Touch Well Before Use: Intuitive and Secure Authentication for IoT Devices

Xiaopeng Li
University of South Carolina
xl4@email.sc.edu

Fengyao Yan
University of South Carolina
fyan@email.sc.edu

Fei Zuo
University of South Carolina
fzuo@email.sc.edu

Qiang Zeng
University of South Carolina
zeng1@cse.sc.edu

Lannan Luo
University of South Carolina
lluo@cse.sc.edu

ABSTRACT

Internet of Things (IoT) are densely deployed in smart environments, such as homes, factories and laboratories, where many people have physical access to IoT devices. How to authenticate users operating on these devices is thus an important problem. IoT devices usually lack conventional user interfaces, such as keyboards and mice, which makes traditional authentication methods inapplicable. We present a *virtual sensing* technique that allows IoT devices to virtually sense user 'petting' (in the form of some very simple touches for about 2 seconds) on the devices. Based on this technique, we build a secure and intuitive authentication method that authenticates device users by comparing the petting operations sensed by devices and those captured by the user wristband. The authentication method is highly secure as physical operations are required, rather than based on proximity. It is also intuitive, adopting very simple authentication operations, e.g., clicking buttons, twisting rotary knobs, and swiping touchscreens. Unlike the state-of-the-art methods, our method does not require any hardware modifications of devices, and thus can be applied to commercial off-the-shelf (COTS) devices. We build prototypes and evaluate them comprehensively, demonstrating their high effectiveness, security, usability, and efficiency.

CCS CONCEPTS

• **Security and privacy → Authentication**; • **Networks → Mobile and wireless security**.

MobiCom '19, October 21–25, 2019, Los Cabos, Mexico
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6169-9/19/10…$15.00
https://doi.org/10.1145/3300061.3345434

KEYWORDS

Internet of Things; authentication; virtual sensing

ACM Reference Format:
Xiaopeng Li, Fengyao Yan, Fei Zuo, Qiang Zeng, and Lannan Luo. 2019. Touch Well Before Use: Intuitive and Secure Authentication for IoT Devices. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom '19), October 21–25, 2019, Los Cabos, Mexico.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3300061.3345434

1 INTRODUCTION

As the cost of Internet of Things (IoT) enabling technologies keeps decreasing, a wide range of devices have become smart. According to Gartner, by 2022 a typical family home could contain 500 smart objects [15]. A smart environment usually has multiple people (e.g., kids and adults, patients and doctors, employees with different duties), who have physical access to the deployed smart devices; but not all of them are supposed to configure the devices and access potentially sensitive information stored in them. Therefore, authenticating IoT users is an important problem [21].

However, authentication for IoT devices is challenging due to their unique characteristics. First, many IoT devices lack traditional user interfaces, such as keypads and displays. Authentication based on *passwords* is thus infeasible for such devices. Second, due to cost constraints, it is probably unrealistic to integrate costly hardware components, such as fingerprint scanners and NFC readers, into them. Thus, authentication via *fingerprint scanning* and *NFC tokens* is not an option for those inexpensive devices. Third, IoT devices are highly diverse in terms of embedded sensors, shape, size, and installation, which makes it challenging to come up with a uniform authentication approach. For example, *voice recognition* can be used for person identification, but it requires microphones, not to mention various attacks against automatic speech recognition systems [4, 47, 71, 73, 74].

According to the characteristics of IoT devices (lack of traditional UIs, cost constraints, and diversity), the following requirements can be naturally derived for the authentication

Table 1: Comparison with some existing techniques.

Method	R1	R2	R3	R4	R5
TouchAuth [70]	✓	✗	✗	✗	✓
ShaVe/ShaCK [43]	✗	✓	✗	✓	✓
Move2Auth [75]	✓	✓	✓	✗	✗
SFIRE [16]	✓	✓	✓	✗	✗
P2Auth	✓	✓	✓	✓	✓

method for IoT devices. It should (**R1**) have no dependency on unusual interfaces or specific sensors; (**R2**) work with commercial off-the-shelf (COTS) devices without requiring hardware modifications; and (**R3**) have no assumption about the sizes and installations of devices. In addition, like authentication for desktops and smartphones, the authentication technique should be (**R4**) secure and (**R5**) reliable.

None of the existing authentication techniques meet all the requirements. Table 1 summarizes the strengths and drawbacks of existing approaches. For example, *TouchAuth* performs authentication by having the user wearing a customized wristband touch an analog-to-digital (ADC) pin of the IoT device [70]. It does not meet **R2**, since it requires hardware modifications of the device to expose an ADC pin. It does not satisfy **R3**, as the approach only works with devices installed indoors. Its security (**R4**) is also questionable (see Section 8).

We notice that some techniques proposed for pairing IoT devices may be adapted to the authentication purpose. By shaking the user's smartphone and the smart object (held together in one hand) [43], the shared movement sequence can be used for authentication. However, it violates **R1** and **R3**, as the approach assumes the smart object contains an accelerometer sensor and the object should be shakable (i.e., small and mobile). Both *Move2Auth* [75] and *SFIRE* [16] establish authentication by moving the user's smartphone around the IoT device and then compare the RSS (Received Signal Strength) changes with the smartphone sensor trace. However, the *proximity*-based approach can be exploited when multiple devices are near the smartphone (or the attacker relays the signals of a remote target device). Thus, they do not satisfy **R4**. Besides, they are not reliable to be used in an environment that has active persons or objects (i.e., fail to satisfy **R5**), as RSS is likely to be interfered with by walking people or moving objects.

We present an authentication method, named ***Pet-2-Auth*** (P2Auth, for short), that satisfies all the requirements. It is built upon a technique called *virtual sensing* that allows IoT devices to virtually sense *critical events* when the user 'pets' (in the form of some very simple operations) the device. For example, a critical event during twisting a knob is the motion changing the twisting direction. Based on that, the authentication method authenticates users by comparing the critical events sensed by devices and those captured by user wristbands. It can be applied to COTS devices without requiring any unusual interfaces, specific sensors or hardware modifications. It can work with devices no matter they are mobile or mounted, large or small, installed indoors or outdoors. This method is highly secure and reliable as physical operations are required, rather than based on proximity, and is resilient to mimicry attacks.[1]

While there are many aspects to describe those authentication operations, such as velocity, displacement, acceleration, and noise, they usually require specific sensors. Our observation is that *every smart device contains a clock*. We thus propose to build the virtual sensing technique on the device clock, and use *timestamps* to describe those key points (i.e., *critical events*) during authentication. For example, many IoT devices have rotary knobs, and our authentication approach only needs the user wearing a wristband (or holding a smartphone) to twist a knob back and forth for a few times. The virtual sensing technique allows the IoT device to describe the twisting operations using a sequence of timestamps corresponding to the direction-changing motions. The timestamps are then compared with the sensor trace captured by the inertial measurement unit (IMU) of the wristband to evaluate their correlation. If they correlate, the user identity is established using the one carried by the wristband. It is worth noting that our approach is *not* based on behavioral biometrics and thus does not rely on any user habits.

We have implemented prototypes applying P2Auth and performed extensive experiments on devices of three most common types of UIs, including knobs, buttons, and touchscreens. The results show that P2Auth (1) achieves high authentication accuracies (e.g., AUC=0.999 for buttons and touchscreens, and AUC=0.997 for knobs); (2) performs the authentication very quickly (less than 2.5s); (3) has low energy consumption; and (4) is resilient to mimicry attacks.

We make the following contributions.

- Based on the unique characteristics of IoT devices, we identify important requirements that should be met by an authentication method for IoT devices.
- We propose a virtual sensing technique, for sensing and describing operations on devices, that is established on the clock embedded in every IoT device. Thus, it does not require any special sensors or hardware modifications of IoT devices.
- Based on the virtual sensing technique, we propose a novel authentication method that only needs a few very simple operations on devices. It is the *first* authentication method that satisfies all the requirements.

[1] Devices that do not support physical operations, like motion sensors, do not have the need for authenticating physical operations and hence are not considered in our work.

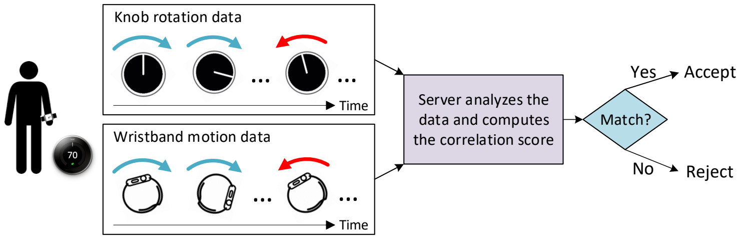
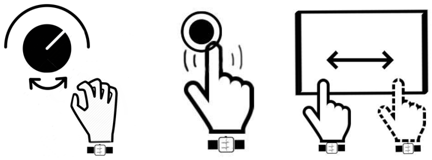


Figure 1: Architecture of P2Auth (twisting a knob as an example).



(a) Twisting. (b) Clicking. (c) Swiping.

Figure 2: Authentication gestures.

- We implement the proposed authentication method in prototypes and comprehensively evaluate them.

The rest of the paper is organized as follows. Section 2 presents the system overview. Section 3 covers the detailed system design and security analysis. Section 4 describes the implementation details. Section 5 introduces how we collect the data and build the datasets. Section 6 presents the evaluation results, and the user study results are discussed in Section 7. The related work is presented in Section 8. The paper is concluded in Section 9.

2 SYSTEM OVERVIEW

Given an IoT device, regardless of its shape, size, installation or embedded sensors, our goal is that a user can authenticate herself through some simple and quick physical operations on the device. Acknowledging that the user interfaces of IoT devices are diverse, here we use a device with a rotary knob to illustrate an overview of the authentication method (and will discuss operations on various UIs in Section 3.1).

Figure 1 shows the architecture of P2Auth, where a user wearing a wristband (such as a smartwatch, a fitness tracker, and a smart ring [29]) twists the knob back and forth for a few times to finish the authentication. In the process, the virtual sensing technique allows the device to makes use of its clock to describe the twisting operations as a sequence of timestamps, while the IMU of the wristband represents the operations as a sequence of accelerometer/gyroscope readings. A server then evaluates whether the two sequences of data correlate. If so, the identity of the current device user is established as the wristband owner.

Is a wristband required? No, the user can opt to hold her smartphone in hand to perform the authentication operations. Our work mainly examines the wristband for two reasons. First, if an employee needs to operate on many devices, wearing a wristband is more convenient than holding a phone. Moreover, while twisting large knobs like the Nest thermostat is not an issue, the user experience of twisting small knobs with a phone held in the hand is not good. Second, the smartphone being held in the hand is closer to fingers and hence can sense the finger operations better than wristbands; therefore, if our system can work well with wristbands, it should also work well with smartphones, which is verified in our evaluation.

Is a server needed? No, the correlation evaluation can be performed by the IoT device or the wristband as well. It is largely an engineering choice depending on which should be trusted and the authentication purposes. If the server is not used, a wristband-to-thing communication channel, such as Bluetooth or Wi-Fi, is needed; plus, either the IoT device shares a key with the wristband of each valid user beforehand, or digital certificates are used to establish a session key to ensure the communication security. All the techniques in Table 1, when used for the authentication purpose, share this requirement. Meanwhile, some other techniques, such as passcodes and fingerprints, do not have this requirement.

Assumptions. (1) The wristband is paired with the server using a secure pairing method [34, 38], and a secret master key is shared between them. So is the IoT device. (2) The communication between the wristband/IoT device and the server is encrypted and private. This can be accomplished with Transport Layer Security (TLS) using a session key derived from the master key, and TLS is feasible on mote-class platforms [70], but other schemes may work as well. (3) When the wristband is taken off, it can detect this, upon which the authentication process is deactivated. When the user puts on the wristband, she needs to authenticate herself to the band using a PIN, after which the authentication process runs in the background. Both techniques are available on smartwatches such as Apple [27] and Android Wear watches [5]. (4) The device has a minimal feedback channel (e.g., an LED or a beeper) [43]. We use it to tell users whether the device status is *locked* (it accepts authentication), *in-authentication*, or *unlocked*. (5) Legacy IoT devices support software updates for adopting our authentication approach. The same or similar assumptions are used in many prior wristband-based authentication works [41, 70].

Threat model. The adversary $\mathcal{A}$ intends to fool the authentication system, such that when $\mathcal{A}$ operates on the target IoT device $\mathcal{D}_a$, the authentication system incorrectly accepts $\mathcal{A}$

as the victim user $\mathcal{V}$ wearing a wristband $\mathcal{W}_v$. We consider the following types of attacks.

Attack-I: When $\mathcal{V}$ is doing non-authentication activities (such as walking, typing, and playing phones), $\mathcal{A}$ launches attacks to make the authentication system incorrectly believe that $\mathcal{A}$'s operations on $\mathcal{D}_a$ correlate with the motions of $\mathcal{W}_v$. For example, $\mathcal{V}$ may walk with arms swinging, and $\mathcal{W}_v$ senses a sequence of motions. At the same time, $\mathcal{A}$ performs authentication operations on $\mathcal{D}_a$ in an attempt to align each operation on $\mathcal{D}_a$ with the motion of $\mathcal{W}_v$.

Attack-II: When $\mathcal{V}$ is performing an authentication on another device $\mathcal{D}_v$, $\mathcal{A}$ may launch attacks on $\mathcal{D}_a$ by mimicking $\mathcal{V}$'s operations. $\mathcal{A}$ may have following three levels of capabilities to launch this type of attacks. (a) $\mathcal{A}$ cannot directly see $\mathcal{V}$'s hand movements due to certain obstructions. (b) $\mathcal{A}$ can clearly see $\mathcal{V}$'s operations on $\mathcal{D}_v$. (c) Besides a clear view, $\mathcal{A}$ is familiar with the authentication system and knows what information is critical for a successful authentication, which helps $\mathcal{A}$ focus on key hand movements of $\mathcal{V}$.

Attack-III: $\mathcal{A}$ can launch energy attacks [69] by performing repetitive interactions with $\mathcal{D}_a$ to initiate authentications. This will cause unnecessary energy consumption on both the IoT device and valid wristbands.

Our work can defend against all the three types of attacks above. The defenses are discussed in Section 3.5.

Attacks beyond scope. $\mathcal{A}$ may be equipped with a camera and a computer assisted by computer-vision techniques to capture and analyze $\mathcal{V}$'s hand movements and reproduce them on $\mathcal{D}_a$ timely. Such attacks can defeat our approach but they require an attacker-controlled camera that captures $\mathcal{V}$'s operations and a robot to reproduce them.

When an authentication process starts, the server needs to contact the wristbands of all the valid users of the IoT device (detailed in Section 3.2), which leaks the information that "someone is trying to authenticate." This leads to privacy leakage and may be exploited by attackers. How to address it is out of the scope of this paper.

Denial-of-Service is possible by jamming the wireless traffic. But if failed authentications occur repetitively, the system can alert security staffs to investigate. Besides, there already have some mature solutions [2, 49] to jamming attacks.

3 SYSTEM DESIGN

In order to implement the "*pet-to-auth*" goal, there are multiple tasks to be resolved. First, how to design usable and effective authentication operations (Section 3.1)? Second, how to trigger the authentication procedure (Section 3.2)? Third, how to represent the authentication operations on the IoT device side and the wristband side (Section 3.3)? Fourth, how to accurately compute the correlation of the data captured on the two sides (Section 3.4)?

Table 2: Physical user interfaces of IoT devices (these devices are surveyed because they are among the most popular IoTs rated by customers [20, 23, 50, 63]).

Interface	Devices

Knob
(6/27)
	Nest Thermostats; Dial Phidget (HIN1101_0);
	August Smart Lock Pro;
	Leviton RC-2000WH Omnistat2;
	Skydrop - Sprinkler Controller;
	Vine Wi-Fi Programmable Thermostat.

Button
(16/27)
	Amazon Echo; AWS IoT Button;
	Triby Smart Speaker; iRobot vacuum;
	Logitech POP Smart Button;
	Ring Video Doorbell Pro; Lockitron;
	Scout Door Lock; August Smart Keypad;
	Chamberlain Wireless Keypads;
	Schlage Connect Smart Deadbolt;
	Samsung SHS-3321 Digital Door Lock;
	Mr. Coffee Smart Coffeemaker;
	Logitech Harmony Elite Smart Controller;
	Emerson Sensi Smart Thermostat;
	Leviton RC-2000WH Omnistat2.

Touchscreen
(8/27)
	Google Home Hub; June Oven;
	Vine Wi-Fi Programmable Thermostat;
	Honeywell Smart Thermostats;
	Hydrawise Smart Irrigation Controller;
	Logitech Harmony Elite Smart Controller;
	Emerson Sensi Touch Smart Thermostat;
	Bosch Smart Thermostats (BCC100).

Multi-UI
(3/27)
	Leviton RC-2000WH Omnistat2;
	Vine Wi-Fi Programmable Thermostat;
	Logitech Harmony Elite Smart Controller.

3.1 Design of Authentication Gestures

An *authentication gesture* is a series of operations performed by a user on the IoT device for authentication. The authentication gesture should be able to be sensed and described by both the wristband and the device. To devise effective and usable authentication gestures, the interaction interfaces of devices should also be taken into consideration.

Table 2 lists some of the popular IoT devices on the market and their UIs. Our survey shows that knobs, buttons, and (usually small) touchscreens are three most common types of UIs. Accordingly, as shown in Figure 2, we design the three authentication gestures. (1) *Twisting knobs back and forth*: when twisting the knob, the micro-controller on the device can detect the amount of current twisting. (2) *Repetitive button clicks*: it will generate consecutive `ButtonDown` and `ButtonUp` events that can be sensed by the device. (3) *Zig-zag swiping on touchscreens*[2]: a touchscreen supports a variety of

[2] Using PINs (or pattern passwords) on a touchscreen might be good for single-user scenarios. But if there are multiple users and the user identity is needed for access control, each user may have to set a username and input it every time. In addition, if there are many devices, remembering many

gestures (i.e., a large design space); after numerous failures (e.g., drawing circles, writing specific characters), we found this gesture can lead to accurate and usable authentication. Plus, it can work well on even very small touchscreens.

These gestures are easy to perform and distinct enough from everyday activities (see Section 3.2). More importantly, each gesture involves abrupt direction/speed changes, which makes it feasible to sense and describe the gesture on both the device side and the wristband side (see Section 3.3).

3.2 Triggering the Authentication Process

Once an IoT device in the locked status detects a *trigger operation*, i.e., *turning a knob*, *pressing a button*, or *swiping on a touchscreen*, it initiates the authentication procedure by contacting the server, which then immediately contacts the wristbands of all permitted users to collect IMU data. This way, the user does *not* need to operate on her wristband to initiate the authentication. Optionally, proximity detection techniques (e.g., based on geolocation, RSSI or Bluetooth) [33, 41] can be deployed to optimize this step by only contacting wristbands within the proximity of the device.

Next, the IMU readings of each wristband are fed to a *binary classifier* (one for each of the authentication gestures) to *determine whether or not the user is performing an authentication gesture.* Only if a wristband passes the classifier checking, does its data participate in the subsequent steps of correlation computation for authentication (Section 3.3-3.4). This step is only used for excluding those users who are doing non-authentication activities (e.g., walking, sleeping, and using computers). It is critical to note that our authentication system does not rely on any behavioral habits of a user (i.e., behavioral biometrics).

We choose SVM to build the classifier, which is effective in the task of activity recognition [25, 52]. Daily activities, such as "walking", "using computers", "using phones", are considered as the negative training samples. We use radial basis function (RBF) as our non-linear kernel, and adopt the standard features described by Kwapisz *et al.* in their work on activity recognition [35]. Specifically, 43 features belonging to the following 6 types are extracted: Average; Standard Deviation; Average Absolute Difference; Average Resultant Acceleration/Gyroscope; Time Between Peaks; and Binned Distribution. We refer the readers to [35] for a detailed description of each type of features.

3.3 Representing Authentication Gestures

When a user performs the authentication gesture, it produces a series of inputs to the device as well as changes in the motion data of the wristband. We first present how to represent an authentication gesture at the device and the wristband. (How to make use of the extracted information for authentication is presented in Section 3.4.) The main challenge is that the diverse IoT devices and wristbands are heterogeneous; however, we need to come up with representations that are feasible on all of them and, moreover, comparable.

3.3.1 *Failed Attempts.*

One may attempt to directly compare the two data sequences (one from the device and the other from the wristband). For button clicking, it is difficult to do so, as they capture completely different information: the device provides very scarce information (i.e., `ButtonDown`/`ButtonUp` and their time), while the wristband IMU provides rich information, e.g., dense acceleration data.

It is possible to extract some motion information (e.g., velocity) from knob twisting; but its motion information is one-dimensional, while the motion data from the wristband is three-dimensional. To compare the two kinds of motion data, we need to find a way to convert the wristband's data into one-dimensional, but there is another challenge—the relative orientation between the wristband and device can greatly vary among users and authentication instances.

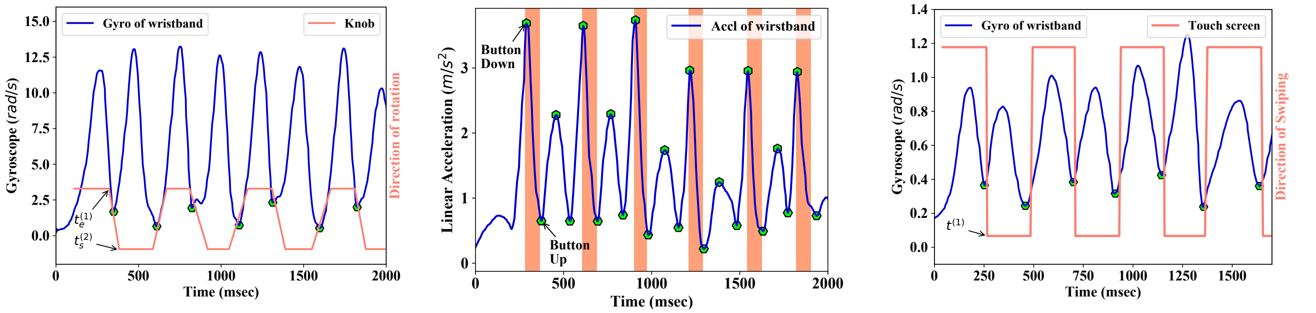
The same issue above is found when comparing the wristband's motion data and the two-dimensional motion data extracted from touchscreen swiping. One alternative way is to extract the movement trajectory from both the touchscreen data and wristband data. For the touchscreen, it is easy to do so by recording the coordinates of each touch point. But fine-grained trajectory inference based on wristband inertial sensor data is still an unresolved problem due to the following reasons [60, 61, 72, 77]: a) the gravity has impacts on the accuracy of orientation projection; b) double integration of the acceleration can cause errors which get worse with time; c) the gyroscope also drifts with time.

3.3.2 *Gesture Representation on IoT devices.*

We propose the *virtual sensing* technique that allows an IoT device to sense and describe an authentication gesture without requiring any specific sensors. A key observation is that a clock exists in every smart device. Virtual sensing thus first extracts *critical events* (event, for short) from an authentication gesture, and then describes each event using a timestamp corresponding to the event occurrence time. This way, an authentication gesture is represented as a series of timestamps. Below we describe virtual sensing for each UI type in detail.

Knob twisting. The event in this case is the abrupt change in the rotation direction. Let $t_e^{(1)}$ be the *end* time of the first rotation and $t_s^{(2)}$ be the *start* time of the subsequent (i.e., second) one. The difference between $t_e^{(1)}$ and $t_s^{(2)}$ corresponds to the short pause for changing the direction, as shown in Figure 3(a) (note that the right y axis indicates the rotation direction: clockwise with larger y value and anti-clockwise

PINs is burdensome (while using the same PIN across devices is not a secure practice). Furthermore, PINs suffer from shoulder-surfing attacks.



(a) Knob twisting. **(b) Button clicking.** **(c) Screen swiping.**

Figure 3: Representations for the three types of authentication gestures.

with smaller y value). For this gesture, we have two series of timestamps of the direction change events derived from the IoT device inputs as follows.

$$\{t_e^{(1)}, t_e^{(2)}, \ldots, t_e^{(n-1)}\} \text{ and } \{t_s^{(2)}, t_s^{(3)}, \ldots, t_s^{(n)}\}$$

Then we combine $t_e^{(i)}, t_s^{(i+1)}$ and replace them with $t^{(i)} = \frac{1}{2}(t_e^{(i)} + t_s^{(i+1)})$ to represent the time of the ith direction change. This way, we obtain $S_D = \{t^{(1)}, t^{(2)}, \ldots, t^{(n-1)}\}$.

Button clicking. A click on button contains two critical events: ButtonDown and ButtonUp, as shown in Figure 3(b), which are represented using two different timestamps: $t_d^{(i)}$ and $t_u^{(i)}$, respectively. We extract two sequences of timestamps for these events.

$$\{t_d^{(1)}, t_d^{(2)}, \ldots, t_d^{(n)}\} \text{ and } \{t_u^{(1)}, t_u^{(2)}, \ldots, t_u^{(n)}\}$$

Since a ButtonDown and a ButtonUp are always coupled, we combine them as a tuple for convenience of data comparison. We thus obtain a sequence of timestamps: $S_D = \{(t_d^{(1)}, t_u^{(1)}), (t_d^{(2)}, t_u^{(2)}), \ldots, (t_d^{(n)}, t_u^{(n)})\}$.

Screen swiping. This gesture produces a more sharp edge at the moment of changing direction than knob twisting, as shown in Figure 3(c). The reason is that the direction change of zig-zag swiping on screen is usually conducted with a shorter pause time. Similarly, we also extract a time sequence $S_D = \{t^{(1)}, t^{(2)}, \ldots, t^{(n-1)}\}$, where $t^{(i)}$ is the end time of the ith swiping.

3.3.3 Gesture Representation on Wristbands. When exploring the gesture representation on wristbands, an interesting finding is that, between the two types of IMU data (i.e., acceleration and gyroscope), one is more useful than the other depending on the gesture.

Knob twisting. A gyroscope can be used for measuring the rate of rotation (i.e., angular velocity), which makes it a good fit for this case. It usually provides three-axis measurements, $[g_x, g_y, g_z]$ along the x, y and z axis, respectively. Instead of examining the signal in each axis, we compute the square root values of the gyroscope data: $g = \sqrt{g_x^2 + g_y^2 + g_z^2}$. The combined gyroscope data yields a reliable performance (see Section 6.2) regardless of the wristband's orientation.

As shown in Figure 3(a), we observe that when the gyroscope decreases to a *valley* point (due to a rotation pause), it corresponds to a direction-changing event captured by the IoT device. We thus extract a sequence of timestamps of all the valleys in gyroscope data, which correspond to the critical events. The time sequence is denoted as: $S_W = \{\hat{t}_v^{(1)}, \hat{t}_v^{(2)}, \ldots, \hat{t}_v^{(m)}\}$.

Button clicking. Given a button clicking, the acceleration data contains salient points corresponding to the two device events: ButtonDown and ButtonUp. When the user's finger moves towards to the button and presses it down (ButtonDown), the acceleration reaches to a peak because the finger's moving speed suddenly decreases to zero. At the moment of ButtonUp, a valley appears in the acceleration because the finger starts to move away from the button. When the finger moves away furthest and stops in the air for a short temporary time, there is a subsequent peak point; and when it starts to move towards to the button again, a subsequent valley.

We thus extract the peak and valley points of each falling edge, as shown in Figure 3(b).

$$\{\hat{t}_p^{(1)}, \hat{t}_p^{(2)}, \ldots, \hat{t}_p^{(m)}\} \text{ and } \{\hat{t}_v^{(1)}, \hat{t}_v^{(2)}, \ldots, \hat{t}_v^{(m)}\}$$

For comparison convenience, we combine each pair of peak and valley that corresponds to a falling edge, and derive the following sequence.

$$S_W = \{(\hat{t}_p^{(1)}, \hat{t}_v^{(1)}), (\hat{t}_p^{(2)}, \hat{t}_v^{(2)}), \ldots, (\hat{t}_p^{(m)}, \hat{t}_v^{(m)})\}$$

Screen swiping. Each direction change of the swiping produces a salient point in the gyroscope data trace. In Fig. 3(c), the point corresponds to a valley, due to a slow down in rotation at the end of a swiping. We obtain a series of timestamps of the valleys in the gyroscope data: $S_W = \{\hat{t}_v^{(1)}, \hat{t}_v^{(2)}, \ldots, \hat{t}_v^{(m)}\}$, where $\hat{t}_v^{(i)}$ is the time of the i-th valley.

3.4 Data Correlation Computation

Once the two timestamp sequences S_D (from the device) and S_W (from the user's wristband) are extracted, the next step is to compute a correlation score between them to determine whether the user should be authorized. We have two observations for evaluating the correlation.

The first one is based on *concurrence*. For each timestamp t in S_D, there should be a corresponding timestamp $\hat{t}$ in S_W that is close to t. A large distance between t and $\hat{t}$ implies a low correlation. Moreover, there should be no extra timestamps in S_W between two consecutive timestamps in S_D.

The second observation is based on *consistency*—the time difference between any two concurring timestamps t and $\hat{t}$ should be consistent. For each authentication, the difference mainly comes from the small time difference between the clocks of the device and the wristband (although we implement the time synchronization, a small difference still exists). Thus, if $\hat{t}$ is ΔT after t at the beginning of the authentication, it should always appear after t for about ΔT time through the authentication. We define a pair $(t, \hat{t})$ that yields a time difference distant from other pairs as an outlier pair, and their time difference as an outlier difference.

We use a machine learning classifier to determine the correlation of two time sequences. Below, we present the feature vector and the machine learning algorithm.

Feature vector. The feature vector consists of features based on the two observations above: (1) *Time difference:* the time difference between each t in S_D and the closest corresponding $\hat{t}$ in S_W; (2) *Secondary time difference* (only for button clicking): the time difference between each t and the second closet corresponding $\hat{t}'$; (3) *Non-correlated event number* (only for knob twisting and screen swiping): the number of extra valley points in S_W between two consecutive device events; (4) *Standard deviation*: standard deviation of the time difference features; (5) *MAD*: median absolute deviation of the time difference features; (6) *Modified z-score*: the modified z-score of the time difference features; to detect an outlier difference, we adopt the modified z-score that is widely used for finding outliers far away from the central point [26]. For each pair of time sequences (S_D, S_W), we compute its feature vector as the input of the classifier.

Classifiers. We consider three widely used classifiers: Support Vector Machine (SVM), k-Nearest Neighbors (kNN) and Random Forest (RF). SVM has been successfully applied to resolve many classification problems [11, 14, 53, 73, 78]. It is capable of deducing non-linear relations between the values in the feature vector to predict a matching score. We use the library of Scikit-learn [48] to train an SVM which implements the Sequential Minimal Optimization (SMO) algorithm [10].

For comparison, we implement kNN and RF. During training, kNN computes the distance between any two samples in the dataset and determines the distance threshold to separate two classes. During testing, each sample is assigned to a class based on the vote of its neighbors. RF is an ensemble method that constructs a number of decision trees and outputs the class label with the most votes from all models.

3.5 Security Analysis

We analyze the resilience of our system to the three types of attacks described in the **Threat model** in Section 2. The experimental results are presented in Section 6.7.

Attack-I: The attacker $\mathcal{A}$ may launch attacks when the victim user $\mathcal{V}$ is walking or using a computer. Our evaluation shows that the binary classifier (see Section 3.2) can effectively defeat such attacks by recognizing the hand movements of $\mathcal{V}$ as not an authentication gesture. Moreover, even the classifier makes a mistake (at a very low probability; see Section 6.1) by classifying it as an authentication gesture, the attack still needs to bypass our defense against Attack-II.

Attack-II (a): This probably corresponds to the most common attack scenario. Without clearly seeing $\mathcal{V}$'s hand movements, it is challenging for $\mathcal{A}$ to determine the exact occurrence time of each input event generated by $\mathcal{V}$ and reproduce them on the target device in *real time*.

Attack-II (b): It provides an ideal attack environment, where $\mathcal{A}$ can clearly see $\mathcal{V}$'s hand interactions with the device, it is still challenging for $\mathcal{A}$ to perfectly mimic $\mathcal{V}$'s actions in *real time*. Average human reaction time is larger than 200ms [17, 30, 44]. Such time difference can be detected by our algorithm.

Attack-II (c): It considers a more powerful attacker who is familiar with our system and knows what events among the device inputs are critical for authentication. Thus, $\mathcal{A}$ may only focus on these input events and try to generate similar events at the same time on the target device. However, it is still difficult for $\mathcal{A}$ to consistently align each input event with that of $\mathcal{V}$ in real time due to human reaction time [17, 30, 44].

Attack-III: To defend against such attacks, an RSSI based proximity estimation is a solution. If a valid user is operating on an IoT device, her wristband should sense relatively strong RSSI from the device. When the server queries the user wristband, it provides the initiating IoT device's ID. A wristband participates in the subsequent authentication only if it senses a high RSSI value from the initiating IoT device. Moreover, if an attacker repetitively triggers authentication failures, the system can warn the security staff, or freeze that IoT device for a while.

4 OTHER IMPLEMENTATION DETAILS

Server. The server performs these main functions: 1) establishing TLS connections with wristbands and IoT devices; 2) running the binary classifier for gesture recognition and

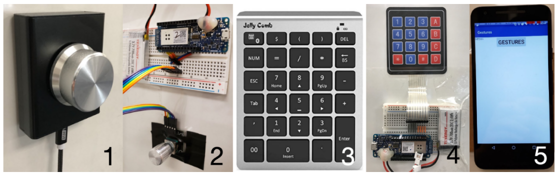


Figure 4: Five devices are used in our experiments, including two knobs (a large knob labeled as 1, and a small one labeled as 2); two keypads (a plastic keypad labeled as 3, and a rubber one labeled as 4; in either case, we only use *one button* for authorization); and a Google Nexus 5 smartphone labeled as 5.

extracting timestamps from the received wristband data; 3) running the authorization classifier and notifying the results. Optionally, the second function can be performed on the wristband to reduce the amount of transmitted data; we leave this for future work.

Wristband. We implement an application to collect the wrist's motion data on the LG W200 smartwatch that runs the latest Android Wear 2.0. The smartwatch has a Bosch BMI160 inertial measurement unit embedded with a triple-axis accelerometer and gyroscope. The original data sampling rate used in our study is set to 100Hz.

IoT devices. A variety of IoT devices/prototypes are used in our experiments, as shown in Figure 4. (1) *Two knob-based devices.* Two different sizes of knobs are used: a large knob labeled as 1, and a small one labeled as 2. The large knob is a volume controller for desktops; we write an interface function to read its rotation data. For the small knob, we use an Arduino board MKR1000 to build its interface. (2) *Two button-based devices.* Two keypads of different materials are used: a plastic keypad labeled as 3, and a rubber one labeled as 4. The plastic one has a Bluetooth module to communicate with the server. An Arduino board MKR1000 is adopted to interface with the rubber keypad, and the data is sent to the server via the Wi-Fi module of MKR1000. (3) *One touchscreen-based device.* The smartphone, Google Nexus 5X manufactured by LG, is used. We implement an application to collect the touch trajectory on the screen and record the coordinates of each touch point in the *xy*-plane of the screen.

5 DATA COLLECTION

We recruit 22 participants: 13 males and 9 females with age ranging from 18 to 41. (a) Each participant is asked to wear a smartwatch and authenticate on each of the three devices for 20 times, including the large knob, the plastic keypad, and the smartphone. Their activity data is used to build Dataset I (Section 5.1), Dataset II (Section 5.2), and Dataset III (Section 5.3). (b) 10 out of the 22 participants perform non-authentication activities, and their activity data is used as negative samples in Dataset I (Section 5.1). (c) 20 out of them take part in the study of mimicry attacks (Section 6.7).

5.1 Dataset I for Gesture Recognition

P2Auth implements a gesture recognizer for each of the three authentication gestures. To study its performance on gesture recognition, 10 participants are asked to wear a smartwatch and perform the four non-authentication activities: (a) walking for 10 minutes; (b) typing two randomly assigned paragraphs on a desktop, which generally takes 5–10 minutes; (c) playing their own smartphones for 10 minutes; and (d) performing any other activities they want for 10 minutes.

This dataset only contains smartwatch data. We have 440 (= 22 × 20) positive samples for each type of authentication gestures. For each positive sample, we randomly take a piece of data that have the same length as the positive sample from one of the above four activities to build the negative sample. We obtain 440 negative samples as well.

5.2 Dataset II for Authorization Classifier

Correlated sensor data pairs. Whenever a participant performs an authentication gesture on a device, we collect one correlated sample consisting of the data pair from the participant's smartwatch and the device. Thus, our dataset contains 1,320 (= 22 × 20 × 3) correlated pairs, each with a label = 1.

Uncorrelated sensor data pairs. Assuming two users (μ_1, μ_2) perform the same authentication gesture at the same time, the time sequence S_{D_1} from μ_1's device and the time sequence S_{W_2} from μ_2's smartwatch constitute an uncorrelated time sequence pair. Note that other users who do not perform an authentication gesture of the same type are mostly filtered by our gesture recognizer and are not considered further.

To build such an uncorrelated sample, we perform *time alignment* for each pair of authentications randomly selected from two users, such that the authentications can be considered as starting at the same time. As studies [17, 30, 44] have demonstrated that the best audio/visual reaction time of human is greater than 50ms (generally between 100–300ms), even for athletes, we shift the timestamps of S_{W_2} to make the starting time difference between S_{D_1} and S_{W_2} within the range of $[-300, -50]$ms or $[50, 300]$ms. We finally generate 1,320 uncorrelated pairs, each with a label = 0.

5.3 Dataset III for Multiple-User Scenario

It is common that multiple users may use the same type of devices at the same time. Note that for devices of different types, the authentication gestures can be used to distinguish users. We built a dataset to exam if P2Auth works well on the multiple-user scenario.

Figure 5 depicts the scenario where *n* users are authenticating to *n* button-based devices at the same time. Take the

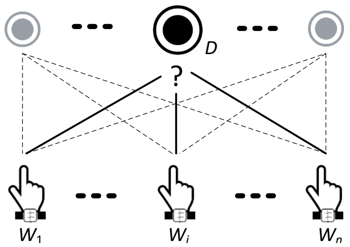


Figure 5: Multiple users simultaneously authenticate to different devices of the same type.

device D as an example, once P2AUTH detects that a user wants to access D, it finds multiple nearby smartwatches performing the same gesture and compares D's input sequence with the motion data of each of these smartwatches to authenticate the right user.

We consider different k-user scenarios with varying k, where $k \in [2, 20]$. For each k-user scenario, given a time sequence S_D of the device D and S_W of the legitimate user's smartwatch, we randomly choose another k-1 users and for each user we select a smartwatch time sequence of the same authentication gesture to construct a sample with k smartwatches. We then perform *time alignment*, such that the k users are considered as authenticating at the same time.

6 EVALUATION

We conducted three in-lab studies to evaluate P2AUTH on its accuracy, stability, efficiency, resilience to attacks, and usability. The first study evaluates the accuracy, stability and efficiency of our system (Section 6.1-6.6). The second study examines its resilience to mimicry attacks (Section 6.7). The last study learns about users' perception of the usability of P2AUTH (presented in Section 7). We also compare our work to existing work on user authentication (Section 6.8). Our studies were approved by IRB at our university.

Metrics. We use False Rejection Rate (FRR) and False Acceptance Rate (FAR) to evaluate the performance of our system. FRR is the fraction of the positive testing data that is misclassified as negative—it tells us how frequently our system denies an authorized user's access. FAR is the fraction of the negative testing data that is misclassified as positive—it tells us how frequently our system accidentally grants an unauthorized individual access to the IoT device. We also report Equal Error Rate (EER) and Area Under The Curve (AUC) of Receiver Operating Characteristics(ROC) curve.

A lower FRR indicates that the system makes fewer mistakes for authorized users, resulting in better *usability* as fewer authorized users need to authenticate again. On the other hand, a lower FAR indicates better effectiveness of the system in preventing adversaries from gaining access. Thus, *a low FAR is good from the security point of view.*

6.1 Accuracy of Gesture Recognition

We use Dataset I in this experiment. We adopt the 10-fold cross-validation to evaluate each gesture classifier. In a 10-fold cross-validation, the data is randomly divided into ten equal-sized pieces. Each piece is used as the testing set with training done on the remaining 90% of the data. The testing results are then averaged over the ten cases. The average false negative rate—the possibility of failing to identify the authentication gesture—is 0.0045, 0.0023, and 0.0045 for button clicking, knob rotation, and screen zig-zag swiping respectively. Take the screen zig-zag swiping as an example, our classifier is able to distinguish zig-zag swipings from other activities including normal operations on the phone (e.g., casual swiping, taping, and scrolling) with an accuracy of 99.55%. Therefore, the gesture classifiers are able to filter out users who are not doing an authentication gesture.

6.2 Accuracy of Authorization Classifier

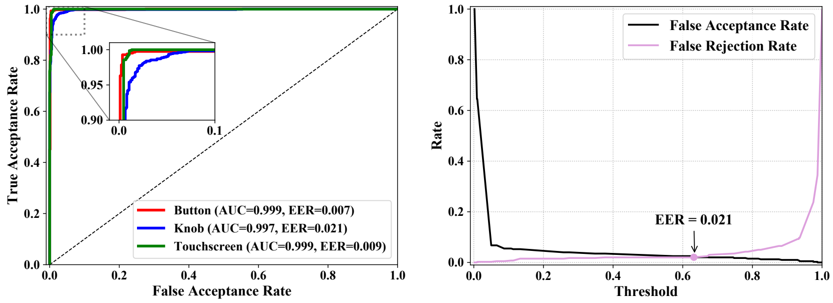
We use Dataset II to evaluate the authentication classifier. We adopt the methodology of Leave-One-Subject-Out (LOSO) which is widely used in evaluating authentication system [9]. In LOSO, we test each subject using the classifier trained with the other 21 subjects' data. We compute the average performance over all the subjects. Through this, we examine whether our classifier is *user independent*—whether it is independent of user profiling and can work for unseen users.

Figure 6(a) shows the average ROC curves of 22 subjects for the three authentication gestures. Specifically, when the threshold is set to 0.53 for buttons, 0.63 for knobs, and 0.87 for touchscreens, P2AUTH obtains an EER of 0.007, 0.021 and 0.009, respectively. Note that the threshold can be set higher for better security but worse usability, and vice verse. Figure 6(b) shows the FAR and FRR of knob authentication by varying the threshold value.

We denote the FAR of the gesture recognizer as e_1 and the FAR of the authorization classifier as e_2. By combining the two components, the overall probability of incorrectly rejecting a legitimate user can be estimate theoretically as $1 - (1 - e_1) * (1 - e_2)$: that would be 0.011, 0.023, and 0.013, respectively, for the three types of authentication gestures.

6.3 Multiple-User Setting

We next consider a more challenging scenario where multiple users use different devices at the same time. For example, user 1 (wearing a wristband $\mathcal{W}_1$) is using a device ($\mathcal{D}_1$) while user 2 (wearing a wristband $\mathcal{W}_2$) is using another device ($\mathcal{D}_2$) at the same time. In this case, the authentication system has to compare the timestamp data of $\mathcal{D}_1$ with that of $\mathcal{W}_1$ and $\mathcal{W}_2$. As the user number increases, the possibility that $\mathcal{D}_1$ is incorrectly identified as being correlated with some user's wristband ($\mathcal{W}_i$) might increase.



(a) ROC curves.

(b) (FAR and FRR) vs. threshold.

Figure 6: (a) The average ROC curves with AUC and EER of 22 subjects. (b) The FAR and FRR with different thresholds for knob authentication.

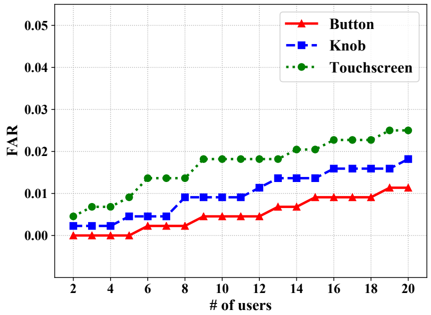


Figure 7: FAR *vs.* number of users for concurrent authentication.

In this scenario, to identify the legitimate user for a device, P2Auth adopts a *ranking-based* mechanism on the correlation scores for all the nearby wristbands that perform the same authentication gesture: 1) the score for the legitimate user should be higher than the threshold determined in Section 6.2; and 2) it is the largest one among all the scores.

Figure 7 shows the FAR when multiple users try to authenticate to the same type of devices at the same time. For instance, in the 10-user scenario, P2Auth achieves an FAR lower than 0.02 with each type of authentication gestures.

When the number of users simultaneously authenticating to the same type of IoT device becomes larger, the possibility that an uncorrelated users satisfies the false acceptance constraints might increase—as a result, the FAR may increase. To handle this, P2Auth can adopt the *collision-based* strategy. That is, if more than one wristbands have a correlation score higher than the threshold, P2Auth rejects all wristbands and requests the users to re-authenticate. The drawback is that the legitimate users have to re-authenticate for each collision, which harms the usability. But if the security has a high weight for a smart environment of large scale, the collision-based strategy is a good fit.

6.4 Stability

We use Dataset II to evaluate the system stability with different parameters and experimental settings.

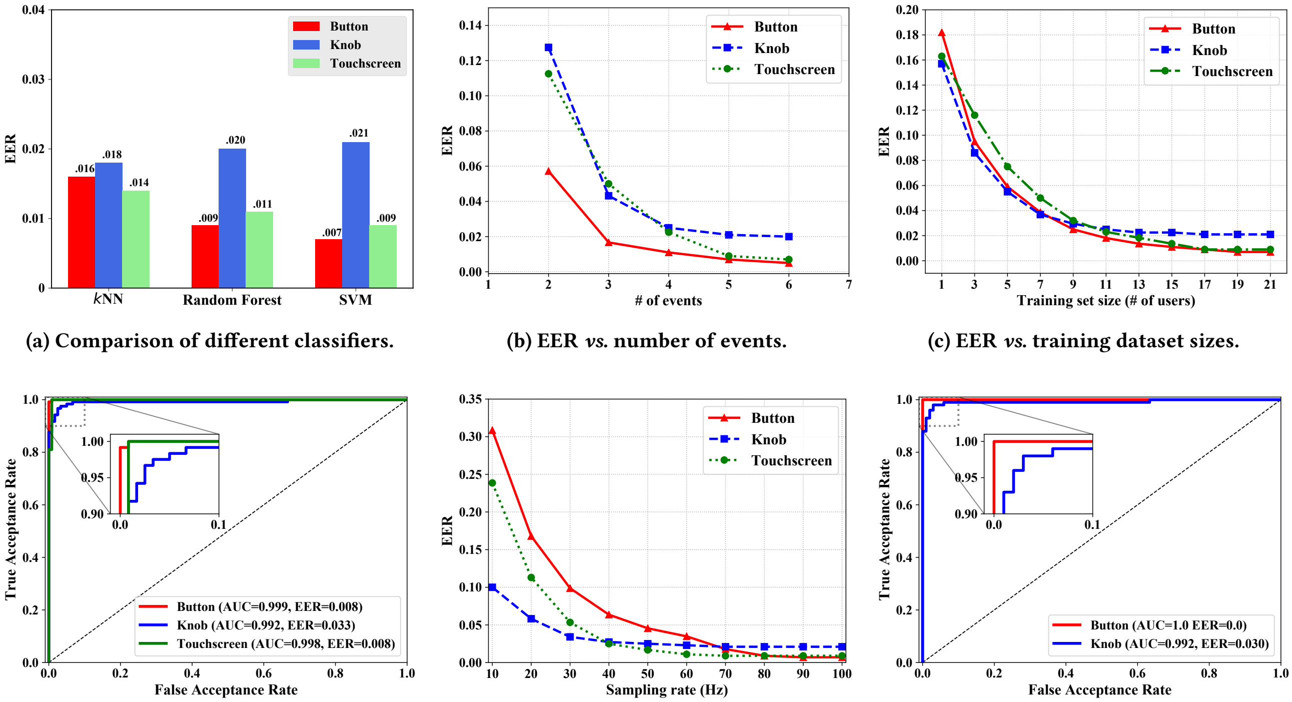
Impact of classifier. We compare the performance of three classifiers, including SVM, kNN and RF, to determine the best one. Parameters of each classifier are tuned to achieve the best performance. For SVM, we examine the linear, polynomial and radial basis function (RBF) kernels, and finally adopt RBF. After the grid search, we set the complexity parameter c as 10, γ as 0.001 for both screen swiping and knob rotation, and c as 5, γ as 0.01 for button clicking. To choose k for kNN, we run tests with k ranging from 1 to 20. The best k is 2 for button clicking, and 4 for the other two types of authentications. For RF classifier, we test different number of trees ranging from 50 to 200, and select the best number as 80 for button clicking, and 100 for the other two.

Figure 8(a) shows the EERs of the three classifiers: SVM performs slightly better than kNN and RF for button- and touchscreen-based devices. So SVM is adopted in our work.

Impact of event number. An event for knob authentication is the direction change of twisting, for touchscreen is the direction change of swiping, and for button is the button clicking. More events provide better security, but also require longer time to authenticate, which hurts usability. Thus, the event number is a trade-off between security and usability.

Figure 8(b) shows the EERs for the three types of authentications with varying number of input events. As expected, EER decreases as the event number increases. We also observe that four events are enough for knob and touchscreen authentication to achieve a satisfactory accuracy, and three button clicks can achieve an EER less than 0.02. We select five events for the three types of devices considering both security and usability. But the number of events can be configured based on the demand; e.g., if the usability has a higher weight, then three clicks for buttons and four events for knobs and touchscreens are appropriate.

Impact of training dataset size. We evaluate the impact of training dataset size on the classifier performance. The training dataset size is defined as the number of users whose samples are used for training. We train the classifier with m ($1 \leq m \leq 21$ in a step of 2) users' data and test it with the data of the rest users ($22 - m$). Each user has 20 correlated time sequence pairs and 20 uncorrelated ones. Figure 8(c) shows the average EER for different number of training users. It can be seen that a satisfactory accuracy can be achieved when a training set size is larger than 13, and the accuracy of the classifiers converges given more than 17 users.



(a) Comparison of different classifiers.

(b) EER *vs.* number of events.

(c) EER *vs.* training dataset sizes.

(d) EER *vs.* installation methods.

(e) EER *vs.* sensor sampling rates.

(f) ROC for unseen devices.

Figure 8: Impact of different parameters and experiment settings.

Impact of installation method. Devices may be installed in different ways for various scenarios. We examine two commonly used installations: (a) mounting to walls and (b) installing/placing on the ground. The 22 participants in Dataset II authenticate to devices mounted to a vertical board (I_a). We then recruit *another* 6 participants to access devices placed on a table (I_b). We use the dataset from I_a to train the SVM classifier and test it on the dataset from I_b. The result in Figure 8(d) shows that different installation methods have little impact on the model performance.

Impact of sampling rate. A higher sampling rate can capture more subtle characteristics of sensor data, but it also introduces higher burdens (e.g., data collection and communication) into the system. The burden mainly comes from the wristbands as we need to collect all nearby wristbands' data and send the data to the server. To find the optimal sampling rate for the wristband, we study the sampling rate ranging from 10Hz to 100Hz at a step of 10Hz by downsampling the original sensor data (100Hz).

Figure 8(e) shows the results. We observe that button clicking requires a sampling rate higher than 70Hz to achieve a good performance, and knob rotation and screen swiping only require a sampling rate higher than 40Hz. We thus select a sampling rate of 80Hz, 60Hz, and 50Hz for button clicking, screen swiping, and knob rotation, respectively.

Unseen devices of the same type. We have two knob-based devices—the large and small knobs, and two button-based devices—the rubber and plastic keypads. We examine whether our model trained on one device can work well on another device with the same type of user interface.

The 22 participants in Dataset II perform rotation operations on the large knob, and clicking operations on the plastic keypad. We recruit *another* 5 participants to authenticate to the small knob and the rubber keypad. We train an SVM classifier using the data from the large knob and test it with the small knob, and train another SVM classifier using the data from the plastic keypad and test it with the rubber keypad. The results in Figure 8(f) show that our model trained on one IoT device works well on another device (of the same category) that has not been seen during the training.

6.5 Feasibility of Using Smartphone

We also test the feasibility of our approach when a user holds a smartphone for authentication. We recruit *another* 5 participants and ask them to hold a Google Nexus 5X smartphone to perform the corresponding authentication gestures to the three types of devices. Note that most smartphones have built-in inertial sensors for motion and orientation tracking [65]; thus our approach can generalize well to most smartphones. We use the data collected from the smartphone

Table 3: Authentication time.

Part		**Time (Std)** ms
Performing gesture	button	1793 (524.8)
	knob	1348 (441.2)
	screen	1036 (365.2)
Transferring 2s Accel./Gyro.		64 (9.4)
Data processing and running SVM		28 (7.6)

Table 4: False Acceptance Rate for mimicry attacks.

Attack	Dev.	1	2	3	4	5	6	7	8	9	10	Avg.
Non-expert	button	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	knob	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.1	0.02
	screen	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.02
Expert	button	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.01
	knob	0.0	0.0	0.1	0.1	0.0	0.0	0.1	0.1	0.0	0.1	0.05
	screen	0.0	0.0	0.0	0.1	0.0	0.0	0.1	0.1	0.0	0.1	0.04

and the IoT devices to build a dataset for testing. We train a SVM classifier using Dataset II, where 22 participants wear a wristband for authentication. We obtain an EER of 0.0, 0.03 and 0.01 for button clicking, knob rotation and screen swiping, respectively. We thus conclude that holding a phone for authentication is feasible and works even better.

6.6 Efficiency and Energy Consumption

We next evaluate the response time of P2Auth. The response time is defined as the time interval beginning at when a user starts the authentication and ending at when P2Auth makes a decision. It mainly consists of three parts: (a) the time for performing an authentication gesture; (b) the time for transmitting data to the server; and (c) the time for data pre-processing and running the algorithm to make an authentication decision. Table 3 shows the time consumed for each part. The total response time is less than 2.5s. Thus, P2Auth can make a decision quickly.

We also measure the energy consumption of our application on the smartwatch. We let the smartwatch read 2.5s acceleration data (or 2.5s gyroscope data) with a sampling rate of 100Hz and send it to the server once every minute for one hour (60 times of authentications are performed). For comparison, we first measure the energy consumption when the screen is always on, which is 37.3 ± 1.5mA per hour. The energy consumption is increase to 48.6 ± 3.6mA per hour when running our application while keeping the screen on. It can infer that our application only consumes about 11.3mA in average per hour ($< 2\%$ of the battery capacity).

6.7 Resilience to Mimicry Attacks

Based on our security analysis in Section 3.5, *Attack-I* can be defeated by the gesture recognizer (Section 5.1), and *Attack-III* can be prevented by proximity estimation and device freezing. For *Attack-II*, *Attack-II* (b) and *Attack-II* (c) are more powerful than *Attack-II* (a). We thus focus on evaluating the resilience of our system to *Attack-II* (b) and *Attack-II* (c).

In this study, we have 10 participants act as victims and another 10 participants act as attackers. We first introduce them the three authentication gestures that need to be performed, and then tell them the purpose of this study: an attacker operates on a device of the same type as the victim's and mimics the victim's hand movements to fool our system.

Table 5: Comparison with existing approaches.

Method	**[EER]|(FAR, FRR)**	**Time**(s)	**Event #**
TouchAuth [70]	- | (0.02, 0.058)	1	-
	- | (0.02, 0.011)	5	-
Zebra [41]	- | (0.0, 0.15)	11	~42
	- | (0.0, 0.10)	50	-
Touchalytics [14]	[0.0—0.04]|-	11–43	11
P2Auth	[0.007—0.021]|-	≤ 2.5	5

In our experiment, we provide the attacker $\mathcal{A}$ with a *clear view* of the victim $\mathcal{V}$'s interactions with the device $\mathcal{D}_v$ by placing $\mathcal{D}_v$ next to $\mathcal{A}$, which models two cases: (a) $\mathcal{A}$ and $\mathcal{V}$ are co-present physically, and (b) $\mathcal{A}$ has access to a video such as a surveillance camera aimed at $\mathcal{D}_v$. We consider our experiment setting as an ideal environment for $\mathcal{A}$, as in a real attack, the environment tends to be more challenging due to visual barriers.

Our experiment contains two sessions. (1) *Attack-II* (b): In the first session, we ask $\mathcal{A}$ (non-expert) to follow $\mathcal{V}$'s operations in real time. (2) *Attack-II* (c): In the second session, $\mathcal{A}$ (expert) is given more information (e. g., following the time of each direction change in knob rotation and screen swiping, and the time of `ButtonDown` and `ButtonUp`) and asked to mimic $\mathcal{V}$'s operations in real time. For each session, $\mathcal{V}$ authenticates to each device for 10 times. Table 4 shows the results. We can see that non-expert attackers achieve a very low success rate; although expert attackers perform a little better, it is still hard for them to mimic victims.

6.8 Comparison with Other Approaches

Table 5 shows the comparison of our work with some related existing work. Our work achieves competitive results compared with TouchAuth [70], and obtains better accuracies than Zebra [41]. Moreover, our approach is more efficient and usable than Zebra and Touchalytics [14] in terms of the response time and the number of events needed for authentication. For example, Zebra needs at least 11 seconds and around 42 input events to determine the user identity, while our system only needs up to 2.5 seconds and 5 events.

7 USER STUDY

This study is to evaluate the usability of P2Auth and compare it with the mobile App, which is commonly used for remote control of smart devices. We adopt Samsung's SmartThings [57] mobile App.

7.1 Procedure

We recruit 30 participants in our university. Most are students with the age ranging from 20 to 35. The experiment takes place in our lab with two areas. In each area, we provide three devices of the three categories.

We first explain the purpose of the study to them, and solicit their informed consent to proceed. We explain the two authentication methods and ask them to try each one for several times. We tell them that the timestamp of each operation they perform will be recorded.

For P2Auth, we introduce the three authentication gestures and ask them to authenticate to the devices by performing the corresponding gestures. Note that the users do not need to unlock the wearables before each authentication.

For the SmartThings App, we create a location (lab) with two areas (rooms), and add three devices into each room in the App. We randomly choose three devices and ask each participant to authenticate to them by following the steps: 1) unlocking the smartphone via a swipe (no password is required); 2) locating the SmartThings App, starting it and logging in using an existing account; and 3) locating the given device in the right room.

Note that each participant uses the two authentication methods in random order. After that, the participants rank the usability for each method by answering the questions in the SUS (System Usability Scale), which is a well-known standard for usability study and consists of a 10 item questionnaire with five response options [3]. Table 6 shows the SUS questions adopted in our user study.

Table 6: SUS questions adopted in the user study.

	Question
1	I would like to use this authentication method frequently.
2	I found the method unnecessarily complex.
3	I thought the method was easy to use.
4	I think that I would need the support of a technical person to be able to use this method.
5	I found the various steps in this method were well designed.
6	I thought there was too much inconsistency in this method.
7	I would imagine that most people would learn to use this method very quickly.
8	I found the method very cumbersome to use.
9	I felt very confident using the method.
10	I needed to learn a lot of things before I could get going with this method.

7.2 Results

SUS scores. With P2Auth, the mean SUS scores for knob rotation, button clicking and screen swiping are 72.92 ± 7.06, 74.16±6.38, and 73.21±6.74, respectively. The mean score for SmartThings App is 68.08 ± 7.22. Thus, P2Auth achieves a higher usability score for our three authentication activities.

Time. We also compare the efficiency of the two authentication mechanisms by measuring the time each user spends on authenticating to a device. For P2Auth, we measure the time for performing each of the three authentication gestures (the same as the time of the first part in Table 3). For SmartThings, we measure the time from the moment when a user unlocks the phone and to the moment when the user correctly locates the device. We do not consider the time used for data communication and running authentication algorithm because we only focus on the time taken for the user to perform the authentication actions.

With P2Auth, the mean time for performing a gesture of knob twisting is 1.49 ± 0.48s, for button clicking 1.69 ± 0.44s, and 1.16 ± 0.31s for screen swiping. With SmartThings, the mean time for locating a device is 16.2±4.2s. Our mechanism is much more efficient.

8 RELATED WORK

8.1 Proximity-Based Approaches

The proximity-based techniques have been adopted to detect a user's proximity to a device, such as ZIA [6], Apple's Auto Unlock [28], and Atama's Sesame [40]. However, such techniques can only confirm whether a user is nearby, but not whether a user is actually *using the device* [41], resulting in unintentional authentications if a user is just passing by the device. Short-range radio communication technologies, such as NFC [12], have been proposed to solve the limitations above. However, the price of a wireless NFC reader is relatively high and its size is large, which is not an option for the inexpensive and small devices.

Finally, *mafia fraud (i.e., radio relay) attacks* have been mature for radio based authentication [13, 19, 67]. How to tackle them is being actively investigated [1, 7, 22].

8.2 Biometrics-Based Approaches

Physiological biometrics. Fingerprint and FaceID have been successfully applied to mobile devices, but it requires hardware modifications to integrate these sensors into each

IoT device. Heart-related information, such as heart deformation [39], electrocardiogram (ECG) [56], photoplethysmogram (PPG) [32, 58], and signal pulse response at the hand palm [42], is also used for user identification. But it requires special sensors integrated into IoT devices as well.

Behavioral biometrics. A body of work uses human behavior information to identify users [8, 14, 18, 24, 37, 45, 51, 54, 55, 59, 66]. E.g., SenseTribute [18] requires the motion information collected from *multiple devices* to correctly identify the user. Moreover, it fails for devices that are not movable or do not have motion sensors. Hallmarks [51] proposes to prepare a dataset recording the gestures of *every* user when she operates on *every* object, and infers the user based on whether the person's current gesture resembles the profiled gesture, which makes it inconvenient to use. In a space with only five users, its accuracy is 63.8%.

Many studies have been conducted on authentication to computers and phones. They infer users based on their interactions with devices, such as keystrokes dynamics [46], mouse movements [76], and touchscreen operations [14, 36, 37, 64]. But they are not applicable to most IoT devices, have low accuracies, and/or require a long interaction time. Our approach does not use any behavioral biometrics, and does not have these limitations.

8.3 Other Approaches

TouchAuth performs authentication by having the user wearing a customized wristband touch an analog-to-digital (ADC) pin of the IoT device [70]. But it requires hardware modifications of the device to expose an ADC pin, and can only work with devices installed indoors. Moreover, hypothetically, attacks may be launched when a victim user touches a malicious device and thus discloses her current potential: assuming the attacker can fake a potential with the same value and impose it onto the victim device, the authentication approach will be fooled.

By holding the smartphone and the smart object (in one hand) and shaking them together [43], the shared movement sequence can be used for authentication. However, the approach assumes the smart object contains an accelerometer sensor and the object should be shakable (i.e., small and mobile). *TAG* [68] requires devices to be equipped with not only an accelerometer sensor but also a vibration motor.

Karapanos *et al.* compare the ambient sound noise recorded by the microphones of two devices (e. g., a smartphone and a computer) to determine the proximity of the two devices, which serves as one authentication factor [31]. However, it requires each IoT device to have a built-in microphone. Moreover, it is difficult to thwart co-located attacks where two devices are close and share the same ambient. Furthermore, researchers show that this approach fails if attackers introduce sound that dominates the ambient noise [62].

Zebra [41] is closely related to our work. Both Zebra and our work can be categorized as "bilateral authentication" [41], which uses information coming from two different sources for authentication. Zebra uses mouse and keyboard inputs to authenticate users at a computer, whereas we consider IoT devices that have neither mice nor keyboards. Our work further differs from Zebra in the following two aspects.

First, the goals are different. Zebra aims at implicit (i.e., without requiring explicit actions from users) authentication. However, this goal also leads to a shortcoming that makes it inappropriate for authenticating IoT users: Zebra has high data collection overhead, requiring at least 11 second data (about 42 user interactions) to correctly verify 85% of true users and identify all adversaries. Our approach has an authentication speed comparable with inputting a 4-digit PIN. We only need 2.5 seconds (5 user interactions) for authentication and achieve AUC ≥ 0.997.

Second, the algorithms are different due to different insights. Zebra segments the motion data and infers the user input events from segments. It assigns a score (1 if the inferred event matches the real event, and 0 otherwise) to each segment before summing the scores up. Ours is a holistic approach where the machine learning based classifier makes the authentication decision based on features derived from the very few but high-entropy timestamps of critical events, in contrast to the inferred low-entropy event labels.

9 CONCLUSION

User authentication for IoT devices supports important applications including access control and device personalization. However, due to lack of conventional UIs, cost constraints, and high diversity, authentication for IoT devices has been difficult to approach. Given that a clock exists in every smart object, we use "timestamps" as a universal language for authentication. The user wearing a wristband or a smart ring (or simply holding her smartphone) performs some very simple operations on the device. Then, the motion trace extracted from the wristband and timestamps from IoTs are converted to features fed into a machine learning model, which achieves AUC ≥ 0.997. Unlike prior state-of-the-art approaches, our method can work with COTS devices without requiring any special sensors or hardware modifications.

ACKNOWLEDGMENTS

We would like to thank our shepherd, Dr. Brad Campbell, and the anonymous reviewers for their insightful comments and constructive suggestions that have helped improve the paper. This project was supported by NSF CNS-1815144, NSF CNS-1856380, and NSF CNS-1850278.

REFERENCES

[1] Gildas Avoine, Muhammed Ali Bingöl, Ioana Boureanu, Srdjan čapkun, Gerhard Hancke, Süleyman Kardaş, Chong Hee Kim, Cédric Lauradoux, Benjamin Martin, Jorge Munilla, Alberto Peinado, Kasper Bonne Rasmussen, Dave Singelée, Aslan Tchamkerten, Rolando Trujillo-Rasua, and Serge Vaudenay. 2018. Security of Distance-Bounding: A Survey. *Comput. Surveys* 51, 5 (2018).

[2] Kemal Bicakci and Bulent Tavli. 2009. Denial-of-Service attacks and countermeasures in IEEE 802.11 wireless networks. *Computer Standards & Interfaces* 31, 5 (2009).

[3] John Brooke. 1996. SUS: A quick and dirty usability scale. In *Usability Evaluation in Industry*. Taylor & Francis, Chapter 21.

[4] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. 2016. Hidden Voice Commands. In *25th USENIX Security Symposium (USENIX Security)*.

[5] Jason Cipriani. 2018. 13 new things you can do with your Android Wear smartwatch. https://www.cnet.com/how-to/tips-and-tricks-for-android-wear-2-0/. (2018). Accessed: 2019-03-04.

[6] Mark D. Corner and Brian D. Noble. 2002. Zero-interaction Authentication. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking (MobiCom)*.

[7] Cas Cremers, Kasper B Rasmussen, Benedikt Schmidt, and Srdjan Capkun. 2012. Distance hijacking attacks on distance bounding protocols. In *IEEE Symposium on Security and Privacy (Oakland)*.

[8] Alexander De Luca, Alina Hang, Frederik Brudy, Christian Lindner, and Heinrich Hussmann. 2012. Touch me once and I know it's you!: Implicit Authentication based on Touch Screen Patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

[9] Michael Esterman, Benjamin J Tamber-Rosenau, Yu-Chin Chiu, and Steven Yantis. 2010. Avoiding non-independence in fMRI data analysis: leave one subject out. *Neuroimage* 50, 2 (2010).

[10] Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. 2005. Working set selection using second order information for training support vector machines. *Journal of machine learning research* 6, Dec (2005).

[11] Huan Feng, Kassem Fawaz, and Kang G. Shin. 2017. Continuous Authentication for Voice Assistants. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking (MobiCom)*.

[12] NFC Forum. 2018. NFC and Contactless Technologies. https://nfc-forum.org/what-is-nfc/about-the-technology/. (2018).

[13] Lishoy Francis, Gerhard P Hancke, Keith Mayes, and Konstantinos Markantonakis. 2011. Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones. *IACR Cryptology ePrint Archive* 2011 (2011).

[14] Mario Frank, Ralf Biedert, Eugene Ma, Ivan Martinovic, and Dawn Song. 2012. Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication. *IEEE Transactions on Information Forensics and Security* 8, 1 (2012).

[15] Gartner. 2014. The Future Smart Home: 500 Smart Objects Will Enable New Business Opportunities. https://www.gartner.com/en/documents/2793317. (2014). Accessed: 2019-03-02.

[16] Nirnimesh Ghose, Loukas Lazos, and Ming Li. 2018. SFIRE: Secret-Free-in-band Trust Establishment for COTS Wireless Devices. In *IEEE International Conference on Computer Communications (INFOCOM)*.

[17] T. P. Ghuntla, H. B. Mehta, P. A. Gokhale, and C. J. Shah. 2012. A Comparative Study of Visual Reaction Time in Basketball Players and Healthy Controls. *National Journal of Integrated Research in Medicine* 3, 1 (2012).

[18] Jun Han, Shijia Pan, Manal Kumar Sinha, Hae Young Noh, Pei Zhang, and Patrick Tague. 2017. Sensetribute: Smart Home Occupant Identification via Fusion Across On-Object Sensing Devices. In *Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments (BuildSys)*.

[19] Gerhard Hancke. 2005. *A practical relay attack on ISO 14443 proximity cards*. Technical Report.

[20] Cynthia Harvey. 2016. 75 Top IoT Devices. https://www.datamation.com/mobile-wireless/75-top-iot-devices-1.html. (2016). Accessed: 2019-02-22.

[21] Weijia He, Maximilian Golla, Roshni Padhi, Jordan Ofek, Markus Dürmuth, Earlence Fernandes, and Blase Ur. 2018. Rethinking Access Control and Authentication for the Home Internet of Things (IoT). In *27th USENIX Security Symposium (USENIX Security)*.

[22] Chong Hee Kim and Gildas Avoine. 2011. RFID Distance Bounding Protocols with Mixed Challenges. *IEEE Transactions on Wireless Communications* 10, 5 (2011).

[23] Software Testing Help. 2019. 18 Most Popular IoT Devices in 2019. https://www.softwaretestinghelp.com/iot-devices/. (2019). Accessed: 2019-03-18.

[24] Mark R. Hodges and Martha E. Pollack. 2007. An 'Object-Use Fingerprint': The Use of Electronic Sensors for Human Identification. In *UbiComp 2007: Ubiquitous Computing*.

[25] Tâm Huynh and Bernt Schiele. 2006. Towards Less Supervision in Activity Recognition from Wearable Sensors. In *IEEE International Symposium on Wearable Computers (ISWC)*.

[26] B. Iglewicz and D.C. Hoaglin. 1993. How to detect and handle outliers. *Milwaukee, WI.: American Society for Quality* (1993).

[27] Apple Inc. 2019. Apple Watch. https://www.apple.com/watch/. (2019). Accessed: 2019-03-04.

[28] Apple Inc. 2019. How to unlock your Mac with your Apple Watch. https://support.apple.com/en-us/HT206995. (2019). Accessed: 2019-02-09.

[29] Motiv Inc. 2019. Motiv Ring. https://mymotiv.com/. (2019). Accessed: 2019-03-10.

[30] Aditya Jain, Ramta Bansal, Avnish Kumar, and K. D. Singh. 2015. A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students. *International Journal of Applied & Basic Medical Research* 5, 2 (2015).

[31] Nikolaos Karapanos, Claudio Marforio, Claudio Soriente, and Srdjan Capkun. 2015. Sound-Proof: Usable Two-Factor Authentication Based on Ambient Sound. In *24th USENIX Security Symposium (USENIX Security)*.

[32] Nima Karimian, Zimu Guo, Mark Tehranipoor, and Domenic Forte. 2017. Human recognition from photoplethysmography (ppg) based on non-fiducial features. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[33] John Krumm and Ken Hinckley. 2004. The NearMe Wireless Proximity Server. In *UbiComp 2004: Ubiquitous Computing*.

[34] Arun Kumar, Nitesh Saxena, Gene Tsudik, and Ersin Uzun. 2009. A comparative study of secure device pairing methods. *Pervasive and Mobile Computing* 5, 6 (2009).

[35] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. 2011. Activity Recognition using Cell Phone Accelerometers. *ACM SIGKDD Explorations Newsletter* 12, 2 (2011).

[36] Lingjun Li, Xinxin Zhao, and Guoliang Xue. 2013. Unobservable Re-authentication for Smartphones. In *Proceedings of the 20th Annual Network Distributed System Security Symposium (NDSS)*.

[37] Xiaopeng Li, Sharaf Malebary, Xianshan Qu, Xiaoyu Ji, Yushi Cheng, and Wenyuan Xu. 2018. iCare: Automatic and User-friendly Child Identification on Smartphones. In *Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications (HotMobile)*.

[38] Xiaohui Liang, Tianlong Yun, Ronald Peterson, and David Kotz. 2017. LightTouch: Securely connecting wearables to ambient displays with

user intent. In *IEEE International Conference on Computer Communications (INFOCOM)*.

[39] Feng Lin, Chen Song, Yan Zhuang, Wenyao Xu, Changzhi Li, and Kui Ren. 2017. Cardiac Scan: A Non-contact and Continuous Heart-based User Authentication System. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking (MobiCom)*.

[40] Atama Designs Ltd. [n. d.]. Sesame 2. https://atama.io/. ([n. d.]). Accessed: 2019-01-31.

[41] Shrirang Mare, Andrés Molina Markham, Cory Cornelius, Ronald Peterson, and David Kotz. 2014. Zebra: Zero-effort bilateral recurring authentication. In *IEEE Symposium on Security and Privacy (Oakland)*.

[42] Ivan Martinovic, Kasper Rasmussen, Marc Roeschlin, and Gene Tsudik. 2014. Authentication Using Pulse-Response Biometrics. In *Proceedings of the 21th Annual Network Distributed System Security Symposium (NDSS)*.

[43] Rene Mayrhofer and Hans Gellersen. 2009. Shake well before use: Intuitive and secure pairing of mobile devices. *IEEE Transactions on Mobile Computing* 8, 6 (2009).

[44] Daniel V. McGehee, Elizabeth N. Mazzae, and G. H. Scott Baldwin. 2000. Driver Reaction Time in Crash Avoidance Research: Validation of a Driving Simulator Study on a Test Track. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*.

[45] Yuxin Meng, Duncan S Wong, Roman Schlegel, et al. 2012. Touch gestures based biometric authentication scheme for touchscreen mobile phones. In *International Conference on Information Security and Cryptology*.

[46] Fabian Monrose and Aviel Rubin. 1997. Authentication via Keystroke Dynamics. In *Proceedings of the 4th ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[47] Dibya Mukhopadhyay, Maliheh Shirvanian, and Nitesh Saxena. 2015. All your voices are belong to us: Stealing voices to fool humans and machines. In *European Symposium on Research in Computer Security*.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, and et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011).

[49] Konstantinos Pelechrinis, Marios Iliofotou, and Srikanth V Krishnamurthy. 2010. Denial of Service Attacks in Wireless Networks: The Case of Jammers. *IEEE Communications Surveys & Tutorials* 13, 2 (2010).

[50] Postscapes. 2019. IoT Devices & Products. https://www.postscapes.com/internet-of-things-award/winners/. (2019). Accessed: 2019-02-22.

[51] Juhi Ranjan and Kamin Whitehouse. 2015. Object Hallmarks: Identifying Object Users Using Wearable Wrist Sensors. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*.

[52] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. 2005. Activity Recognition from Accelerometer Data. In *Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence*.

[53] Kimberly Redmond, Lannan Luo, and Qiang Zeng. 2019. A cross-architecture instruction embedding model for natural language processing-inspired binary code analysis. *The NDSS Workshop on Binary Analysis Research (BAR)*.

[54] Napa Sae-Bae, Kowsar Ahmed, Katherine Isbister, and Nasir Memon. 2012. Biometric-rich gestures: a novel approach to authentication on multi-touch devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

[55] Hataichanok Saevanee and Pattarasinee Bhatarakosol. 2008. User authentication using combination of behavioral biometrics over the touchpad acting like touch screen of mobile device. In *International Conference on Computer and Electrical Engineering*.

[56] Sairul I Safie, John J Soraghan, and Lykourgos Petropoulakis. 2011. Electrocardiogram (ECG) biometric authentication using pulse active ratio (PAR). *IEEE Transactions on Information Forensics and Security* 6, 4 (2011).

[57] Samsung. [n. d.]. SmartThings. https://www.samsung.com/global/galaxy/apps/smartthings/. ([n. d.]). Accessed: 2019-01-15.

[58] Abhijit Sarkar, A Lynn Abbott, and Zachary Doerzaph. 2016. Biometric authentication using photoplethysmography signals. In *IEEE 8th International Conference on Biometrics Theory, Applications and Systems*.

[59] Mohamed Shahin, Ahmed Badawi, and Mohamed Kamel. 2007. Biometric authentication using fast correlation of near infrared hand vein patterns. *International Journal of Biological and Medical Sciences* 2, 3 (2007).

[60] Sheng Shen, Mahanth Gowda, and Romit Roy Choudhury. 2018. Closing the Gaps in Inertial Motion Tracking. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom)*.

[61] Sheng Shen, He Wang, and Romit Roy Choudhury. 2016. I Am a Smartwatch and I Can Track My User's Arm. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*.

[62] Babins Shrestha, Maliheh Shirvanian, Prakash Shrestha, and Nitesh Saxena. 2016. The Sounds of the Phones: Dangers of Zero-Effort Second Factor Login Based on Ambient Audio. In *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[63] Santosh Singh. 2019. Top 20 IoT Platforms in 2018. https://internetofthingswiki.com/top-20-iot-platforms/634/. (2019). Accessed: 2019-03-17.

[64] Zdeňka Sitová, Jaroslav Šeděnka, Qing Yang, Ge Peng, Gang Zhou, Paolo Gasti, and Kiran S Balagani. 2016. HMOG: New behavioral biometric features for continuous authentication of smartphone users. *IEEE Transactions on Information Forensics and Security* 11, 5 (2016).

[65] Michael Stanley and Jongmin Lee. 2018. *Sensor Analysis for the Internet of Things*. Morgan & Claypool Publishers.

[66] Jing Tian, Chengzhang Qu, Wenyuan Xu, and Song Wang. 2013. KinWrite: Handwriting-Based Authentication Using Kinect. In *Proceedings of the 20th Network and Distributed System Security Symposium (NDSS)*.

[67] José Vila and Ricardo J. Rodríguez. 2015. Practical Experiences on NFC Relay Attacks with Android. In *Radio Frequency Identification*.

[68] Wei Wang, Lin Yang, and Qian Zhang. 2016. Touch-and-guard: secure pairing through hand resonance. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*.

[69] Sheng Wei, Jong Hoon Ahnn, and Miodrag Potkonjak. 2013. Energy Attacks and Defense Techniques for Wireless Systems. In *Proceedings of the 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks*.

[70] Zhenyu Yan, Qun Song, Rui Tan, Yang Li, and Adams Wai Kin Kong. 2019. Towards Touch-to-Access Device Authentication Using Induced Body Electric Potentials. *arXiv preprint arXiv:1902.07057* (2019).

[71] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, XiaoFeng Wang, and Carl A. Gunter. 2018. CommanderSong: A Systematic Approach for Practical Adversarial Voice Recognition. In *27th USENIX Security Symposium (USENIX Security)*.

[72] Sangki Yun, Yi-Chao Chen, and Lili Qiu. 2015. Turning a Mobile Device into a Mouse in the Air. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*.

[73] Qiang Zeng, Jianhai Su, Chenglong Fu, Golam Kayas, Lannan Luo, Xiaojiang Du, Chiu C. Tan, and Jie Wu. 2019. A multiversion programming inspired approach to detecting audio adversarial examples. In

Proceedings of the 49th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN).

[74] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. 2017. Dolphinattack: Inaudible voice commands. In *Proceedings of the 24th ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[75] Jiansong Zhang, Zeyu Wang, Zhice Yang, and Qian Zhang. 2017. Proximity Based IoT Device Authentication. In *IEEE International Conference on Computer Communications (INFOCOM)*.

[76] Nan Zheng, Aaron Paloski, and Haining Wang. 2011. An Efficient User Verification System via Mouse Movements. In *Proceedings of the 18th ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[77] Pengfei Zhou, Mo Li, and Guobin Shen. 2014. Use It Free: Instantly Knowing Your Phone Attitude. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking (MobiCom)*.

[78] Fei Zuo, Xiaopeng Li, Patrick Young, Lannan Luo, Qiang Zeng, and Zhexin Zhang. 2019. Neural Machine Translation Inspired Binary Code Similarity Comparison beyond Function Pairs. In *Proceedings of the 26th Network and Distributed System Security Symposium (NDSS)*.