2GesturePIN: a secure PIN-based authentication for smartwatches

Meriem Guerar, Luca Verderame, Mauro Migliardi, Francesco Palmieri, and Alessio Merlo

***Abstract*—Smartwatches are becoming increasingly ubiquitous, 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. PINs and Pattern Lock have been widely used in smartwatches for user authentication. However, those types of passwords are not robust against various forms of attacks, such as side channel, phishing, smudge, shoulder surfing and video recording attacks even when the TEE is used. Thus, the user's security and privacy are in risks without a strong authentication scheme in place. In this work, we propose 2GesturePIN, a new authentication method that allows users to authenticate securely to their smartwatches and sensitive services through solely two gestures. It leverage the rotating bezel or crown which are the most intuitive ways to interact with a smartwatch. 2GesturePIN enhances the resilience of the regular PIN to common attacks while maintaining a high level of usability.**

***Index Terms*—Smartwatch, Authentication, TrustZone, NFC services, PIN, Bezel.**

I. 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 a lot of sensitive information is now stored on them. Besides health and fitness tracking, smartwatches can be used for making payments, identification, ticketing, and controlling access to critical services and infrastructures. This trend is amplified by the introduction of Near-Field Communication (NFC) technology which allows devices to act as a smart card. As such, the protection of smartwatches sensitive data became of paramount importance.

One solution to protect user's sensitive data from malware is through Trusted Execution Environment (TEE) because it ensures that data is stored, processed and protected within a totally trusted environment that malware cannot tamper with.

TEE has been widely deployed in many smartphones and, recently, it has been ported also on smartwatches. For example, Samsung Gear S2 and S3 contain Knox which is a mobile security platform that provides a trusted execution environment based on the ARM TrustZone technology [1].

M. Guerar, L. Verderame, and A, Merlo are with the Department of Informatics, Bioengineering, Robotics and Systems Engineering, University of Genoa, Genoa, Italy e-mail: {alessio.merlo, luca.verderame, meriem.guerar}@unige.it.

M. Migliardi is with the Department of Information Engineering, University of Padua, Padua, Italy e-mail: mauro.migliardi@unipd.it.

F. Palmieri is with Department of Computer Science, University of Salerno, Salerno, Italy e-mail: fpalmieri@unisa.it.

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 to enable the access to sensitive information: such mechanism is vulnerable to several attacks (e.g, side channel, phishing, shoulder surfing and video recording attacks) and can be easily bypassed [2], [3], [4]. Therefore, such mechanism is currently too unreliable for security-sensitive operations (e.g, Payment using Samsung pay, Apple pay and Android pay) on smartwatches. The adoption of pattern locks as an alternative to PIN does not provide stronger security guarantees [2], [5].

Although TEE offers a trusted user interface (UI) which ensures that malware running on the device cannot steal data displayed or typed on the screen, this is not sufficient to prevent data leakage from other components of the device that interact with the TEE (e.g., accelerometer, gyroscope, and orientation sensors), as they allow to carry out side channel attacks [6]. Another security issue is that although the TEE locks the screen whenever a trusted application wants to use it, the user cannot understand which app is displaying the PIN pad (i.e., if they are prompted by a trusted application in the TEE or from a malware app in the untrusted part of the OS that mimics the trusted one). This is due to the fact that the screen is shared among all apps installed on the smartwatch, independently from being hosted in the TEE or not. Furthermore, 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.

In this paper we propose an authentication framework (*2GesturePIN*) able to enhance both the security and the usability of PIN input on smartwatches. The novelty of 2GesturePIN is to leverage the bezel of the smartwatch as a secure hardware (i.e., the bezel is controlled by the TEE) to input four-PIN digits through solely two gestures. In this way, the user is not required to tap on small-size touch screens. This does not only improve the usability, but it also enhances the security of the regular PIN code against shoulder surfing, video recording, phishing and motion-based side channel attacks, as we show later in the paper. Hence, the 2GesturePIN framework can be used as an authentication mechanism for security critical NFC-based operations on smartwatches.

A typical usage scenario would be using 2GesturePIN to secure the access to an e-wallet that allows the user to open her house, car and office doors as well as making payment in store, paying parking and public transport tickets, just to cite

Fig. 1. 2GesturePIN user interface.

a few. In such a scenario, the user has to authenticate to the smartwatch and then she can perform any operation by simply tapping her smartwatch on NFC devices, without the need to carry multiple cards and keys or look for her smartphone in her pocket each time she wants to perform a transaction. At the same time, while a stolen card can be automatically used by the thief to impersonate the user (e.g., access a building or small-value payment), the stolen smartwatch cannot be used unless the thief has the owner's PIN as well.

The rest of this paper is structured as follows: In section 2 we introduce 2GesturePIN framework; in Section 3 we describe the smart lock access control use case; in Section 4 we present threat model and perform a security analysis of our scheme; in Section 5 we present related work. Finally, section 6 concludes this paper.

II. 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 *i)* the usability of the authentication process and *ii)* the resiliency against a wide range of security threats.

In this section we will provide: *i)* a brief description of TEE security features, *ii)* a description of the 2GesturePIN authentication process from the user perspective (Section II-B), *iii)* a description of the architecture of the 2GesturePIN Framework (Section II-C), *iv)* a detailed discussion of the whole authentication flow (Section II-D) and, finally, *v)* a description of a prototype implementation for Tizen OS and SierraTEE enabled smartwatches (Section II-E).

A. Trusted Execution Environment

The concept of Trusted Execution Environment (TEE) has been standardized by GlobalPlatform in 2011 [7]. 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 (e.g. dedicated storage, dual mode CPUs) and software (e.g. 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 (e.g., secure storage and biometric sensors) along with other resources shared with the REE (e.g., 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, e.g. its private key and certificate, hardwired in a read-only memory.

Vasudevan et al. [8] summarize the security requirements that a TEE has to fulfill:

- Isolated Execution ensuring that applications that resides in the TEE (called Trusted Applications - TA) execute completely isolated from and unhindered by any other application.
- Secure Storage protecting persistently stored data (e.g. cryptographic keys).
- Remote Attestation enabling remote parties to ascertain that they are dealing with a given trusted application on a specific TEE-device.
- Secure Provisioning enabling communication by remote parties with a specific trusted application, thereby protecting integrity and confidentiality of transmitted data.
- Trusted Path, i.e. a channel allowing i) the user to send data to the TEE and ii) the TEE to send back data to the user; the channel 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 includes Intel Trusted Execution Technology [9] and Texas Instrument MShield [10].

B. 2GesturePIN User Authentication

Differently from 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 wheel 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, e.g. 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 his 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. This implies that, thanks to the 2GesturePIN authentication method, the user is able to input his four-PIN digits with solely two simple gestures (i.e rotating the wheel through the bezel) instead of typing it in small sized touch screen.

An example of 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 in order to match the second PIN digit (3) of the outer wheel. Then, 2GesturePIN

Fig. 2. 2GesturePIN authentication method.

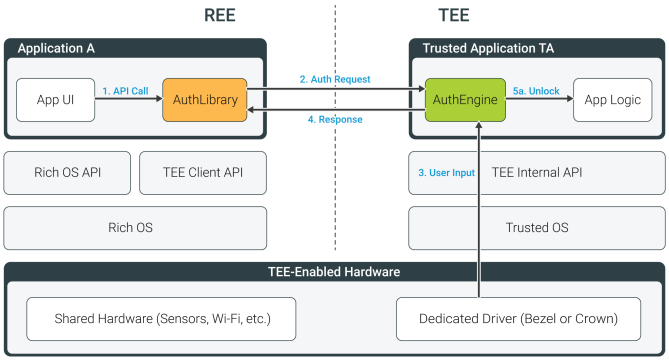

Fig. 3. 2GesturePIN Framework Architecture.

computes a second screen with a novel 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, the user rotates the the number 4 to match number 0.

C. 2GesturePIN Architecture

The 2GesturePIN Framework is composed by two main modules: 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 *i)* perform authentication requests (step 1 in Figure 3), *ii)* render the authentication interface according to the inputs provided by the *AuthEngine* and *iii)* 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 the means of the *AuthLibrary*.

The core fuctionalities of the *AuthEngine* are:

- *2GesturePIN Authentication Management*. The engine enforces the authentication flow described in the Section II-B, manages the session's lifecycle and is able to unlock the TA in case of successfull authentication (step 5a in Figure 3). For a detailed description of the authentication steps please refer to Section II-D;
- *Trusted Input Management*. The *AuthEngine* handles the Trusted Path to the dedicated hardware, e.g. the bezel of the smartwatch, in order 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 (e.g. 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, the device D relies on embedded TrustZone crypto libraries, using a 512 bit SHA-2 algorithm [11]. 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.

D. 2GesturePIN Authentication Flow

Figure 4 depicts the authentication process of 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 his smartwatch an application that requires the 2GesturePIN authentication mechanism. The application is composed by the Application UI (A) that resides in the

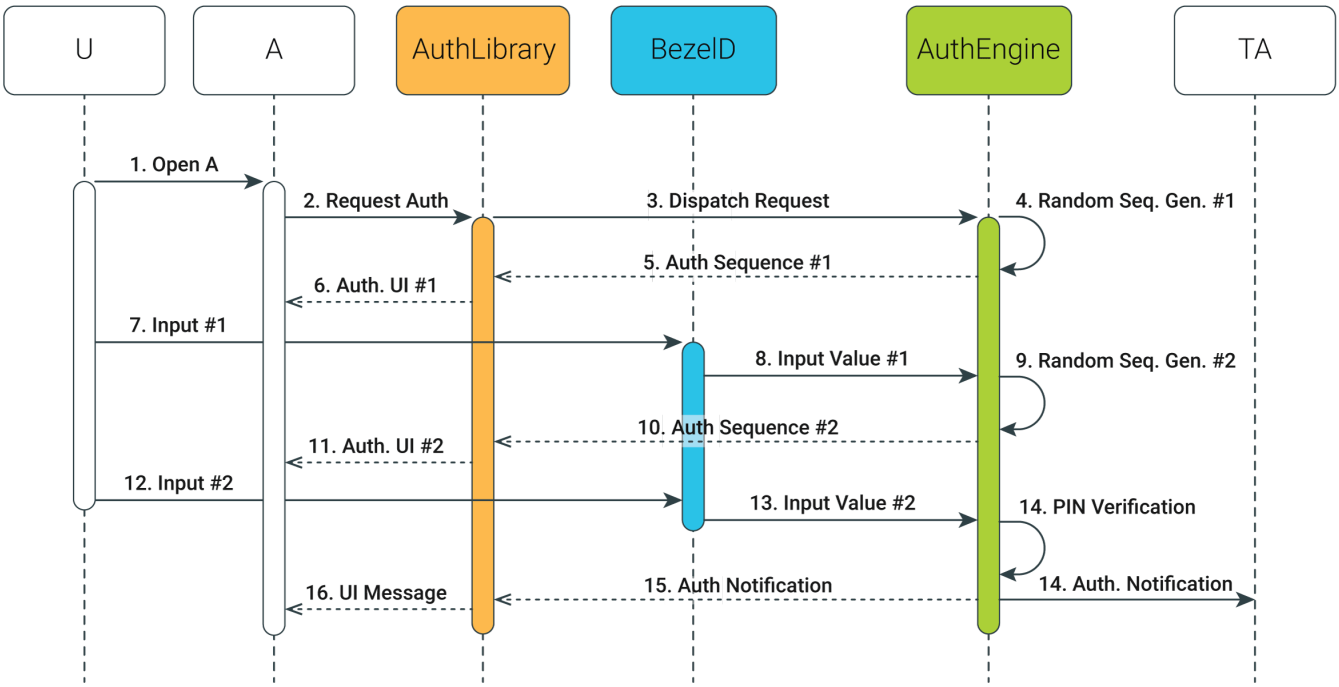

Fig. 4. 2GesturePIN authentication flow.

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 his PIN and confirms its choice with the side button. After the confirmation, the Bezel driver (BezelD), that resides in the TEE, gets the rotation degree of the bezel and sends this information to the *AuthEngine* (steps 7 and 8).

The *AuthEngine* computes the second random sequence of the wheel's numbers (step 9) necessary for the input of the second pair 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).

E. 2GesturePIN Implementation

One potential implementation of 2GesturePIN framework is through SierraTEE [12] which provides an open source implementation of TEE environment compatible with Global Platform standards and ARM TrustZone. SierraTEE supports multiple operating systems including any 64-bit platform utilizing ARM Cortex-A53 processors which are used by Samsung Gear S3 smartwatches. 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 OS 2.3.2 as well as on Android Wear OS 2.0 with no dependencies to the TEE.

We developed the applications using Tizen Studio 3.0 and Android Studio 3.1 respectively. The test equipment consisted of the Samsung Gear S3 Frontier LTE smartwatch from Samsung, which is based on Tizen 2.3.2 and fitted out with a hardware rotating bezel, a 360x360 pixels screen, a Dual-core 1.0 GHz, 768MB RAM and 4GB internal memory. For Android Wear, we used the LG Watch Style W270 smartwatch. Released at the beginning of 2017, it runs Android wear 2.0 and it features a Quad-Core 1.1 GHz Qualcomm Snapdragon Wear 2100 CPU, with 512 MB of RAM and 4 GB of internal memory and 360x360 pixels screen. Instead of the rotating bezel, this smartwatch is equipped with a rotating crown on the side.

As shown in the figure 5, 2GesturePIN app has two modes, training and test, where the first mode allows participants to get familiar with the concept and the latter may be used for usability tests, in which we record the input time and the error rate on ten consecutive attempts. The application menu features also a settings button where we can update the PIN, along with other settings such as toggling the colors use, this option displays different colors on the wheel portions where each digit has always the same color. We noticed that using fixed colors on the wheel helps the users quickly locate

Fig. 5. 2GesturePIN screenshots in Tizen emulator

numbers and therefore authenticate faster.

III. Threat model and Security Analysis

A. 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 (e.g Trusted User Interface) used currently by sensitive applications to protect PIN entry. As mentioned above, although TEE protects PIN entry from malware that try to record the screen or touch events during authentication by providing a trusted path to the screen, the PIN code or Pattern lock still vulnerable to the following attacks:

Side Channel attacks. In many recent studies, researchers have considered smartwatches as a side channel to infer secrets entered on external devices such as smartphones [13], [14] and ATMs [15], [16] thus not taking into account the smartwatch as a venue for authentication; on the contrary, in [2], authors show the feasibility of inferring the user's password entered into the smartwatch itself through touch screen. Since the malware is not able to record the screen and touch events during the users PIN input, another kind of attack has been emerged which leverage the other shared resources between the REE and the TEE (e.g., accelerometer [6], the camera and the microphone [17], and the Gyroscope [18], [19]) to infer the users input. The idea behind this attack is that the devices micro-movements caused by the user's tap on the touchscreen are quite different depending on the tap' location. Therefore, using 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 his smartwatch "Snoopy"[2], a fitness or gaming app which is in fact a Trojan app that eavesdrop 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 crowd-sourced data to infer the user's passwords.

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 [20]. 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 (e.g,. Payment app) to trick the user to input his 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 [21]. However, unlike smartphones, the current smartwatches in the market are not equipped with a LED. Therefore, implementing this mechanism on smartwatches requires additional hardware. In order to show how our security mechanism deal with this issue, we assume that the smartwatch is infected by a malicious application that pretends to be 2GesturePIN.

Guessing attacks. In order to evaluate the security of 2GesturePIN against Guessing attack, we assume that an attacker is able to get the physical access to the smartwatch and she will 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 (i.e. coffee shop, library, bus, 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. The observation can be made once or multiple times. In addition, we assume that the shoulder surfer is reasonably close to the user and she is able to 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.

B. 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.

a) Side channel attack: As described above, smartwatches could be infected by a malicious software masked as a legitimate application (e.g 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. In addition, The bezel driver resides in the TEE which prevents the malicious software from reading the bezel input. Since 2GesturePIN is the first work that uses the bezel for smartwatch authentication, there is not any work in the literature that aim to detect the bezel movement through sensors as side channel. However,

Fig. 6. 2GesturePIN applications.

even in that case, 2GesturePIN is designed in such a way that the correct PIN digits as well as all the other nine numbers on the wheel turn with the same degree according to the bezel movement and this degree is different at each step and authentication session due to the randomization of the wheel numbers position. Hence, 2GesturePIN provides indirect input that does not reveal any useful information related to the PIN digits and thus it is resilient to any kind of 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 6, to authenticate the user to his smartphone or ATM account using 2GesturePIN, the user has to slide the seekbar to rotate the Wheel. However, since the user input is different each time the user authenticate due the randomization of wheel numbers positions, the smartwatch couldnt reveal any useful information about the users PIN. The same thing holds for smart door lock: using 2GesturePIN instead of keypad lock provides enhanced security against this category of side channel attacks, in fact the degree of rotation of the knob is different for each authentication session. It is important to mention that unlocking the smart lock using the smartwatch as described in the use case provides two factor authentication and thus it is more secure than using 2GesturePIN directly in smart lock.

b) Phishing attack: As mentioned above, a malicious application in the REE can pretend to be 2GesturePIN to trick the user into entering his PIN digits. Using a Trusted User Interface does not provide a protection against this attack because the screen is shared between REE and TEE and thus, the user can not make difference between UI displayed by REE or TEE. However, 2GesturePIN framework enables the user to figure out that this application is fake easily by noticing that the wheel is not moving according to the bezel movement or not moving at all. This is due to the fact that the access to the bezel is granted only to the trusted applications and thus prevents the malicious application in REE from acquiring any data about the bezel movement. Hence, the phishing app can not trick the user by pretending to be a trusted application that requires 2GesturePIN authentication. Therefore, 2GesturePIN is resilient to phishing attacks.

c) 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 didn't 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 (i.e bezel). Therefore, we added this section in the security analysis to highlight how 2GesturePIN resists this attack. Since, 2GesturePIN framework leverages the bezel as means of PIN input and ensures that the bezel could be accessible only to the TEE, it prevents any automated process from performing a brute force attack. This is due to the fact that malicious codes in REE is unable to physically rotate the bezel required for the PIN input or simulate the bezel movement by manipulating the bezel driver which makes 2GesturePIN resilient to this attack.

d) Guessing attack: The chance that a guessing attack succeeds depends on the size of the password space. 2GesturePIN uses the standard four-PIN digits in such a way the ten numbers in the wheel can be matched to ten numbers in two different input steps, therefore there are 100 possible combinations for each step. Hence, the number of possible password combination is 10000. Similar to the regular PIN, the probability of guessing successfully the users PIN in three attempts using 2GesturePIN is 0.0003.

e) Shoulder surfing attack: In order 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. There is not any indication of which combination among them represents the correct part of the four-PIN digits. In addition, 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 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.

f) 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 real world scenario, it is not always easy to perform such an attack without the user's awareness. The best recording of 2GesturePIN authentication session gives an attacker two sets of ten combinations of numbers in which she can 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 is important to notice that 2GesturePIN is more resilient against this type of attacks with respect to regular PIN and pattern lock, where, in the same conditions, the recording will reveal the user's PIN or the users pattern with certainty.

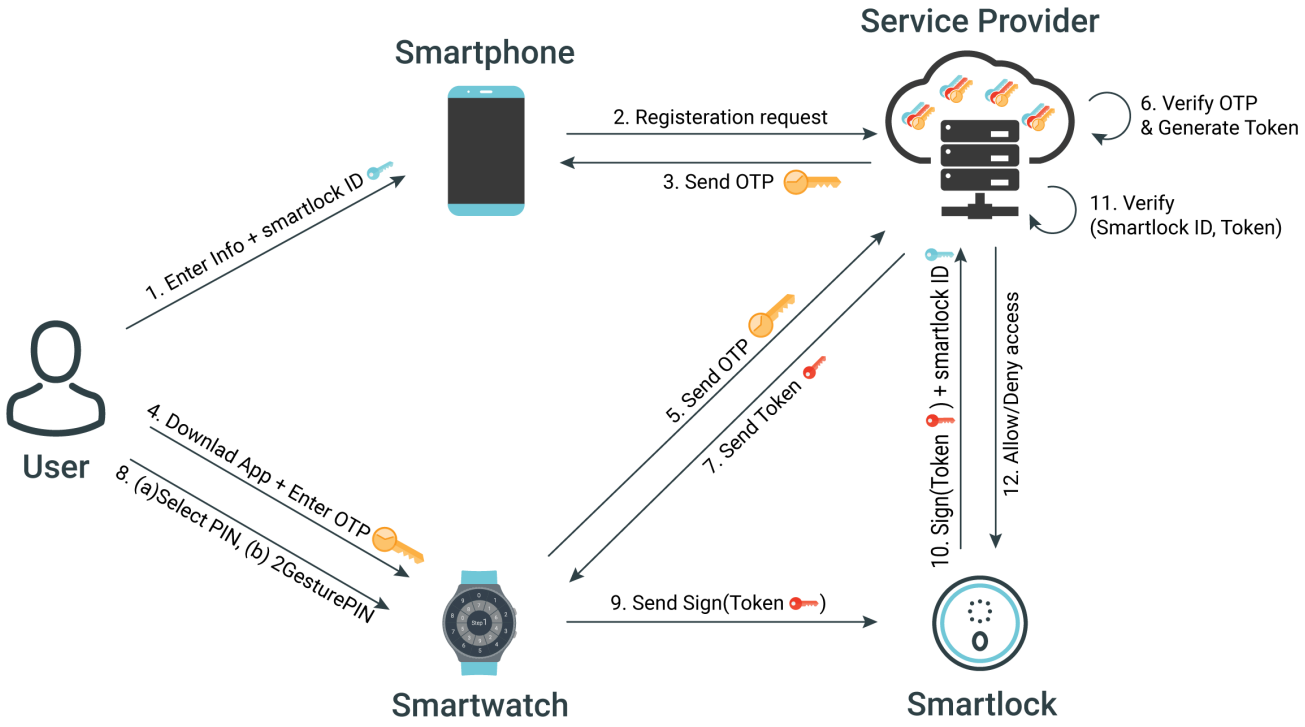

Fig. 7. Use case:smart lock access control.

C. Comparison of 2GesturePIN security strength to regular PIN

Since the regular 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 important to highlight why using 2GesturePIN to protect sensitive applications on 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 the Table I, 2GesturePIN is resilient against phishing, side channel and shoulder surfing attacks and it provides enhanced level of security to video-recording attack comparing to the regular PIN (i.e., the probability that the attacker reveal the PIN successfully with 2GesturePIN is 0.01 instead of 1 using the regular PIN). In addition, it provides the same security level as the regular PIN against brute force and guessing attack. Hence, 2GesturePIN is a good alternative to regular PIN for protecting the sensitive services on smartwatch.

TABLE I
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
Video-recording attacks	Medium	Weak
Gessing attacks	Same	Same

IV. 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 example, the users save times by using their smartwatch rather than looking for their smartphones or keys on their pockets. On the other hand, using smartwatches enable a two factor authentication comparing to smart cards (i.e., access control and small-amount payment use cases) and physical keys. If someone lost his car key in a parking, this key can be used directly to gain unauthorised access to the car, while in the case of the smartwatch, the access to the key is protected by an authentication method.

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 (e.g., 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 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 install a smart lock of a specific service provider. Then, she register herself the first time to the SP portal and enroll the smartwatch though installing a TA of the same SP which offers 2GesturePIN framework. After a successful login using 2GesturePIN, this application allows the user to lock or unlock the door. Figure 7 depicts the overall scenario.

A. Smartwatch Enrollment

In order to initiate the registration process, the user visits the website of service provider (SP) through his smartphone or personal computer and starts the registration process. During this phase, the user enters his information (e.g 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 (e.g 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 his smartwatch (D) and enter the OTP using the same input mechanism of 2GesturePIN described in Section II-D (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 [22].

The user then selects a four-PIN digits in 2GesturePIN settings to protect the access to this token (step 8(a)). The PIN initialization is performed as follow: 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 [23] 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*.

B. Smart Lock Authentication

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

When the user taps his smartwatch on the smart lock, the TA, by the means of 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 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 [24].

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 case of the session expiration, the smartwatch prompts the user a new request for authentication (step 8b).

V. Related work

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 (e.g Apple Pay, Google Pay or Samsung Pay) or controlling access to critical services and infrastructures (e.g 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], [25], they are the predominant types of authentication mechanism today.

A. Behavioral-biometrics based authentication methods

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, [26], [27] designed a motion-based authentication method able to authenticate a user by performing a gestures with a wrist worn device or smartwatch, after building the user's behavioural profile by collecting data from device sensors. A similar approach has been taken by SnapAuth [28], 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 (i.e., 15). Johnston and Weiss [29] 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, users 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 [30]. Authors pointed out that gait recognition is highly efficient and recommended to authenticate users in a transparent and continuous manner. Results are positive, however 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 his walk or is not walking. A different approach has been taken by [31]: TapMeIn let the user authenticate by tapping a specific rhythm on the smartwatchs touchscreen. Results are significantly promising, with an accuracy of 98.7%, however tests were performed in a lab with a limited dataset, which may favour the classification process. All works presented above are related to a specific behavior of a human while performing some tasks, such as hand movement, gait, and rhythmic tapping, which may present some limitations [32]. 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 persons walking style and persons typing behavior changes considerably throughout a day with different states of mind such as excited, tired, etc. These limitations related to human behavior nature among others might be the main barriers to solely rely on a behavioral system.

B. Knowledge based authentication methods

Other researchers instead worked on increasing security around the current authentication methods: PIN and pattern lock. Research has focused on smartphones (e.g. [33], [34], [35], [36], [37], [38], [39]), ATM (e.g. [40], [41], [42], [43]) and recently smartwatches, where the small screen size introduces usability concerns.

A novel PIN based authentication method is Personal Identification Chord (PIC) [44], 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, however the usability study shows PIC as slightly slower and more error prone than PIN. Furthermore, PIC is not resilient to side-channel and shoulder-surfing attacks. In [45], authors introduced a two factor authentication method, called Draw- a-PIN. To authenticate, the user is required to draw his PIN digits sequentially on the touchscreen instead of typing it. Beside 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 with respect to shoulder-surfing resilience, the usability study of its implementation on a smartwatch [25] showed that it is not usable to unlock the smartwatch due to its high error rate and long authentication time (i.e Overall Average Error rate 20.65% , Overall Average authentication time 7356 ms). Analogous method to TapMeIn [31] is Beat-PIN [46], where a PIN is represented by a sequence of beats recorded when the user taps on the smartwatch touchscreen (i.e, a beat is the time between the instance the user touches the screen and the instance the user lifts his 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 [47]. The variations of the lights values read by the ambient light sensor are used to build sequences representing particular User Interface (UI) events, such as single-click, double-click, 1-sec-hold, etc. These events are used to enter the PIN (e.g. 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 events 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 dark environment.

Similar to 2GesturePIN, VibraInput [48] and DialA [49] 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 touch the screen, the vibration starts and the user has to remember the letter that correspond 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 his finger. Since the outer wheel contains multiple occurrences of this indicator, another round is required to identify the PIN digit. Thus, beside the overhead of memorizing an additional secret, VibraInput requires eight gestures to input four PIN digits. In DialA [49], the user has to use the letter that he heard through an earphones as an indicator to input the four-PIN digits. The rotation and commitment are conducted via another small scroll wheel at the bottom of the smartphone screen in order to prevent a direct input. However this method is not suitable for smartwatches for two reasons. First because the user is required to wear an earphones and connect it to the smartwatch through bluetooth (i.e., if it is not connected) which is impractical and take a long time. Second, because using another small wheel is not suitable for small-size screen such as smartwatches and rotating the wheel directly makes the user susceptible to side channel, shoulder surfing and video recording attacks. In addition, unlike 2GesturePIN, DialA requires four gestures.

VI. Conclusion

Nowadays, wearable transactions are becoming very popular due to the facilities provided by the NFC technology. However, the authentication step is often considered as the weakest link in the security of these transactions due to the increase of security threats. This paper introduced a novel PIN-based authentication method for smartwatches through two bezel 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. In future work, in order to have a complete assessment of 2GesturePIN usability, we intend to carry on a study with a significant number of participants testing both the bezel based implementation and the crown based one.

REFERENCES

[1] A. Technologies, Arm security technology building a secure system using trustzone technology, Tech. rep. (2005).
URL http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf

[2] C. X. Lu, B. Du, H. Wen, S. Wang, A. Markham, I. Martinovic, Y. Shen, A. Trigoni, Snoopy: Sniffing your smartwatch passwords via deep sequence learning, IMWUT 1 (2017) 152:1–152:29.

[3] A. Brandon, M. Trimarchi, Trusted display and input using screen overlays, in: 2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig), 2017, pp. 1–6. doi:10.1109/RECONFIG.2017.8279826.

[4] H. Khan, U. Hengartner, D. Vogel, 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, ACM, New York, NY, USA, 2018, pp. 164:1–164:10. doi:10.1145/3173574.3173738.
URL http://doi.acm.org/10.1145/3173574.3173738

[5] G. Ye, Z. Tang, D. Fang, X. Chen, W. Wolff, A. J. Aviv, Z. Wang, A video-based attack for android pattern lock, ACM Trans. Priv. Secur. 21 (4) (2018) 19:1–19:31. doi:10.1145/3230740.
URL http://doi.acm.org/10.1145/3230740

[6] E. Owusu, J. Han, S. Das, A. Perrig, J. Zhang, Accessory: Password inference using accelerometers on smartphones, in: Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications, HotMobile '12, ACM, New York, NY, USA, 2012, pp. 9:1–9:6. doi:10.1145/2162081.2162095.
URL http://doi.acm.org/10.1145/2162081.2162095

[7] G. Platform, The trusted execution environment: Delivering enhanced security at a lower cost to the mobile market, White Paper February.

[8] A. Vasudevan, E. Owusu, Z. Zhou, J. Newsome, J. M. McCune, Trustworthy execution on mobile devices: What security properties can my mobile platform give me?, in: International Conference on Trust and Trustworthy Computing, Springer, 2012, pp. 159–178.

[9] INTEL, Intel trusted execution technology, Tech. rep. (2012).
URL https://www.intel.it/content/www/it/it/architecture-and-technology/trusted-execution-technology/trusted-execut

[10] J. Srage, J. Azema, M-shield mobile security technology, TI White pape r http://focus. ti. com/pdfs/wtbu/ti mshield whitepaper. pdf.

[11] W. Penard, T. van Werkhoven, On the secure hash algorithm family.

[12] Sierraware, Sierratee trusted execution environment.
URL https://www.sierraware.com/open-source-ARM-TrustZone.html

[13] A. Sarkisyan, R. Debbiny, A. Nahapetian, Wristsnoop: Smartphone pins prediction using smartwatch motion sensors, in: 2015 IEEE International Workshop on Information Forensics and Security (WIFS), 2015, pp. 1–6. doi:10.1109/WIFS.2015.7368569.

[14] A. Maiti, M. Jadliwala, J. He, I. Bilogrevic, (smart)watch your taps: Side-channel keystroke inference attacks using smartwatches, in: Proceedings of the 2015 ACM International Symposium on Wearable Computers, ISWC '15, ACM, New York, NY, USA, 2015, pp. 27–30. doi:10.1145/2802083.2808397.
URL http://doi.acm.org/10.1145/2802083.2808397

[15] C. Wang, X. Guo, Y. Chen, Y. Wang, B. Liu, Personal pin leakage from wearable devices, IEEE Transactions on Mobile Computing 17 (3) (2018) 646–660. doi:10.1109/TMC.2017.2737533.

[16] C. Wang, X. Guo, Y. Wang, Y. Chen, B. Liu, Friend or foe?: Your wearable devices reveal your personal pin, in: AsiaCCS, 2016.

[17] L. Simon, R. Anderson, Pin skimmer: Inferring pins through the camera and microphone, in: Proceedings of the Third ACM Workshop on Security and Privacy in Smartphones & Mobile Devices, SPSM '13, ACM, New York, NY, USA, 2013, pp. 67–78. doi:10.1145/2516760.2516770.
URL http://doi.acm.org/10.1145/2516760.2516770

[18] L. Cai, H. Chen, Touchlogger: Inferring keystrokes on touch screen from smartphone motion, in: Proceedings of the 6th USENIX Conference on Hot Topics in Security, HotSec'11, USENIX Association, Berkeley, CA, USA, 2011, pp. 9–9.
URL http://dl.acm.org/citation.cfm?id=2028040.2028049

[19] Z. Xu, K. Bai, S. Zhu, Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors, in: Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks, WISEC '12, ACM, New York, NY, USA, 2012, pp. 113–124. doi:10.1145/2185448.2185465.
URL http://doi.acm.org/10.1145/2185448.2185465

[20] R. van Rijswijk, E. Poll, Using trusted execution environments in two-factor authentication: comparing approaches, in: Proceedings of the Open Identity Summit 2013 (OID 2013), Lecture Notes in Informatics, Gesellschaft for Informatik, 2013, pp. 20–31.

[21] B. Cornillault, F. Dahan, Secure mode indicator for smart phone or pda.
URL Texas Instruments Inc

[22] B. Ozdenizci, K. Ok, V. Coskun, A tokenization-based communication architecture for hce-enabled nfc services, Mobile Information Systems 2016.

[23] A. Armando, A. Merlo, L. Verderame, Trusted host-based card emulation, in: High Performance Computing & Simulation (HPCS), 2015 International Conference on, IEEE, 2015, pp. 221–228.

[24] S. Blake-Wilson, A. Menezes, Unknown key-share attacks on the station-to-station (sts) protocol, in: International Workshop on Public Key Cryptography, Springer, 1999, pp. 154–170.

[25] Smartwatches locking methods: A comparative study, in: Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017), USENIX Association, Santa Clara, CA, 2017.
URL https://www.usenix.org/conference/soups2017/workshop-pr

[26] J. Yang, Y. Li, M. Xie, Motionauth: Motion-based authentication for wrist worn smart devices, in: 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), 2015, pp. 550–555. doi:10.1109/PERCOMW.2015.7134097.

[27] A. Lewis, Y. Li, M. Xie, Real time motion-based authentication for smartwatch, in: 2016 IEEE Conference on Communications and Network Security (CNS), 2016, pp. 380–381. doi:10.1109/CNS.2016.7860521.

[28] A. Buriro, B. Crispo, M. Eskandri, S. Gupta, A. Mahboob, R. Van Acker, Snapauth: A gesture-based unobtrusive smartwatch user authentication scheme, in: A. Saracino, P. Mori (Eds.), Emerging Technologies for Authorization and Authentication, Springer International Publishing, Cham, 2018, pp. 30–37.

[29] A. H. Johnston, G. M. Weiss, Smartwatch-based biometric gait recognition, in: 2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS), 2015, pp. 1–6. doi:10.1109/BTAS.2015.7358794.

[30] N. Al-Naffakh, N. Clarke, F. Li, P. Haskell-Dowland, Unobtrusive gait recognition using smartwatches, in: 2017 International Conference of the Biometrics Special Interest Group (BIOSIG), 2017, pp. 1–5. doi:10.23919/BIOSIG.2017.8053523.

[31] T. Nguyen, N. Memon, Tap-based user authentication for smartwatches, Computers & Security 78 (2018) 174 – 186. doi:https://doi.org/10.1016/j.cose.2018.07.001.
URL http://www.sciencedirect.com/science/article/pii/S0167

[32] M. Ehatisham-ul Haq, M. A. Azam, J. Loo, K. Shuang, S. Islam, U. Naeem, Y. Amin, Authentication of smartphone users based on activity recognition and mobile sensing, Sensors 17 (9). doi:10.3390/s17092043.
URL http://www.mdpi.com/1424-8220/17/9/2043

[33] M. Guerar, M. Migliardi, A. Merlo, M. Benmohammed, F. Palmieri, A. Castiglione, Using screen brightness to improve security in mobile social network access, IEEE Transactions on Dependable and Secure Computing 15 (4) (2018) 621–632. doi:10.1109/TDSC.2016.2601603.

[34] E. von Zezschwitz, A. De Luca, B. Brunkow, H. Hussmann, Swipin: Fast and secure pin-entry on smartphones, in: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI '15, ACM, New York, NY, USA, 2015, pp. 1403–1406. doi:10.1145/2702123.2702212.
URL http://doi.acm.org/10.1145/2702123.2702212

[35] M. Guerar, M. Migliardi, A. Merlo, M. Benmohammed, B. Messabih, A completely automatic public physical test to tell computers and humans apart: A way to enhance authentication schemes in mobile devices, in: 2015 International Conference on High Performance Computing Simulation (HPCS), 2015, pp. 203–210. doi:10.1109/HPCSim.2015.7237041.

[36] M. Guerar, A. Merlo, M. Migliardi, Completely automated public physical test to tell computers and humans apart: A usability study on mobile devices, Future Generation Computer Systems 82 (2018) 617 – 630. doi:https://doi.org/10.1016/j.future.2017.03.012.
URL http://www.sciencedirect.com/science/article/pii/S0167

[37] A. Bianchi, I. Oakley, V. Kostakos, D. S. Kwon, The phone lock: Audio and haptic shoulder-surfing resistant pin entry methods for mobile devices, in: Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction, TEI '11, ACM, New York, NY, USA, 2011, pp. 197–200. doi:10.1145/1935701.1935740.
URL http://doi.acm.org/10.1145/1935701.1935740

[38] T. Kwon, S. Na, Tinylock: Affordable defense against smudge attacks on smartphone pattern lock systems, Computers & Security 42 (2014) 137 – 150. doi:https://doi.org/10.1016/j.cose.2013.12.001.
URL http://www.sciencedirect.com/science/article/pii/S0167

[39] M. Guerar, A. Merlo, M. Migliardi, Clickpattern: A pattern lock system resilient to smudge and side-channel attacks, JoWUA 8 (2) (2017) 64–78.
URL http://isyou.info/jowua/papers/jowua-v8n2-4.pdf

[40] M. Guerar, M. Benmohammed, V. Alimi, Color wheel pin: Usable and resilient ATM authentication, J. High Speed Networks 22 (3) (2016) 231–240. doi:10.3233/JHS-160545.
URL https://doi.org/10.3233/JHS-160545

[41] A. D. Luca, E. von Zezschwitz, H. Hußmann, Vibrapass: secure authentication based on shared lies, in: CHI, ACM, 2009, pp. 913–916.

[42] A. De Luca, K. Hertzschuch, H. Hussmann, Colorpin: Securing pin entry through indirect input, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '10, ACM, New York, NY, USA, 2010, pp. 1103–1106. doi:10.1145/1753326.1753490.
URL http://doi.acm.org/10.1145/1753326.1753490

[43] D. Nyang, A. Mohaisen, J. Kang, Keylogging-resistant visual authentication protocols, IEEE Transactions on Mobile Computing 13 (11) (2014) 2566–2579. doi:10.1109/TMC.2014.2307331.

[44] I. Oakley, J. H. Huh, J. Cho, G. Cho, R. Islam, H. Kim, 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, ACM, New York, NY, USA, 2018, pp. 75–87. doi:10.1145/3196494.3196555.
URL http://doi.acm.org/10.1145/3196494.3196555

[45] T. V. Nguyen, N. Sae-Bae, N. Memon, Draw-a-pin: Authentication using finger-drawn pin on touch devices, Computers & Security 66 (2017) 115 – 128. doi:https://doi.org/10.1016/j.cose.2017.01.008.
URL http://www.sciencedirect.com/science/article/pii/S0167404817300123

[46] B. Hutchins, A. Reddy, W. Jin, M. Zhou, M. Li, L. Yang, 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, ACM, New York, NY, USA, 2018, pp. 101–115. doi:10.1145/3196494.3196543.
URL http://doi.acm.org/10.1145/3196494.3196543

[47] H. Yoon, S. Park, K. Lee, Exploiting ambient light sensor for authentication on wearable devices, in: 2015 Fourth International Conference on Cyber Security, Cyber Warfare, and Digital Forensic (CyberSec), 2015, pp. 95–100. doi:10.1109/CyberSec.2015.27.

[48] T. Kuribara, B. Shizuki, J. Tanaka, Vibrainput: Two-step pin entry system based on vibration and visual information, in: CHI '14 Extended Abstracts on Human Factors in Computing Systems, CHI EA '14, ACM, New York, NY, USA, 2014, pp. 2473–2478. doi:10.1145/2559206.2581187.
URL http://doi.acm.org/10.1145/2559206.2581187

[49] M.-K. Lee, H. Nam, D. K. Kim, Secure bimodal pin-entry method using audio signals, Computers Security 56 (2016) 140 – 150. doi:https://doi.org/10.1016/j.cose.2015.06.006.
URL http://www.sciencedirect.com/science/article/pii/S0167404815000929

Meriem Guerar Biography text here.

Mauro Migliardi Biography text here.

Luca Verderame Luca Vallerini is a Master Student in Computer Engineering at the University of Padua.

Francesco Palmieri Biography text here.

Alessio Merlo Biography text here.