 kHz), due to the imperfection of each device, the total sending time of every k packets will differ within the millisecond range. Therefore, devices could use this difference as a random seed to generate their session keys. For example, every time a device gets k packets (sending in one second), it checks whether the RSS is in fluctuation. If so, the reading of milliseconds of the current time will be tailed to the old random seed to get a new random seed, $\hat{r}_{seed} = r_{seed}||t_k$.

Here, we analyze the randomness of our random seed by the randomness of time readings. Every time we update our random seed, we record the time reading in milliseconds range (2^3 bits). Then, we apply the NIST suite of statistical tests on our recording dataset (8 bits, 10^6 updates). As we can see in Table 1, all these p-values are greater than 0.01, which indicates the high probability that our dataset is randomly generated.

TABLE 1
P-Value of Several NIST Statistical Tests for our Random Seed

NIST test	p-value
Frequency	0.739918
BlockFrequency(m=128)	0.506438
CumulativeSums(forward/reverse)	0.372123/0.698499
Runs	0.500798
LongestRun	0.180609
Rank	0.155209
FFT	0.122325
Non Overlapping(m=9,B=00000001)	0.647302
Overlapping(m=9)	0.110434
Universal	0.302109
Approximate Entropy(m=10)	0.172934
Random Excursions(x=±1)	0.164011
Random Excursions Variant(x=-1)	0.445935
Serial(m=16)	0.107192
Linear Complexity(M=500)	0.449602

5 Security Analysis

In this section, we will discuss why our scheme is secure enough for device pairing. Before the analysis, we tested 12 IoT device locations inside and outside of an office/ apartment.

RSS/SNR Coefficient in Different Locations. First of all, similar to [6]: physical boundaries (e.g., walls) draw a natural trust boundary between legitimate devices and an *Adv*'s device present outside a wall, we realize that an adversary *Adv* cannot get as high signal strengths as legitimate devices do, due to signal fading. Different propagation media cause different distortion and attenuation of signals. Also, compared to air, the walls are not conducive for signal transmission and therefore are bound to induce a non-negligible signal strength attenuation. In Table 2, the experiments results are presented.

As we can see in Table 2, the SNR decays rapidly when the receiver is placed outdoors. This decay is caused by distance attenuation and the signal reflection, meaning that even if an *Adv* were to use an enhanced receiver to collect more accurate RSS trajectory data with manually-generated interferences, the walls would still prevent AP from exposing itself to *Adv*.

Moving on, we now analyze the security of our scheme by considering how the scheme resists against the following attacks.

TABLE 2
RSS and SNR in Different Locations

Location	Distance from AP	RSS (dBm)	SNR (db)	noise level (dBm)
Office	1	-29	24.78	-53
	2	-32	23.60	-55
	3	-33	24.05	-57
	4	-37	23.74	-60
	5 (outdoor)	-60	11.95	-72
	6 (outdoor)	-65	10.03	-75
Apartment	1	-26	25.12	-51
	2	-29	24.47	-53
	3	-31	24.02	-55
	4	-35	22.18	-57
	5 (outdoor)	-62	10.78	-73
	6 (outdoor)	-67	10.01	-77

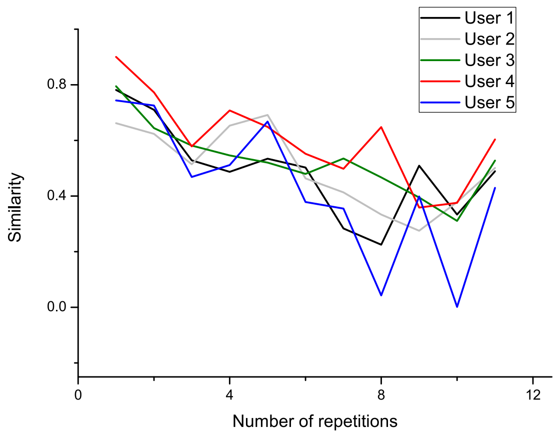


Fig. 7. Similarity checking among different users with similar references generate pattern.

1) *Deducing attack*. We first define a security game between a challenger C and an adversary A:

(State-IND).

Setup: A gets the number of SNR (not exceed 12).

Phase 1: A records some real-world dataset of RSS $R = \{r_1, \cdots, r_t\}$ and sends it to C. C responds with the real state (stable, fluctuate) of each r_i.

Challenge: A chooses a pair of RSS records (r_0, r_1), where r_0 is recorded under a stable environment and r_1 is recorded with manually-generated inferences. A sends the pair to C. C flips a coin $b \in \{0,1\}$ and returns $R_b = r_b + \eta + loss$, where η is the noise ranging around $\{r_b$+SNR$\}$ and loss is the recording loss ranging around$\{-2$dbm$\}$.

Phase 2: A records some new dataset of RSS (differ with the dataset used in phase 1) and gets their real state from C.

Guess: According to R_b, A makes a guess of b'.

We say that the scheme is secure if any polynomial-time A in the above game has at most a negligible advantage.

$$Adv^{State-IND}_{scheme,A}(1^l) = \left| Pr[b' \sim b] - \frac{1}{2} \right| \leq negl(l) \tag{2}$$

where $negl(l)$ denotes a negligible function in l.

Based on Table 2, the noise level of *Adv* is similar to the range of fluctuations, which could result in a stable RSS record similar to an unstable one. This is advantageous as *Adv* could not tell noises from normal RSS results.

Besides, as we can see in Fig. 7, after recording user U's RSS trajectory data with manually-generated interferences, we repeatedly record some other users' RSS data with interferences generated in the same pattern as U did. It turns out that even if other users know how the human generates interferences, the similarity between their RSS data and U's data is not high enough (most time the similarity is less than 70 percent) to pass the similarity checking.

Therefore, *Adv* can not launch a deducing attack to pass similarity checking.

2) *Impersonating attack*. In our pairing scheme, *Adv* may impersonate a legitimate device/AP by launching a MITM attack. Since devices should reveal their RSS data for similarity checking, we use a commit-reveal architecture with time limits to prevent the possible harm caused by RSS trajectory data exposure. We get the following lemma.

Theorem 1. *Our scheme is secure against* Adv*'s impersonating attack.*

Proof. First, we define the probability that *Adv* can successfully pass similarity checking:

$$p_A = max_{C_A}(pr[sim(S_A, S_D) \geq t])$$

Here, C_A is *Adv*'s commitment, C_A=$\texttt{commit}((S_A, e_A); k_A)$ where k_A is the secret involved in commitment generation and S_D is committed by devices/Helper.

Since devices will not reveal their raw RSS data during committing procedure and we have shown that *Adv* cannot deduce legitimate device's RSS trajectory data, the probability that *Adv* can successfully pass similarity checking is:

$$p = max_{k \in \mathbb{K}}(pr[Dis(Dec(C_A, k_A), S_D) \leq d])$$

When C_A and S_D are determined, *Adv* needs to have the ability to find a suitable k_A so that $Dec(C_A, k_A)$ and S_D have sufficient similarity. Since the commitment protocol is collision resistant, it is hard for *Adv* to find k_A, p is negligible. Also, the existence of the credential e_i in legitimate device D_i's commitment C_i is considered as a secret to *Adv*, which keeps the freshness of commitment C_i and makes *Adv* cannot perform a MitM attack by commitment replay to get devices' RSS data. Additionally, when devices find that AP's broadcasting is finished, the starting time t_{start} and the ending time t_{end} of accepting commitments are determined in our protocol. The only way that *Adv* can alter t_{start} and t_{end} to launch an MitM attack is to disguise himself as AP and keep broadcasting to all pairing-ready devices includes 'Helper', which can be easily detected by 'Helper' since 'Helper' has already connected to AP. Hence, our scheme is secure against *Adv*'s impersonating attack. □

3) *Radio Interference Attacks*. If *Adv* were to use a very powerful signal to interfere with AP's communication, it will lead to a pairing failure. As described in the threat model, *Adv* could easily jam AP's communication, due to the fact that all radio-based pairings are commonly vulnerable to radio interference attacks. However, launching such attacks could be easily perceived by a user. Likewise, *Adv* still cannot get any useful data through these means.

6 Evaluation

In this section, we present the evaluation of our scheme in two aspects: the correlation of sensed RSS datasets and the usability of our scheme. All experiments were conducted with the following equipment: four desktop computers equipped with an Intel-5300 Wireless Network Adapter as devices, one laptop with FAST-FW54U wireless USB Adapter as AP, and one laptop equipped with an Intel-1535 Wireless Network Adapter acting as the *Adv*. Lastly, we focus on the reliability of our scheme under the situation where another outdoor IoT device is also in pairing mode.

6.1 Correlation

As we have introduced before, some manually-generated interferences are added to enlarge the fluctuation for a

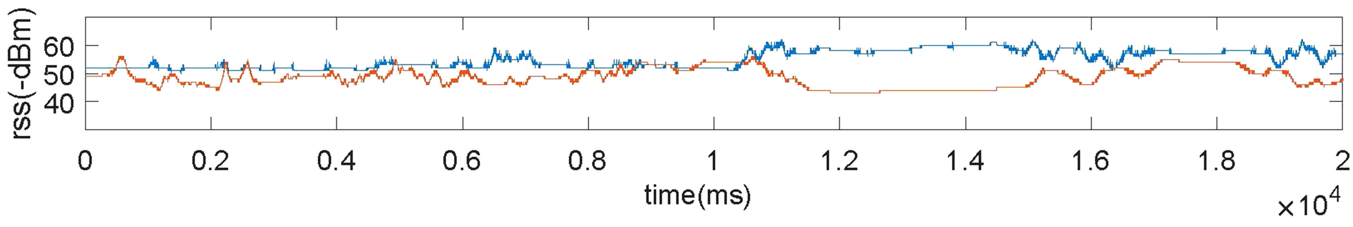


Fig. 8. RSS trajectory without manually-generated interferences.

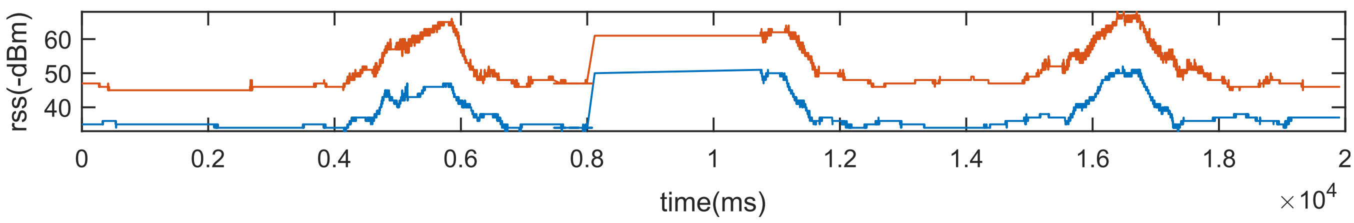


Fig. 9. RSS trajectory with manually-generated interferences.

better pairing accuracy. Without manually-generated interferences, the correlation coefficient between two devices (with a distance of 2m apart) in a normal environment could be really low [17]: the correlation coefficient only hits about 0.53, as shown in Fig. 8.

However, when we apply certain human actions (using Tin foil to wave over AP's transmitting component) while AP is broadcasting, we get the rather surprising results of a perfect match, as seen in Fig. 9.

In order to know how the distance factor could influence the sensed RSS dataset and how the similarity would change between two recorded RSS dataset, we conduct some experiments in a real room. Specifically, the room size is about $20m^2$ (4mx5m), so we set the distance between devices from 1 to 5 meters, respectively (the distance between AP and device is set to 2.5m, see Fig 10). As we can see in Table 3, in the stable situation, the overall trend of the correlation coefficient is inversely proportional to the distance between devices because the transmission power decreases. However, the RSS dataset remains a high similarity rate with the help of human intervention. According to the result shown in Table 3, a distance below 5 meters just has a slight impact over the sensed RSS dataset since the disturbance caused by human is much stronger than what distance does. That is, as long as the fluctuation caused by human intervention is strong enough, the sensed RSS datasets in the same time period can be highly correlated.

From here, we then try to understand how the correlation coefficient will change with the varying distance between AP and other devices. The experiment setup is shown in Fig. 11. Here we set the distance between devices to 2m (to get the best correlation coefficient, see Table 3). Then, we move the devices away from AP, step-by-step (still, manually-generated inferences for pairing are included). The result is shown in Table 4. The correlation coefficients in all tested distances reach 0.75, which is then chosen as a threshold for us to tell whether two devices are in the same indoor environment.

After testing the impact of distance, we then evaluated our scheme in a one-member home and a nine-member office, as shown in Fig. 12. To evaluate the reliability of our

TABLE 3
The Effect of Distance Between Devices on Correlation Coefficient

Distance (m)	Without Intervention	With
0	0.9527	0.8994
1	0.6503	0.8684
2	0.5388	0.9339
3	0.2854	0.83
4	0.4738	0.8181
5	-0.2602	0.7658

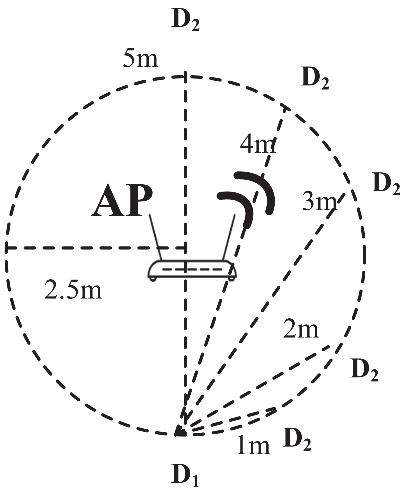


Fig. 10. The environment details for different distance between devices.

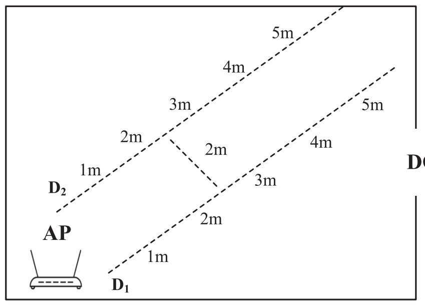


Fig. 11. The setup details for different distance between device and AP.

TABLE 4
Impact of Distance Between Device and AP

Distance from AP (m)	Correlation Coefficient	Distance from AP (m)	Correlation Coefficient
0	0.9218	0.5	0.9184
1	0.9079	1.5	0.8917
2	0.9213	2.5	0.8225
3	0.7619	3.5	0.8407
4	0.8323	4.5	0.8284
5	0.8059		

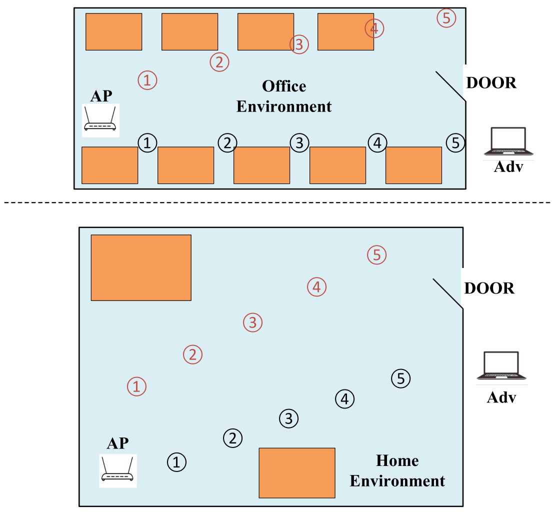
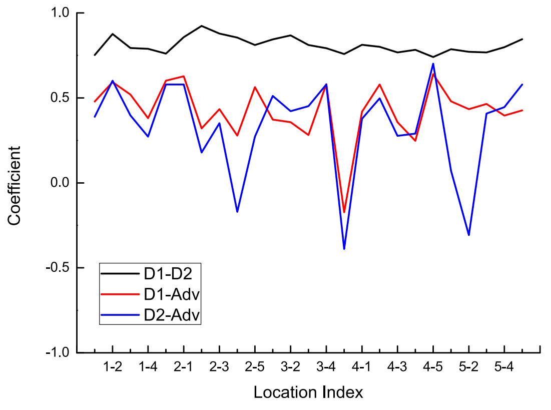


Fig. 12. Details of two testbeds.

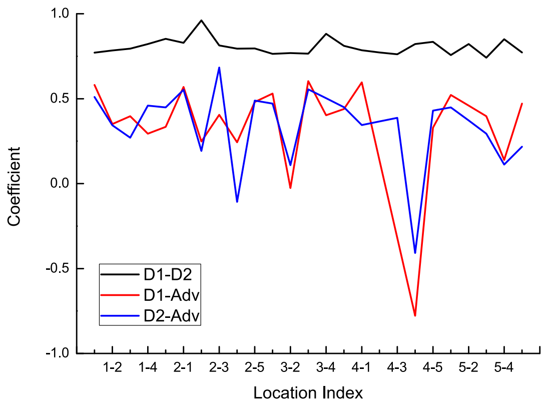
scheme, *Adv* with his outdoor device was placed outside the door, in each testbed.

The whole test result can be seen in Fig. 13. The location index represents the locations that device D_1 and device D_2 are placed at. For example, the index "2-4" means that device D_1 is at position No.2 (black) while device D_2 is at position No.4 (red) in Fig. 12.

All of the above results show that, once an appropriate threshold is chosen, devices outside the house cannot get a high similarity RSS trajectory and thus can not pass the Helper's similarity checking. On the other hand, the indoor legitimate devices still can be paired successfully.



(a) Correlation Coefficient results in an Office



(b) Correlation Coefficient results in a single family House

Fig. 13. Coeff results.

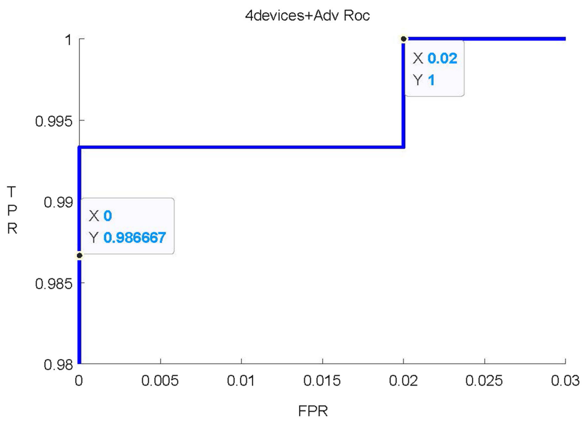


Fig. 14. Tell *Adv* from four legitimate devices.

6.2 Usability

While we were able to get a threshold that allowed us to distinguish indoor devices from those outside, to further evaluate the usability of our scheme, we should also measure the pairing rate ($\frac{\text{successful pairing attemps}}{\text{total pairing attemps}}$) and the pairing time of our scheme, in a couple of real-world scenarios.

First of all, we wanted to determine the threshold to discern indoor devices from outdoor ones. In order to determine this, we ran some more experiments (10 seconds for each experiment, repeated 50 times) using the same layout shown in Fig. 12 with more pairing devices. The ROC curve we get is presented in Fig. 14. When we set the correlation coefficient threshold to 0.7, the total FPR is 0.0213, while the TPR could hit 1. However, this brings a higher rate of false positives, which could badly harm the security of our setup. Therefore, we raised the threshold to 0.75, and by doing so, the FPR decreased to 0.0098, while the TPR remained at 0.9547, which could still be an practical threshold to detect illegitimate devices.

We then evaluated the relationship between pairing time and the pairing rate. Results are shown in Fig. 15. The minimum time we needed to reach an average TPR of 80% was about 8s (waving hands around AP). When the pairing time went up to 20s, we obtained a high TPR of about 99% and a low FPR of about 0.1%. However, as the pairing time goes up, the time required to extract information from raw data would increase accordingly. We use a Raspberry Pi 3

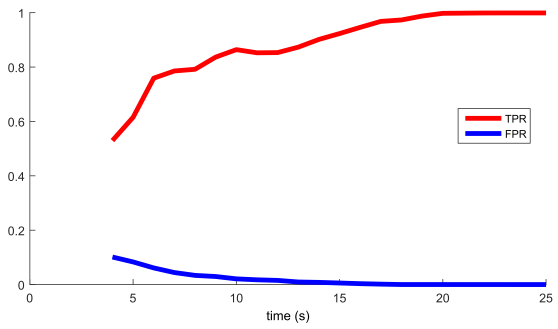


Fig. 15. FPR/TPR results.

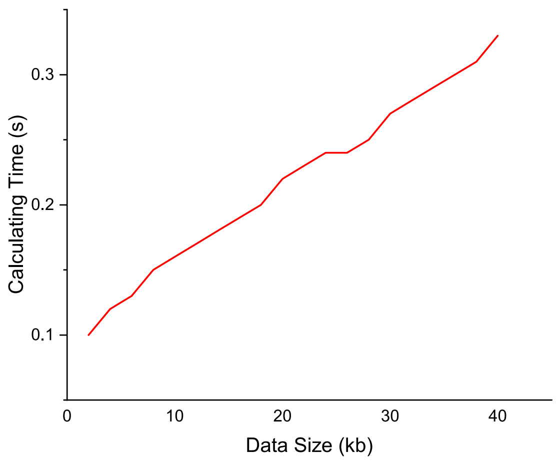


Fig. 16. Computational overhead of RSS data processing.

(1.2Ghz, 1GB RAM) analog IoT device to see how many resources the calculation steps will cost. Luckily, as we can see in Fig. 16, when the pairing time reached 20 seconds (the raw data we got is about 40kb), it takes only 0.33s to calculate special points (as secret). We believe that is acceptable for users to run a program for about 10s to pair all devices together, with one operation. However, if there was a need to achieve higher security, we could increase the pairing time to 20s.

7 Conclusions and Discussion

Pairing UI-constrained IoT devices is challenging, especially when multiple such devices should be paired simultaneously. To solve this problem, we proposed an RSS-based multi-device pairing scheme, where the pairing time is reduced and the maximum distance allowed for successful pairing between devices prolonged.

However, there are some limitations of our scheme. For example, our scheme requires a helper device to connect to AP's network in advance to record the RSS data used to determine the similarity during the similarity checking process. Also, since our scheme cannot be performed without a "Helper" device, the first device to join AP's network needs to have an additional sensing capability to realize a secure pairing procedure. A smartphone should suffice as a helper device. Furthermore, since there is no standard for hand waving, users may not be able to properly add manually interference to RSS data, but this can be checked by the helper, which can remind the user.

References

[1] E. Ronen, A. Shamir, A.-O. Weingarten, and C. O'Flynn, "IoT goes nuclear: Creating a ZigBee chain reaction," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 195–212.

[2] M. D. Donno, N. Dragoni, A. Giaretta, and A. Spognardi, "DDoS-capable IoT malwares: Comparative analysis and mirai investigation," *Secur. Commun. Netw.*, vol. 2018, pp. 1–30, 2018.

[3] J. Mao, S. Zhu, and J. Liu, "An inaudible voice attack to context-based device authentication in smart IoT systems," *J. Syst. Archit.*, vol. 104, 2020, Art. no. 101696.

[4] M. Rostami, A. Juels, and F. Koushanfar, "Heart-to-heart (H2H)," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 1099–1112.

[5] M. Miettinen, N. Asokan, T. D. Nguyen, A.-R. Sadeghi, and M. Sobhani, "Context-based zero-interaction pairing and key evolution for advanced personal devices," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 880–891.

[6] J. Han et al., "Do you feel what i hear? enabling autonomous IoT device pairing using different sensor types," in *Proc. IEEE Symp. Secur. Privacy*, 2018, pp. 836–852.

[7] S. A. Chaudhry, M. S. Farash, N. Kumar, and M. H. Alsharif, "PFLUA-DIOT: A pairing free lightweight and unlinkable user access control scheme for distributed IoT environments," *IEEE Syst. J.*, vol. 16, no. 1, pp. 309–316, Mar. 2022.

[8] J. Zong et al., "Smart user pairing for massive mimo enabled industrial IoT communications," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2020, pp. 207–212.

[9] S. Jana, S. N. Premnath, M. Clark, Sneha K. Kasera, N. Patwari, and S. V. Krishnamurthy, "On the effectiveness of secret key extraction from wireless signal strength in real environments," in *Proc. IEEE 15th Trans. Mobile Comput.*, vol. 12, no. 5, pp. 917–930, May 2013.

[10] N. Saxena, J.-E. Ekberg, K. Kostiainen, and N. Asokan, "Secure device pairing based on a visual channel," in *Proc. IEEE Symp. Secur. Privacy*, 2006, pp. 306–313.

[11] L. Liu, Z. Han, L. Fang, and Z. Ma, "Tell the device password: Smart device Wi-Fi connection based on audio waves," *Sensors*, vol. 19, no. 3, 2019, Art. no. 618.

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

[13] L. Cai, K. Zeng, H. Chen, and P. Mohapatra, "Good neighbor: Ad hoc pairing of nearby wireless devices by multiple antennas," in *Proc. Proc. 18th Annu. Netw. Distrib. Syst. Secur. Conf.*, 2011, Art. no. 16532.

[14] Z. Li, Q. Pei, I. Markwood, Y. Liu, and H. Zhu, "Secret key establishment via RSS trajectory matching between wearable devices," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 3, pp. 802–817, Mar. 2018.

[15] T. Zhang et al., "Tap-to-pair," *Proc. ACM Interactive Mobile, Wearable Ubiquitous Technol.*, vol. 2, no. 4, pp. 1–21, Dec. 2018.

[16] X. Li, Q. Zeng, L. Luo, and T. Luo, "T2Pair: Secure and usable pairing for heterogeneous IoT devices," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2020, pp. 309–323.

[17] J. Zhang, Z. Wang, Z. Yang, and Q. Zhang, "Proximity based IoT device authentication," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.

[18] N. Ghose, L. Lazos, and M. Li, "Secure device bootstrapping without secrets resistant to signal manipulation attacks," in *Proc. IEEE Symp. Secur. Privacy*, 2018, pp. 819–835.

[19] N. Ghose, L. Lazos, and M. Li, "SFIRE: Secret-free-in-band trust establishment for COTS wireless devices," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 1529–1537.

[20] C. Cremers, Kasper B. Rasmussen, B. Schmidt, and S. Capkun, "Distance hijacking attacks on distance bounding protocols," in *Proc. IEEE Symp. Secur. Privacy*, 2012, pp. 113–127.

[21] B. Wang, Y. Sun, T. Q. Duong, L. D. Nguyen, and N. Zhao, "Security enhanced content sharing in social IoT: A directed hypergraph-based learning scheme," *IEEE Trans. Veh. Technol*, vol. 69, no. 4, pp. 4412–4425, Apr. 2020.

[22] J. Sengupta, S. Ruj, and S. D. Bit, "End to end secure anonymous communication for secure directed diffusion in IoT," in *Proc. 20th Int. Conf. Distrib. Comput. Netw.*, 2019, pp. 445–450.

[23] Y. Song, Z. Cai, and Z.-L. Zhang, "Multi-touch authentication using hand geometry and behavioral information," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 357–372.

[24] S. Pirbhulal, H. Zhang, W. Wu, S. C. Mukhopadhyay, and Y. Zhang, "Heartbeats based biometric random binary sequences generation to secure wireless body sensor networks," *IEEE Trans. Biomed. Eng.*, vol. 65, no. 12, pp. 2751–2759, Dec. 2018.

[25] N. Patwari, J. Croft, S. Jana, and S. K. Kasera, "High-rate uncorrelated bit extraction for shared secret key generation from channel measurements," *IEEE Trans. Mobile Comput.*, vol. 9, no. 1, pp. 17–30, Jan. 2010.

[26] J. Hua, H. Sun, Z. Shen, Z. Qian, and S. Zhong, "Accurate and efficient wireless device fingerprinting using channel state information," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 1700–1708.

[27] C. Ruiz, S. Pan, A. Bannis, M.-P. Chang, H. Y. Noh, and P. Zhang, "IDIoT: Towards ubiquitous identification of IoT devices through visual and inertial orientation matching during human activity," in *Proc. IEEE/ACM 5th Int. Conf. Internet Things Des. Implementation*, 2020, pp. 40–52.

[28] K. Chetty, G. Smith, and K. Woodbridge, "Through-the-wall sensing of personnel using passive bistatic WiFi radar at standoff distances," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 4, pp. 1218–1226, Apr. 2012.

[29] F. Adib and D. Katabi, "See through walls with WiFi!," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, 2013, pp. 75–86.

[30] Q. Pu, S. Jiang, and S. Gollakota, "Whole-home gesture recognition using wireless signals (demo)," *ACM Special Int. Group Data Commun.*, vol. 43, no. 4, pp. 485–486, 2013.

[31] B. Kellogg, V. Talla, and S. Gollakota, "Bringing gesture recognition to all devices," in *Proc. Netw. Syst. Des. Implementation*, 2014, pp. 303–316.

[32] F. Adib, Z. Kabelac, D. Katabi, and R. C. Miller, "3D tracking via body radio reflections," in *Proc. Netw. Syst. Des. Implementation*, 2014, pp. 317–329.

[33] K. Ali, Alex X. Liu, W. Wang, and M. Shahzad, "Keystroke recognition using WiFi signals," in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw.*, 2015, pp. 90–102.

[34] H. Abdelnasser, M. Youssef, and K. A. Harras, "WiGest: A ubiquitous WiFi-based gesture recognition system," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 1472–1480.

[35] H. Abdelnasser, K. A. Harras, and M. Youssef, "UbiBreathe: A Ubiquitous non-invasive WiFi-based breathing estimator," in *Proc. 16th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2015, pp. 277–286.

[36] W. Wang, X. Wang, D. Feng, J. Liu, Z. Han, and X. Zhang, "Exploring permission-induced risk in android applications for malicious application detection," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 11, pp. 1869–1882, Nov. 2014.

[37] W. Wang, M. Zhao, and J. Wang, "Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 8, pp. 3035–3043, 2019.

[38] W. Wang, Y. Li, X. Wang, J. Liu, and X. Zhang, "Detecting android malicious apps and categorizing benign apps with ensemble of classifiers," *Future Gener. Comput. Syst.*, vol. 78, pp. 987–994, 2018.

[39] W. Wang, Y. Shang, Y. He, Y. Li, and J. Liu, "BotMark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors," *Inf. Sci.*, vol. 511, pp. 284–296, 2020.

[40] W. Wang, Alex X. Liu, M. Shahzad, K. Ling, and S. Lu, "Understanding and modeling of WiFi signal based human activity recognition," in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw.*, 2015, pp. 65–76.

Heng Ye received the PhD degree with Beijing Jiaotong University, since 2016. His research interests include differential privacy, privacy-preserving data sharing, and IoT.

Qiang Zeng received the bachelor's and master's degrees from Beihang University, and PhD degree from Penn State University. He is an Associate Professor in the Department of Computer Science with George Mason University. He is a recipient of an NSF CAREER Award. His main research interest is Computer Systems Security, with a focus on Cyber-Physical Systems, Internet of Things, and Mobile Computing. He also works on Adversarial Machine Learning.

Jiqiang Liu (Senior Member, IEEE) received the PhD degree from Beijing Normal University, in 1999. He now works with Beijing Jiaotong University as a professor as well as Dean of school of soft engineering. He has published more than 120 research papers. His research interests include trusted computing, privacy preserving, and security protocol.

Xiaojiang Du (Fellow, IEEE) received the BS degree from Tsinghua University, Beijing, China, in 1996, MS and PhD degree in electrical engineering from the University of Maryland, College Park, in 2002 and 2003, respectively. He is the Anson Wood Burchard Endowed-Chair Professor in the Department of Electrical and Computer Engineering with Stevens Institute of Technology. He was a tenured professor with Temple University between August 2009 and August 2021. His research interests are security, wireless networks, and systems. He has authored more then 500 journal and conference papers in these areas, including the top security conferences IEEE S&P, USENIX Security, and NDSS. He has been Awarded more than 8 million US Dollars research grants from the US National Science Foundation (NSF), Army Research Office, Air Force Research Lab, the State of Pennsylvania, and Amazon. He won the Best Paper Award with several conferences, such as IEEE ICC 2020, IEEE GLOBECOM 2014 and the Best Poster Runner-up Award with the ACM MobiHoc 2014. He serves on the editorial boards of three IEEE journals. He is an ACM Distinguished Member, and an ACM Life Member.

Wei Wang (Member, IEEE) received the PhD degree from Xi'an Jiaotong University, in 2006. He is currently a full professor with school of computer and information technology, Beijing Jiaotong University, China. He was a post-doctoral researcher with the University of Trento, Italy, from 2005 to 2006. He was a post-doctoral researcher with TELECOM Bretagne and with INRIA, France, from 2007 to 2008. He was also a European ERCIM Fellow with the Norwegian University of Science and Technology (NTNU), Norway, and with the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, from 2009 to 2011. He has authored or coauthored more than 100 peer-reviewed articles in various journals and international conferences. His main research interests include mobile, computer, and network security. He is an Elsevier "highly cited Chinese Researchers". He is an Editorial Board Member of Computers & Security and a Young AE of Frontiers of Computer Science.