Secure UHF RFID Authentication With Smart Devices

Ang Li, *Student Member, IEEE*, Jiawei Li, *Graduate Student Member, IEEE*, Yan Zhang, *Member, IEEE*, Dianqi Han, *Member, IEEE*, Tao Li, *Member, IEEE*, and Yanchao Zhang, *Fellow, IEEE*

***Abstract*— Commodity ultra-high-frequency (UHF) RFID authentication systems only provide weak user authentication, as RFID tags can be easily stolen, lost, or cloned by attackers. This paper presents the design and evaluation of SmartRFID, a novel UHF RFID authentication system to promote commodity crypto-less UHF RFID tags for security-sensitive applications. SmartRFID explores extremely popular smart devices and requires a legitimate user to enroll his smart device along with his RFID tag. Besides authenticating the RFID tag as usual, SmartRFID verifies whether the user simultaneously possesses the associated smart device with both feature-based machine learning and deep learning techniques. The user is considered authentic if and only if passing the dual verifications. Comprehensive user experiments on commodity smartwatches and RFID devices confirmed the high security and usability of SmartRFID. In particular, SmartRFID achieves a true acceptance rate of above 97.5% and a false acceptance rate of less than 0.7% based on deep learning. In addition, SmartRFID can achieve an average authentication latency of less than $2.21\,\mathrm{s}$, which is comparable to inputting a PIN on a door keypad or smartphone.**

***Index Terms*— UHF RFID, smart device, smartwatch, authentication, access control.**

I. Introduction

RADIO Frequency Identification (RFID) technology has been widely used in personnel/object identification and access control. A typical RFID system consists of RFID readers, a backend server, and RFID tags with each assigned to a unique user. An RFID reader keeps sending radio signals to detect and query any RFID tag within its transmission range. Once receiving an RFID reader's query, an RFID tag responds with the stored authentication information which can be a simple Electronic Product Code (EPC) or a cryptographic message. The RFID reader forwards the response to the backend server for final verification. A valid response corroborates the legitimacy of the RFID tag carrier who is then permitted access to the protected area or system such as a building/room, computer system, or parking lot.

Passive ultra-high frequency (UHF) RFID is dominating the RFID market [1]. A UHF RFID system operates the frequency band between $860\,\mathrm{MHz}$ and $960\,\mathrm{MHz}$. UHF RFID readers transmit modulated RF signals to interrogate tags. Passive UHF RFID tags are batteryless and communicate with RFID readers by backscattering their signals. Specifically, the passive UHF tag receives the energy via signals propagated from the reader antenna. Once the signals reach the tag, the energy travels through the tag antenna to activate the chip or integrated circuit (IC). The remaining energy is modulated with the chip data and sent back via the tag antenna to the reader antenna in the form of electromagnetic waves. The response is then detected and decoded at the reader. In contrast to low-frequency and high-frequency tags, passive UHF tags have a much longer reading distance up to $12\,\mathrm{m}$ and are also much cheaper. Passive UHF tags are thus favorable choices in many contexts such as in hospitals for the nursing staff pushing a stretcher to transport a patient across different areas, in a logistics center for employees moving pallets, in office buildings for people with a physical impairment, and in large indoor environments to track the whereabouts of users.

Using passive UHF RFID in security-sensitive applications has been largely hindered by the lack of cryptographic support on most commodity products. As far as we know, NXP's UCODE DNA RAIN RFID tags [2] are the only products that support cryptographic authentication checks. These tags cost about \$1 each [3] and are much more expensive than regular crypto-less tags typically costing $5¢$ to $15¢$. Therefore, **crypto-less passive UHF RFID tags** are still the mainstream products in the RFID market. We focus on such tags and may omit the term "crypto-less passive UHF" whenever no confusion may arise hereafter. The security of passive UHF RFID systems has been receiving growing attention from the academia and industry. For instance, some recent publications in IEEE TWC [4], [5], [6] propose elegant solutions to different RFID security issues. As another example, given that the RSSI and phase information of backscattered RFID signals are available on commodity RFID readers [7], RFID signals

Manuscript received 27 September 2022; revised 31 October 2022; accepted 24 November 2022. Date of publication 12 December 2022; date of current version 12 July 2023. This work was supported in part by the U.S. National Science Foundation under Grant CNS 2055751 and Grant 1933069. The associate editor coordinating the review of this article and approving it for publication was J. Xie. *(Corresponding author: Yanchao Zhang.)*

Ang Li, Jiawei Li, and Yanchao Zhang are with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: anglee@asu.edu; jwli@asu.edu; yczhang@asu.edu).

Yan Zhang is with the Electrical and Computer Engineering Department, The University of Akron, Akron, OH 44325 USA (e-mail: yzhang1@uakron.edu).

Dianqi Han is with the Computer Science and Engineering Department, The University of Texas at Arlington, Arlington, TX 76019 USA (e-mail: dianqi.han@uta.edu).

Tao Li is with the Computer and Information Technology Department, Indiana University-Purdue University Indianapolis, Indianapolis, IN 46202 USA (e-mail: tli6@iupui.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TWC.2022.3226753.

Digital Object Identifier 10.1109/TWC.2022.3226753

have been explored to recognize human gestures to achieve gesture-based user authentication [8], [9], [10]. This paper also explores the rich information contained in backscattered RFID signals to design more secure RFID authentication systems.

Commodity RFID authentication systems are particularly vulnerable to **adversarial RFID tags**, each of which refers to a legitimate RFID tag or its clone controlled by the adversary. In particular, RFID-based ID badges can be easily lost/stolen and possessed by the adversary. In addition, most commodity RFID tags do not support cryptographic operations. A capable adversary can overhear the unencrypted reader-tag communications and then exploit the sniffed tag information to make clones. The adversary can use an adversarial RFID tag to impersonate the legitimate user and gain access to protected physical or electronic resources. As a case study, we easily cloned the UHF RFID tags in the parking decals and used them to successfully enter the gated garage of our institution.

Given the growing ubiquity of human-carried smart devices such as smartphones and smartwatches, it is a natural question whether smart devices can be explored to protect RFID authentication systems from adversarial RFID tags. One plausible solution is to directly use a commercial two-factor authentication system such as Duo [11]. In particular, a Duo-like technique would involve the user holding the smart device, unlocking the screen, finding the Duo app, and manually approving a login notification that is pushed by the backend RFID server after validating the tag information. Involving non-trivial user effort, this approach has poor usability especially for people with fat fingers or weak vision and greatly diminishes the convenience of UHF RFID authentication. Additionally, it is shown in [12] that the average authentication time for the Duo-like approach is about $16.1\,\mathrm{s}$. Furthermore, the Duo-like approach is vulnerable to synchronized jamming attacks [13]. Another plausible solution is to verify the physical proximity of the user's enrolled smart device and the RFID reader via a short-range wireless channel such as NFC or Bluetooth. This method, however, requires additional NFC or Bluetooth capabilities that are not available on most commodity RFID readers. Additionally, it requires pairing the smart device with the RFID reader, which may severely hinder its usability especially if the legitimate user needs to access many protected areas (e.g., a nurse in a big hospital) with each having a distinct RFID reader. Moreover, this method is vulnerable to known wireless man-in-the-middle and relaying attacks between the smart device and RFID reader [14], [15], [16], [17], which make it difficult to verify their physical proximity. It is also worth noting that such attacks render it infeasible to use smart devices alone for proximity-based access control to protected electrical and physical resources.

In this paper, we propose **SmartRFID**, a smart device-aided RFID authentication system to promote commodity RFID tags for otherwise inapplicable security-sensitive applications. Each legitimate SmartRFID user registers his/her RFID tag and also smart device during system enrollment. In each subsequent authentication instance, SmartRFID verifies two factors for any RFID user: (1) the tag Electronic Product Code (EPC) as in a traditional RFID authentication system and (2) the physical proximity of the paired RFID tag and smart device. The user is considered authentic if and only if both authentication factors can be validated. Although SmartRFID explores user-carried smart devices as well, it involves minimal user effort, is much less time-consuming and invulnerable to synchronized jamming attacks in contrast to the Duo-like solution; it is not subject to wireless MiM attackers either.

SmartRFID checks the coexistence of the paired RFID tag and smart device by verifying the correlation between the smart device's inertial accelerometer data and the backscattered RFID signals. Each legitimate user in SmartRFID is required to tap or shake the RFID tag multiple times for a short duration (say, $2\,\mathrm{s}$) with the same hand holding the paired smart device. The intentional hand movement can be detected by the smart device's inertial accelerometer. It can also induce phase changes in the backscattered RFID signals that are readily available on commodity RFID readers. The backend server collects and compares the acceleration and phase data. If a strong acceleration-phase correlation could be found, the backend server trusts that the user simultaneously holds the paired tag and smart device and thus considers him/her authentic.

There are two main obstacles to implement SmartRFID.

- **First, a capable attacker can perform synchronized hand movement with the victim user who may often unknowingly wave his/her smart device**. This synchronization attack is feasible if the attacker can visually observe the victim user either in person (e.g., as a malicious co-worker) or through a live video feed from a spy camera. The attacker can then pass authentication due to the strong acceleration-phase correlation. SmartRFID thwarts this attack by requiring that each user signal his/her authentication intention with a sequence of taps or shakes on the RFID tag. In contrast to totally random hand movement, such intentional hand gestures are much less unlikely to accidentally perform by the legitimate user. Additionally, our method has high usability because both taps and shakes are very easy to perform. We develop novel machine learning-driven algorithms to recognize each individual tap or shake gesture from noisy acceleration and RFID-phase data, respectively.
- **Second, the acceleration and phase data are of different types and cannot be directly compared**. We tackle this challenge by first extracting two time series of data from the acceleration and phase data, respectively. Then we explore and compare two methods to verify their correlation. The first method explores conventional machine learning techniques—including Support Vector Machine (SVM), Random Forest (RF), and k-nearest neighbors (k-NN)—to build a feature-based correlation detection module. The second method uses a two-branch deep neural network with two CNN (Convolutional Neural Network) layers and one LSTM (Long Short-Term Memory) layer in each branch.

We prototype SmartRFID on commodity smartwatches and UHF RFID systems and evaluate its security and usability through detailed user experiments involving 20 volunteers. The true and false acceptance rates with the feature-based correlation detection module are 97.3% and 3.5%, respectively, and those with the deep correlation network are 97.56% and 0.66%, respectively. Additionally, the feature-based correlation

detection module achieves an Area Under The Curve (AUC) score as high as 0.995 and an equal error rate (EER) as low as 0.018, while the deep correlation network module achieves an AUC score as high as 0.998 and EER as low as 0.017. We also show that SmartRFID can achieve an average authentication latency of less than 2.21 s, which is comparable to inputting a PIN on a door keypad or smartphone. Finally, a user survey confirms that SmartRFID is very easy and convenient to use.

SmartRFID can significantly boost the security of commodity UHF RFID authentication systems with high usability and deployability. In terms of **security**, in addition to cloning or stealing the legitimate RFID tag, the attacker needs to steal the associated smart device which is carried by the user (e.g., on the wrist), is normally protected by a login password, and receives stronger user attention/protection. In addition, how the legitimate user interacts with his/her smart device and RFID tag can be a self-chosen gesture password difficult for the attacker to guess or emulate. With regard to **usability and deployability**, smart devices required by SmartRFID have been prevalent in daily life and most target application contexts of UHF RFID systems. For example, many organizations (e.g., the authors' affiliation) enforce Duo-like mobile two-factor authentication and require each affiliate to have a smartphone-like smart device. So SmartRFID does not involve any extra smart device other than what the user already has in most cases. Furthermore, SmartRFID users only need to perform simple tap or shake gestures with their smart devices, and the resulting acceleration data are automatically uploaded to the system server through a software module on the smart device.

The rest of this paper is organized as follows. Section II outlines the system model and workflow. Section III presents the adversary model. Section IV details the SmartRFID design. Section V presents the evaluation results. Section VI reviews the related work. Section VII concludes this paper.

II. System Model and Overview

SmartRFID aims to add a second authentication factor to a commodity UHF RFID authentication system which consists of a server, RFID readers, and passive crypto-less UHF RFID tags. For convenience only, we use Bob as an exemplary legitimate user to outline the system model and operations. Bob has an ID badge with an embedded RFID tag and wears it on a lanyard or clips it to his clothes. We do not differentiate the ID badge with the RFID tag hereafter. Bob also has a smart device (e.g., a smartphone, smartwatch, or fitness tracker) with a standard accelerometer. He installs a SmartRFID app on the smart device and also creates his app username and password for the server to recognize him. Bob enrolls both his smart device and RFID tag with the server.

We assume that the server can always communicate with Bob's smart device either through a direct WiFi or cellular channel or with the RFID reader as a relay. All the communication messages between Bob's smart device and the server are actually handled by the SmartRFID app. So we assume a secure end-to-end TLS-like channel between the SmartRFID app and server. When the server receives authenticated messages from a SmartRFID app instance logged into under Bob's username and password, it trusts that the messages are indeed from Bob's smart device. This assumption is also required by a Duo-like two-factor authentication system [11].

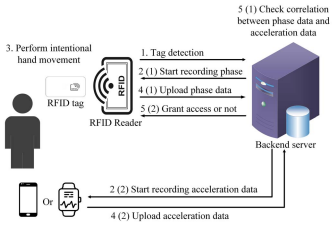


Fig. 1. Workflow of SmartRFID.

Fig. 1 shows the SmartRFID system workflow. Each step are detailed in late sections.

1. The RFID reader acquires the tag information with a traditional RFID interrogation protocol such as EPC Gen 2 and sends it to the backend server for verification.
2. After validating the tag information, the server notifies the RFID reader to report the phase information of backscattered signals and Bob's smart device to record its acceleration data.
3. Bob taps or shakes his tag randomly or according to a predetermined pattern within a short duration with the same hand holding or wearing the paired smart device.
4. The RFID reader and Bob's smart device submit phase data and acceleration data to the server, respectively.
5. The server checks the correlation between received phase and acceleration data with either of two approaches or both. The first approach uses a feature-based correlation detector based on traditional machine learning techniques, and the second uses a deep correlation network built upon deep learning techniques. If the server finds a strong acceleration-phase data correlation, it considers the RFID user indeed Bob and grants him access to protected physical or electronic resources (e.g., a building, server room, computer, or vehicle) Bob is entitled to use. Otherwise, the user is denied access.

SmartRFID is intended to be highly usable by minimizing the additional effort required of RFID users. In particular, Bob does not need to perform any action other than taping or shaking his RFID tag. All the communications between the server and the SmartRFID app are automatically conducted without involving Bob's effort. For example, Bob does not need to manually initiate the authentication session or respond to the server's acceleration-data collection request via the app. In addition, Bob need not know when the server starts to collect phase and accelerometer data.

III. Adversary Model

Rather than providing very strong security, SmartRFID aims to secure commodity RFID authentication systems which are otherwise vulnerable to tag spoofing and cloning. We thus consider a reasonable adversary, denoted by $\mathcal{A}$, that possesses an authentic copy of Bob's RFID tag, be it lost, stolen, or cloned. We assume that Bob's smart device is protected by a password as in common scenarios. Consider a smartwatch as

an example. Each time Bob puts on his smartwatch, he must input the password to unlock it and activate all the apps therein—including the SmartRFID app—which remain active until the smartwatch leaves Bob's wrist and are then automatically locked. Therefore, $\mathcal{A}$ still cannot bypass SmartRFID by stealing both Bob's smartwatch and RFID tag as long as the smartwatch's password mechanism is secure.

$\mathcal{A}$ is assumed to know how SmartRFID works and uses the RFID tag to initiate an authentication session. $\mathcal{A}$ aims to impersonate Bob for gaining illegitimate access to a protected area or system. Since the tag information (e.g., EPC) is authentic and verifiable, the backend server proceeds to pull the accelerometer data from Bob's smart device and the phase data from the RFID reader $\mathcal{A}$ is interacting with. We assume that $\mathcal{A}$ can observe Bob's hand movement in real time either in person (e.g., as a malicious bystander, shoulder surfer, or coworker) or through a live feed from a spy camera, so $\mathcal{A}$ can perform synchronized hand movement on the RFID tag. If the server verifies a strong acceleration-phase data correlation to mistake $\mathcal{A}$ for Bob, $\mathcal{A}$ succeeds.

We have two additional assumptions. First, the server cannot use secure localization techniques to tell whether Bob's smart device, the RFID tag under his name and used by $\mathcal{A}$, and the RFID reader $\mathcal{A}$ attempts to cheat are at the same location. The reason is that secure localization techniques often rely on many assumptions which may not hold in practice. Second, we do not consider denial-of-service attacks on SmartRFID, in which $\mathcal{A}$ seeks to induce wrong phase measurements and thus authentication failures by signal interference.

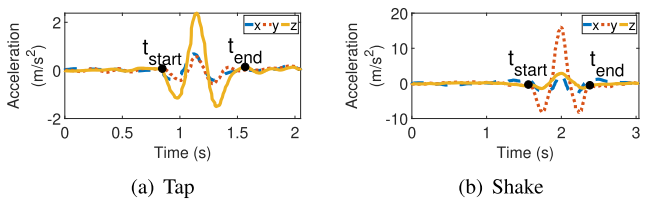
IV. SmartRFID System Design

In this section, we first describe two permissible hand gestures in SmartRFID—taps and shakes—which serve as the basic line of defense. Then we overview the SmartRFID modules. Finally, we detail the design of each module.

A. Intentional Gestures in SmartRFID

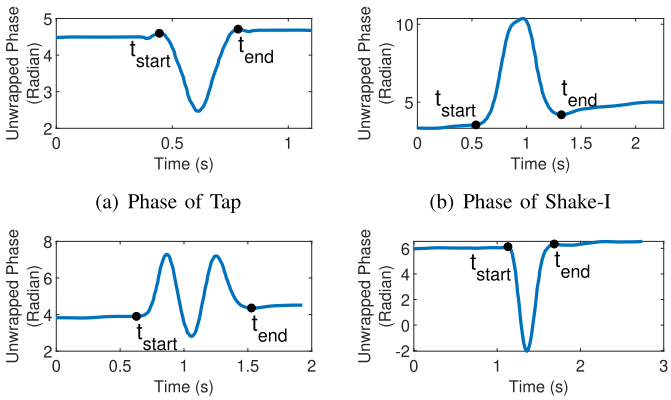
Passive UHF RFID tags communicate with the RFID reader by backscattering the reader's signals. The backscattered signal's phase is available on commodity RFID readers such as Impinj R420 [7]. According to [18], the phase can be expressed as $\phi = (\frac{4\pi df}{c} + \phi_{\text{reader}} + \phi_{\text{card}}) \mod 2\pi$, where $2d$ is the round-trip propagation distance between the reader and card, f is the signal frequency, c is the speed of light, ϕ_{reader} denotes the phase rotation due to the reader's transmit and receive circuits, and ϕ_{card} represents the phase rotation caused by the RFID card's reflection characteristics. Tapping or shaking the RFID card can change its circuit impedance and also signal propagation, leading to some additional phase rotation which is explored in SmartRFID.

In SmartRFID, each legitimate user signals his authentication intention by performing specific hand gestures. Assume that Bob touches his RFID tag with the same hand holding the associated smart device. Such physical interactions can induce both phase and acceleration data changes that can be strongly correlated. A common user's hand/arm movement is mostly unintentional and slow with a repeatable pattern (e.g., walking or running). If SmartRFID allows random hand movement during an authentication session, the adversary $\mathcal{A}$ would have abundant opportunities to visually observe Bob and then perform synchronized hand movement with the cloned or stolen RFID tag to successfully impersonate Bob with a higher chance. So SmartRFID requires each legitimate user to perform intentional hand movement comprising either shakes or taps to explicitly indicate his authentication intention.



(a) Tap (b) Shake

Fig. 2. Acceleration of intentional gestures.



(a) Phase of Tap (b) Phase of Shake-I

(c) Phase of Shake-II (d) Phase of Shake-III

Fig. 3. Phase of intentional gestures.

- *Tap*: Use the hand holding or wearing the associated smart device to place a finger a few centimeters (e.g., 2 cm) above the RFID tag, then quickly touch the tag, and lift the finger.
- *Shake*: Quickly move the RFID tag up and down with the hand holding or wearing the associated smart device.

These two hand gestures both have been widely adopted in various applications on touchscreen devices and virtual reality equipment. Therefore, RFID users can perform them effortlessly. Fig. 2 and Fig. 3 show the processed 3-axis acceleration data and unwrapped phase data of each intentional gesture naturally performed by a volunteer. We use $[t_{\text{start}}, t_{\text{end}}]$ to represent a tap or shake event, where t_{start} and t_{end} denote the time that the user starts and finishes a tap or shake gesture. We can observe that each gesture can induce significant phase and acceleration changes. Moreover, the two gestures lead to distinct phase and acceleration data patterns, so they can be reliably identified from highly noisy phase and accelerometer data. In addition, we can see that the phase change induced by shaking a tag exhibits three possible patterns relating to how the tag is shaken, as shown in Fig. 3. These observations all drive our system design illustrated later.

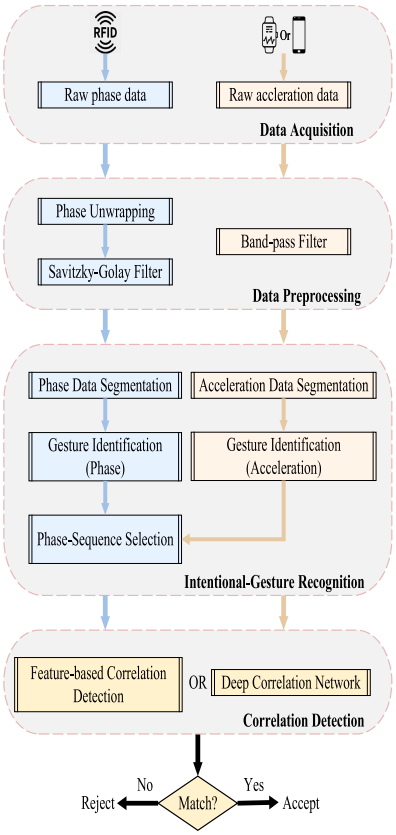


Fig. 4. SmartRFID system modules.

B. SmartRFID System Modules

SmartRFID comprises four main modules shown in Fig. 4. In the Data Acquisition module, the server instructs the RFID reader and the enrolled smart device to record and submit phase and acceleration data, respectively. In the Data Preprocessing module, the server uses various techniques to process the noisy raw phase and acceleration data. In the Intentional-Gesture Recognition module, the server extracts and recognizes tap and shake gestures from the processed phase and acceleration data, respectively, to determine whether a user performs intentional hand movement. In particular, it extracts a timestamp vector that records the start and end time of each recognized tap or shake gesture, as well as a fine-grained data vector that contains all the processed phase/acceleration data associated with intentional hand movement. In the final Correlation Detection module, the system explores two different approaches to check whether there is a strong phase-acceleration data correlation. The feature-based correlation detection submodule examines the correlation between the phase and acceleration timestamp vectors with feature-based machine learning techniques. The deep correlation network checks if the two fine-grained data vectors have a strong correlation with deep learning techniques. SmartRFID declares the user (in)authentic based on either module or both according to the system preference.

C. Data Acquisition

An authentication session starts when the reader receives a tag response to its query. In particular, the reader keeps sending queries to detect nearby tags. When the user (legitimate or not) approaches the reader, he performs intentional hand movement per the SmartRFID requirement until passing authentication or failing after a long time. The tag automatically answers the reader's query by backscattering its stored authentication information (e.g., the unique EPC). The reader-tag communications follow a standard EPC Gen2 RFID protocol.

After receiving the tag response via the reader, the server searches its database. If a matching user (i.e., Bob in our example) is found, the server sends a "Start Sensing" command to both the reader and Bob's enrolled smart device. The overall latency from the RFID tag responding until the "Start Sensing" command reaches the reader and smart device is usually very short and well below 0.5 s per the EPC Gen2 standard and our experiments. So it is safe to assume that Bob just starts performing hand interactions with his tag. Once receiving "Start Sensing", the RFID reader and smart device begin to record and submit the phase and acceleration data, respectively. The server issues a "Stop Sensing" command to the reader and smart device after authenticating the user or failing to do so after a sufficiently long time (say, $5\,\mathrm{s}$).

D. Data Preprocessing

After receiving raw phase and acceleration data, the server feeds them into the Data Preprocessing module. Since the data are of different types, they are separately handled.

As shown in Fig. 4, the raw phase data go through phase unwrapping and Savitzky-Golay filtering in sequence. In particular, the signal phases reported by the reader are wrapped around $[0, 2\pi]$ and thus cannot truly reflect how they change over time. So we correct each wrapped phase angle (measured in radians) by adding or subtracting 2π if absolute jumps between consecutive phase values are greater than or equal to π. Then we use a Savitzky-Golay smoothing filter [19] to smooth the unwrapped phase data and also remove random noise. This filter performs a local polynomial regression in a subset of adjacent points based on least squares fitting. It can preserve some important distribution features such as relative maxima, making it more suitable than other filtering methods for our case.

For the acceleration data, we adopt a band-pass filter to eliminate noise and interference. The frequency of uncontrolled shakes and burst noise is generally much higher than that of hand movement. Additionally, the frequency of interference caused by gravity is very low. According to [20], $0.5\,\mathrm{Hz}$ is sufficient to eliminate the gravity's impact from raw acceleration data. Moreover, since average human reaction time is about 284 ms [21], an user can perform a tap or shake gesture no more than five times per second. Therefore, we apply a Butterworth filter with cutoff frequencies of $0.5\,\mathrm{Hz}$ and $5\,\mathrm{Hz}$ to all three axes of acceleration data to remove the noise and interference caused by gravity.

E. Intentional-Gesture Recognition

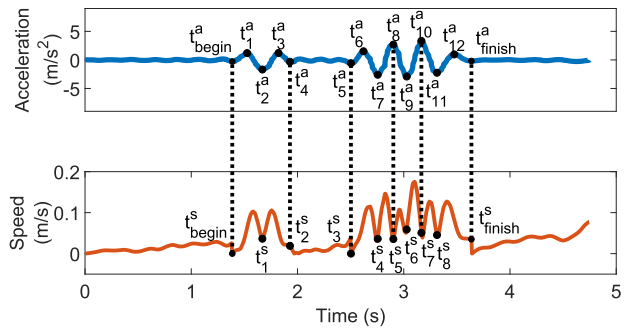
This module aims to recognize individual tap or shake gestures from the processed phase and acceleration data. The following three steps are performed in sequence.

Step 1 (Data Segmentation): This step aims to segment the processed phase and acceleration data into individual tap or shake gestures. The output is a timestamp sequence containing the start and end timestamps of each tap or shake gesture. Due to space constraints and similar observations, we give two intentional hand movement patterns performed within a short period (say, 4 s) as examples: (1) *Tap-4*: tap an RFID tag four times, and (2) *Shake-4*: shake an RFID tag four times.

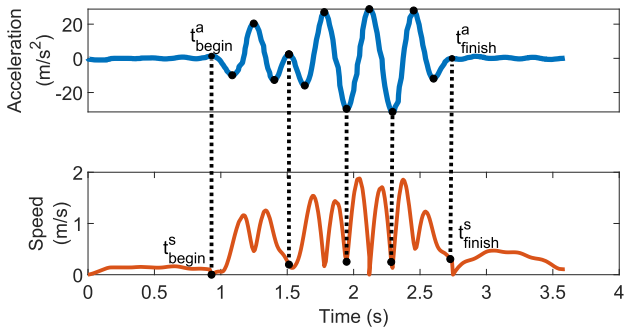
1) Acceleration-Data Segmentation: An individual tap gesture involves the user first putting his finger a few centimeters above the RFID tag, then starting to move until touching the tag, and finally raising his finger back to near the starting point. During this process, the user's hand speed first increases from zero and then decreases to zero when his finger touches the tag; afterwards, it increases and then decreases to zero again when his finger returns to near the starting point. Therefore, the start and end time of a tap gesture is the first and third time when the hand speed is zero. We have similar observations for each shake gesture.

However, we cannot directly leverage the speed data derived from 3-axis acceleration data to identify the start and end time of each tap or shake gesture. The reason is that the directly derived speed data are very noisy and inaccurate in most cases due to accelerometer measurement errors and cumulative errors caused by integration. Although the direct speed data are not quite useful, we can still find some good samples to help us locate "zero" (speed) points. For this purpose, we first estimate the hand speed at each timestamp from 3-axis acceleration data. In particular, we calculate velocity by integrating acceleration along each axis and thus get three velocity vectors $[v_x, v_y, v_z]$. Then we calculate the hand speed as the square root value of the velocity data as $s = \sqrt{{v_x}^2 + {v_y}^2 + {v_z}^2}$. The lower panels in Fig. 5(a) and Fig. 5(b) show the hand-speed changes when a user performs Tap-4 and Shake-4, respectively. Subsequently, we find all "zero" points between the begin and finish time of the intentional hand movement and then remove those irrelevant ones when the hand is static for a sufficiently long time (e.g., between t_2^s and t_3^s). So we can get a set of relevant "zero" points and extract their timestamps. For example, in the lower panel of Fig. 5(a), we have a timestamp vector $[t^s_{\text{begin}}, t^s_1, t^s_2, t^s_3, t^s_4, t^s_5, t^s_5, \ldots, t^s_{\text{finish}}]$. According to our observation, we choose the first and third elements of every three adjacent elements as the start and end time of each tap gesture and thus obtain a sequence of timestamps for Tap-4 as $[t^s_{\text{begin}}, t^s_2, t^s_3, t^s_5, t^s_5, t^s_7, t^s_7, t^s_{\text{finish}}]$. Similarly, we can get a timestamp vector for Shake-4 as shown in Fig. 5(b).

We proceed to identify the corresponding "zero" points. As shown in Fig. 5, the start and end timestamps of each tap or shake gesture in the speed curve correspond to some turning points like t_4^a and t_{10}^a (the upper panel in Fig. 5(a)). According to our preliminary experiments, the first and last tap (or shake) gestures both have four or five turning points, while the two middle ones can both have three, four or five turning points. The examples in Fig. 5 show five turning points for the first gesture, four and three for the two middle ones, and four for the last one. Based on these observations, we design



(a) Tap-4

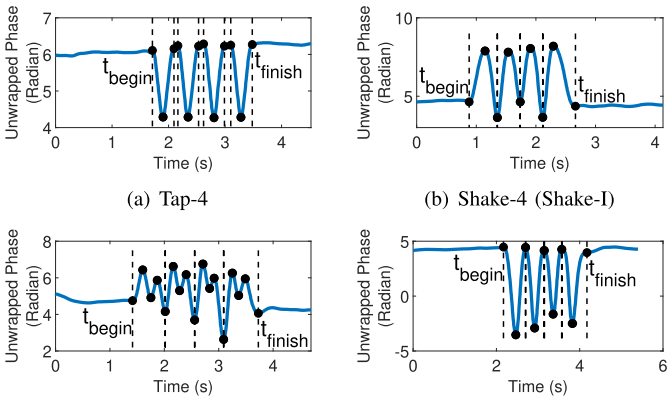


(b) Shake-4

Fig. 5. Processed acceleration and speed data.

an iterative approach to extract a start and end time of each intentional gesture from the processed acceleration data.

We first select the axis that is the most sensitive to intentional hand movement as the main axis for data segmentation. In particular, we compute the standard deviation (STD) of acceleration data along each axis and then select the axis that has the largest STD as the main axis. Next, we try to find the turning points that are related to each tap or shake gesture by using the acceleration data along the main axis. Specifically, we compute the first derivative of the acceleration data and find all turning points which we use to segment the data. Then we compute the acceleration difference between the first and last points for each segment. Afterwards, we find those segments whose acceleration difference is larger than an empirical threshold value τ and use their first and last points as candidate turning points. Subsequently, we iteratively identify the start and end time of each gesture from the candidate set. In particular, since the first gesture of intentional hand movement has at least four turning points, we select the first and fourth ones from the candidate set as its potential start and end points. Then the following iterative process is executed. We check if the acceleration difference of the following segmentat is above a threshold θ. If so, we use timestamps of the two points as the start and end time of the gesture; we also use the first and third turning points of the next three as the potential start and end points of the next gesture. Otherwise, the end timestamp of the following segment is considered the end time of the gesture, and the potential start and end points of the next gesture is set to the first and fourth points of the next four.



(a) Tap-4 (b) Shake-4 (Shake-I)

(c) Shake-4 (Shake-II) (d) Shake-4 (Shake-III)

Fig. 6. Processed phase data of Tap-4 and Shake-4.

Fig. 5 shows an example. We get a sequence of timestamps from processed acceleration data as $T_a = [(t_s^1, t_e^1), (t_s^2, t_e^2), \ldots, (t_s^n, t_e^n)]$, where t_s^i and t_e^i $(1 \leq i \leq n)$ denote the start and end timestamps of the ith tap or shake, respectively. Furthermore, we extract a three-row acceleration data matrix A between t_s^1 and t_e^n to represent intentional hand movement, in which each row corresponds to the processed timestamped acceleration data of one axis.

2) Phase-Data Segmentation: As shown in Fig. 3(a), a tap gesture induces an inverted bell-shaped phase pattern. Additionally, our experiments reveal that a shake gesture may lead to three phase patterns: a bell shape (*shake*-I), two consecutive bell shapes (*shake*-II), and an inverted bell shape (*shake*-III). Figs. 3(b) to 3(d) shows three examples of a Shake-4 event, each comprising *shake*-I, *shake*-II, and *shake*-III patterns only. In practice, a Shake-4 event may induce an arbitrary combination of the three phase patterns. In addition, each bell-shaped or inverted bell-shaped phase pattern has three turning points, as shown in Fig. 6.

Based on these observations, we design the phase-data segmentation algorithm as follows. In the first step, we use similar operations in the previous algorithm to find all turning points and extract the timestamp of the first and third ones as the start and end time of each (inverted) bell-shaped pattern. For the Tap-4 event, we can directly use the start and end time of each inverted bell-shaped pattern as the start and end timestamps of each tap. For the Shake-4 event, it is not straightforward to determine if two adjacent bell-shaped patterns correspond to one shake-II case or two shake-I cases. To solve this issue, we first output a set of timestamp sequence that represents a possible segmentation: $S_p = \{T_{p_1}, T_{p_2}, \ldots, T_{p_s}\}$, where $T_{p_i} = [\hat{t}_s^{i1}, \hat{t}_e^{i1}, \hat{t}_s^{i2}, \hat{t}_e^{i2}, \ldots, \hat{t}_s^{in}, \hat{t}_e^{in}]$. Then we infer the maximum length of S_p. Given m bell-shaped patterns, the number of *shake*-I and *shake*-II lie in $[0, \lfloor \frac{m}{2} \rfloor]$ and $[m - \lfloor \frac{m}{2} \rfloor, m]$, respectively. Given i *shake*-II, the number of *shake*-I is $m-2i$, so the maximum length of S_p is $\sum_{i=0}^{\lfloor \frac{m}{2} \rfloor} \binom{m-i}{i}$. In addition, we can use other information such as the maximum and minimum number of intentional gestures to further shorten the list. The most feasible phase sequence in S_p is to be chosen in Step 3.

TABLE I
LIST OF FEATURES

#	Feature Name	Description
1	Mean	$\mu = \frac{1}{N}\sum_{i=1}^{N} d(i)$
2	Standard deviation	$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(d(i)-\mu)^2}$
3	Mean absolute deviation	$D_x = \frac{1}{N}\sum_{i=1}^{N} \mid d(i) - \mu \mid$
4	Skewness	$s = \frac{\frac{1}{N}\sum_{i=1}^{N}(d_i-\mu)^3}{(\sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i-\mu)^2})^3}$
5	Kurtosis	$k = \frac{\frac{1}{N}\sum_{i=1}^{N}(d_i-\mu)^4}{(\frac{1}{N}\sum_{i=1}^{N}(d_i-\mu)^2)^2}$
6	Root-Mean-Square (RMS)	$d_{rms} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \mid d_i \mid^2}$
7	Max	$d_{max} = max(d)$
8	Min	$d_{min} = min(d)$
9	Number of peaks	$c_{peaks} = peaks(d)$
10	Number of valleys	$c_{valleys} = valley(d)$
11	Variance	$v = \frac{1}{N-1}\sum_{i=1}^{N} \mid d_i - \mu \mid^2$
12	Energy	$E_d = \sum_{i=1}^{N} \mid d_i \mid^2$
13	Peak difference	$d_p = d_{max} - d_{min}$
14	Pearson correlation coefficient	$\rho = \frac{cov(A,B)}{\sigma_A \sigma_B}$

Step 2 (Gesture Identification): We utilize feature-based machine learning techniques to identify tap or shake gestures from the phase and acceleration data segments.

3) Feature Extraction: We explore 13 features in total (#1 to #13 in Table I) to represent the phase segment. In addition, we extract #1 to #13 features for each axis and compute the Pearson correlation coefficient (PCC) (#14 feature) between every two axes for the acceleration data segment. Hence, we can obtain a 13-feature vector and a 42-feature vector from the phase and acceleration data segments, respectively.

4) Intentional-Gesture Detector:: Since we cannot collect profiles of all potential gestures, we adopt one-class support vector machine (OC-SVM) [22] to train two gesture detectors to determine if a phase or acceleration data segment represents an intentional gesture (i.e., tap or shake). OC-SVM maps input data into a higher dimensional feature space via a kernel and finds the maximal margin hyperplane which best separates the training data from the origin. It has been successfully applied in a wide variety of application areas such as anomaly detection [23]. In this paper, we select a Radial Basis Function (RBF) kernel and a sigmoid kernel for the phase and acceleration data, respectively.

5) Tap or Shake Classifier:: Once a data segment is identified as an intentional gesture, we further build classifiers from training datasets with known labels to infer its gesture type as either tap or shake. In particular, since the two intentional gestures can induce four possible phase patterns (see Fig. 3), we train a multi-class classifier to recognize the gesture for each phase segment. In addition, we use a binary classifier to recognize the gesture for each acceleration segment. SmartRFID can work with many existing machine learning techniques. In this paper, we consider three lightweight supervised machine learning techniques, including Support Vector Machine (SVM), Random Forest (RF), and k-nearest neighbors (k-NN). The performance of these techniques are compared in Section V.

Step 3 (Phase-Sequence Selection): Since the Phase-Data Segmentation submodule outputs a set S_p of potential timestamp sequences, SmartRFID needs to determine which one is feasible. We leverage the timestamp sequence T_a and its corresponding gesture-type vector from the acceleration data to find the most feasible timestamp sequence in S_p. More specifically, we first identify a subset from S_p, denoted by S'_p, such that each sequence in S'_p has the same gesture-type vector as that of T_a. Since the phase-data and acceleration-data collection processes may not be perfectly synchronized, we further derive a relative-time sequence for T_a and also one for each sequence in S'_p, in which each element is updated by subtracting the first timestamp. Next, we compute three metrics between the relative-time sequence of T_a and that of each sequence in S'_p, including the PCC, average element-wise difference, and the element-wise difference variance. Then a phase sequence in S'_p with a higher PCC, lower average element-wise difference, and lower element-wise difference variance is ranked higher for each metric accordingly. Finally, we select the sequence in S'_p with the highest average rank as the most feasible phase-data sequence, denoted by $T_p = [\hat{t}^1_e, \hat{t}^2_s, \hat{t}^2_e, \ldots, \hat{t}^n_s, \hat{t}^n_e]$. As in Acceleration-Data Segmentation, we also generate a fine-grained phase data vector ϕ that includes all the processed timestamped phase data between $\hat{t}^1_e$ and $\hat{t}^n_e$ to fully characterize intentional hand movement.

F. Correlation Detection

Finally, we explore two different ways to check the correlation between the phase sequence and acceleration sequence, based on traditional feature-based machine learning methods and deep learning techniques, respectively.

1) Feature-Based Correlation Detection: We use traditional machine learning classifiers to check if T_p and T_a correlate. Since the phase and acceleration data are supposed to be induced by the same intentional hand movement, the duration of each gesture and interval between two adjacent gestures should be highly consistent in T_p and T_a. As in Phase-Sequence Selection, we first obtain the two relative-time data sequences, denoted by $T^r_p = [0, \hat{t}^1_{e'}, \hat{t}^2_{s'}, \hat{t}^2_{e'}, \ldots, \hat{t}^n_{s'}, \hat{t}^n_{e'}]$ and $T^r_a = [0, t^1_{e'}, t^2_{s'}, t^2_{e'}, \ldots, t^n_{s'}, t^n_{e'}]$. Then we compute the difference between T^r_p and T^r_a as $D = [0, \hat{t}^1_{e'} - t^1_{e'}, \hat{t}^2_{s'} - t^2_{s'}, \hat{t}^2_{e'} - t^2_{e'}, \ldots, \hat{t}^n_{s'} - t^n_{s'}, \hat{t}^n_{e'} - t^n_{e'}]$. Next, based on aforementioned observation, we extract the mean, standard deviation, mean absolute deviation, max, and min from D to generate a feature vector for (T_p, T_a). Afterwards, we use the resulting feature vector to train a binary classifier based on any established machine learning technique such as SVM, RF, and k-NN which are compared in Section V. During each authentication session, the server explores the same process to extract a feature vector and then test it with the classifier.

2) Deep Correlation Network: We also explore deep learning to build a *deep correlation network* to check the correlation between the phase and acceleration data. Fig. 7 shows the architecture of our deep correlation network whose inputs are the first-order derivatives of the phase and acceleration data, denoted by ϕ' and A', respectively. We pad zeros in the end (if needed) to make all the input vectors/matrices have the same length. The network has two branches that can extract features from the phase and acceleration data, respectively. Both network branches consist of two one-dimensional (1D) convolutional layers and a Long Short Term Memory (LSTM) layer. Each 1D convolutional layer is followed by a batch normalization Layer, a Rectified Linear Unit (ReLU) activation layer, and a dropout layer. The two 1D convolutional layers extract features from the input vector/matrix, and the LSTM layer encodes the sequence of features to a vector representation. The two branches produce two feature matrices representing the two input data vectors, respectively. We then compute the matrix product of the two feature matrices to quantify their similarity in each time step. Afterwards, we perform a max pooling over the matrix product and then use two fully connected (FC) layers with sigmoid activation to produce a value between 0 and 1, which can be interpreted as a correlation probability. If the output value is above 0.5, we consider T_p and T_a correlated and otherwise uncorrelated.

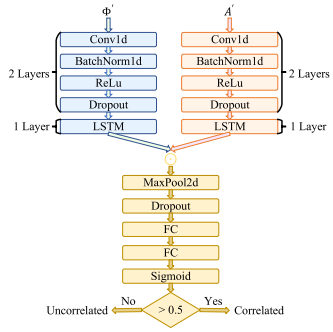


Fig. 7. Deep correlation network.

V. Performance Evaluation

In this section, we evaluate the usability and security of SmartRFID. Following the adversary model in Section III, we considered two types of attackers. **Type-1** attackers can observe the victim's hand movement in real time and then attempts synchronized hand movement with the victim. **Type-2** attackers can additionally record and practice the victim's hand movement multiple times beforehand.

A. Experiment Setup and Performance Metrics

We prototyped SmartRFID with commodity devices. As shown in Fig. 8, we used an Impinj Speedway R420 RFID reader equipped with a circularly polarized RFID antenna and connected it to a Dell Precision laptop that acts as the backend server. The RFID reader continuously sent queries at a rate of 100 reads/second. In addition, we used Zebra's passive RFID cards as users' access cards in our experiments. But our system is applicable to any other commodity RFID tags or cards such as Aline ALN-9740 RFID tags and Omni-ID's Adept 650P RFID cards. To collect acceleration data, we implemented Android application and Tizen application on Huawei Watch 2 that runs Android Wear 2.1 and Samsung Galaxy Watch that runs Tizen 5.0, respectively. The accelerometer sampling

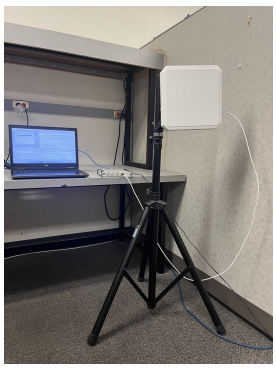


Fig. 8. Experimental setup.

rate was set to 100 Hz. We also implemented a Java application based on the Octane SDK together with the reader to record the phase of the backscattered signals. The application on the smartwatch and the phase-recording application are both part of the Data Acquisition submodule in SmartRFID. In addition, we implemented Data Preprocessing, Intentional-Gesture Recognition, and Feature-based Correlation Detection using Matlab on the laptop. Finally, we constructed the deep correlation network in PyTorch 1.7 and trained the model on Dell 7920 Tower with NVIDIA Quadro GP100 16GB GPU.

In SmartRFID, users need to perform intentional hand movement which should be not only easy and quick to perform by legitimate users but also hard to imitate by attackers. We evaluated the following six types of intentional hand movement events in our experiments: (1) Tap-4/5/6: tap an RFID card four/five/six times, respectively; and (2) Shake-4/5/6: shake an RFID card four/five/six times, respectively. In addition, there were random human activities such as walking during the experiments.

The main performance metrics we use include the True Acceptance Rate (TAR), False Acceptance Rate (FAR), Receiver Operating Characteristics (ROC) curve, Equal Error Rate (EER), and Area Under The Curve (AUC).

B. Data Collection

With the IRB approval from our institution, we recruited 20 participants for the experiments, including 3 females and 17 males aged between 20 and 35. We collected the following three training datasets (Dataset I to III) and two test datasets (Datasets IV and V).

1) Dataset-I for Intentional Gesture Recognition: We constructed a gesture-profile dataset to recognize tap and shake gestures in intentional hand movement. In particular, we asked a participant $\mathcal{P}$ to perform 1,000 taps and also 1,000 times of each of Shake-I, Shake-II, and Shake-III gestures (see Fig. 3). Then we fed the resulting raw phase and acceleration data into the Data Preprocessing module to obtain the phase and acceleration profiles of the tap and shake gestures, respectively. Finally, we collected 4,000 phase data samples and also 4,000 acceleration data samples. The gesture-profile dataset is not user-specific.

2) Dataset-II for Feature-Based Correlation Detection: and **Dataset-III for Deep Correlation Network.** We further asked $\mathcal{P}$ to perform each intentional hand movement 700 times to collect phase and acceleration data samples. So we collected 4,200 positive paired phase-acceleration data samples, each of which is labeled as 1. In addition, we asked $\mathcal{P}$ and another participant $\mathcal{P}'$ to perform each intentional hand movement 10 times. A phase data sample from $\mathcal{P}$ and an acceleration data sample from $\mathcal{P}'$ constitute an uncorrelated phase-acceleration pair, leading to $60 \times 60 = 3,600$ negative paired phase-acceleration samples. Furthermore, we asked $\mathcal{P}'$ to act as a Type-2 attacker and mimic the actions of $\mathcal{P}$ in real time. $\mathcal{P}$ was required to perform each intentional hand movement 100 times, so we collected another 600 negative paired phase-acceleration samples and thus 4,200 in total which are labeled as 0. Next, we randomly chose 200 from the 700 positive paired samples of each intentional hand movement and 1,200 negative paired samples from all negative paired samples. Then we fed them into the Data preprocessing and Intentional-Gesture Recognition modules to build Dataset-II that contains 1,200 positive and 1,200 negative paired phase-acceleration samples. In addition, we input all raw data into Data Preprocessing and Intentional-Gesture Recognition modules to extract pairs of phase and acceleration data vectors. The first derivatives in such phase-acceleration vector pairs were used to construct Dataset-III that have 4,200 positive and negative paired phase-acceleration samples.

3) Dataset-V for Intentional-Gesture Detector: Five participants were asked to perform four daily activities within the transmission range of the RFID reader: walking, using a computer, using a phone, and any other physical activities. All such activities led to unintentional hand gestures other than tap and shake gestures. Each participant wore the smartwatch and a Vulcan RFID wristband to perform these activities. After obtaining the raw data, we used the Data Preprocessing module to remove noise. Then we used our iterative algorithm and the Phase-Data Segmentation submodule to extract the acceleration and phase data segments as negative samples. We collected 800 negative gesture samples in total.

4) Dataset-VI for User and Attacker Emulation: All volunteers acted as either legitimate users or attackers to generate this dataset. Each volunteer was first asked to practice the aforementioned 6 intentional hand movement events multiple times until he/she was familiar with them. Afterwards, we required them to perform each intentional hand movement 20 times. So we collected 2,400 positive paired phase-acceleration samples in total. In addition, 6 out of the 20 participants served as both Type-1 and Type-2 attackers. Each attacker was physically co-located with the victim and could clearly observe the victim's hand movement. For each volunteer, the other 5 were regarded as his/her victims. Each victim-attacker pair performs each intentional hand movement 10 times, leading to $50 \times 6 \times 2 = 600$ negative paired phase-acceleration samples.

C. Model Training

We used Dataset-I to train two OC-SVM models for the phase and acceleration data, respectively. Additionally, we trained SVM, RF, and k-NN on Dataset-I and Dataset-II

for intentional gesture recognition and feature-based correlation detection. We adopted 10-fold cross-validation to evaluate each model. In particular, we applied OC-SVM and SVM with four kernel functions, including linear, polynomial, RBF, and sigmoid. We also used the grid search method to find the best parameters. The implementations of OC-SVM and SVM relied on LibSVM [24]. For RF, we tested the number of decision trees ranging from 5 to 100. For k-NN, we tested the number of neighbors ranging from 1 to 40 and various distance metrics such as Standardized Euclidean distance. The parameter settings are summarized as follows.

1) Tap/Shake Gesture Detector: We chose RBF and the sigmoid function as kernel functions for phase and acceleration data, respectively. We set the parameter ν to 0.05 and γ to 0.1 for phase data, and ν to 0.1 and γ to 0.4 for acceleration data.

2) Tap/Shake Gesture Classifier: For SVM, we used the polynomial function and RBF as the kernel functions for the phase and acceleration data, respectively. We set γ to 3 and degree to 3 for phase data, and γ to 0.01 for acceleration data. For RF, we set the number of decisions to 22 and 5 for the phase and acceleration data. For k-NN, we chose Standardized Euclidean distance as the distance metric for both data types. The k value is set to 9 and 5 for phase and acceleration data, respectively.

3) Feature-Based Correlation Detection: For SVM, we selected RBF as the kernel function with γ and c set to 2 and 2^{19}, respectively. We also chose 31 decision trees for the RF classifier. For k-NN, we used Manhattan distance as distance metric and set k to 7.

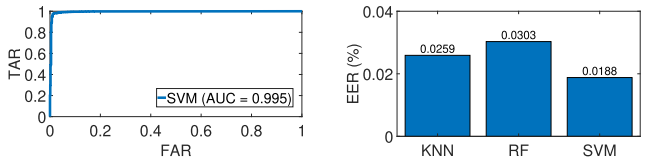
4) Deep Correlation Network: We used dataset-III to train the deep correlation network. The number of hidden units in each 1D convolutional layer and LSTM layer was set to 128. The kernel size of 1D convolutional layer was 3 with stride of 1. The dropout rate was set to 0.2 for all dropout layers and the max pooling window size was 10. We adopted two FC layers with 120 and 80 units, respectively. We trained the network by minimizing binary cross entropy between the actual label and the output using the Adam optimizer. It took approximately 20 minutes to train the deep neural network on 200 epochs.

D. Intentional-Gesture Recognition

We used Dataset-V and Dataset-VI to evaluate intentional-gesture detectors. The TPRs are 94.17% and 99.88% for phase-based and acceleration-based intentional-gesture detectors, respectively, and the corresponding FPRs are 1.57% and 0.02%, respectively. The results show that our intentional-gesture detectors can rule out unintentional gestures, so SmartRFID can naturally detect attackers who perform random hand movement with overwhelming probability. In addition, we applied intentional tap/shake gesture classifiers on Dataset-VI to evaluate the classification accuracy that is defined as the ratio of true positives plus true negatives to the total number of samples. Table II shows that SVM outperforms RF and NB for the phase data, and all the three classifiers can achieve 100% accuracy for the acceleration data. We further compared the average prediction time and found that SVM took less time than RF and NB. Therefore, we chose SVM as the intentional tap/shake gesture classifier for both phase and acceleration data in subsequent experiments.

TABLE II
CLASSIFICATION ACCURACY FOR INTENTIONAL-GESTURE RECOGNITION

Data Type	SVM	RF	KNN
Phase	98.875%	97.75%	97.12%
Acceleration	100%	100%	100%



(a) ROC of SVM (b) EER

Fig. 9. ROC and EER comparison for feature-based correlation detection.

TABLE III
FAR FOR TYPE-1 AND TYPE-2 ATTACKERS

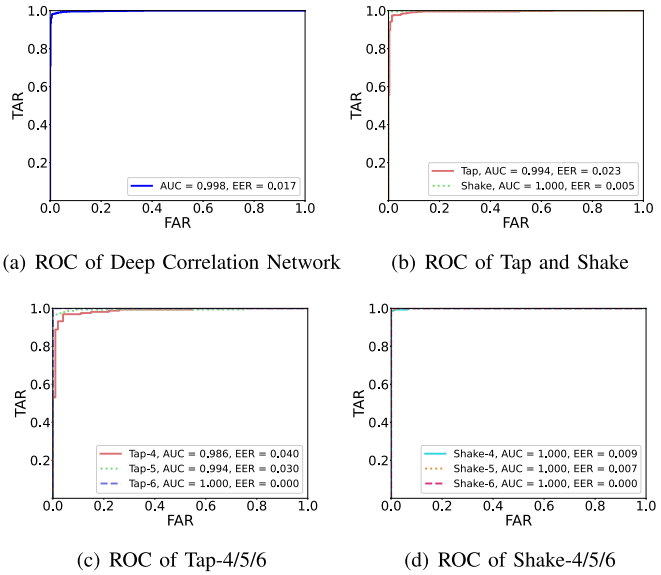
	Tap-4	Tap-5	Tap-6	Shake-4	Shake-5	Shake-6
Type-1	0%	0%	0%	0%	0%	0%
Type-2	1%	1%	1%	0%	0%	0%

E. Feature-Based Correlation Detection

We fed raw data samples from Dataset-VI into Data Preprocessing and Intentional-Gesture Recognition modules to extract timestamp vectors. We then compared the EERs of SVM, RF, and kNN classifiers in Fig.9(b). Since the EER of SVM is lower than those of RF and k-NN, we chose SVM for the Feature-Based Correlation Detection module. Moreover, Fig. 9(a) shows the ROC curve of the SVM classifier. Since the ROC curve is located at the top-left corner, we can simultaneously achieve a very high TAR and a very low FAR. Specifically, the TAR of feature-based correlation detection is 97.3%. We also evaluated the resilience of the SVM classifier to Type-1 and Type-2 attackers. The FARs under Type-1 and Type-2 attackers are 0.78% and 6.3%, respectively. It is also not surprising to see that Type-2 attackers have a higher success rate than Type-2 attackers. To sum up, SmartRFID using feature-based correlation detection can correctly distinguish legitimate users from Type-1 and Type-2 attackers with very high probability.

F. Deep Correlation Network

We evaluated the deep correlation network on Dataset-VI. As in Section V-E, we obtained phase and acceleration data vectors from the raw data. Then we computed the first derivatives of both data vectors and input them into the pre-trained deep correlation network to check if the two data vectors correlate. Fig. 10(a) shows the ROC curve of the deep correlation network. Specifically, the TAR of the deep correlation network is 97.56%,



(a) ROC of Deep Correlation Network (b) ROC of Tap and Shake

(c) ROC of Tap-4/5/6 (d) ROC of Shake-4/5/6

Fig. 10. ROC and AUC of deep correlation network.

indicating that it can recognize legitimate users with very high probability. In addition, the overall FAR of the deep correlation network is 0.67%. In particular, the FARs under Type-1 and Type-2 attackers are 0% and 1%, respectively. The results demonstrate that the deep correlation network can detect Type-1 and Type-2 attackers with overwhelming probability.

Tap and shake gestures have slightly different performance. As shown in Fig. 10(b), the shake gesture has better performance than the tap gesture. One possible reason is that the shake gesture can induce larger amplitude changes than the tap gesture, which makes the correlation detection easier and the attacker emulation harder. Let us further examine the ROC curves for all six types of intentional hand movement. As shown in Fig. 10(c) and Fig. 10(d), the deep correlation network performs better as the number of taps or shakes increases. This indicates that the more gestures the user performs, the easier for the deep correlation network to check the acceleration-phase correlation, the harder for attackers to mimic the victim's intentional hand movement. Moreover, Table III shows that all Type-1 attackers in our experiments can be detected by the deep correlation network. Additionally, although Type-2 attackers can occasionally mimic the victim's tap-4/5/6 gestures, it is still very difficult for them to bypass the deep correlation network. Moreover, none of them can mimic the victim's shake gestures.

As we can see, the deep correlation network is better in distinguishing legitimate users from Type-1 and Type-2 attackers. The reason is that feature-based correlation detection only considers time information, while the deep correlation network additionally considers the data-change pattern. We can also notice that the deep correlation network needs more training data than feature-based correlation detection. Which method(s) to use in SmartRFID thus depends on practical security requirements and the availability of training data.

TABLE IV
USABILITY SCORES

	Mean	Standard Deviation	Min	Median	Max
Q1	4.62	0.51	4	5	5
Q2	4.5	0.75	3	5	5
Q3	4.63	0.74	3	5	5
Q4	4.88	0.35	4	5	5

G. Authentication Latency

We also studied the authentication latency of SmartRFID. The authentication latency can be broken into three parts: the time to perform an intentional hand movement event, the network delay to transmit accelerometer data to the server, and the response time that the system needs to make a decision. In our experiments, the average time for a tap and a shake are about 271 ms and 408 ms, respectively. Additionally, the average performing time of Tap-4/5/6 and Shake-4/5/6 events ranges from $1.33\,\mathrm{s}$ to $2.74\,\mathrm{s}$. The overall average performing time is about $1.99\,\mathrm{s}$. Moreover, the average network delay for transferring accelerometer data of $5\,\mathrm{s}$ is about 48 ms. Furthermore, the average response time for Tap-4/5/6 and Shake-4/5/6 events varies between $0.157\,\mathrm{s}$ and $0.176\,\mathrm{s}$. The overall average response time is $0.167\,\mathrm{s}$. Hence, the average authentication latency ranges from $1.535\,\mathrm{s}$ to $2.964\,\mathrm{s}$. The overall average authentication latency is $2.205\,\mathrm{s}$, which is comparable to inputting a PIN on a door keypad.

H. Usability Studies

We also surveyed the same 20 volunteers about their experience using SmartRFID. Specifically, we asked each volunteer (Q1) whether it is easy to learn using SmartRFID, (Q2) if SmartRFID is easy to use, (Q3) if tap and shake gestures are easy to perform, and (Q4) if intentional hand movement is easy to memorize. Each participant was asked to give a score ranging from one (lowest) to five (highest) for each question and was not allowed to give all five. The average scores are listed in Table IV. The results clearly indicate that SmartRFID is very easy to use and more preferable than traditional RFID-based authentication due to its high security.

VI. RELATED WORK

A. Mobile Authentication

There is prior work using smartwatch-like wrist wearables as secure tokens for mobile authentication. ZEBRA [25] continuously verifies a user's identity when he/she works on a computer by comparing two sequences of operations (e.g., typing and scrolling) captured by his/her wristband and the computer. A user is authenticated if the majority of the two operation sequences match. SAW [26] requires users to tap a key on a keyboard multiple times or wiggle a mouse for a few seconds with their wristband hand to unlock their desktops. Users are considered legitimate if the difference is less than a predefined threshold between the two time sequences of keystrokes or wiggles that are extracted from the desktop and the built-in motion sensors of their wristbands. WristUnlock [27] asks a user to raise his/her smartphone

naturally with his/her wristband hand to unlock the smartphone by checking the correlation between acceleration data from the wrist wearable and smartphone. As another example, Pet-2-Auth [28] authenticates users on IoT devices that have buttons, knobs, or touchscreen, such as Nest Thermostat and Amazon Echo. Users click on buttons or twist knobs multiple times with their wristband hands. Pet-2-Auth then uses a SVM classifier to check the correlation between the two timestamp sequences of these actions extracted from the user's wristband and the target IoT device. SmartRFID differs significantly from the aforementioned work in the application context and data processing techniques. Specifically, SmartRFID focuses on commodity UHF RFID systems that have a very different user interaction mode. In particular, RFID systems are not equipped with any interaction interface, so we can only leverage raw RFID signals. SmartRFID explores new algorithms to extract intentional-gestures related data from noisy phase data and uses SVM classifiers to recognize them. Moreover, we explore and design a cross-modal deep neural network to check the correlation between phase and accelerometer data.

Biometric authentication methods have been widely explored for mobile authentication as well. They can be classified into two categories: physiological and behavioral [29]. Physiological biometric authentication relies on unique biological traits such as retinas, irises, voices, facial characteristics, and fingerprints to authenticate users. Behavioral biometric authentication identifies users by their unique patterns exhibited when they interact with a mobile device [30], [31], [32], [33], [34], [35], [36]. These schemes all aim to secure smart devices themselves and are orthogonal to SmartRFID which explores commodity smart devices to secure commodity UHF RFID systems.

B. RFID Security

RFID security has also been extensively studied. Many cryptographic authentication protocols can prevent illegal reading of tags [37], [38], [39], but these schemes cannot be applied to commodity crypto-less UHF RFID tags. Czeskis [40] et al. proposed to mitigate replay attacks by requiring RFID users to handshake their cards with a build-in accelerometer during the authentication session. RF-Cloak [41] protects RFID systems from eavesdropping attacks by randomizing modulation and wireless channels. Hu-Fu [42] is a physical-layer authentication method for passive RFID tags by leveraging inductive coupling of two adjacent tags and signal randomization. RF-Mehndi [43] authenticates an RFID card and its holder's identity simultaneously by exploring the backscattered signal phase changes induced by the holder's fingertip on a carefully design passive tag array. More recently, RF-Rhythm [44] identifies an RFID user's identity by requiring the user to perform a sequence of taps on his/her card according to a self-chosen secret melody. RCID [45] is a new fingerprinting scheme for RFID tags based on wideband backscatter. WearRF-CLA [46] combines wearables and RFID tags to achieve secure and usable continuous location authentication. Orthogonal to the above work, SmartRFID explores pervasive smart devices to enhance the security of RFID authentication systems without requiring any hardware modification. In [4], Tan et al. design a protocol to authenticate RFID tags. In [5], Saad et al. proposes a new approach to inject noise-like signals at the reader end to prevent attackers from eavesdropping tag information. In [6], Tan et al. presents two protocols to accurately and efficiently monitor the RFID tags for missing tags. By comparison, SmartRFID does not require any modification on the reader end. SmartRFID can be applied to generic UHF RFID systems with user-carried commodity smart devices and RFID tags.

The schemes in [35] and [36] authenticate users by exploring an off-body RFID tag array to sense users' daily activities such as door knocking or walking patterns. Both schemes can be considered behavioral biometric RFID authentication techniques and require each RFID user to get involved in the intensive model-training process. In addition, they require a customized off-body RFID tag array and thus cannot be applied to generic UHF RFID systems. By comparison, the model training in SmartRFID is generic to all users and does not require individual user involvement. SmartRFID also applies to generic UHF RFID systems with user-carried commodity smart devices and RFID tags.

VII. Conclusion

In this paper, we presented the design and evaluation of SmartRFID, a novel UHF RFID authentication system. SmartRFID explores pervasive user-carried smart devices to protect commodity crypto-less UHF RFID tags from spoofing and cloning and thus greatly enhances their applicability to security-sensitive contexts. We designed novel feature-based machine learning techniques and also deep learning techniques to check the coexistence of an RFID tag and its associated smart device on the RFID user. Comprehensive user experiments on commodity RFID devices and smartwatches confirmed the high security and usability of SmartRFID.

Acknowledgment

The authors would like to thank anonymous reviewers for their constructive and helpful advice.

References

[1] (2019). *Global Radio Frequency Identification (RFID) Market: Transforming the Healthcare Industry*. [Online]. Available: https://rb.gy/lucy0r

[2] (2020). *MIFARE SAM AV3 for NTAG 5, ICODE DNA and UCODE DNA*. [Online]. Available: https://www.nxp.com/docs/en/application-note/AN12698.pdf

[3] (2020). *SL3S5002N0FUD/00CZ by NXP Semiconductor*. [Online]. Available: https://www.arrow.com/en/products/sl3s5002n0fud00cz/nxp-semiconductors

[4] C. C. Tan, B. Sheng, and Q. Li, "Secure and serverless RFID authentication and search protocols," *IEEE Trans. Wireless Commun.*, vol. 7, no. 4, pp. 1400–1407, Apr. 2008.

[5] W. Saad, X. Zhou, Z. Han, and H. V. Poor, "On the physical layer security of backscatter wireless systems," *IEEE Trans. Wireless Commun.*, vol. 13, no. 6, pp. 3442–3451, Jun. 2014.

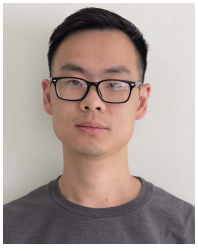
[6] C. C. Tan, B. Sheng, and Q. Li, "Efficient techniques for monitoring missing RFID tags," *IEEE Trans. Wireless Commun.*, vol. 9, no. 6, pp. 1882–1889, Jun. 2010.

[7] (2019). *Impinj Speedway Revolution R420 UHF RFID Reader*. [Online]. Available: https://www.atlasrfidstore.com/impinj-speedway-revolution-r420-uhf-rfid-reader-4-port/

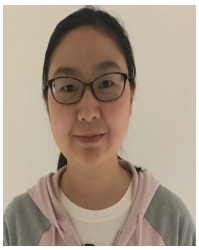
[8] Y. Zou, J. Xiao, J. Han, K. Wu, Y. Li, and L. M. Ni, "GRfid: A device-free RFID-based gesture recognition system," *IEEE Trans. Mobile Comput.*, vol. 16, no. 2, pp. 381–393, Feb. 2017.

[9] C. Wang et al., "Multi—Touch in the air: Device-free finger tracking and gesture recognition via COTS RFID," in *Proc. IEEE INFOCOM*, Honolulu, HI, USA, Apr. 2018, pp. 1691–1699.
[10] S. Zhang, C. Yang, X. Kui, J. Wang, X. Liu, and S. Guo, "Reactor: Real-time and accurate contactless gesture recognition with RFID," in *Proc. SECON*, Boston, MA, USA, Jun. 2019, pp. 1–9.
[11] (2020). *Duo*. [Online]. Available: https://www.duosecurity.com/product/methods/duo-mobile
[12] K. Reese, T. Smith, J. Dutson, J. Armknecht, J. Cameron, and K. Seamons, "A usability study of five two-factor authentication methods," in *Proc. SOUPS*, Santa Clara, CA, USA, Aug. 2019, pp. 357–370.
[13] M. Jubur, P. Shrestha, N. Saxena, and J. Prakash, "Bypassing push-based second factor and passwordless authentication with human-indistinguishable notifications," in *Proc. AsiaCCS*, Jun. 2021, pp. 447–461.
[14] J. Wang, F. Hu, Y. Zhou, Y. Liu, H. Zhang, and Z. Liu, "BlueDoor: Breaking the secure information flow via BLE vulnerability," in *Proc. ACM MobiSys*, Toronto, ON, Canada, Jun. 2020, pp. 286–298.
[15] S. Jasek, "Gattacking Bluetooth smart devices," in *Proc. Black Hat USA Conf.*, Las Vegas, NV, USA, Jul. 2016.
[16] S. Akter, S. Chellappan, T. Chakraborty, T. A. Khan, A. Rahman, and A. B. M. A. A. Islam, "Man-in-the-middle attack on contactless payment over NFC communications: Design, implementation, experiments and detection," *IEEE Trans. Depend. Sec. Comput.*, vol. 18, no. 6, pp. 3012–3023, Nov. 2021.
[17] L. Francis, G. P. Hancke, K. Mayes, and K. Markantonakis, "Practical relay attack on contactless transactions by using NFC mobile phones," *IACR Cryptol. ePrint Arch.*, vol. 2011, p. 618, May 2011.
[18] (2013). *Speedway Revolution Reader Application Note: Low Level User Data Support*. [Online]. Available: https://support.impinj.com/hc/en-us/articles/202755318-Application-Note-Low-Level-User-Data-Support
[19] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures," *Anal. Chem.*, vol. 36, no. 8, pp. 1627–1639, Jul. 1964.
[20] Y. Fujiki, "IPhone as a physical activity measurement platform," in *Proc. CHI EA*, Atlanta, GA, USA, Apr. 2010, pp. 4315–4320.
[21] *Reaction Time Statistics*. (2019). [Online]. Available: https://www.humanbenchmark.com/tests/reactiontime/statistics
[22] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, Jul. 2001.
[23] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognit.*, vol. 58, pp. 121–134, Oct. 2016.
[24] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, Apr. 2011.
[25] S. Mare, A. M. Markham, C. Cornelius, R. Peterson, and D. Kotz, "ZEBRA: Zero-effort bilateral recurring authentication," in *Proc. IEEE Symp. Secur. Privacy*, May 2014, pp. 705–720.
[26] S. Mare, R. Rawassizadeh, R. Peterson, and D. Kotz, "SAW: Wristband-based authentication for desktop computers," in *Proc. UbiComp*, Singapore, Oct. 2018, pp. 1–29.
[27] L. Zhang, D. Han, A. Li, T. Li, Y. Zhang, and Y. Zhang, "WristUnlock: Secure and usable smartphone unlocking with wrist wearables," in *Proc. IEEE CNS*, Washington, DC, USA, Jun. 2019, pp. 28–36.
[28] X. Li, F. Yan, F. Zuo, Q. Zeng, and L. Luo, "Touch well before use: Intuitive and secure authentication for IoT devices," in *Proc. ACM MobiCom*, Los Cabos, Mexico, Oct. 2019, pp. 1–17.
[29] (2020). *What are biometrics—A Complete Guide*. [Online]. Available: https://www.miteksystems.com/blog/what-are-biometrics-a-complete-guide
[30] J. Tian, C. Qu, W. Xu, and S. Wang, "KinWrite: Handwriting-based authentication using Kinect," in *Proc. NDSS*, San Diego, CA, USA, Feb. 2013, pp. 1–18.
[31] F. Monrose and A. Rubin, "Authentication via keystroke dynamics," in *Proc. ACM CCS*, Zürich, Switzerland, 1997, pp. 48–56.
[32] H. Feng, K. Fawaz, and K. G. Shin, "Continuous authentication for voice assistants," in *Proc. MobiCom*, Snowbird, UT, USA, Oct. 2017, pp. 343–355.
[33] Y. Wang, Y. Wang, and T. Tan, "Combining fingerprint and voiceprint biometrics for identity verification: An experimental comparison," in *Proc. ICBA*, Hong Kong, Jul. 2004, pp. 663–670.
[34] C. Shi, J. Liu, H. Liu, and Y. Chen, "Smart user authentication through actuation of daily activities leveraging WiFi-enabled IoT," in *Proc. ACM MobiHoc*, Chennai, India, Jul. 2017, pp. 1–10.
[35] A. Huang, D. Wang, R. Zhao, and Q. Zhang, "Au-ID: Automatic user identification and authentication through the motions captured from sequential human activities using rfid," in *Proc. UbiComp*, London, U.K., Sep. 2019, pp. 1–26.
[36] C. Feng, J. Xiong, L. Chang, F. Wang, J. Wang, and D. Fang, "RF-identity: Non-intrusive person identification based on commodity RFID devices," in *Proc. UbiComp*, Sep. 2021, pp. 1–23.
[37] L. Kulseng, Z. Yu, Y. Wei, and Y. Guan, "Lightweight mutual authentication and ownership transfer for RFID systems," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, Mar. 2010, pp. 1–5.
[38] T. Li, W. Luo, Z. Mo, and S. Chen, "Privacy-preserving RFID authentication based on cryptographical encoding," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 2174–2182.
[39] L. Yang, Q. Lin, C. Duan, and Z. An, "Analog on-tag hashing: Towards selective reading as hash primitives in Gen2 RFID systems," in *Proc. ACM MobiCom*, Snowbird, UT, USA, Oct. 2017, pp. 301–314.
[40] A. Czeskis, K. Koscher, J. R. Smith, and T. Kohno, "RFIDs and secret handshakes: Defending against ghost-and-leech attacks and unauthorized reads with context-aware communications," in *Proc. ACM CCS*, New York, NY, USA, 2008, pp. 479–490.
[41] H. Hassanieh, J. Wang, D. Katabi, and T. Kohno, "Securing rfids by randomizing the modulation and channel," in *Proc. NSDI*, Santa Clara, CA, USA, Mar. 2015, pp. 235–249.
[42] G. Wang et al., "Towards replay-resilient RFID authentication," in *Proc. ACM MobiCom*, New Delhi, India, Oct. 2018, pp. 385–399.
[43] C. Zhao et al., "RF-Mehndi: A fingertip profiled RF identifier," in *Proc. IEEE INFOCOM*, Paris, France, Apr. 2019, pp. 1513–1521.
[44] J. Li et al., "RF-Rhythm: Secure and usable two-factor RFID authentication," in *Proc. IEEE INFOCOM*, Jun. 2020, pp. 2194–2203.
[45] J. Li, A. Li, D. Han, Y. Zhang, T. Li, and Y. Zhang, "RCID: Fingerprinting passive RFID tags via wideband backscatter," in *Proc. IEEE INFOCOM*, May 2022, pp. 700–709.
[46] A. Li, J. Li, D. Han, Y. Zhang, T. Li, and Y. Zhang, "WearRF-CLA: Continuous location authentication with wrist wearables and UHF RFID," in *Proc. ACM AsiaCCS*, May 2022, pp. 508–520.

Ang Li (Student Member, IEEE) received the B.E. degree in network engineering from Guangxi University, China, in 2010, and the M.S. degree in computer science from Beihang University, China, in 2014. He is currently pursuing the Ph.D. degree in computer engineering with Arizona State University. His research interests include security and privacy in social networks, machine learning, wireless networks, and mobile computing.

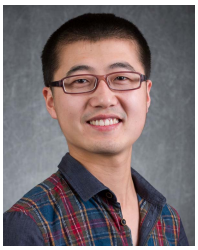


Jiawei Li (Graduate Student Member, IEEE) received the B.E. degree in telecommunication engineering from the Nanjing University of Posts and Telecommunications in 2013. He is currently pursuing the Ph.D. degree in computer engineering with Arizona State University. His research interests include security and privacy issues in wireless networks and wireless sensing.

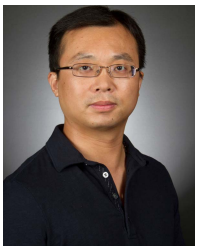


Yan Zhang (Member, IEEE) received the Ph.D. degree in computer engineering from Arizona State University. She is currently an Assistant Professor with the Department of Electrical and Computer Engineering, The University of Akron. Her research interests include cybersecurity and privacy issues in mobile and networked systems, including the Internet of Things, AI/ML-powered wireless and mobile systems, mobile sensing, and mobile crowdsourcing.

Dianqi Han (Member, IEEE) received the Ph.D. degree in computer engineering from Arizona State University. He is currently an Assistant Professor with the Department of Computer Science and Engineering, The University of Texas at Arlington. His research interests include security and privacy in networked systems, including AI-powered cybersecurity, the Internet of Things (IoT), wireless security and privacy, mobile computing and security, and smart and connected health.



Tao Li (Member, IEEE) received the B.E. degree in software engineering from Hangzhou Dianzi University in 2012, the M.S. degree in computer science and technology from Xi'an Jiaotong University in 2015, and the Ph.D. degree in computer engineering from Arizona State University in 2020. He is currently an Assistant Professor with the Department of Computer and Information Technology, Indiana University-Purdue University Indianapolis (IUPUI). His research interests include security and privacy issues in networked/mobile/distributed systems, smart sensing, and wireless networks.



Yanchao Zhang (Fellow, IEEE) received the B.E. degree in computer science and technology from the Nanjing University of Posts and Telecommunications in 1999, the M.E. degree in computer science and technology from the Beijing University of Posts and Telecommunications in 2002, and the Ph.D. degree in electrical and computer engineering from the University of Florida in 2006. He is currently a Professor with the School of Electrical, Computer and Energy Engineering, Arizona State University. His research interests include network and distributed system security, wireless networking, and mobile computing. He received the U.S. NSF CAREER Award in 2009 and is an IEEE Fellow for contributions to wireless and mobile security. He also chaired the 2020 ARO Workshop on Autonomous and Proactive Defenses in Wireless Networks, the 2017 IEEE Conference on Communications and Network Security (CNS), the 2016 ARO Workshop on Trustworthy Human-Centric Social Networking, the 2015 NSF Workshop on Wireless Security, and the 2010 IEEE GLOBECOM Communication and Information System Security Symposium. He is/was on the Editorial Boards of IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON CONTROL OF NETWORK SYSTEMS, and IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY.