Received: 31 May 2019 | Revised: 16 August 2019 | Accepted: 28 August 2019
DOI: 10.1002/cpe.5549

WILEY

RESEARCH ARTICLE

Securing PIN-based authentication in smartwatches with just two gestures

Meriem Guerar[1] | Mauro Migliardi[2] | Francesco Palmieri[3] | Luca Verderame[1] | Alessio Merlo[1]

[1]Department of Informatics, Bioengineering, Robotics and Systems Engineering, University of Genoa, Genoa, Italy
[2]Department of Information Engineering, University of Padua, Padua, Italy
[3]Department of Computer Science, University of Salerno, Salerno, Italy

Correspondence
Francesco Palmieri, Department of Computer Science, University of Salerno, 84084 Salerno, Italy.
Email: fpalmieri@unisa.it

Summary

Smartwatches are becoming increasingly ubiquitous as they offer new capabilities to develop sophisticated applications that make daily life easier and more convenient for consumers. The services provided include applications for mobile payment, ticketing, identification, access control, etc. While this makes modern smartwatches very powerful devices, it also makes them very attractive targets for attackers. Indeed, PINs and Pattern Lock have been widely used in smartwatches for user authentication. However, such authentication methods are not robust against various forms of cybersecurity attacks, such as side channel, phishing, smudge, shoulder surfing, and video-recording attacks. Moreover, the recent adoption of hardware-based solutions, like the Trusted Execution Environment (TEE), can mitigate only partially such problems. Thus, the user's security and privacy are at risk without a strong authentication scheme in place. In this work, we propose 2GesturePIN, a new authentication framework that allows users to authenticate securely to their smartwatches and related sensitive services through solely two gestures. 2GesturePIN leverages the rotating bezel or crown, which are the most intuitive ways to interact with a smartwatch, as a dedicated hardware. 2GesturePIN improves the resilience of the regular PIN authentication method against state-of-the-art cybersecurity attacks while maintaining a high level of usability.

KEYWORDS

authentication, bezel, NFC services, PIN, smartwatch, trustzone

1 | INTRODUCTION

In recent years, the market for smartwatches has been steadily growing. From being merely an extension of a smartphone, these devices became more independent, and store a lot of sensitive information. Indeed, besides health and fitness tracking, smartwatches can be used for making payments as well as for the identification, ticketing, and controlling access to critical services and infrastructures. Furthermore, this trend has been amplified by the introduction of Near-Field Communication (NFC) technology, which allows devices to act as a smart card. In such a scenario, the user can directly authenticate to the smartwatch using a PIN code and then use an e-wallet to unlock a door or pay in a store. Such a solution avoids the burden of carrying multiple cards and keys or look for the smartphone in the pocket each time. Furthermore, while a stolen card can be directly used by the thief to impersonate the user (eg, to access a building or perform a small-value payment), the stolen smartwatch cannot be used unless the thief has the owner's PIN as well. As such, the protection of smartwatches' sensitive data and the user PIN code assumes paramount importance.

Fortunately, the recent introduction of Trusted Execution Environments (TEE) in mobile devices has enabled an improvement in the protection of user's sensitive data, especially from untrusted apps or malware. Thanks to a segregated environment that runs in parallel with the standard operating system, the TEE guarantees that data is stored and processed within an isolated area that malware cannot tamper with. Furthermore, the TEE offers a Trusted User Interface (TUI), which ensures that untrusted apps on the device cannot steal data displayed or typed on the screen. Indeed, TEE has been widely deployed in many smartphones and, recently, it has also been ported on smartwatches. For example, Samsung Gear S2 and S3 contain Knox, a mobile security platform that provides a trusted execution environment based on the ARM TrustZone technology.[1]

Unfortunately, the adoption of TEE comes with a security drawback, namely, the access to sensitive data stored in the TrustZone is the weakest link for the security of NFC transactions.

This is due to the usage of PIN codes for enabling access to sensitive information. Indeed, such a mechanism is vulnerable to several attacks (eg, side channel, phishing, shoulder surfing, and video-recording attacks) and, thus, it can be easily bypassed.[2-4] Therefore, the traditional PIN authentication mechanism is unreliable for security-sensitive operations (eg, payment using Samsung Pay, Apple Pay, and Google Pay) on smartwatches. Unfortunately, the recent adoption of pattern locks as an alternative to PIN does not provide stronger security guarantees, as discussed in the works of Lu et al[2] and Ye et al.[5]

On the other hand, the usage of the TUI does not prevent security issues, like side-channel attacks[6] that affects other components (eg, accelerometer and gyroscopes) interacting with the TEE. Moreover, although the TUI can reserve the screen whenever a trusted application needs it, the smartwatch screen is still shared among all the apps, either trusted or not. As such, the user cannot understand which app is displaying the PIN pad, ie, if the current screen belongs to a trusted app or a malware that mimics the trusted interface.

Besides, the input of the user's PIN code on smartwatches generates usability concerns, as a smartwatch screen is far smaller than a smartphone one. Thus, the design of an authentication method for smartwatches requires to deal with both security and usability concerns.

To mitigate such issues, we proposed in our previous work[7] a preliminary idea of an authentication framework (*2GesturePIN*) able to enhance both the security and the usability of PIN input on smartwatches. In this work, we consolidate and extend such idea by providing both a security and usability analysis, as well as discussing its implementation on real smartwatches and pointing out some actual use cases.

The novelty of 2GesturePIN is to leverage the bezel of the smartwatch as TEE dedicated hardware to input four-PIN digits through two screens of concentric wheels randomly numbered from 0 to 9. The user is required in each step to rotate the inner wheel to match the first and the second pair of her PIN code, respectively.

Thanks to the 2GesturePIN framework, the user can authenticate to the smartwatch with solely two gestures, without the need to tap a small-size touch screen, which is usually uncomfortable and error-prone. Furthermore, 2GesturePIN enhances the security of the PIN input mechanism to a set of state-of-the-art attacks like shoulder surfing, video recording, phishing, and motion-based side-channel attacks.

Hence, the 2GesturePIN framework can improve the security of the authentication mechanism to critical NFC-based operations on smartwatches, ie, to secure access to an NFC e-wallet for ticketing, access control, and payment.

The rest of this paper is structured as follows. Section 2 introduces the 2GesturePIN framework; Section 3 presents the threat model and the security analysis of the 2GesturePIN framework. The benefits of using 2GesturePIN in a smart lock access control use case are described in Section 4, while Section 5 presents the results of a usability experiment that compares 2GesturePIN and the most common methods on Samsung Gear Sport smartwatch. Finally, Section 6 presents related work, while Section 7 concludes this paper.

2 | INTRODUCING THE 2GESTUREPIN FRAMEWORK

The 2GesturePIN authentication framework is a user-centric security solution for smartwatch-based sensitive applications such as payment and access control. 2GesturePIN leverages smartwatches TEE security features and a novel authentication mechanism in order to enhance (1) the usability of the authentication process and (2) the resiliency against a wide range of security threats.

In this section, we will provide (1) a brief description of TEE security features, (2) a description of the 2GesturePIN authentication process from the user's perspective (Section 2.2), (3) a description of the architecture of the 2GesturePIN Framework (Section 2.3) and, finally, (4) a detailed discussion of the whole authentication flow (Section 2.4).

2.1 | Trusted execution environment

GlobalPlatform has standardized first the concept of Trusted Execution Environment (TEE) in the work of Global Platform.[8] A TEE is a separated execution environment that runs alongside the normal operating system, called Rich Execution Environment (REE), and provides security services to that rich environment. The TEE combines both hardware (eg, dedicated storage, dual-mode CPUs) and software (eg, secure kernel, separated drivers) facilities in order to enhance the security of the mobile device.

Moreover, the TEE has dedicated resources that are accessible only to the TEE (eg, secure storage and biometric sensors) along with other resources shared with the REE (eg, screen and sensors) which are locked by the TEE when a trusted application wants to use them. Hence, communication between the shared resources and the TEE is secure and confidential. Each TEE holds its own cryptographic resources, ie, its private key and certificate, hardwired in read-only memory.

Vasudevan et al[9] summarized the security requirements that a TEE has to fulfill.

- *Isolated Execution*, ensuring that applications that reside in the TEE (called Trusted Applications - TA) execute completely isolated from and unhindered by any other application.
- Secure Storage, which protects persistently stored data (eg, cryptographic keys).
- *Remote Attestation*, enabling remote parties to ascertain that they are dealing with a given trusted application on a specific TEE-device.

FIGURE 1 2GesturePIN user interface

- *Secure Provisioning*, which enables communication by remote parties with a specific trusted application, thereby protecting the integrity and confidentiality of transmitted data.
- *Trusted Path*, ie, a channel allowing (1) the user to send data to the TEE and (2) the TEE to send back data to the user. The Trusted Path protects against eavesdropping and tampering.

The most used implementation of TEE for mobile devices, called TrustZone, has been deployed by ARM[1] for the ARM Cortex Processor family. Other TEE implementations include Intel Trusted Execution Technology[10] and Texas Instrument MShield.[11]

2.2 | 2GesturePIN user authentication

As opposed to a traditional PIN-input interface that uses a digital keypad, the 2GesturePIN UI consists of two concentric wheels with ten equally sized sectors, as shown in Figure 1. The outer wheel contains numbers from 0 to 9 in a fixed order, while the inner one is numbered randomly from 0 to 9 at each session and step. Furthermore, the inner wheel can rotate according to a TEE dedicated hardware movement, eg, the movement of bezel or crown.

In order to authenticate, the user is required to rotate the inner wheel so that the first digit of her PIN matches the second one. In the same way, in the second step of the authentication process, the user rotates again a new randomly generated inner wheel to match the third PIN digit with the fourth PIN digit. Therefore, thanks to the 2GesturePIN authentication method, the user can input her four-PIN digits with solely two simple gestures (ie, rotating the wheel through the bezel) instead of typing it in small-sized touch screen.

An example of an authentication process using the 2GesturePIN Framework is shown in Figure 2 in the hypothesis that the user's PIN code is "7340."

In the first gesture, the user uses the bezel to rotate the first PIN digit (7) in the inner wheel to match the second PIN digit (3) of the outer wheel. Then, 2GesturePIN computes a second screen with a new inner wheel with a random arrangement of the numbers. In the second gesture, the same process is repeated with the third and fourth PIN digits, so the user rotates the number 4 to match number 0.

2.3 | 2GesturePIN architecture

The 2GesturePIN Framework is composed by two main modules, ie, the *AuthLibrary* and the *AuthEngine* as depicted in Figure 3.

The **AuthLibrary** is the bridge for the authorization procedure between the REE application and the *AuthEngine* embedded in the corresponding Trusted Application. The *AuthLibrary* provides a comprehensive set of APIs to (1) perform authentication requests (step 1 in Figure 3), (2) render

FIGURE 2 2GesturePIN authentication method

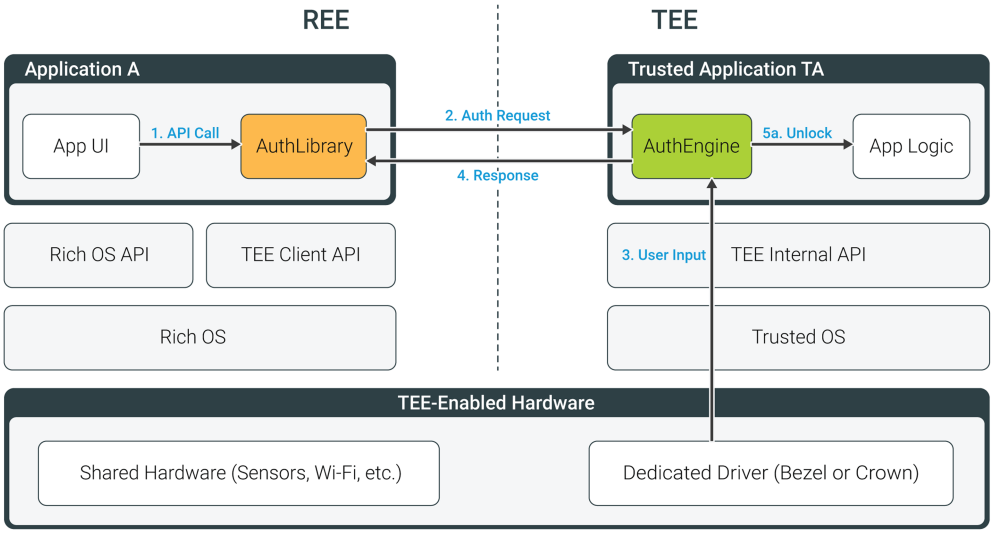

FIGURE 3 2GesturePIN framework architecture

the authentication interface according to the inputs provided by the *AuthEngine*, and (3) notify the user about successful/denied authentications (step 4 in Figure 3).

The **AuthEngine** contains the core of the 2GesturePIN authentication process. The engine is embedded in the TA that resides in the TEE of the smartwatch and is entirely separated from the REE. The only interaction with the engine is by means of the *AuthLibrary*.

The core functionalities of the *AuthEngine* are as follows.

- *2GesturePIN Authentication Management*. The engine enforces the authentication flow described in Section 2.2, manages the session's lifecycle, and is able to unlock the TA in case of successful authentication (step 5a in Figure 3). For a detailed description of the authentication steps, please refer to Section 2.4.
- *Trusted Input Management*. The *AuthEngine* handles the Trusted Path to the dedicated hardware, eg, the bezel of the smartwatch, to obtain the user input (step 3 in Figure 3). In this paper, we leverage the bezel as the preferred input method, but the rotating crown can be used as well if the smartwatch is not enabled with a rotating bezel.
- *Storage of User Credentials*. The *AuthEngine* stores the credential information of the user (eg, the PIN or the access token). This functionality relies on the secure storage features of the TEE, sealing the information with a secret key K_a derived from the device-specific private key and the ID of the application, computed using the hash of its source code. To compute hash values, device D relies on embedded TrustZone crypto libraries, using a 512-bit SHA-2 algorithm.[12] Since K_a is per-application and per-device, the *AuthEngine* ensures that each stored information will only be accessible to the corresponding application for the specific device.

In order to benefit from 2GesturePIN's features, an application needs to import the *AuthLibrary* inside the REE Application and the *AuthEngine* in the corresponding TA Application. The choice of developing the 2GesturePIN Framework as a set of libraries allows the seamless integration into existing applications without requiring OS modifications or bending of the application's logic.

2.4 | 2GesturePIN authentication flow

Figure 4 depicts the authentication process implemented in the 2GesturePIN framework. For this description, we selected the smartwatch bezel as the dedicated hardware to exploit the trusted path to the TEE. Furthermore, the user (U) has installed on her smartwatch an application that requires the 2GesturePIN authentication mechanism. The app is composed by the Application UI (A) that resides in the REE and contains the 2GesturePIN Library (AuthLib), and the corresponding trusted part (TA), placed in the TEE, with the 2GesturePIN Engine embedded inside (AuthEngine).

Once U taps on the application A icon (step 1), an authentication request is sent to the AuthLib (step 2) and then dispatched to the *AuthEngine* (step 3).

At this point, the *AuthEngine* computes the first random sequence of the wheel's numbers that will be used to retrieve the first pair of digits of the user's PIN (step 4). The sequence is then sent to AuthLib that renders the interface, as showed in Figure 2, and displayed it to U (steps 5 and 6).

Once the authentication screen is displayed, U rotates the bezel of the smartwatch in order to match the first pair of digits of her PIN. The Bezel driver (BezelD), which resides in the TEE, sends a rotary event to the *AuthEngine* at each detent point detected during the bezel rotation (steps 7 and 8).

The *AuthEngine* starts a timer with a predefined period of time (eg, 700 ms) upon receiving each event. The user's input is confirmed when this period is up. The *AuthEngine* computes the second random sequence of the wheel's numbers (step 9) necessary for the input of the second pair

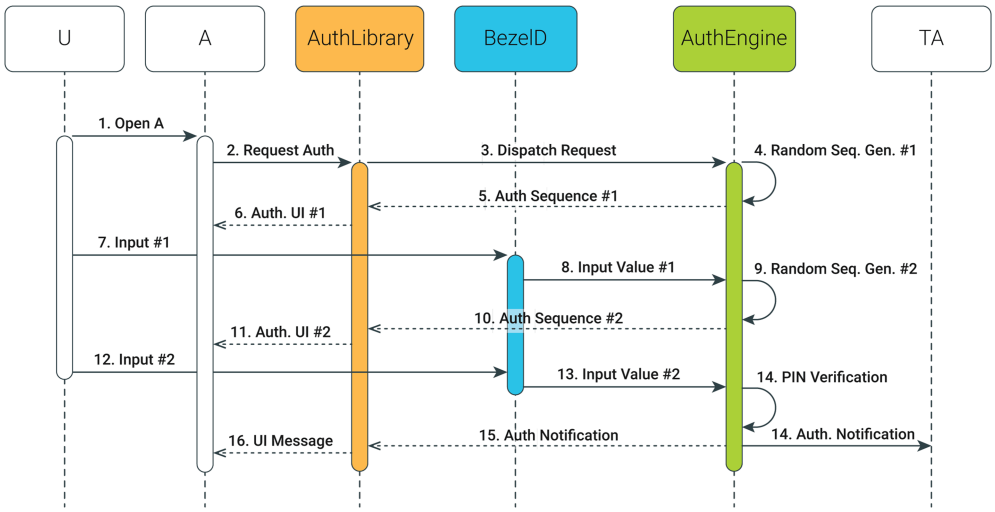

FIGURE 4 2GesturePIN authentication flow

of PIN digits (steps 10 to 13). After the second interaction with U, the *AuthEngine* is able to check if the input provided matches the User's PIN (step 14).

At the end of the process, a notification, either success or failure, is sent to the TA (step 14) and to the user through the smartwatch interface (steps 15 and 16).

3 | THREAT MODEL AND SECURITY ANALYSIS

3.1 | Assumptions and threat model

In this paper, we suggest the 2GesturePIN framework for smartwatches, which enables user-friendly authentication between end-users and trusted applications that reside in the TEE of those devices.

We assume that 2GesturePIN framework leverages the same TEE security features (eg, Trusted User Interface) used currently by sensitive applications to protect PIN entry. As discussed in the previous sections, despite the TUI may protect the PIN entry from the recording of the screen or touch events, the PIN code or Pattern lock is still vulnerable to the following attacks.

Side Channel attacks. Many recent studies consider smartwatches as a side-channel to infer secrets entered on external devices such as smartphones,[13,14] and ATMs.[15,16] However, they are not taking into account the smartwatch as a venue for authentication. On the contrary, in the work of Lu et al,[2] authors showed the feasibility of inferring the user's password entered into the smartwatch itself through the touch screen. Since the malware is not able to record the screen and touch events during the user's PIN input, another kind of side-channel attacks has emerged that leverage other shared resources between the REE and the TEE (eg, accelerometer,[6] the camera and the microphone,[17] and the Gyroscope[18,19]). The idea behind this attack is that the device's micro-movements caused by the user's tap on the touchscreen are quite different depending on the tap' location. Therefore, using a trusted path to the screen is not enough to protect the regular PIN input against these attacks. In this work, we assume that the user installs on her smartwatch "Snoopy,"[2] a fitness or gaming app, which is, in fact, a Trojan app that eavesdrops motion data when users type or swipe their passwords on smartwatches. Snoopy periodically upload the extracted motion data to the cloud, where it leverages deep neural networks trained with crowdsourced data to infer the user's passwords. It is worth noticing that such kind of complex attacks on mobile are also hard to detect with static and dynamic analysis techniques at the state of the art (see, eg, the works of Armando et al[20,21] for some references).

Phishing attacks. An open issue shared by many TEEs implementations, including ARM TrustZone, is how the users can be sure that they are dealing with the trusted application when both trusted and untrusted applications share the same display.[22,23] It is considered a severe problem because nothing prevents the malicious application in the REE from displaying a UI with content similar to a trusted application (eg, Payment app) to trick the user into typing her PIN or Pattern.

To mitigate this attack, the TEE has to provide local attestation. Local attestation should enable the user to check whether the UI displayed on the screen is actually from a trusted application in the TEE or from some phishing app in the REE. One potential solution is to use a LED as secure hardware to indicate to the user that the device is operating in trusted mode.[24] However, unlike smartphones, the current smartwatches on the market are not equipped with a LED. Therefore, implementing this mechanism on smartwatches requires additional hardware. To show how our security mechanism deals with this issue, we assume that the smartwatch is infected by a malicious application that pretends to be 2GesturePIN.

Guessing attacks. To evaluate the security of 2GesturePIN against Guessing attack, we assume that an attacker can physically access the smartwatch and thus try to guess the correct four-PIN digits to gain access.

Shoulder surfing and recording attacks. Performing authentication using the PIN or Pattern Lock in public environments (ie, coffee shop, library, bus, and class) exposes the user to shoulder surfing attack. Usually, if an attacker observes the user over her shoulder when she is entering her PIN or Pattern, she would be able to reveal the user's secret from the first attempt because both of these methods use direct input.

In order to evaluate the security of 2GesturePIN against this attack, we assume that the user types her PIN code using 2GesturePIN in a public space where the risk of being observed by someone is high. In addition, we assume that the shoulder surfer is reasonably close to the user, and she can observe the entire authentication session.

In the video-recording attack scenario, we assume that an attacker records the entire 2GesturePIN authentication session using her smartphone camera to watch it later and to try identifying the user's PIN. On the other hand, since the surveillance cameras are available in a growing number of public and private spaces, we assume that an attacker may obtain multiple recordings of the same user during the authentication from surveillance cameras. Yet, in the latter scenario, we consider that the surveillance cameras may capture only the user hand during the input while it is extremely unlikely to have a clear image of the content displayed on the screen due to the small size of the smartwatch screen, the lighting, and the position of the cameras. Therefore, the attacker has to try reconstructing the PIN from the video recordings of the user's hand without the content displayed on the screen.[25]

3.2 | 2GesturePIN security analysis

In the following section, we discuss how 2GesturePIN provides enhanced security against the attacks mentioned earlier by exploiting a non-invasive configuration of the TEE, to ensure a trusted path to the bezel, and the novel authentication mechanism.

Side channel attack

As described above, smartwatches could be infected by a malicious software masked as a legitimate application (eg, fitness or gaming app) that leverages motion sensors as side-channel to infer the PIN typed on the touch screen. 2GesturePIN provides adequate protection against this kind of attack by using the bezel rather than the touch screen as a new way of PIN input. Besides, the bezel driver resides in the TEE, which prevents malicious software from reading the bezel input. At present, there is not any work in the literature that aims at detecting the bezel movement through sensors as side-channel. However, even if a methodology had been devised, 2GesturePIN is designed in such a way that both the correct PIN digits and all the other nine numbers on the wheel turn the same degree. Furthermore, the rotation degree is different at each step and each authentication session because of the randomization of the wheel numbers position. Hence, the 2GesturePIN framework does not reveal any useful information related to the PIN digits, and thus, it is resilient to any state-of-the-art sensor-based side-channel attacks.

Although stealing the PIN through smartwatch when the users authenticate to their smartphones[13,14] or ATM machines[15,16] is out the scope of this paper, we wanted to mention that using 2GesturePIN on these devices provides protection against this category of side-channel attacks as well. As shown in Figure 5, to authenticate the user to her smartphone or ATM account using 2GesturePIN, the user has to slide the seek bar to rotate the Wheel. However, thanks to the randomization of the wheel numbers position, the smartwatch could not reveal any useful information about the PIN since the user input is always different. The same thing holds for smart door lock, ie, using 2GesturePIN instead of keypad lock provides enhanced security against this category of side-channel attacks. Indeed, the degree of rotation of the knob is different for each authentication session.

Phishing attack

In a phishing attack, a malicious application in the REE can pretend to be 2GesturePIN to trick the user into entering her PIN digits. However, due to the sharing of the screen between REE and TEE, the user cannot distinguish between benign and fake UIs. However, 2GesturePIN framework helps the user to detect malware UIs by noticing that the wheel is not moving according to the bezel movement or not moving at all. Indeed, since 2GesturePIN uses the bezel as dedicated hardware, only trusted apps can collect data of the bezel rotation. Hence, the phishing app cannot trick the user by pretending to be a trusted application that requires 2GesturePIN authentication. Therefore, 2GesturePIN is resilient to phishing attacks.

FIGURE 5 2GesturePIN: use cases

Brute force attack

A Brute Force Attack is a password cracking method that uses an automated process to try all possible character combinations until the password is found. It is important to indicate that we did not mention this attack in the threat model because using a trusted path to screen protects the regular PIN from this attack. However, 2GesturePIN uses a different way of input (ie, the bezel). Therefore, we added this section in the security analysis to highlight how 2GesturePIN resists this attack. By using the bezel as dedicated hardware only accessible to the TEE, 2GesturePIN framework can prevent any automated process from performing a brute force attack. Indeed, malicious codes in the REE are unable to physically rotate the bezel required for the PIN input or simulate the bezel movement by manipulating the bezel driver.

Guessing attack

The chance that a guessing attack succeeds depends on the size of the password space. Since 2GesturePIN uses a standard four-PIN code, there are 100 possible inputs for each step, ie, the combination of the two wheels with ten numbers each. Hence, the number of possible password combination is 10000. Similar to the regular PIN, the probability of guessing the user's PIN successfully in three attempts using 2GesturePIN is 0.0003.

Shoulder surfing attack

To enhance the security of the PIN method against shoulder surfing attack, 2GesturePIN provides an indirect input of the user's PIN through the smartwatch bezel. During the authentication, the user is required to rotate the bezel to match two PIN digits for each of the two steps. However, turning the wheel in fact match two sets of ten numbers including the two PIN digits. Therefore, there is not an indication of which combination among them represents the correct part of the four-PIN digits. Besides, the randomization of the wheel numbers prevents shoulder surfer from successfully input the PIN by turning the wheel at the same observed position in the previous session without the knowledge of the PIN digits. Hence, observing the user during 2GesturePIN authentication process is not enough to reveal the PIN digits. Thus, 2GesturePIN enhances the security of the regular PIN against shoulder surfing attack.

Video-recording attack

An even more serious threat than looking over someone shoulder is using an external recording device such as a smartphone to record the entire authentication session. However, in a real-world scenario, it is not always easy to perform such an attack without the user's awareness. An attacker may prefer to use the surveillance cameras to avoid attracting the user's attention and to try reconstructing the PIN through the spatiotemporal dynamics of the user's hand.[25] Using the surveillance cameras, an attacker can easily get multiple recordings of the same user without her awareness; for instance, a user that buys a coffee using her smartwatch every morning from the closest coffee shop to her office equipped with a surveillance camera is susceptible to such an attack.

2GesturePIN provides resilience to both single and multiple recording attacks using the surveillance cameras because the user input is different from each authentication thanks to the randomization of the wheel numbers positions. Thus, an attacker could not reveal any useful information about the user's PIN by recording the user's fingertip or hand movements one or multiple times during the authentication, in contrast to the regular PIN and Pattern lock where the position of numbers on the keypad and the position of dots in the screen are known.[25,26] On the contrary, if an attacker records the entire authentication session using her smartphone, the best recording of 2GesturePIN only reveals two sets of ten combinations of numbers. Using such a piece of information, the attacker can only derive a list of 100 possible combinations, including the correct four-PIN digits. Hence, the probability that an attacker successfully guess the user's PIN is 0.01. It worth noticing that 2GesturePIN increase the resiliency against single video-recording attacks w.r.t. regular PIN and pattern lock, where, in the same conditions, the recording will reveal the user's PIN or the user's pattern with certainty. An attacker could reveal the user's PIN through an intersection analysis if she was able to record both the user's input and the screen content of the same user multiple times without her awareness. However, we consider such an attack very unlikely in a real-world scenario.

3.3 | Comparison of 2GesturePIN security strength to regular PIN

Since the standard PIN authentication method is the most predominant method of user authentication and both this method and 2GesturePIN use a four-PIN digits as a secret, it is essential to highlight why using 2GesturePIN to protect sensitive applications on a smartwatch is better than the regular PIN.

As mentioned above, TEE provides a Trusted UI to protect the regular PIN against brute force, touch-logger, and spyware-based screen recording attacks. Nevertheless, it still vulnerable to side-channel, phishing, shoulder surfing, and video-recording attacks which motivates us to suggest 2GesturePIN as an alternative.

As shown in Table 1, 2GesturePIN is resilient against phishing, side-channel, and shoulder surfing attacks. Moreover, it is able to increase the level of security against a perfect video-recording attack of the entire authentication session comparing to the regular PIN. Indeed, the probability that the attacker reveal the PIN successfully with 2GesturePIN is 0.01 instead of 1 using the regular PIN. Besides, 2GesturePIN is resilient to both single and multiple video-recording attacks through surveillance cameras which record only the user's input. Finally, it provides the same security level as the regular PIN against brute force and guessing attacks.

TABLE 1 Comparison of the security strength

	2GesturePIN	The regular PIN
Brute force attacks	Strong	Strong
Phishing attacks	Strong	Weak
Side-channel attacks	Strong	Weak
Shoulder surfing attacks	Strong	Weak
Single Video-recording attacks (User's input and screen)	Medium	Weak
Single and Multiple Video-recording attacks (User's input)	Strong	Weak
Guessing attacks	Same as PIN	PIN Strength

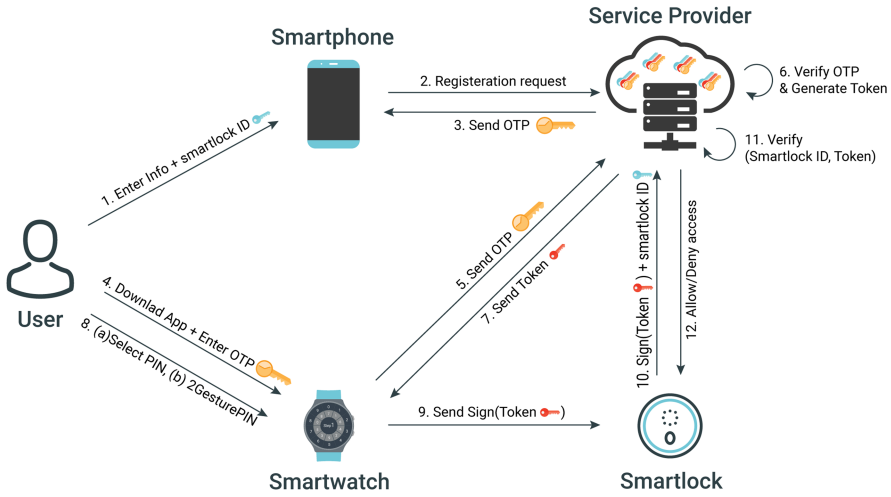

FIGURE 6 2GesturePIN: access through smart locks

Hence, 2GesturePIN is a good alternative to regular PIN for protecting the sensitive services on a smartwatch.

4 | USE CASE SCENARIO: SMART LOCK ACCESS CONTROL

Nowadays, NFC-enabled smartwatches are capable of replacing all users smart cards that they have on their wallet as well as their physical keys. In addition, they provide numerous advantages comparing to both smartphones and smart cards, especially in potentially crowded places such as in public transportation or hospitals. For instance, users can save time by using their smartwatch rather than looking for their smartphones or keys on the pockets. On the other hand, using smartwatches enable a two-factor authentication comparing to smart cards (ie, access control and small-amount payment use cases) and physical keys. As an example, if someone lost her car key, such a key can be used directly to gain unauthorized access to the car, while in the case of the smartwatch, the access to the key is further protected by PIN code.

Access control systems are promising use cases of NFC technology. Therefore, we consider the case of using the smartwatch as a key to unlock any door's smart lock (eg, home, office, car, etc). While this makes these devices very attractive by facilitating the user's daily life, it also exposes them to high-security risk. This is why, usually, access permission occurs after touching the smartwatch to an NFC reader and authentication of the user. Unlike the PIN and Pattern lock, 2GesturePIN provides an enhanced level of security against common attacks and a user-friendly authentication method for sensitive services. The rest of this section will present how the 2GesturePIN framework can be used to unlock a door's smart lock. First, the user installs a smart lock of a specific service provider. Then, she registers herself the first time to the SP portal and enrolls the smartwatch through installing a TA of the same SP, which offers 2GesturePIN framework. After a successful login session using 2GesturePIN, the TA allows the user to lock or unlock the door. Figure 6 depicts the overall scenario.

4.1 | Smartwatch enrollment

To initiate the registration process, the user visits the website of the service provider (SP) through her smartphone or personal computer and starts the registration process. During this phase, the user enters her information (eg, name, surname, phone number) as well as the smart lock ID (step 1).

SP receives the data (step 2) and transmits to the user a security code (eg, a One-Time Password) that allows the enrollment and initialization of the Smartwatch application (step 3).

The user downloads the wearable application that contains the 2GesturePIN Framework from SP on her smartwatch (D) and enters the OTP using the same input mechanism of 2GesturePIN described in Section 2.4 (step 4).

The wearable application sends the OTP to the SP (step 5). If the OTP verification succeeds, SP generates the Access Token (*Token*) for the user (step 6) and sends it to the smartwatch (step 7). Although we do not need specific assumptions on the token generation procedure, we suggest the usage of Random Number Generator algorithms for creating token values as described in the work of Ozdenizci et al.[27]

The user then selects a four-digit PIN in 2GesturePIN settings to protect the access to this token (step 8(a)). The PIN initialization is performed as follows: the user uses the first number of the OTP as an indicator to input all the PIN digits sequentially. Thus, in contrast to the authentication, the PIN initialization requires four gestures.

For this protocol, the smartwatch and the SP need to rely on a secure communication channel. To do so, the *AuthEngine* implements the enrollment protocol defined in the work of Armando et al[28] that allows mutual authentication between the server and the device and secrecy of the exchanged payload. At the end of this registration process, the User's PIN, the first number of OTP, and the Access Token are securely stored in the TEE by the *AuthEngine*.

4.2 | Smart lock authentication

After the enrollment phase of Section 4.1, the user authenticates using 2GesturePIN as described in Section 2.4 (step 8(b)). Now, the smartwatch is ready to be used to unlock the desired smart locks.

When the user taps her smartwatch on the smart lock, the TA, using the *AuthEngine* library, computes the access proof for the smart lock in the form of

$Msg_{auth} : Enc_{SP}\{Cert_D, Sgn_D\{U, D, SL, Token, T_d\}\}$.

The entire message, encrypted with the public key of *SP*, is composed by the certificate of the smartwatch ($Cert_D$) and a payload signed with the private key of the smartwatch (Sgn_D). The payload generated by the device is composed by the access token (*Token*), the identities of the user (*U*), the device (*D*), and the smart lock (*SL*) along with a timestamp T_d in order to ensure message freshness and resiliency against replay attacks. Since the certificate of the device is included in the encrypted part of the message, this exchange is resistant to Public Key Substitution UKS attack, described in the work of Blake-Wilson and Menezes.[29]

Msg_{auth} is then sent to the smart lock through the NFC interface of the device (step 9). The smart lock forwards Msg_{auth} as well as its ID to the service provider (step 10). SP decrypts the message and checks the signature of the smartwatch, the freshness of the message, and whether the specific device *D*, associated with the user *U* is authorized to access the smart lock *SL* with that specific *Token* (step 11).

Finally, the SP sends an allow or deny access response to the smart lock (step 12) according to the verification process. In the case of the session expiration, the smartwatch prompts the user a new request for authentication (step 8b).

5 | 2GESTUREPIN USABILITY EXPERIMENT

One potential implementation of 2GesturePIN framework is through SierraTEE,[30] which provides an open-source implementation of a TEE environment compatible with Global Platform standards and ARM TrustZone. However, the actual presence of the TEE that is required to make the solution secure is meaningless from the usability point of view. Therefore, for usability test purpose, we implemented the 2GesturePIN on Tizen 4.0.0.4 with no dependencies to the TEE. We developed the application using Tizen Studio 3.3. The test equipment consisted of the Samsung Gear Sport smartwatch from Samsung, which is based on Tizen 4.0.0.4 and fitted out with a hardware rotating bezel, a 360x360 pixels screen, a Dual-core 1.0 GHz, 768 MB RAM, and 4 GB internal memory.

As shown in Figure 7, 2GesturePIN app has two modes, ie, Training and Test. The Training mode allows participants to get familiar with the concept and try the authentication method. The Test mode, instead, can be used for usability tests, in which we record the input time and the error rate on ten consecutive attempts.

The application also features a settings button where users can update the PIN code, and enable or disable the usage of colors. Such a specific option displays different colors on the wheel portions where each digit always has the same color. We noticed that using fixed colors on the wheel helps the users to locate numbers and therefore decrease the overall authentication time.

FIGURE 7 2GesturePIN on Samsung Gear Sport

TABLE 2 Comparison of 2GesturePIN with the common authentication methods

Authentication method	Authentication time (s)	Error Rate (%)
The regular PIN	2.37	5
Pattern Lock	1.23	4
Bezel-based PIN	10.25	0
2GesturePIN	2.95	0

5.1 | Usability experiment results

The study was conducted on 10 participants, with an average age of 33 years (range: 20-53). In the beginning, participants were provided with a detailed explanation of the framework and its usage. After the introduction, each user was asked to set a four-digit PIN code, and then she was encouraged to practice the unlocking until she felt familiar with the concept. Afterward, users were required to enter their PIN digits for ten times using the test mode. Thus, we were able to track 100 authentication sessions performed by 10 participants. The authentication time was measured from the moment the participant starts rotating the bezel to match the two first PIN digits to the validation of the user's input in the second gesture.

The same procedure was followed to evaluate the usability of the regular PIN, Android Pattern lock, and the bezel-based PIN authentication method provided by the last version of Tizen OS.

Table 2 shows the average authentication time and error rate of 2GesturePIN along with the other methods. Test results show that Pattern Lock has the fastest authentication time since most of the participants select a simple pattern. 2GesturePIN had approximately the same authentication time as the regular PIN, while it is about three times faster than bezel-based PIN authentication.

We argue that the main reason is that 2GesturePIN requires two gestures instead of four and uses a shorter time for the number confirmation. In terms of error rate, 2GesturePIN and Bezel-based PIN authentication methods have the lowest error rate among the evaluated methods, and this is due to the use of the bezel as a mean of input rather than the touch screen. Hence, 2GesturePIN is a usable alternative for PIN input on a smartwatch.

6 | RELATED WORK

Protecting any kind of information from unauthorized access has been a subject of security research for many years and numerous schemes have been developed over time to secure strategic data and transactions from several menaces, ranging from insider threats to common attacks to authentication mechanisms and infrastructures.[31,32] The diffusion of new smart devices is creating new opportunities and opening new perspectives for developing new enhanced and more usable authentication schemes, which will be better capable to face these challenges.

In particular, authentication on smartwatches is mainly used to set a lock screen to prevent unauthorized access to the device, and it is usually disabled by default. It is, however, required if the user wants to take advantage of mobile payment systems (eg, Apple Pay, Google Pay or Samsung Pay) or to control the access to critical services and infrastructures (eg, smart home Locker, smart cars locker, smart health services, and infrastructures) in order to further tighten security around these systems. Although the known security and usability issues of the regular PIN and Pattern lock on smartwatches,[2,33] they are the predominant types of authentication mechanism today.

6.1 | Behavioral-biometrics-based authentication methods

Biometrics is often considered the silver bullet for authentication and several biometrics-based schemes have been developed to secure important information and strategic data.[34] This last motivated many researchers to take advantage of smartwatches rich-sensing capabilities to design behavioral-biometrics-based authentication methods as an alternative.

For instance, Yang et al[35] and Lewis et al[36] designed a motion-based authentication method able to authenticate a user by performing a gesture with a wrist-worn device or smartwatch, after building the user's behavioral profile by collecting data from device sensors. A similar approach has been taken by SnapAuth,[37] where the authentication is performed by a finger-snapping gesture. This system achieved 82.34% True Acceptance Rate (TAR) at 34.12% False Accept Rate (FAR) using one-class MLP as the classifier, on very limited training samples (ie, 15). Johnston and Weiss[38] studied the feasibility of using smartwatches for gait-based biometrics. Their study shows that gait is not sufficient to be used as a sole means of identification of individuals; instead, it is seen as a potentially valuable component in a multimodal biometric system. Authors have also pointed out some limitations of their work; for instance, user's data were collected the same day, thus not representing a real-world scenario, and their preliminary works showed that results degrade significantly when data are collected on different days. A gait-based approach for continuous authentication has been investigated by Al-Naffakh et al.[39] Authors pointed out that gait recognition is highly efficient and recommended to authenticate users in a transparent and continuous manner. Although such results are positive, the gait recognition, and authentication were performed only in a controlled environment; thus, results may differ in a real-life scenario.

None of these studies discuss the case where the user is not recognized by her walk or is not walking. A different approach has been taken by Nguyen and Memon[40]; TapMeIn let the user authenticate by tapping a specific rhythm on the smartwatch's touchscreen. Results are significantly promising, with an accuracy of 98.7%. However, tests were performed in a lab with a limited dataset, which may favor the classification process.

All works presented above are related to the specific behavior of a human while performing some tasks, such as hand movement, gait, and rhythmic tapping, which may present some limitations.[41] For example, multiple users may have the same hand-waving patterns. Wearing an outfit, such as a trench coat or a footwear, may change a person's walking style and person's typing behavior changes considerably throughout a day with different states of mind such as excited or tired. Indeed, these limitations related to human behavior nature might be the main barriers to the adoption of solutions that rely on a behavioral system solely.

6.2 | Knowledge-based authentication methods

Other researchers instead worked on increasing security around the currently most used authentication methods, ie, PIN and pattern lock. Research has focused on smartphones,[42-48] ATM,[49-52] and, recently, smartwatches, where the small screen size introduces usability concerns.

A novel PIN-based authentication method is Personal Identification Chord (PIC),[53] where the user can enter ten different inputs using only four big on-screen buttons. The recall study shows that both PIN and PIC achieve high recall rates and input accuracy, although the usability study shows that the PIC is slightly slower and more error-prone than the PIN-based authentication method. Furthermore, PIC is not resilient to side-channel and shoulder-surfing attacks. In the work of Nguyen et al,[54] the authors introduced a two-factor authentication method, called Draw-a-PIN. To authenticate, the user is required to draw her PIN digits sequentially on the touchscreen instead of typing it. Besides the correctness of the PIN, Draw-a-PIN uses the drawing behavior of the user as an additional security layer. While Draw-a-PIN provides some advantages for shoulder-surfing resilience, the usability study of its implementation on a smartwatch[33] showed that it is not usable to unlock the smartwatch due to its high error rate and long authentication time (ie, Overall Average Error rate 20.65%, Overall Average authentication time 7356 ms). An approach similar to TapMeIn[40] is Beat-PIN,[55] where a PIN is represented by a sequence of beats recorded when the user taps on the smartwatch touchscreen (ie, a beat is the time between the instance the user touches the screen and the instance the user lifts her finger from the screen). However, Beat-PIN does not use the user's typing behavior, and thus, it is less robust against shoulder surfing attack. Beat-PIN achieved an Equal Error Rate of 7.2% with an authentication time of 1.7 seconds. A sensors-based authentication is given by Yoon et al.[56] The variations of the light's values read by the ambient light sensor are used to build sequences representing particular User Interface (UI) events, such as single-click, double-click, and 1-sec-hold. These events are used to enter the PIN (eg, a three events PIN could be single-click 1-sec-hold single-click). Besides its vulnerability to brute force and side-channel attacks, this method is not usable because the input of solely a three-event long PIN requires approximately between 9 and 10 seconds. In addition, using the ambient light sensor for the PIN input makes the input impossible in a dark environment.

Similar to 2GesturePIN, VibraInput[57] and DialA[58] utilize two concentric wheels with ten equally sized sectors as a user interface for PIN authentication on smartphones. However, in contrast to 2GesturePIN, the outer wheel in DialA contains ten different letters and in VibraInput contains four repetitive letters, while each letter represents a vibration pattern. In addition to the four-PIN digits, the user has to remember four vibrations pattern and their corresponding letters. When the user touches the screen, the vibration starts, and the user has to remember the letter that corresponds to this vibration pattern in order to use it as an indicator to input the PIN digit. The vibration stops as soon as the user left her finger. Since the outer wheel contains multiple occurrences of this indicator, another round is required to identify the PIN digit. Thus, besides the overhead of memorizing an additional secret, VibraInput requires eight gestures to input four PIN digits. In DialA,[58] the user has to use the letter that she heard through earphones as an indicator to input the four-digit PIN. The rotation and commitment are conducted via another small scroll wheel at the bottom of the smartphone screen in order to prevent direct input. However, this method is not suitable for smartwatches for two reasons. First, the user is required to wear earphones and connect it to the smartwatch through Bluetooth (ie, if it is not connected), which is impractical and take a long time. Second, because using another small wheel is not suitable for the small-size screen such as smartwatches and rotating the wheel directly makes the user susceptible to side-channel, shoulder surfing, and video-recording attacks. Besides, unlike 2GesturePIN, DialA requires four gestures. Recently, Samsung has been rolling out Tizen 4 to the Samsung Gear S3 Classic, Gear S3 Frontier, and Gear Sport smartwatches. This software update allows the PIN input through the bezel.[59] Initially, a blue indicator is in the number zero. To login, the user has to move the indicator to the PIN digits sequentially. The number will be selected if the user did not move the bezel for a certain amount of time illustrated visually by changing the number background color to blue progressively. However, unlike 2GesturePIN, this method is vulnerable to shoulder surfing, video-recording, and phishing attacks. This is due to the fact that it utilizes a visual indication to select the PIN digits and the access to the bezel is not dedicated only to the trusted applications. Moreover, it requires four gestures to input the PIN digits rather than two, which leads to a long authentication time.

Hence, 2GesturePIN represents a notable enhancement both in terms of security and usability in comparison to these methods.

7 | CONCLUSION

Nowadays, wearable devices are becoming very popular due to the facilities provided by the NFC technology, which enables smartwatches to host access control, ticketing, and payments services. However, the user authentication to the wearable devices is by far the weakest link in the security of those services. In fact, due to the weakness of existing PIN authentication mechanism to security threats, eg, phishing or spyware based-recording attacks, malware could easily bypass the user authentication and exploit the ticketing or the payment service of the

user. Unluckily, the introduction of the TEE and Trusted User Interfaces is not sufficient to adequately protect the critical smartwatch services. Furthermore, the usage of a small screen to input a PIN code is usually error-prone and uncomfortable.

To mitigate those security and usability issues, this paper introduced 2GesturePIN Framework, a novel PIN-based authentication method for smartwatches. 2GesturePIN leverages the bezel of the device to input four-PIN digits through two rotation gestures. The security analysis showed that the proposed scheme is resilient against brute force, phishing attacks, side channel, shoulder surfing, and video-recording attacks. From a usability point-of-view, the results of our experiment show that 2GesturePIN is a usable alternative for PIN input on smartwatches. Moreover, this paper explains the benefits of the 2GesturePIN framework in an NFC access control scenario.

In future works, we intend to (1) refine the prototype implementation, (2) extend the 2GesturePIN framework to other OSes, and (3) further evaluate the usability of the framework performing a study with a statistically significant number of participants.

ORCID

Francesco Palmieri https://orcid.org/0000-0003-1760-5527

REFERENCES

1. ARM Technologies. *Arm Security Technology: Building a Secure System Using TrustZone Technology*. Technical Report. Cambridge, UK: ARM Limited; 2005. http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf
2. Lu CX, Du B, Wen H, et al. Snoopy: sniffing your smartwatch passwords via deep sequence learning. *Proc ACM Interact Mob Wearable Ubiquitous Technol*. 2018;1(4). Article No. 152.
3. Brandon A, Trimarchi M. Trusted display and input using screen overlays. In: Proceedings of the 2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig); 2017; Cancun, Mexico.
4. Khan H, Hengartner U, Vogel D. Evaluating attack and defense strategies for smartphone pin shoulder surfing. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18); 2018; Montreal, Canada.
5. Ye G, Tang Z, Fang D, et al. A video-based attack for Android pattern lock. *ACM Trans Priv Secur* 2018;21(4). Article No. 19.
6. Owusu E, Han J, Das S, Perrig A, Zhang J. ACCessory: password inference using accelerometers on smartphones. In: Proceedings of the 12th Workshop on Mobile Computing Systems & Applications (HotMobile '12); 2012; San Diego, CA.
7. Guerar M, Verderame L, Migliardi M, Merlo A. 2GesturePIN: securing pin-based authentication on smartwatches. In: Proceedings of the 28th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2019); 2019; Napoli, Italy.
8. Global Platform. The trusted execution environment: delivering enhanced security at a lower cost to the mobile market. White paper. February 2011.
9. Vasudevan A, Owusu E, Zhou Z, Newsome J, McCune JM. Trustworthy execution on mobile devices: What security properties can my mobile platform give me? In: *Trust and Trustworthy Computing: 5th International Conference, TRUST 2012, Vienna, Austria, June 13-15, 2012. Proceedings*. Berlin, Germany: Springer; 2012:159-178.
10. Intel. *Intel Trusted Execution Technology*. Technical Report. Santa Clara, CA: Intel; 2012. https://www.intel.it/content/www/it/it/architecture-and-technology/trusted-execution-technology/trusted-execution-technology-security-paper.html
11. Srage J, Azema J. M-Shield mobile security technology. Texas instruments white paper. 2005.
12. Penard W, van Werkhoven T. On the secure hash algorithm family. In: *Cryptography in Context*. 2008:1-18.
13. Sarkisyan A, Debbiny R, Nahapetian A. WristSnoop: smartphone PINs prediction using smartwatch motion sensors. In: Proceedings of the 2015 IEEE International Workshop on Information Forensics and Security (WIFS); 2015; Rome, Italy.
14. Maiti A, Jadliwala M, He J, Bilogrevic I. (Smart)watch your taps: side-channel keystroke inference attacks using smartwatches. In: Proceedings of the 2015 ACM International Symposium on Wearable Computers (ISWC '15); 2015; Osaka, Japan.
15. Wang C, Guo X, Chen Y, Wang Y, Liu B. Personal pin leakage from wearable devices. *IEEE Trans Mob Comput*. 2018;17(3):646-660.
16. Wang C, Guo X, Wang Y, Chen Y, Liu B. Friend or foe?: Your wearable devices reveal your personal pin. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (ASIACCS); 2016; Xi'an, China.
17. Simon L, Anderson R. PIN skimmer: inferring PINs through the camera and microphone. In: Proceedings of the 3rd ACM Workshop on Security and Privacy in smartphones & Mobile Devices (SPSM '13); 2013; Berlin, Germany. http://doi.acm.org/10.1145/2516760.2516770
18. Cai L, Chen H. TouchLogger: inferring keystrokes on touch screen from smartphone motion. In: Proceedings of the 6th USENIX Conference on Hot Topics in Security; 2011; San Francisco, CA. http://dl.acm.org/citation.cfm?id=2028040.2028049
19. Xu Z, Bai K, Zhu S. TapLogger: inferring user inputs on smartphone touchscreens using on-board motion sensors. In: Proceedings of the 5th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WISEC '12); 2012; Tucson, AZ. http://doi.acm.org/10.1145/2185448.2185465
20. Armando A, Costa G, Verderame L, Merlo A. Securing the "bring your own device" paradigm. *Computer*. 2014;47(6):48-56.
21. Armando A, Costa G, Merlo A, Verderame L. Enabling BYOD through secure meta-market. In: Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks; 2014; Oxford, UK.
22. van Rijswijk RM, Poll E. Using trusted execution environments in two-factor authentication: comparing approaches. In: *Proceedings of the Open Identity Summit 2013 (OID 2013)*. Bonn, Germany: Gesellschaft for Informatik; 2013:20-31. *Lecture Notes in Informatics*.
23. Aonzo S, Merlo A, Tavella G, Fratantonio Y. Phishing attacks on modern Android. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security; 2018; Toronto, Canada.
24. Dahan FB, Cornillault B. Secure mode indicator for smart phone or PDA. US patent 8,479,022. 2013.
25. Shukla D, Kumar R, Serwadda A, Phoha VV. Beware, your hands reveal your secrets! In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14); 2014; Scottsdale, AZ. http://doi.acm.org/10.1145/2660267.2660360

26. Ye G, Tang Z, Fang D, et al. Cracking Android pattern lock in five attempts. Proceedings 2017 Network and Distributed System Security Symposium (NDSS'17); 2017; San Diego, CA.
27. Ozdenizci B, Ok K, Coskun V. A tokenization-based communication architecture for HCE-enabled NFC services. *Mob Inf Syst*. 2016;2016.
28. Armando A, Merlo A, Verderame L. Trusted host-based card emulation. In: Proceedings of the 2015 International Conference on High Performance Computing & Simulation (HPCS); 2015; Amsterdam, The Netherlands.
29. Blake-Wilson S, Menezes A. Unknown key-share attacks on the station-to-station (STS) protocol. In: *Public Key Cryptography: Second International Workshop on Practice and Theory in Public Key Cryptography, PKC'99 Kamakura, Japan, March 1-3, 1999 Proceedings*. Berlin, Germany: Springer; 1999:154-170.
30. Sierraware. *SierraTEE Trusted Execution Environment*. Technical Report. Cupertino, CA: Sierraware Inc.; 2101. https://www.sierraware.com/open-source-ARM-TrustZone.html
31. Ogiela L, Ogiela MR. Insider threats and cryptographic techniques in secure information management. *IEEE Syst J*. 2015;11(2):405-414.
32. Ogiela MR, Ogiela U. Secure information management in hierarchical structures. In: *Advanced Computer Science and Information Technology: Third International Conference, AST 2011, Seoul, Korea, September 27-29, 2011. Proceedings*. Berlin, Germany: Springer; 2011:31-35.
33. Nguyen T, Memon N. Smartwatches locking methods: a comparative study. In: Proceedings of the 33th Symposium on Usable Privacy and Security (SOUPS 2017); 2017; Santa Clara, CA. https://www.usenix.org/conference/soups2017/workshop-program/way2017/nguyen
34. Ogiela L, Ogiela MR, Ogiela U. Efficiency of strategic data sharing and management protocols. In: Proceedings of the 2016 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS); 2016; Fukuoka, Japan.
35. Yang J, Li Y, Xie M. MotionAuth: motion-based authentication for wrist worn smart devices. In: Proceedings of the 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops); 2015; St. Louis, MO.
36. Lewis A, Li Y, Xie M. Real time motion-based authentication for smartwatch. In: Proceedings of the 2016 IEEE Conference on Communications and Network Security (CNS); 2016; Philadelphia, PA.
37. Buriro A, Crispo B, Eskandri M, Gupta S, Mahboob A, Van Acker R. SNAPAUTH: a gesture-based unobtrusive smartwatch user authentication scheme. In: Saracino A, Mori P, eds. *Emerging Technologies for Authorization and Authentication*. Cham, Switzerland: Springer International Publishing; 2018:30-37.
38. Johnston AH, Weiss GM. Smartwatch-based biometric gait recognition. In: Proceedings of the 2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS); 2015; Arlington, VA.
39. Al-Naffakh N, Clarke N, Li F, Haskell-Dowland P. Unobtrusive gait recognition using smartwatches. In: Proceedings of the 2017 International Conference of the Biometrics Special Interest Group (BIOSIG); 2017; Darmstadt, Germany.
40. Nguyen T, Memon N. Tap-based user authentication for smartwatches. *Comput Secur*. 2018;78:174-186. http://www.sciencedirect.com/science/article/pii/S0167404818303778
41. Ehatisham-ul Haq M, Azam MA, Loo J, et al. Authentication of smartphone users based on activity recognition and mobile sensing. *Sensors*. 2017;17(9). http://www.mdpi.com/1424-8220/17/9/2043
42. Guerar M, Migliardi M, Merlo A, Benmohammed M, Palmieri F, Castiglione A. Using screen brightness to improve security in mobile social network access. *IEEE Trans Dependable Secure Comput*. 2018;15(4):621-632.
43. von Zezschwitz E, De Luca A, Brunkow B, Hussmann H. SwiPIN: fast and secure PIN-entry on smartphones. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15); 2015; Seoul, Republic of Korea. http://doi.acm.org/10.1145/2702123.2702212
44. Guerar M, Migliardi M, Merlo A, Benmohammed M, Messabih B. A completely automatic public physical test to tell computers and humans apart: a way to enhance authentication schemes in mobile devices. In: Proceedings of the 2015 International Conference on High Performance Computing & Simulation (HPCS); 2015; Amsterdam, The Netherlands.
45. Guerar M, Merlo A, Migliardi M. Completely automated public physical test to tell computers and humans apart: a usability study on mobile devices. *Future Gener Comput Syst*. 2018;82:617-630. http://www.sciencedirect.com/science/article/pii/S0167739X17303709
46. Bianchi A, Oakley I, Kostakos V, Kwon DS. The phone lock: audio and haptic shoulder-surfing resistant pin entry methods for mobile devices. In: Proceedings of the 5th International Conference on Tangible, Embedded, and Embodied Interaction (TEI '11); 2011; Funchal, Portugal.
47. Kwon T, Na S. TinyLock: affordable defense against smudge attacks on smartphone pattern lock systems. *Comput Secur*. 2014;42:137-150.
48. Guerar M, Merlo A, Migliardi M. ClickPattern: a pattern lock system resilient to smudge and side-channel attacks. *JoWUA*. 2017;8(2):64-78. http://isyou.info/jowua/papers/jowua-v8n2-4.pdf
49. Guerar M, Benmohammed M, Alimi V. Color wheel pin: usable and resilient ATM authentication. *J High Speed Netw*. 2016;22(3):231-240.
50. De Luca A, Von Zezschwitz E, Hußmann H. Vibrapass: secure authentication based on shared lies. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems; 2009; Boston, MA.
51. De Luca A, Hertzschuch K, Hussmann H. ColorPIN: securing PIN entry through indirect input. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10); 2010; Atlanta, GA.
52. Nyang D, Mohaisen A, Kang J. Keylogging-resistant visual authentication protocols. *IEEE Trans Mob Comput*. 2014;13(11):2566-2579.
53. Oakley I, Huh JH, Cho J, Cho G, Islam R, Kim H. The personal identification chord: a four buttonauthentication system for smartwatches. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security (ASIACCS '18); 2018; Incheon, Republic of Korea.
54. Nguyen TV, Sae-Bae N, Memon N. DRAW-A-PIN: authentication using finger-drawn PIN on touch devices. *Comput Secur*. 2017;66:115-128.
55. Hutchins B, Reddy A, Jin W, Zhou M, Li M, Yang L. Beat-PIN: a user authentication mechanism for wearable devices through secret beats. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security (ASIACCS '18); 2018; Incheon, Republic of Korea. http://doi.acm.org/10.1145/3196494.3196543
56. Yoon H, Park S-H, Lee K-T. Exploiting ambient light sensor for authentication on wearable devices. In: Proceedings of the 2015 4th International Conference on Cyber Security, Cyber Warfare, and Digital Forensic (CyberSec); 2015; Jakarta, Indonesia.
57. Kuribara T, Shizuki B, Tanaka J. Vibrainput: two-step PIN entry system based on vibration and visual information. In: Proceedings of the CHI '14 Extended Abstracts on Human Factors in Computing Systems; 2014; Toronto, Canada. http://doi.acm.org/10.1145/2559206.2581187

58. Lee M-K, Nam H, Kim DK. Secure bimodal PIN-entry method using audio signals. *Comput Secur*. 2016;56:140-150. http://www.sciencedirect.com/science/article/pii/S0167404815000929
59. Samsung Gear S3 frontier software update. https://www.verizonwireless.com/support/samsung-galaxy-gear-s3-frontier-update/. Accessed August 6, 2019.

How to cite this article: Guerar M, Migliardi M, Palmieri F, Verderame L, Merlo A. Securing PIN-based authentication in smartwatches with just two gestures. *Concurrency Computat Pract Exper*. 2019;e5549. https://doi.org/10.1002/cpe.5549