A feasibility study on pairing a smartwatch and a mobile device through multi-modal gestures

Dario Bernardi

School of Science

Thesis submitted for examination for the degree of Master of Science (Technology), Security and Mobile Computing.

Espoo 27.5.2019

Thesis supervisors:

Prof. Mario Di Francesco
Prof. Elena Dubrova

Thesis advisor:

D.Sc. Rainhard Findling

Aalto University
School of Science

AALTO UNIVERSITY
SCHOOL OF SCIENCE

ABSTRACT OF THE
MASTER'S THESIS

Author: Dario Bernardi

Title: A feasibility study on pairing a smartwatch and a mobile device through multi-modal gestures

Date: 27.5.2019 Language: English Number of pages: 6+53

Department of Computer Science

Professorship: Mobile Cloud Computing

Supervisors: Prof. Mario Di Francesco, Prof. Elena Dubrova

Advisor: D.Sc. Rainhard Findling

Pairing is the process of establishing an association between two personal devices. Although such a process is intuitively very simple, achieving a straightforward and secure association is challenging due to several possible attacks and usability-related issues. Indeed, malicious attackers might want to spoof the communication between devices in order to gather sensitive information or harm them. Moreover, offering users simple and usable schemes which attain a high level of security remains a major issue. In addition, due to the great diversity of pairing scenarios and equipment, achieving a single, usable, secure association for all possible devices and use cases is simply not possible.

In this thesis, we study the feasibility of a novel pairing scheme based on multi-modal gestures, namely, gestures involving drawing supported by accelerometer data. In particular, a user can pair a smart-watch on his wrist and a mobile device (e.g., a smart-phone) by simply drawing with a finger on the screen at the device.

To this purpose, we developed mobile applications for smart-watch and smart-phone to sample and process sensed data in support of a secure commitment-based protocol. Furthermore, we performed experiments to verify whether encoded matching-movements have a clear similarity compared to non-matching movements.

The results proved that it is feasible to implement such a scheme which also offers users a natural way to perform secure pairing. This innovative scheme may be adopted by a large number of mobile devices (e.g., smart-watches, smart-phones, tablets, etc.) in different scenarios.

Keywords: Mobile devices, Security, Pairing, Signal, Processing, Multi-modal Gestures

Acknowledgements

Thanks to my family and, above all, my parents for their trust in me and their daily, unceasing, affectionate support.
To all my wonderful friends in Italy, for making me feel at home although thousands of kilometers away.
To the new friends in Sweden and Finland, amazing persons and travel companions.

Thanks to my supervisor at Aalto, Professor Mario Di Francesco, for giving me the opportunity to work on this topic, for his meticulous feedback and continuous encouragement.
To my advisor, Doctor Rainhard Findling, for his talent, expertise, attention and patience during my struggles along the path.
To my supervisor at KTH, Professor Elena Dubrova, for helping me with final corrections and submission process at my home university.

Last but not least, thanks to Sweden and Finland, tiny, peaceful, powerful, visionary countries, for giving me the opportunity to study abroad in stimulating and stunning environments, and to know new cultures, friends and brilliant minds.

At the end of the day, this has been one of the best experiences of my life.

Espoo, 27.5.2019

Dario Bernardi

Abbreviations and Acronyms

API	Application Programming Interface
CBC	Cipher Block Chaining
CKP	Candidate Key Protocol
DH	Diffie Hellman
FFT	Fast Fourier Transform
FM	Frequency Modulation
IFFT	Inverse Fast Fourier Transform
IV	Initialization Vector
LPF	Low-Pass Filter
LURD	Left Up Right Down
KDE	Kernel Density Estimate
MAC	Message Authentication Code
MANA	MANual Authentication
MITM	Man In The Middle
NMK	Network Membership Key
OOB	Out Of Band
RF	Radio Frequency
SHA	Secure Hash Algorithm
ShaVe	Shaking for Verification
ShaCK	Shaking to Construct a Key
TEK	Temporary Encryption Key
TV	TeleVision
USB	Universal Serial Bus
WLAN	Wireless Local Area Network

Contents

1 Introduction

Mobile devices have become the most widely used technology in our daily life, to the point of being considered an extension of the human body. These are not only represented by smart-phones and tablets, but also by wearable devices, such as smart-watches.

Among wearable devices, these in particular are seeing a tangible growth reaching 120.2 million units by 2022 [1]. The enormous success of smart-watches is due to the high number of applications developed during the last years. In fact, they are not only used for a faster glance to notifications, but also for performing secure payments [2, 3, 4], tracking body health and activities [5, 6, 7, 8], and controlling devices remotely [9, 10, 11]. Moreover, the scientific research is continuously studying possible new applications involving smartwatch sensors, for instance, smartwatch-based finger-writing [12] and smartwatch-based diet monitoring [13].

To make the user's information ubiquitous and synchronized among mobile devices, producers and service providers are constantly improving and enriching devices inter-communication within the boundaries of their ecosystems. On the other hand, they must ensure that user's information does not leak to unwanted third parties. To this purpose, establishing a secure communication between devices is fundamental for protecting data from intruders. Therefore, secure pairing – the process of establishing a secure association between two personal devices – plays a crucial role.

1.1 Research topic

Ordinary users are often annoyed when setting up pairings, as the process may be cumbersome or complicated [14]. For instance, asking users to type on one device a short pin-code shown by the other device involved in the pairing is considered time-consuming and error-prone. Thus, achieving an association that ensures both security and usability [15] is challenging.

In the last decade, several studies focused on finding such association, also driven by the growing number of sensors with which mobile devices are equipped, like fingerprint readers, capacitors, accelerometers, gyroscopes, microphones, and so forth. Some of these studies considered the possibility to perform pairing based on listening to the same ambient audio [16, 17, 18, 19, 20, 21] or radio signals [22, 23], requiring limited user intervention. Others introduced novel pairing protocols based on shaking devices together [24, 25, 26, 27] or performing simultaneous drawings on their screens [28]. However, due to the great diversity of pairing scenarios and equipment, achieving a single, usable, secure association for all possible devices and use cases is simply not possible.

1.2 Research goals and methodology

Given the above-mentioned challenges, we aim to study the feasibility of a novel pairing scheme based on multi-modal gestures, namely gestures involving drawing supported by accelerometer data. In particular, a user can pair a smart-watch on his wrist and a mobile device (e.g., a smart-phone) by simply drawing with a finger on the screen of the mobile device.

This research is challenging since it is hard to verify that finger and wrist movements translate into comparable signals from the smart-watch and the smart-phone. In fact, it is necessary to find an effective approach to compare accelerometer data (i.e., raw acceleration) against drawing data (i.e., coordinates on the screen).

To this end, the major goals of this thesis are the following:

- Accurately sample wrist and finger movements.
- Design a pre-processing pipeline to filter, transform and encode collected data.
- Evaluate whether our approach is effective to distinguish matching pairings from non-matching ones.

In this study, we adopt a qualitative research methodology and use an empirical method [29], investigating previous research on secure pairings and exploring strategies for implementing the above-mentioned novel scheme. Moreover, we operate with an inductive approach. In fact, after implementing the needed applications and supporting tools, we perform ad hoc experiments to infer the effectiveness of our approach from a statistical analysis of obtained results. If the statistical inference is successful, the feasibility and security of the new protocol are proved.

1.3 Contributions

The contributions of this work are the following:

- Developing mobile applications to sample wrist and finger movements at smart-watch and smart-phone.
- Designing a processing pipeline to filter and transform collected data.
- Introducing a novel method for encoding the processed data.
- Performing ad hoc experiments to simulate pairings in a simplified test-bed.
- Analyzing results to prove the effectiveness of our approach and demonstrating the feasibility and security of the novel pairing scheme based on multi-modal gestures.

1.4 Structure of the thesis

The rest of the thesis is organized as follows. Chapter 2 surveys existing strategies to perform secure key establishment between two devices as well as secure protocols that use *fuzzy data* as secrets for pairing. Chapter 3 describes context and strategies for sampling and pre-processing data to perform pairing through multi-modal gestures. Chapter 4 presents an experimental evaluation of the proposed scheme. Finally, Chapter 5 concludes the thesis by summarizing the related achievements and outlining directions for future works.

2 Secure device paring

In the last decade, secure pairing gained increasing attention due to the exceptional growth in both mobile devices and short-range wireless communication technologies, such as Bluetooth, WiFi and ZigBee.

Pairing is the process of establishing an *association* between two personal devices. Although such a process is intuitively very simple, achieving a straightforward and secure association is challenging due to several possible attacks [30] and usability-related issues [14]. Indeed, each device may be equipped with different hardware capabilities, or may perform secure association in completely different scenarios with application-specific security requirements. Thus, a single, general-purpose *association model* (i.e., the procedure within the association which requires direct involvement of the user) which attains a high level of security, usability and user experience [15] for all possible use cases is simply not possible.

Nowadays, most solutions for secure device pairing rely on *user-conditioning* for triggering the *association mode*. In other words, the involved devices do not accept any association request without any user-specific action (e.g., the association mode is prompted only pressing a specific button).

Moreover, to accomplish a secure association the two devices must perform mutual *identification* and *authentication* without having any pre-shared information so as to exclude passive malicious eavesdroppers and active MITM (Man-In-The-Middle) attackers [31]. The final goal is to establish and share a secret key with the intended party so as to encrypt their intercommunication and, if needed, perform access control.

2.1 Key establishment methods

Ensuring high security standards while establishing a shared secret is fundamental to adopt secure protocols in real scenarios. On the other hand, the complexity and diversity of these scenarios has led researchers to find different strategies to perform secure key establishments.

In this section we detail these strategies, starting from the conceptually easiest one, namely having a dedicate unspoofable channel to transmit the key, to the more complex which make use of symmetric or asymmetric cryptography for agreeing on a shared secret.

As shown in Figure 1, key establishment can be performed with different strategies [30]: either by transferring the key over an OOB (Out-Of-Band) channel (i.e., an unspoofable and secure channel physically under the direct control of the user, usually bidirectional) or by using more complex key agreement protocols.

In the former case (P1), one device selects a key that is transferred directly to the second device (e.g., using a USB stick or a direct physical connection).

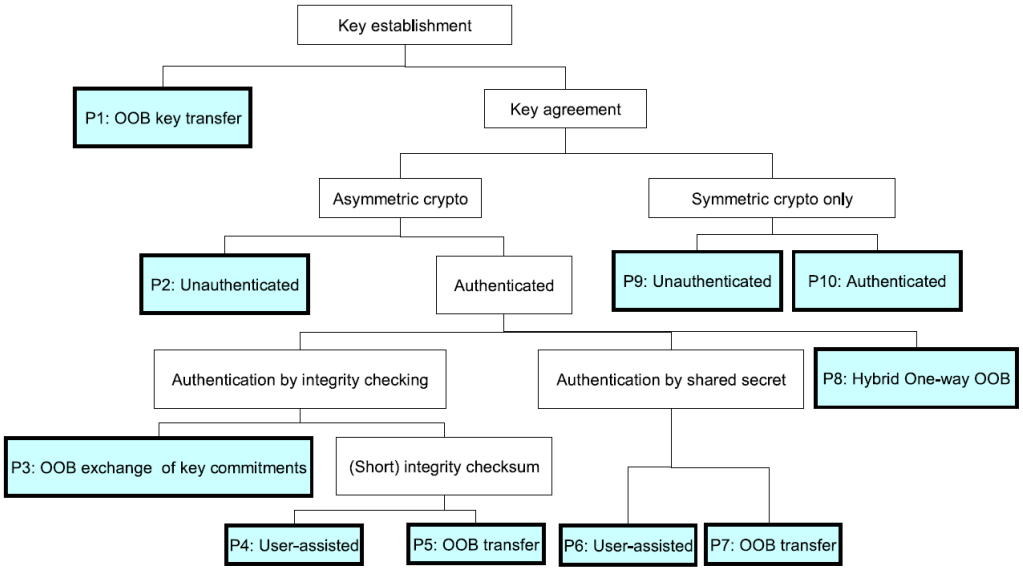


Figure 1: Classification of key establishment protocols [30].

The security of this procedure relies completely upon the security of the OOB channel for the key transfer.

The latter case includes several scenarios that involve either symmetric or asymmetric cryptography. These cases are detailed next.

2.1.1 Key agreements with symmetric cryptography

Key agreement using *symmetric cryptography* is adopted by HomePlugAV [32], a standard for in-home power-line communications that supports broadband multimedia applications. HomePlugAV provides both unauthenticated and authenticated symmetric key agreement modes.

The unauthenticated key agreement (P9), so-called *simple connect mode*, requires the user to trigger the association mode firstly on one of the two devices, so-called device, which is responsible to generate a NMK (Network Membership Key). Subsequently, the user conditions the second device. Before exchanging information, they agree on tone maps, namely a mechanism to compensate interference in the power-line channel, hence helping to detect possible MITM attackers as well. The two parties exchange nonces and generate a TEK (Temporary Encryption Key) by hashing them together. Eventually, the controller device encrypts the NMK by using the TEK and sends the secret material to the second device.

On the other hand, the authenticated key agreement, so-called *secure mode* (P10), requires the user to firstly type into the controller the pre-labeled passkey that comes with the device to pair. In this scenario, sufficiently long passkeys (e.g., at least 12 alphanumeric characters) are necessary to exclude possible exhaustive searches of the pre-labeled secret. Afterward, the controller identifies the new device, generates and encrypts an NMK that is eventually sent to the

other party.

Both HomePlugAV Secure mode and Simple connect mode narrow down possible passive attacks thanks to the inherent difficulty to filter the signal noise in locations outside the building.

2.1.2 Key agreements with asymmetric cryptography

Key agreements using *asymmetric cryptography* are based on a DH (Diffie-Hellman) key-exchange [33]. Since this key-exchange is known to be vulnerable to MITM attacks [34], these agreements must perform actions to detect and exclude possible active attackers, by correctly identifying and authenticating the intended party.

Then again, in some scenarios, the DH key-exchange is performed without authenticating the two parties at a later time (P2). Indeed, one of the devices participating in the pairing process (e.g., headphones using Bluetooth Secure Simple Pairing model [35]) may be equipped neither with display nor with a keypad.

Conversely, DH keys can be authenticated by using a short integrity checksum of the messages exchanged during the initial phase. On both devices, if the checksums result identical the authentication succeeds. In this respect, Figure 2 shows the verification phase between two parties which already performed the DH exchange. Functions h and f are both collision-resistant and one-way cryptographic hash functions (e.g., SHA-256 [36]). The function f is also mapped to a short human-readable string (e.g., 6 digits). The degree symbol (°) indicates that the received information may differ from the original due to a possible active attacker tampering the communication. In addition, this scheme makes use of commitments, that is, a hash of secret material which is disclosed to the counterpart at a later stage. The plain-text message that discloses the secret material previously sent in the commitment is called commitment opening. After opening the commitment, if Device B fails the verification of the commitment the protocol aborts.
The checksum comparison (i.e., $Va = Vb°$, $Va° = Vb$ in Figure 2) may be performed directly by the user (P4) (e.g., in the Bluetooth Secure Simple Pairing model [35] the user ensures that both devices display the identical 6-digit numeric value) or by using a physical OOB channel (P5) (e.g., through a USB stick or a direct physical connection). On the other hand, an active attacker may alter the communication and deny the pairing success. However, in order to deceive both devices, the attacker has to learn about devices' public keys in advance (i.e., PKa and PKb), and choose Ra and Rb so that $f(PKa, PKb°, Ra, Rb°) = f(PKa°, PKb, Ra°, Rb)$. Since the attacker is not able to affect the result of the hash function, the chance of a successful attack is 2^{-n}, where n is the length of the evenly distributed output of f.

In addition to these methods, the two devices may use an OOB channel (P3) to verify the commitments to the public keys exchanged during the DH phase. In other words, the two devices authenticate the other party by solely sending

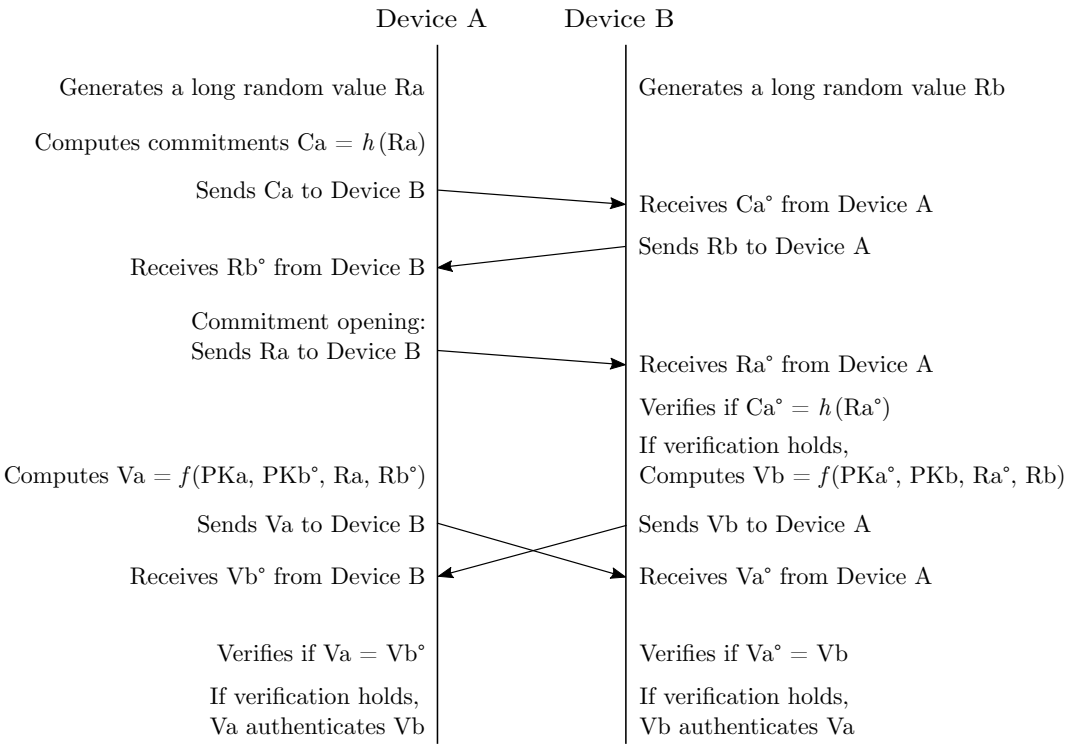


Figure 2: Verification of public keys by short integrity checksum.

the commitments (i.e., public keys or their hashes) over the OOB channel. In this scenario, the security critically depends upon the OOB channel being unspoofable, and upon strong cryptographic hash functions.

Instead of using commitments to public keys over an OOB channel, the two devices can authenticate each other by including a pre-shared short secret S in their commitments. The secret may be input on both devices by the user. Alternatively, one device may show the secret and ask the user to enter it on the second device (P6).

In addition to these, the protocol may use an OOB channel for transferring the secret S between the two devices (P7). In this respect, Suomalainen at al. describe a variant [30] of the MANA-III protocol [37] that uses a k-round authentication protocol of public keys. Assuming that the two devices already obtained the one-time secret, they split S into k pieces. As shown in Figure 3, given i=1...k, both devices demonstrate the knowledge of the i^{th} part of S by including S_i into their commitments during each round.

In addition to the secret portion, for each round commitments like $\mathrm{C_i a} = h(1, \mathrm{PKa}, \mathrm{PKb}^\circ, \mathrm{S_i}, \mathrm{R_i a})$ are composed by an identifier (e.g., digit *1* for device A and *2* for Device B), devices' public keys, and a long random number. The identifier is needed for excluding mirroring attacks.

To obtain S_i, a MITM can fool Device A by using garbage data instead of $C_i b$ from Device B, and retaining $R_i a$ in the next message by Device A. The attacker can brute-force S_i by knowing the identifier of Device B, PKa, PKb, and $R_i b$. However, after successfully computing S_i, she would still need the other *k-1*

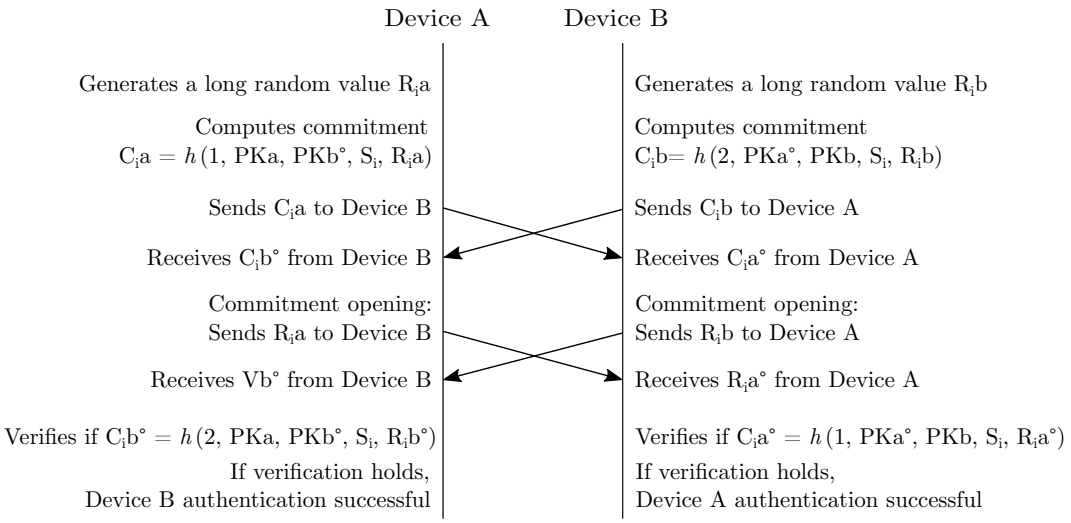


Figure 3: Verification of public keys by pre-shared short secret at the i^{th} round.

portions for compromising the protocol. As a result, given X={probability to discover the whole S}, and Y={probability to discover the single S_i portion}, the resulting conditional probability for a successful attack is:

$$P(X \mid Y) = \frac{P(X \cap Y)}{P(Y)} = \frac{\frac{1}{2^n}}{\frac{1}{2^{\frac{n}{k}}}} = \frac{1}{2^{n-\frac{n}{k}}}$$

Moreover, even when computing the secret correctly, the attacker would be able to compromise the protocol only once as long as S is a *one-time* passkey. However, as pointed out by Sethi at al. [28], the round-based MANA-III protocol is limited due to the difficulty to ensure that both devices divide the secret S in identical portions, and the entropy does not downgrade in any of these. Moreover, the protocol must demonstrate that any portion S_i has no dependency on other portions S_j, with $j>i$, that might reveal secret information. Furthermore, solutions involving commitments strictly rely on the timing of the messages. Indeed, before the opening phase, both devices need to ensure their commitment reaches to other party. Otherwise, an active attacker would need just a few seconds to exhaustively search for the S_i portion, thereby compromising the whole protocol.

As a solution, the check on correct timing can be delegated to the user. On the other hand, there is no intuitive user interface that denies the user to use shortcuts for transmitting plain-text random numbers before the actual delivery of the commitments. For this reason, Sethi at al. [28] introduce a one-round *time-based* commitment scheme that uses a pre-shared secret and timers for the authenticating the devices after the DH exchange, enforcing the correct timing of the messages. Accordingly, as shown in Figure 4, the two devices perform a mutual synchronization and set a t_0 reference time, based on a specific event triggered by the user. Depending on the user precision in the

synchronization, t_0 might differ in the two devices (for simplicity, $t_{A,0}$ and $t_{B,0}$ are simultaneous in Figure 4). Moreover, the devices agree on timers Δ_1 (i.e. maximum time interval to receive the commitment) and Δ_2 (i.e. maximum time interval to receive the commitment opening). The timers are experimentally set to let the exchanged material arrive, including possible network delays in modern networks and subtle imprecision in the synchronization. Subsequently, the devices generate their commitments by hashing their identifier, the secret S, a fresh long random number, and the key generated during the DH phase. If the devices do not receive any commitment within a $t_0 + \Delta_1$ time frame, they abort the pairing process. Otherwise, using the key generated during the DH phase, they open their commitments by encrypting and transmitting their identifier, the random number, and the secret S. The encryption is needed to limit statistical attacks. In case the devices do not receive any encrypted material within time $t_0 + \Delta_1 + \Delta_2$, they abort the pairing process. Otherwise, they perform a verification of the received commitments and, if positive, they authenticate the other party.

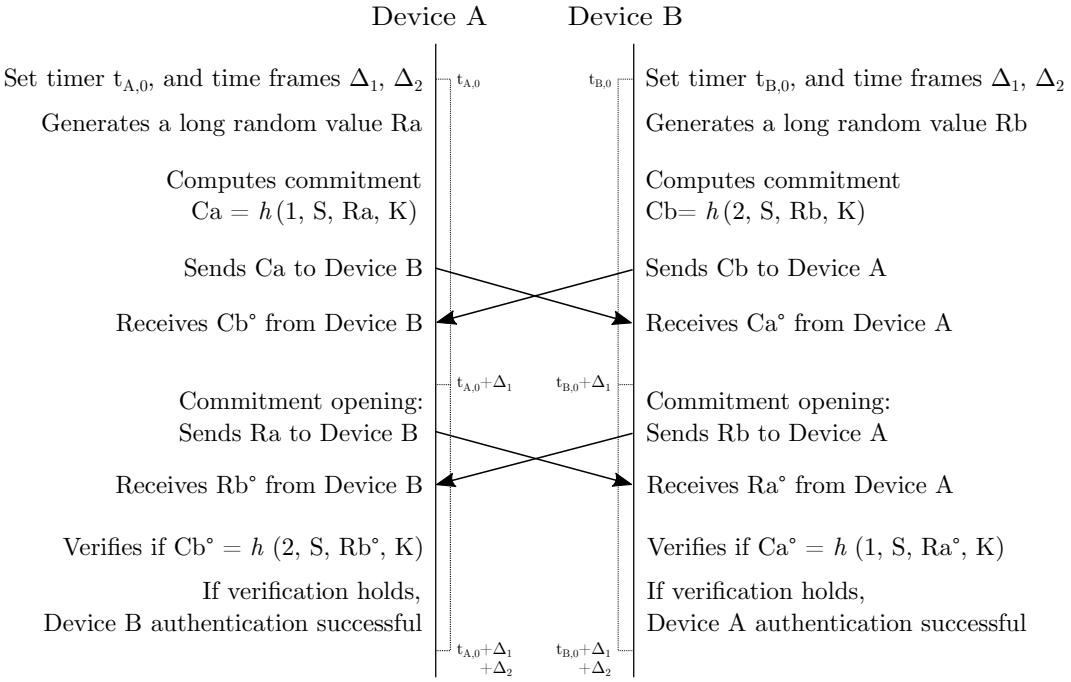


Figure 4: Time-based commitment protocols.

As explained in Section 2.2, the literature has proposed a considerable amount of key agreement protocols based on authenticated asymmetric cryptography using a pre-shared secret. To simplify the *association model*, hence reducing the direct involvement of the user in the pairing process, many of these protocols use *fuzzy* secrets for deriving the pre-shared short secrets. In other words, the two devices derive exactly the same secret S for authenticating each other by using material they both know only approximately (e.g., audio or radio signals in the surrounding environment).

In addition to the previous methods, it is possible to authenticate the two devices by using a hybrid authentication in case the used OOB channel is one-way only (P3). As shown in Figure 5, in this scenario Device A sends a commitment of his public key and a long random value along with a shared secret *S*. After ensuring the material has been delivered, Device A opens its commitment by transmitting *PKa* and *Ra* on the in-band channel. Once Device B obtains this information, it performs the verification using the previously received material. If verification is successful, Device B sends back on the in-band channel a message authentication code (i.e $MAC = c(message,key)$) of devices' public keys and random numbers by using the secret *S*. Therefore, the protocol is considered secure as long as the OOB channel is unspoofable, and c and f are strong cryptographic functions.

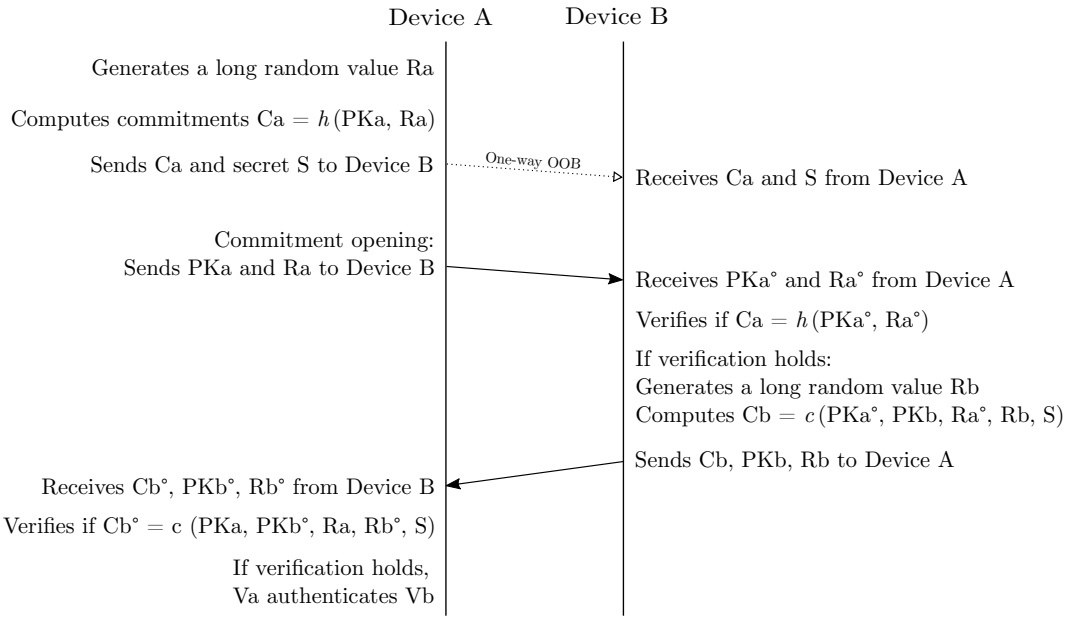


Figure 5: Hybrid authentication.

2.2 Pairing using fuzzy data

During the last years, thanks to the enormous interest towards mobile devices by commercial actors, research has been motivated to discover novel, simple, user-friendly ways to perform secure pairing between devices.

As mentioned later in this section, a large class of these protocols utilizes *fuzzy data* as secrets for pairing. *Fuzzy data* is information that devices involved in the pairing process know only approximately, and refer to data gathered by devices' sensors listening to the same noisy environment. The aim is to independently derive the same exact cryptographic material out of the fuzzy data.

Nowadays mobile devices (e.g., smart-phones, smart-watches, etc.) are equipped with a growing number of highly accurate sensors, hence offering

researchers the chance to invent new protocols using fuzzy data. Among these, we present the most interesting ones which gather fuzzy data from devices' microphones (Section 2.2.1), radio interfaces (Section 2.2.2), accelerometers (Section 2.2.3), and touch-screens (Section 2.2.4).

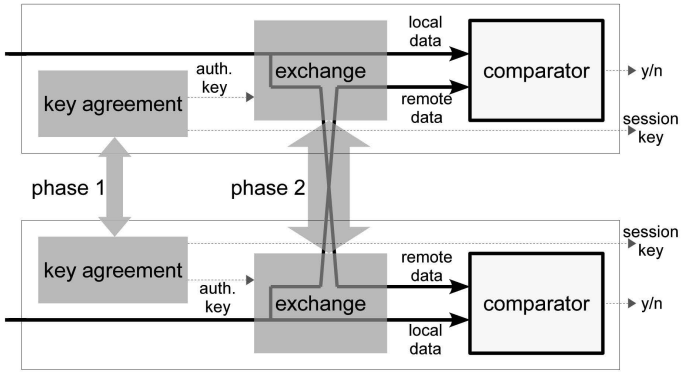


Figure 6: Fuzzy data used for session key authentication [24].

As shown in Figure 6, fuzzy data can be used for authenticating the session key derived from the previous DH agreement (P6 and P7 in Figure 1). For example, the MANA-III protocol variant described in Figure 3 may use fuzzy data gathered by sensors as a pre-shared secret S. After the commitment opening, both devices independently compare the received portion of secret S with their own version.

For these reasons, the main challenge of this scenario is to establish a suitable *metric* to correctly compare the own version with the received one, and an appropriate *threshold* that ideally zeros the false-positives and minimize the number of false-negatives.

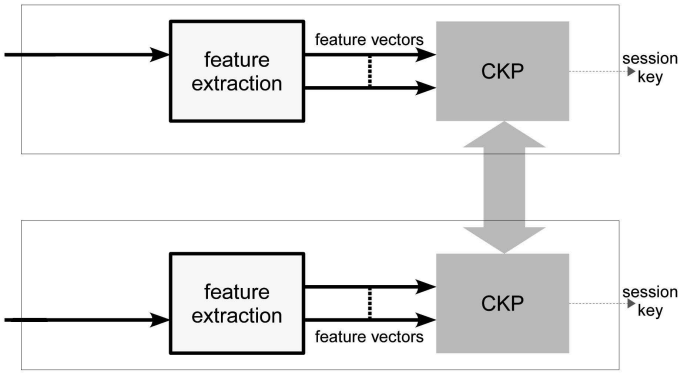


Figure 7: Fuzzy data used for shared session key construction [24].

On the contrary, as depicted in Figure 7, the so-called *fuzzy cryptography* avoids the key agreement phase, using fuzzy data for direct extraction of the session key (P9 in Figure 1). In other words, both devices use multiple steps so as to interactively build their secret material. In particular, they

firstly exchange fuzzy data feature vectors (i.e., vectors that contain important information about the related raw data), usually along with correction codes. Then, the CKP (Candidate Key Protocol) continuously seeks for matching feature candidates at a sufficient rate. If successful, the CKP eventually builds the same strong session key on both devices.

In any case, using fuzzy data for generating secrets is challenging. In fact, the security of this model strictly depends upon the robustness of generated key in terms of its entropy. On the other hand, calculating the entropy of the generated secrets is often complicated or even impossible. Moreover, different device may have sensors with very diverse characteristics and accuracy making complicated to utilize this model.

2.2.1 Fuzzy data on audio signals

As investigated by different studies [16, 17, 18], a successful way to extract fuzzy data and generate secret material is to utilize audio signals as OOB channel. Since audio channels are inherently range-restricted, they naturally reduce the attack surface of malicious intruders.

Sigg et al. [19, 20] introduced a secure pairing protocol that uses fuzzy cryptography based on spontaneous audio-sensoring through devices' microphones.

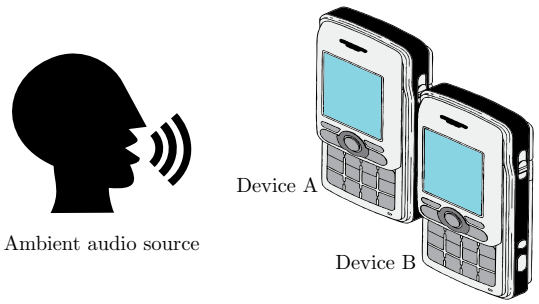


Figure 8: Spontaneous audio-sensoring protocols.

Firstly, the devices perform a proximity scan via Bluetooth so as to detect possible target devices for pairing. Then, the two devices involved in the pairing listen to the surrounding ambient audio signal. In order to adjust possible recording delays and to effectively perform synchronization, the devices use a short preset audio pattern for detecting the actual beginning of the original signal, discarding possible noise. Eventually, they compute an audio fingerprint, that is, a digital summary of the original signal able which identifies it with high probability. Possible errors in fingerprints are corrected by using correction codes. Once corrected, the fingerprints are used as a seed to generate the final key for encrypting the subsequent devices' communication. The security of the protocols relies on the fact that the closer the devices are the more similar the fingerprints result, and that the entropy in the ambient audio scenarios is high.

Similarly, Goodrich et al. [21] developed Loud and Clear, a human-assisted pairing system that uses an audio channel for authenticating previously exchanged public keys, for instance, over Bluetooth or 802.11 WLANs [38]. In

this scenario, one device must be provided with an audio interface (e.g., a speaker). The rationale is to ease the alphanumeric hash verification of the public keys which, in most of the cases, results annoying and cumbersome for the users. In other words, the hash is translated into words that complete pre-built, syntactically-correct, English-like templates. For achieving this, the entire hash of the public key is split into 10-bit portions, each of them converted into an integer which generates a human-readable word (e.g., SHA-1 hash [36] with entropy of 80 bits is divided into 8 portions, resulting in 8 different words).

As an example, let us consider a unidirectional authentication between two smart-phones. The first device (i.e., personal device) transmits its public key to the other party over a Bluetooth channel and displays the sentence to match. After receiving the public key, the second device (i.e., target device) performs the hash of it and generates the sentence. Then, the second device reads aloud the sentence through its speaker. If verification holds, the personal device authenticates the target.

To ensure strong security, this scheme must ensure that slightly different 10-bit sequences (e.g., difference of only 1 bit) produce as much phonetically-distant as possible words. Moreover, the entropy strictly depends upon the length of the output of the hash function, which impacts on the number of words to be generated.

2.2.2 Fuzzy data on radio signals

A different approach was proposed by Amigo [22], a protocol that relies on WiFi signals only. Amigo uses fuzzy data out of WiFi signals for extracting secret material in order to authenticate a previously exchanged DH key, as shown in Figure 6.

In contrast, ProxiMate by Mathur et al. [23] extracts the cryptographic key from different kinds of ambient RF (Radio Frequency) signals (e.g., WiFi Access Point, cellular base radio station, TV signal, or FM radio station). Moreover, ProxiMate follows the fuzzy cryptography paradigm (Figure 7), hence it extracts the session key directly from radio signals.

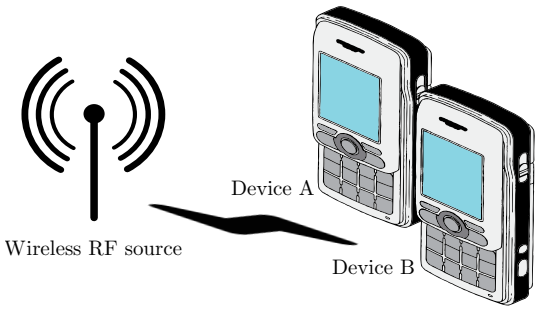


Figure 9: Proximity pairing protocols based on radio signals.

The reliability of this scheme is based on the high correlation of signals in near proximity. Moreover, although wireless signals are affected by reflections

and scatterings in the time domain, their temporal variation does not affect the correlation at times $t + \delta$, as long as δ is small. Furthermore, the authors demonstrate this schema is even resistant to attacks where attackers have control of the wireless source.

More specifically, the two devices periodically sample and demodulate the ambient radio signal. Then, the first device encodes the measurements at pre-defined time intervals so as to ensure bit independence in the encoding. Subsequently, the first device sends a snippet of the generated code along with a correction code to the other party. The second device uses the snippet for time synchronization. After performing the synchronization, the second device encodes its measurements, similarly to the first device. The correction code is used to derive the identical key seed. Eventually, both devices generate the same exact key for encrypting the subsequent communication by using the seed.

2.2.3 Fuzzy data on movement signals

In the last decade, a large family of studies regarding device movement sensoring [24, 39, 40, 25, 41, 42, 26, 27] has been conducted in order to infer practical applications using acceleration data.

Among these, many studies focused on exchanging secret material by shaking devices together. This approach has multiple advantages. Firstly, shaking devices together requires minimal user involvement with no need for previous training and specific user interface. In addition, compared to wireless or audio channels, shaking devices together naturally creates a range-restricted channel that enforces user control over devices. Furthermore, acceleration movements are difficult to tamper [24]. Moreover, acceleration sensors are nowadays integrated into the vast majority of mobile devices. As a result, this approach can be implemented in most devices, hence improving security as well as usability.

Figure 10: Shaking devices together [24].

Mayrhofer and Gellersen [24] introduced a novel secure pairing schema based on shaking devices together in which the generated secret material can be used in two different flavors, ShaVe (Shaking for Verification) and ShaCK (Shaking to Construct a Key). The former uses the secret material for verifying a previously exchanged session key, as shown in Figure 6. The latter follows the fuzzy cryptography paradigm, according to Figure 7.

Both cases are challenging due to the likely lack of temporal alignment of acceleration time series. Moreover, in most cases, acceleration time series result similar but not identical at the two devices, due to the different accuracy of sensors and due to plausible slightly different shaking movement. For these reasons, as shown in Figure 11, in both scenarios devices need temporal and spatial alignment after acceleration data sampling.

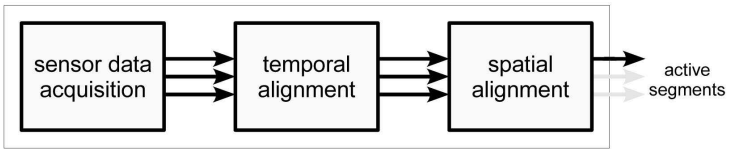


Figure 11: Acquisition and pre-processing of acceleration data on both devices [24].

In other words, data are independently gathered by the two devices at a preset sampling rate (e.g., f=100-600 Hz), considering all the three axes. To detect the actual beginning of user shaking and to start the authentication process, both ShaVe and ShaCK take into account the variance of the acceleration. Basically, whenever devices observe that the variance exceeds a pre-established threshold in a specific time frame, an authentication active segment is in place. On the other hand, the end of an active segment is detected differently in ShaVe and ShaCK. The former always considers time frames of fixed length (e.g., 3 seconds). The latter seeks for the drop of user motion under a certain level. Furthermore, since sensors' calibration may vary in each and every device, spatial alignment is needed for the following correct comparison of segments. To do so, samples are firstly normalized. Then, the acceleration along the three axes is combined into a single dimension by computing the magnitude of the vectors.

After pre-processing of acceleration data, ShaVe implements the authentication protocol based on the MANA-III variant depicted in Figure 3. During the DH phase the two devices, say Device A and Device B, generate two secure keys K^{Auth} and K^{Sess}. Given the knowledge on one of these keys, it is computationally unfeasible to infer the other. Then, both devices group their acceleration active segments so as to obtain vectors a and b, where $len(a) = n$ and $len(b) = m$. Both devices prepend one additional block containing a local identifier, Id_A and Id_B (e.g., their Bluetooth MAC address), for excluding mirroring attacks. By using K^{Auth} and a random IV (Initialization Vector) in CBC (Cipher Block Chaining) mode, they compute the cryptograms $c = Enc(K^{Auth}, IV, Id_A|a)$ and $d = Enc(K^{Auth}, IV, Id_B|b)$ respectively, where $len(c) = n + 1$ and $len(d) = m + 1$. After computing the encrypted material, they split c and d in two halves (i.e., A_1 and A_2, B_1 and B_2) which are transmitted in two rounds in order to avoid attackers to gain knowledge about any part of the plain-text. After receiving the cryptograms from their counterpart, both devices resemble the halves and verify whether the active segments match.

In any case, they communicate success or failure to the other device.

In contrast, devices in ShaCK mode do not perform the DH agreement (Figure 7). As a result, the encryption key K_{ab} needs to be extracted through the CKP directly. For this reason, devices hash their acceleration feature vectors h. Device A sends to Device B its $h + s$, where s is a random salt value. Once received, Device B checks existing matching vectors against its own partner-specific list of feature vectors. If multiple matching feature vectors are found, once sufficient entropy has been gathered, Device B concatenates all the matching vectors along with a publicly-known constant C and hashes them to build a candidate key K_{ab}, and transmits it to Device A. Then, if no messages have been lost, Device A should be able to generate the construct the same K_{ab} and verify the correctness of the material obtained by Device B. In any case, Device A acknowledges success or failure to Device B.

Both ShaVe and ShaCK bring their own advantages and disadvantages. Indeed, even though ShaVe provides stronger security as it is based on an extensively studied cryptographic design, ShaCK is computationally lightweight and more interactive. Moreover, since ShaCK provides a less restrictive design, it is suitable in cases when shorter shaking is needed.

In subsequent research, Findling et al. [41] extended the study of Mayrhofer and Gellersen [24] using conjoint device shaking for transferring the authentication state among devices over a pre-established secure channel. In other words, given an existing secure channel over which the two devices already encrypt their communication, the token-device (e.g., a smart-watch that is most probably unlocked for long periods of time) transfers its authentication state to the other device (e.g., a smart-phone) so as to automatically unlock it. Basically, the two devices independently and continuously measure and derotate their 3D accelerations. If the variance of the acceleration magnitude exceeds a preset threshold within a sliding time window (e.g., 2 seconds), the devices detect an active segment and trigger the verification phase. Both measurements are then aggregated on one device, usually the locked device. The receiving device applies a bandpass filtering and a frequency collapsing function. Subsequently, it computes a scalar metric value out the two active segments. Eventually, according to pre-established individual frequency weights, the target device decides whether to accept or reject the unlocking.

2.2.4 Fuzzy data on screen coordinates

As already introduced in Section 2.1, in their research Sethi et al. [28] proposed a novel pairing protocol based on fuzzy data gathered from devices equipped with touch-screens or touch-pads (e.g., smart-phones, printers, etc.). Basically, as shown in Figure 12, the user draws simultaneously on both touch-sensitive surfaces by using two fingers of the same hand. If the two drawings are considered similar, the protocol allows the pairing.

Drawings are mainly characterized by x and y spatial coordinates, and do not have a natural time axis. Moreover, APIs (Application Programming Interface)

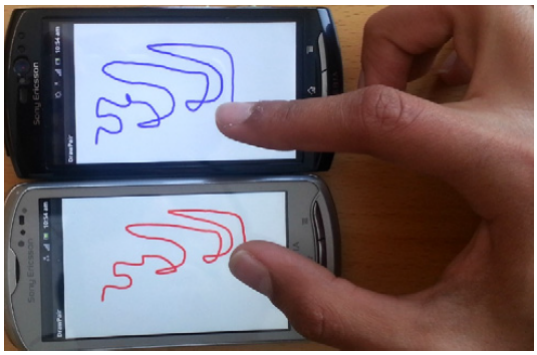


Figure 12: Synchronized drawings on two distinct devices [28].

for obtaining spatial coordinates as a function of time do not sample at a consistent frequency. Thus, the protocol would need mechanisms to approximate existing interpolation errors and to ensure continuous synchronization between devices, becoming cumbersome to implement. Furthermore, there are no reliable ways to calculate drawing entropy in the existing literature.

For these reasons, the authors of [28] discarded the fuzzy cryptography approach (Figure 7) and decided to use the secret material extracted from drawings for authenticating a previously agreed session key. They proposed a one-round time-based commitment scheme with approximate synchronization which does not require to split the secret, hence reducing the amount of exchanged messages and the complexity of protocol implementation.

The scheme depicted in Figure 13 is an extension of the protocol shown in Figure 4 and consists of four different phases.

The two devices firstly agree on a session key K^{Sess} through a DH exchange. As soon as the user finishes to draw and lifts his fingers from the touch-sensitive surfaces, both parties extract the secret material from the recorded drawings, as described later in this section, and take a reference time (i.e., $t_{A,0}$ for Device A, $t_{B,0}$ for Device B). The reference times might differ on the two devices depending on the user precision in lifting fingers from the surfaces. Moreover, the two devices agree on two timers, Δ_1 (i.e., the maximum time interval for delivering the commitment) and Δ_2 (i.e., the maximum time interval for correcting possible differences between $t_{A,0}$ and $t_{B,0}$). Since the security of commitment-based protocols significantly depends on the order of the messages, timers are a fundamental characteristic of this protocol. That is, the commitment opening must be performed after receiving the counterpart's commitment.

For simplicity, we now consider Device A only (Device B works similarly). Device A picks a long fresh random number r_A (e.g., 128 bits long). It computes its commitment by hashing a local identifier Id_A so as to exclude mirroring attacks (e.g., its own Bluetooth Media Access Control address), the extracted secret material v_A, the previously generated random number r_A, and the session key K^{Sess}. As soon as the commitment is computed, Device A transmits it to Device B. In the meanwhile, Device A waits for the counterpart's commitment.

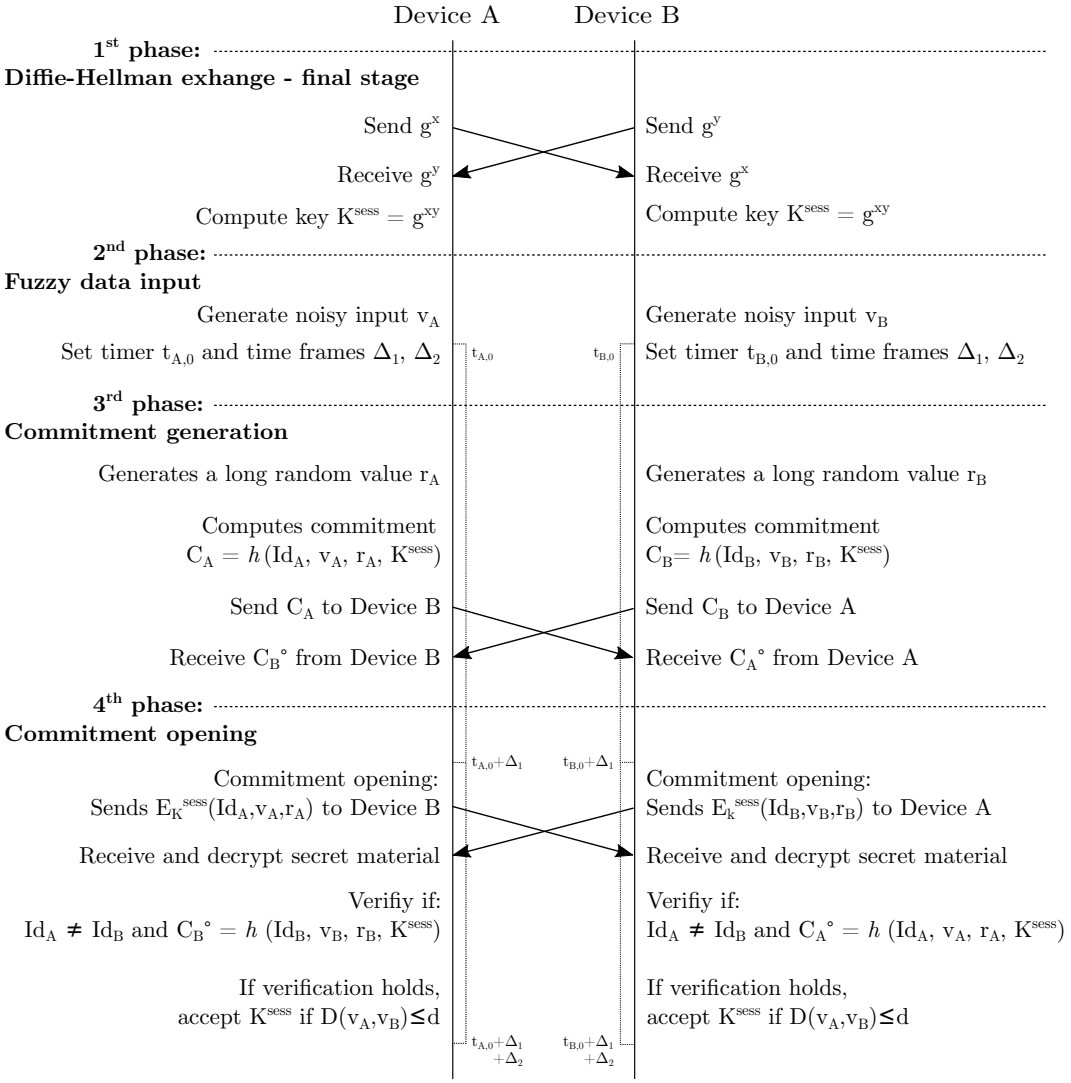
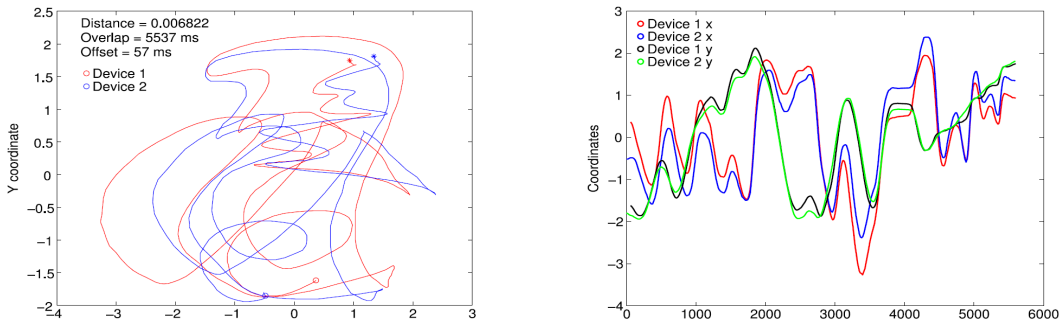


Figure 13: One-round time-based commitment protocol with implicit synchronization [28].

If Device A does not receive any data from Device B within $t_{A,0} + \Delta_1$ time interval, it aborts the process.

After receiving the hash from Device B, Device A opens its commitment by encrypting Id_A, v_A and r_A with K^{Sess}, and transmitting the secret material. Although not effective against active attackers, in this phase the encryption is needed for excluding passive statistical attacks. Concurrently, Device A waits for the commitment opening from Device B. If no data are received within $t_{A,0} + \Delta_1 + \Delta_2$ time, the pairing process is aborted. Otherwise, once it receives the material, Device A verifies that $Id_A \neq Id_B$ for excluding mirroring attacks yet another time. Moreover, having all the information for computing the counterpart's commitment, it verifies that the calculated commitment matches the one received in the previous phase. If verification is successful, Device A also proves that the similarity between the secret material extracted from both

drawings is less or equal a pre-established threshold (i.e., $D(v_A, v_b) \leq d$).



(a) (b)

Figure 14: Location metric: (a) centered and scaled drawings; (b) x and y coordinates as function of time [28].

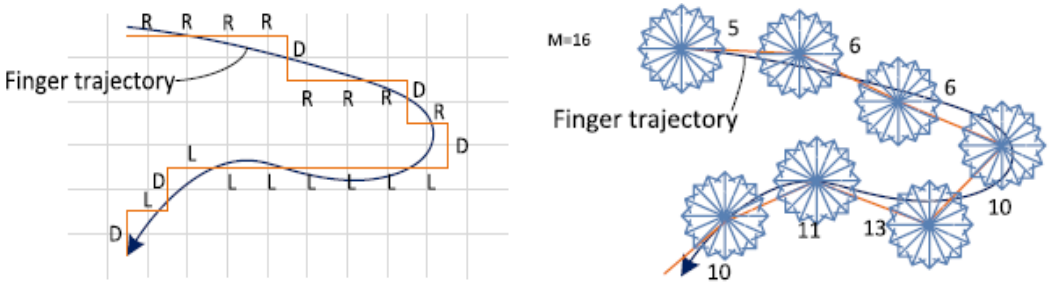
As a consequence, a proper distance metric D and threshold d must be introduced in order to discard non-matching movements (i.e., not similar enough to perform secure pairing) and accept the matching ones. For this reason, the study inspected possible ways to compare the two drawings.

The authors firstly studied a *location metric* (Figure 14), inspecting x and y coordinates as a function of time. Figure 14a shows two matching drawings after performing centering and scaling. Coordinates x and y in Figure 14b are interpolated at a frequency of 1 KHz. The distance between drawings in the location metric is calculated as the Euclidean distance of the difference between vectors of x coordinates (i.e., $\vec{x}_A - \vec{x}_B$) and y coordinates (i.e., $\vec{y}_A - \vec{y}_B$), as follows:

$$D(drawing_A, drawing_B) = \frac{1}{n}(\sum_{n=1}^{n}((x_{A,i} - x_{B,i})^2 + (y_{A,i} - y_{B,i})^2))^{\frac{1}{2}}$$

The resulting distribution of pairs (Figure 16a) of matching drawings (green) against non-matching drawings (red) demonstrates that location metric is suitable for clearly distinguishing successful pairings from failing ones. The sharp division between the two sets eases the detection of the threshold d, which is experimentally chosen in the separation space between the two curves.

On the other hand, the location metric is computationally expensive due to the cost of searching for the best time offset to align the two drawings. Thus, instead of focusing on spatial coordinates, the authors of [28] studied a *movement metric* by inspecting velocity and direction of drawings as a function of time. Then again, as shown in (Figure 16b), this metric does not offer a clear distinction between matching drawings (green) against non-matching ones (red). The cause is mainly ascribable to the touch-screen APIs which do not offer sufficiently frequent readings for a precise calculation of velocities and accelerations.



(a) (b)

Figure 15: Trajectory-based metrics: (a) LURD metric; (b) Angle string metric [28].

Therefore, Sethi et al. [28] focused on metrics which are not based on rigorous time synchronization and are not heavily affected by drawings' inaccuracy. Based on previous studies [43], they introduced a *LURD (Left Up Right Down) metric*. The rationale is to encode polygons, that is, to find a string representation of the drawings. The aim is to compare the similarity rate of the two resulting strings by using a proper *edit distance*. In other words, the edit distance is the minimum cost for transforming one string into the other in order to make them exactly the same. As shown in Figure 15a, the authors ideally divide the touch-sensitive surface by sketching a grid. Whenever the trajectory of the finger crosses the grid, the intersection points are isolated and the resulting string is built by translating the finger movement using a 4-character alphabet (i.e., L=left-ward, U=up-ward, R=right-ward, D=down-ward).

Even though the distribution depicted in Figure 16c demonstrates that LURD metric (Figure 15a) is suitable for encoding trajectories, the authors refined it by increasing the number of directions in the intersection points in order to obtain more accurate encoded strings. Therefore, as shown in figure Figure 15b, in the *Angle string metric* for each drawing intersection point the surrounding area is divided into 16 even portions (i.e., each portion is 360/16=22.5 degrees). Consequently, portions are encoded using an alphabet composed by decimal numbers from 1 to 16. However, after conducting several experiments, the authors verified that the ideal number of portions for the Angle string metric (Figure 15b) is 32. Then, trajectories are encoded by detecting on which portion they lay between two consecutive points.

The distribution plot (Figure 16d) highlights the effectiveness of the Angle string metric in terms of a negligible number of false-positives and sharp separation between the two sets for a straightforward detection of threshold d. Moreover, the Angle string metric increases the implementation efficiency as it does not rely on rigorous time synchronization.

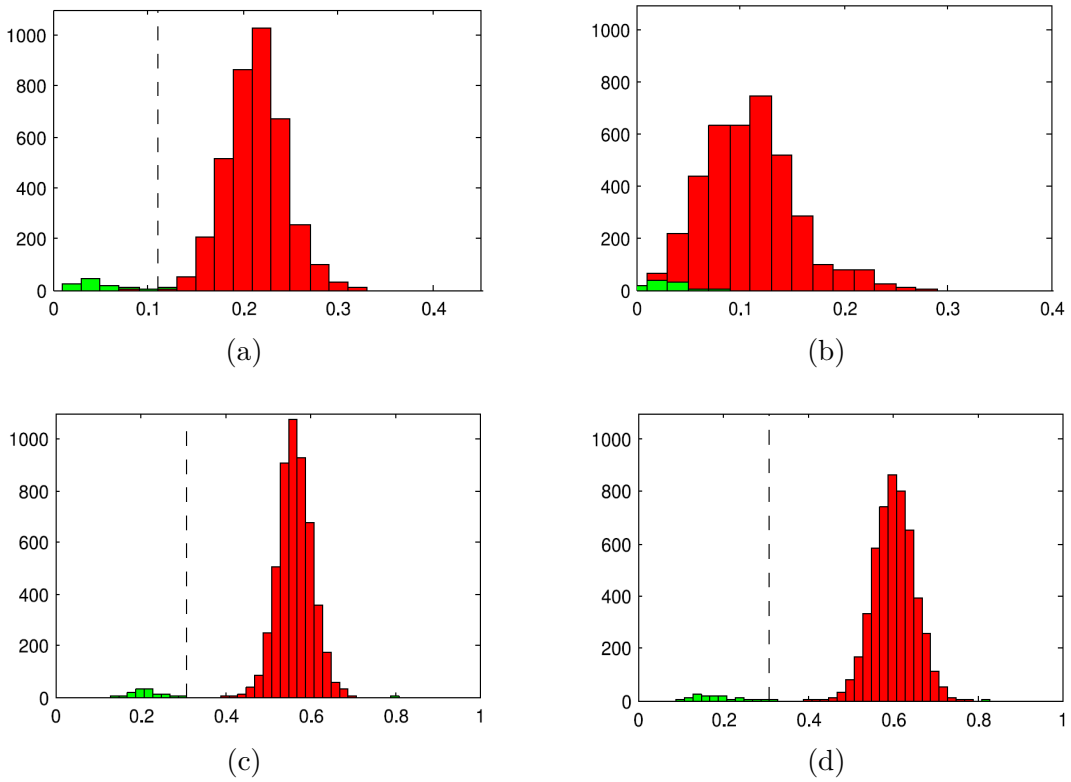


Figure 16: Distribution of matching drawings (green) and non-matching drawings (red) using different metrics: (a) Location metric (Euclidean distance); (b) Movement metric; (c) LURD distance; (d) ANGLE string distance [28].

3 Secure multi-modal gestures

Secure pairing protocols introduced by Sethi et al. [28] using simultaneous drawings (Section 2.2.4), and Mayrhofer and Gellersen [24] based on shaking devices together (Section 2.2.3) are especially interesting for their usability and solution design.

Our goal is to integrate these two protocols and to find a hybrid solution that involves accelerometer data along with screen coordinates. In particular, this thesis aims to verify the feasibility of a new usable scheme based on fuzzy data (Section 2.2) to securely pair a smartwatch and a mobile device, say a mobile phone, through multi-modal gestures.

As shown in Figure 17, multi-modal gestures indicate those gestures involving drawing supported by accelerometer data. In other words, the user can pair a smart-watch on his wrist and a smart-phone by simply drawing with a finger on the screen of the mobile device.

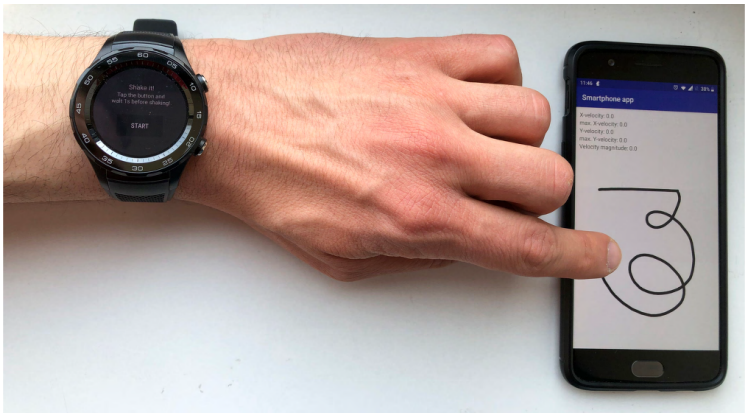


Figure 17: Pairing through multi-modal gestures.

This new scheme can be generally applied to perform pairing between one device provided with an accelerometer and another equipped by a touch-sensitive surface (e.g., a smart-phone and a tablet, a smart-watch and a tablet, etc.).

However, in the above-mentioned protocols [28, 24] the approach for comparing signals is simple since they seek for similarities between sensors of equal nature (i.e., two accelerometers or two touch-screens). In our case, verifying that finger and wrist movements translate into comparable signals from smart-watch and smart-phone is challenging due to the different nature of sensor data. That is, we need to find an effective approach to compare accelerometer data (i.e., raw acceleration) against drawing data (i.e., coordinates on the screen).

For clarity, the drawing depicted in Figure 18 will be considered in the following sections. The drawing represents a suitable example for easier detection of possible similarities between smart-phone and smart-watch signals since it implies clear acceleration peaks around vertices.

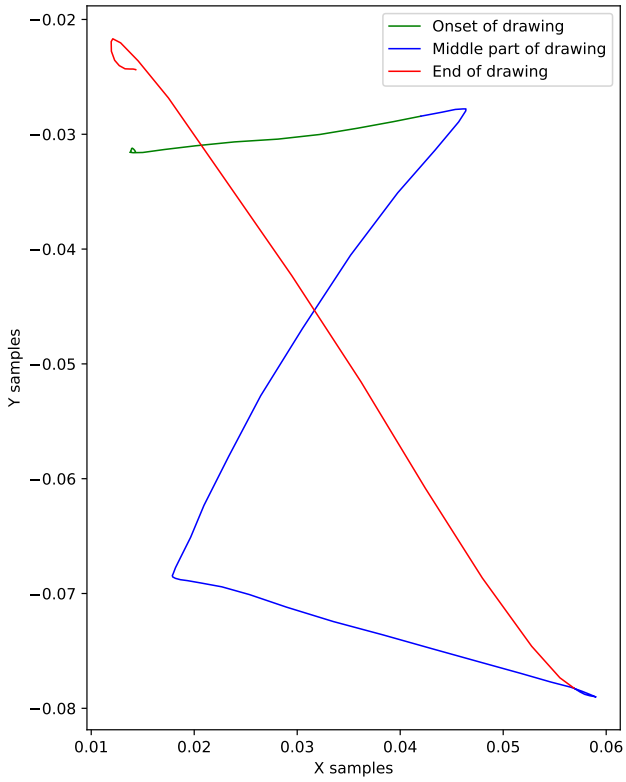


Figure 18: Drawing example on smart-phone screen. The green segment represents the initial part of the finger movement on the screen, whereas blue and red segments illustrate the middle and final part of the drawing respectively.

3.1 Background

To effectively study a novel secure pairing protocol, we firstly need to consider possible attack vectors which may compromise it.

This study assumes that the two devices have access to one in-band channel (e.g., a wireless link) and one OOB channel (i.e., the imaginary link built by the movement of the smart-watch and the screen drawing).

The former is deemed to be insecure as vulnerable to MITM active attacks and to passive eavesdrops of the communication. The latter is supposes to be secure and unspoofable since location-limited and under the physical direct control of the user, but error-prone due to inherent external noise and possible different sensor calibration on the devices.

In this context, the final goal of the attacker is to successfully access the confidential information between the two devices.

For these reasons, the combination between wrist movement and finger

drawing over the OOB channel represents the first building block for agreeing on a secret.

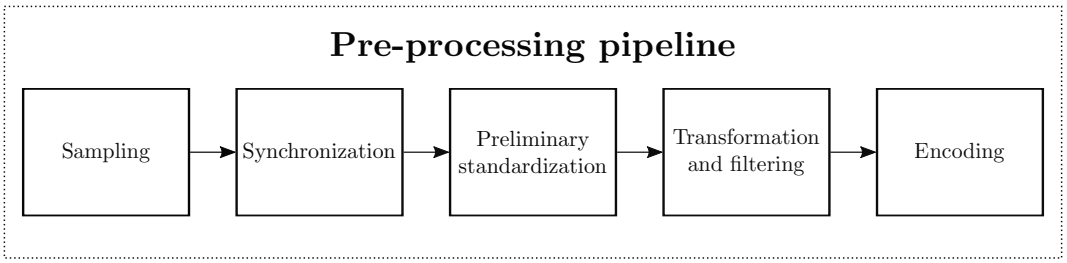


Figure 19: Pipeline of steps for processing signals extracted from smart-watch and smart-phone. The pipeline outputs encoded signals.

As shown in Figure 19, to extract a secret from the two devices we utilize a pipeline of steps for processing and eventually encoding signals to make them mutually comparable.

In other words, once signals are sampled (Section 3.2) they need to be synchronized (Section 3.3) to eliminate possible clock skews. In addition, in the preliminary standardization (Section 3.4), the aim is to remove the gravity contribution from the smart-watch sample. Then, data are conveniently transformed and filtered to make them comparable (Section 3.5). Namely, we filter smart-watch samples to obtain velocity and screen coordinates from the acceleration. In addition, we filter smart-phone samples to compute velocity and acceleration from screen coordinates. After transformation, data are encoded (Section 3.6) to produce their bit-string representation.

The pre-processing pipeline in Figure 19 is necessary for subsequent steps described in Chapter 4, where we study a comparison metric function to compute a similarity ratio and a threshold for distinguishing non-matching movements (i.e., not similar enough to perform secure pairing) from the matching ones. Once similarity ratio and threshold are found, we are able to apply this information to the key agreement described in Figure 13. In other words, the encoded signals from the pre-processing pipeline represent the noisy inputs v_A and v_B on the two devices, whereas the comparison metric and threshold serve as function D and parameter d respectively.

3.2 Sampling

We sample raw acceleration along x, y and z axes in the time domain from smart-watch at a frequency of 100 Hz approximately, whereas we sample drawing x and y screen coordinates in the time domain from smart-phone at a frequency of 50 Hz roughly.

These frequencies represent the maximum sampling rate that our devices support. Equipment, APIs and implementation details are given in Section 4.1.2 and Section 4.1.3 for smart-watch and smart-phone respectively.

However, as detailed later in Section 3.5.1, we are interested to study frequencies in the range of 0-20 Hz, which better describe movements of human limbs. Therefore, according to the Nyquist–Shannon sampling theorem [44, 45], in order not to lose any piece of information from the continuous-time movement signals, we need to sample data at least at the double of the maximum frequency we are interested to (i.e., 40 Hz). As a result, both devices sample at an adequate frequency for our goal.

3.3 Data synchronization

Data synchronization represents the first important step for subsequent detection of possible signal similarities.

To this purpose, it is necessary to find a proper event that uniquely determines the onset of the pairing process at the smart-watch. In fact, although the drawing has clear onset and termination events (i.e., whenever the finger touches the screen and lifts), the accelerometer is continuously sampling. Consequently, data synchronization is only performed at the smart-watch.

Once such an event is clearly detected, the benefit is twofold. First, it becomes straightforward to discard insignificant information from meaningful acceleration data from smart-watch samples (i.e., data recorded by the accelerometer before the actual tap on the screen and after the finger lifting from the screen). Second, clear detection of the onset event allows eliminating clock discrepancies. As detailed in Section 4.1, we use an Android smart-phone and a WearOS smart-watch for our study. These systems use the NTP protocol [46] for synchronizing their clocks. Although NTP offers a potential accuracy to the tens of microseconds while synchronizing with the NTP servers, in the vast majority of the cases the clocks of the two devices present a significant time skew.

Based on the findings of Mayrhofer and Gellersen [24] described in Section 2.2.3, we use the magnitude of the raw acceleration on the smartwatch for detecting the onset of the pairing. The idea is based on the fact that the initial tap on the screen usually corresponds to a wide movement of the wrist, hence generating a peak of magnitude on the accelerometer.

For simplicity purpose, we conduct our study by firstly holding the wrist in a quiet position above the other device, and then performing a strong tap on the screen, provoking a clear high peak of magnitude before continuing to draw.

We compute the magnitude as the 2-norm (Euclidean norm) of x, y and z raw acceleration components, as follows:

$$m = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

Therefore, we make use of the *Smoothed z-score algorithm* [47] to perform a peak detection, that is to verify the presence of a significant increase of magnitude. The algorithm computes a moving mean considering a certain *lag*

(i.e., number of consecutive data-points). In our experiments, we consider a *lag* of 4 data-points. In addition, whenever a new data-point is examined, the algorithm calculates the standard deviation and the z-score [48], that is the distance from the mean in terms of number of standard deviations. In case the z-score exceeds a chosen threshold (in our experiments *threshold* is set to 9), the algorithm returns a peak. Moreover, the influence of the new data-point on the previous ones is set to 0.

As shown in Figure 20, the Smoothed z-score algorithm signals in presence of high peaks of magnitude, discarding weaker magnitude values and noise. Thus, it is experimentally possible to verify that the actual first tap on the screen corresponds to the maximum value of magnitude within the first three positive peaks (i.e., within the first three series of value 1 returned by the Smoothed z-score algorithm).

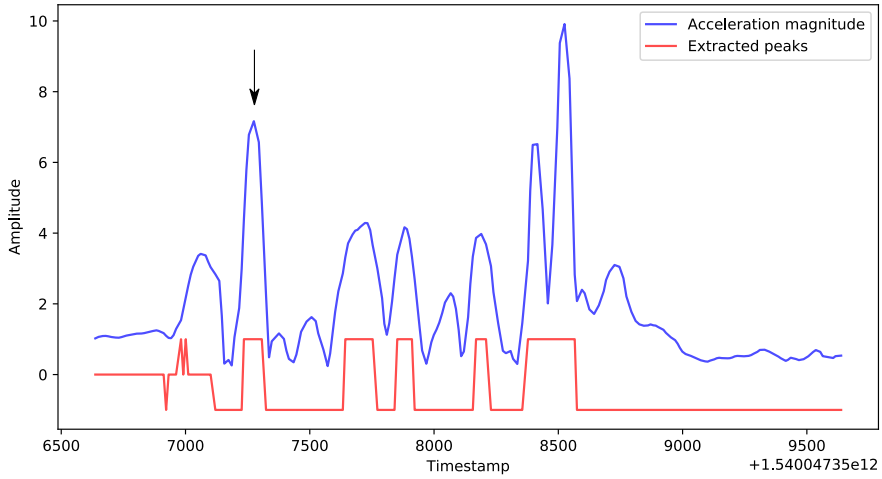


Figure 20: Tap detection on smart-watch data.

3.4 Preliminary standardization

As shown in Figure 17, in order to minimize the impact of roll, pitch and yaw [49] on acceleration data, we perform the drawing by keeping the smart-watch as parallel as possible to the ground and the finger perpendicular to the smart-phone.

In this scenario, we expect the gravity to impact almost exclusively on the z axis of the smart-watch. Obviously, the contribution of the gravity on the smart-phone is null since we sample x and y screen coordinates only. Thus, removing the contribution of gravity from smart-watch samples and computing the linear acceleration results straightforward. Moreover, although the raw acceleration on all three axes is needed for the magnitude calculation in the Data synchronization process (Section 3.3), after computing the linear acceleration

we can ignore the smart-watch z axis and study possible similarities between smart-watch's and smart-phone's x and y axes only.

After detecting the actual first tap on smart-watch samples as described in Section 3.3, we obtain the linear acceleration by firstly calculating the mean of the raw acceleration on x and y axes considering the meaningful information only (i.e., the data recorded by the accelerometer before the actual tap on the screen and after the finger lifting from the screen). Eventually, we simply subtract the mean from x and y raw accelerations.

3.5 Transformation and filtering

Transformation and filtering represents the core step of the pre-processing pipeline shown in Figure 19. In other words, the goal is to transform data on both devices to reach a common state for an easier comparison of signals. Moreover, data on both devices need to be filtered for eliminating noise and user movement imprecision.

Section 3.5.1 describes the transformation step at the smart-watch, namely the process of integration and filtering of data. Similarly, Section 3.5.2 details the transformation step at the smart-phone, that is the process of differentiation and filtering of data.

3.5.1 Smart-watch integration and filtering

After computing linear accelerations along x and y axes on the smart-watch, we aim to calculate velocities and screen coordinates along these two axes.

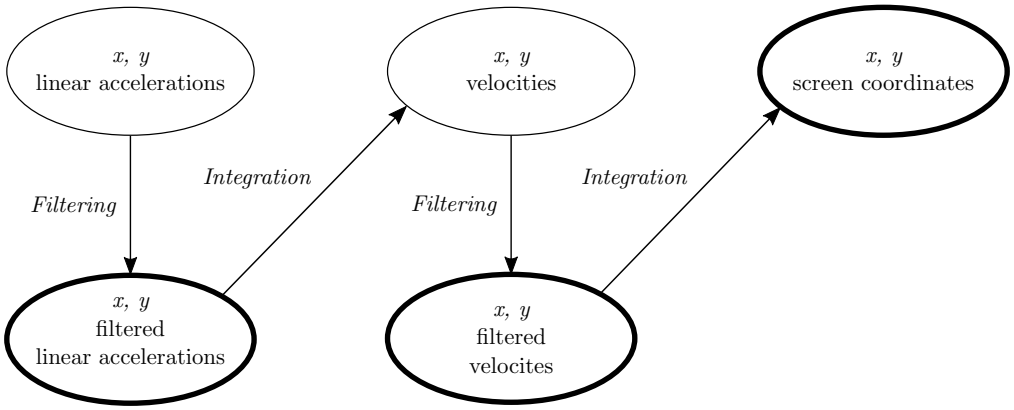


Figure 21: Integration and filtering steps to calculate velocities and screen coordinates from raw accelerations.

To do so, as depicted by Figure 21, we firstly need to remove useless frequencies out of the signals and retain meaningful information about the wrist movement. As a matter of fact, accelerometers are highly sensible and able to sample at frequencies higher than 100 Hz. In the vast majority of the cases, their signals are affected by noise and prone to collect imperceptible or unintended user's wrist movements.

Several studies on human body motion have been conducted to identify the frequency range that conveniently describes deliberate limb movements. Winter [50] asserts that meaningful frequencies for describing human motion are approximately in the range of 0-10 Hz. On the other hand, Bouten et al. [51] affirm that, for assessing daily human activities, it is sufficient that body-fixed accelerometers record in the range of 0-20 Hz. In their study about pairing devices by shaking them together [24] (see Section 2.2.3), Mayrhofer and Gellersen obtained better results taking into consideration frequencies up to 40 Hz. Then again, although the ShakeUnlock protocol [41] is based on the findings of Mayrhofer and Gellersen, its authors found better results in the range of 0-16 Hz. Based on these findings, for conducting our experiments we decide to use a cut-off frequency of 20Hz.

As shown in Figure 22, we convert the accelerations along x and y axes from the time domain to their representation in the frequency domain by using the FFT (Fast Fourier Transform) algorithm. Then, we attenuate frequencies that exceed the cut-off frequency through an LPF (Low-Pass Filter), obtaining a filtered signal in the frequency domain. By using the IFFT (Inverse Fast Fourier Transform) algorithm, the filtered signal is converted back to the time domain.

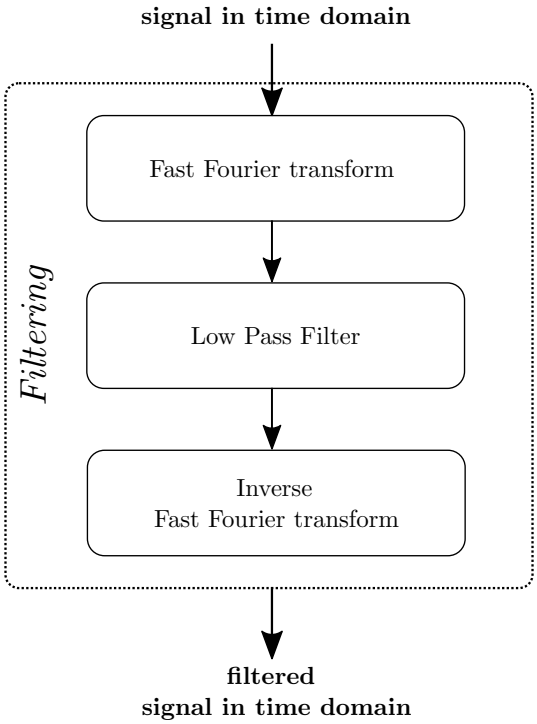


Figure 22: Filtering steps for attenuating high frequencies in input signals.

Consequently, we determine velocities by integrating the accelerations using the *composite trapezoidal rule*. Let the acceleration be defined in the interval $[t_a, t_b]$, that it respectively the first and the last timestamp of the recorded sample. Let x_k be a partition of $[t_a, t_b]$ where $t_a = x_0 < x_1 < ... < x_{n-1} < x_n = t_b$ and

Δx_k the duration of the k^{th} time interval, namely $\Delta x_k = x_k - x_{k-1}$. Then, for each of the axes:

$$v_{x,k} = \int_{t_a}^{t_b} f(x)dx \approx \sum_{n=1}^{N} \frac{f(x_{k-1}) + f(x_k)}{2} \Delta x_k$$

Once the velocities along the x and y axes are obtained, we can filter and integrate them the same exact way as before to compute x and y spatial coordinates of the wrist movement.

3.5.2 Smart-phone differentiation and filtering

At the smart-phone, the goal is to calculate x and y velocities and accelerations by utilizing the drawing screen coordinates.

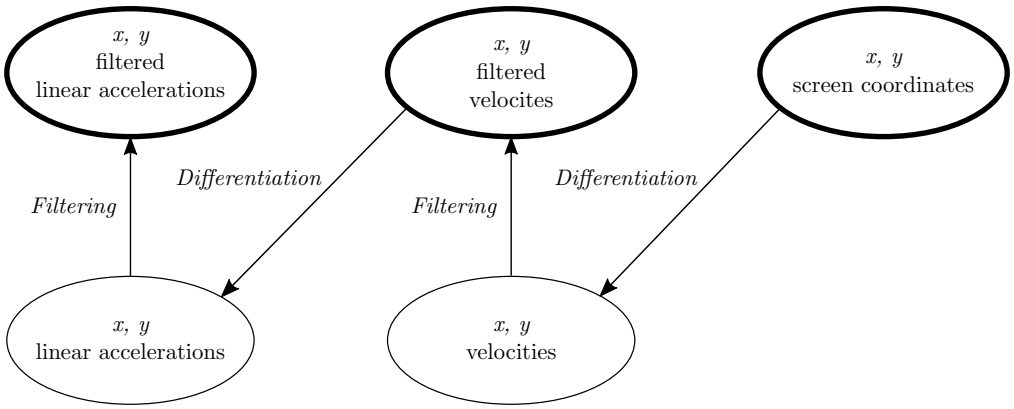


Figure 23: Differentiation and filtering steps to calculate velocities and accelerations from screen coordinates.

To do so, as depicted in Figure 23, we firstly obtain the velocity on the x and y axes by computing the derivative of the screen coordinates. Let the screen coordinates be defined in the time interval $[t_a, t_b]$, that it respectively the first and the last timestamp of the recorded sample. Let t_k be a partition of $[t_a, t_b]$ where $t_a = t_0 < t_1 < ... < t_{n-1} < t_n = t_b$ and Δt_k the duration of the k^{th} time interval, namely $\Delta t_k = t_k - t_{k-1}$. In addition, let Δx_k be the spatial interval between two consecutive data-points, namely $\Delta x_k = f(t_k) - f(t_{k-1})$. Then, for each consecutive couple of data-points in the sample, both for x and y axes:

$$v_{x,k} = \frac{\Delta x_k}{\Delta t_k}$$

Once velocities are computed, we filter the signals for two main reasons. In the first place, by filtering we ensure that signals do not contain any possible noise that may come from the sampling process. Moreover, by using this method we align both pre-processing phases, on smart-watch and smart-phone. For these reasons, the filtering is performed similarly to the method used for

smart-watch signals, as shown in Figure 22. In other words, the signals are converted to their representation in the frequency domain by using the FFT algorithm. Then, an LPF attenuates the undesired high frequencies. Eventually, through the IFFT algorithm, the filtered signals are converted back to their representation in the time domain.

At this stage, the accelerations along x and y are computed differentiating the velocities calculated during the previous step. Let Δv_k be the difference of velocities of two consecutive data-points, namely $\Delta v_k = f'(t_k) - f'(t_{k-1})$. Then, for each consecutive couple of data-points, both for x and y axes:

$$a_{v,k} = \frac{\Delta v_k}{\Delta t_k}$$

Then, the accelerations are filtered as mentioned above for the previous step. As a result, we eventually have the whole comparable information needed for detecting possible similarities between smart-phone and smart-watch signals.

3.6 Encoding

Once data on both devices are in a suitable state for comparison, the next step is to encode signals from smart-watch and smart-phone to let devices independently and asynchronously perform the similarity check.

The goal is to connect our study with the findings of the Commitment-based device-pairing protocol with synchronized drawings [28], as explained in Section 2.2.4. In other words, we aim to find a signal encoding method complaint with the key agreement depicted in Figure 13. Moreover, the encoding method has to be independent of the amplitude of the signals, namely not prone to modify its output in the presence of high or low sensor sensitivity.

To this purpose, we introduce a novel and simple method for encoding signals. Individually on both devices, velocity (or acceleration) x and y components are rendered on the same plot to obtain their drawing-like representation. Figure 24 shows an example obtained by interpolating velocity components on a single device starting from the onset data-point (i.e., green node) to the last data-point in time (i.e., red node). As a matter of fact, our approach differs from the study by Sethi et al. [28] depicted in Figure 15a and Figure 15b, since the authors encode drawings by taking into considerations screen coordinates in time only.

Subsequently, we assign a 2-bit code to each quadrant of the Cartesian coordinate system, as depicted in Figure 25, by using the Grey-code [52] bit-sequence ordering principle, where one sequence differs from the closer ones by flipping one single bit only. We choose this method since it is straightforward and reasonably simple to implement. Moreover, it does not present strict rules for its applications and suits perfectly our goals.

Afterward, we overlap the Cartesian coordinate system on the first data-point of the drawing-like graph, as shown in Figure 26. In addition, we draw a vector that joins the first data-point with the third one, namely a vector with $jump = 2$. We decide to use the $jump$ vector since it provides a good

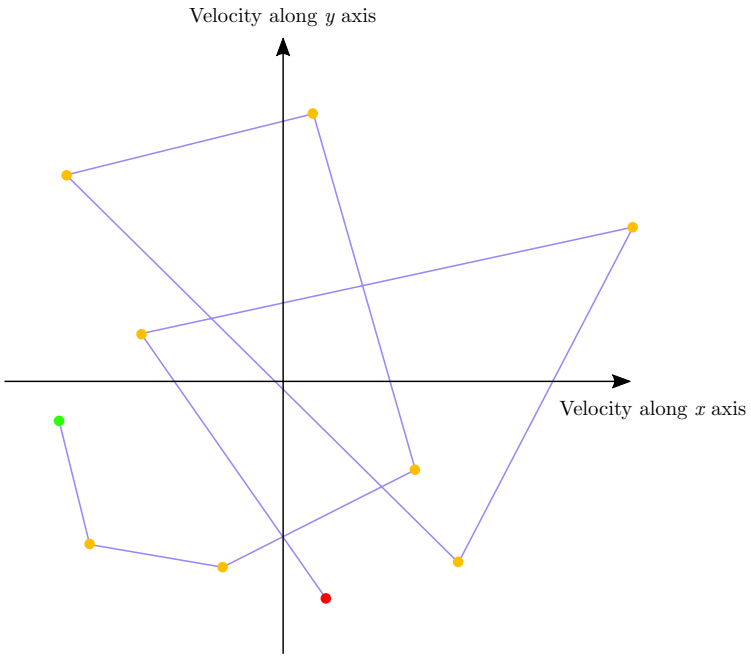


Figure 24: Example of velocity components plotted together obtaining a drawing-like graph.

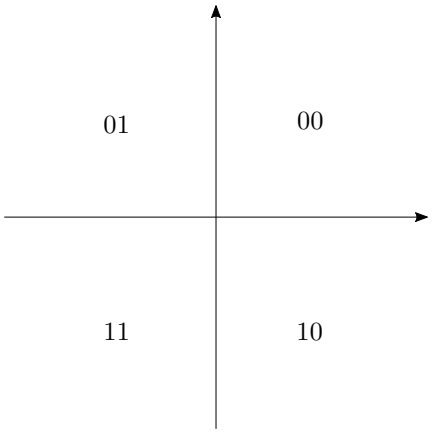


Figure 25: 2-bit Grey-code ordering for signal encoding.

approximation of the drawing-like graph, and it reduces the impact of possible outliers on the resulting code.

Therefore, according to the quadrant over which the jump vector lays on, the corresponding two bits are extracted. In the example depicted by Figure 26, the first two extracted bits are “10” since the jump vector lays on the fourth quadrant. The same procedure is repeated by overlapping the Cartesian coordinate system on the second data-point (see Figure 27) with a jump vector connecting the second and the fourth data-point. In this case, the extracted two bits are “00”, as the vector lays on the first quadrant.

Consequently, by repeating this procedure data-point by data-point until the

jump vector connects the third-last data-point to the last one and combining all the 2-bit extracted codes in sequence, we eventually obtain the overall encoded signal. The overall code extracted from the drawing in Figure 24 is "10 00 00 01 10 10 01 11".

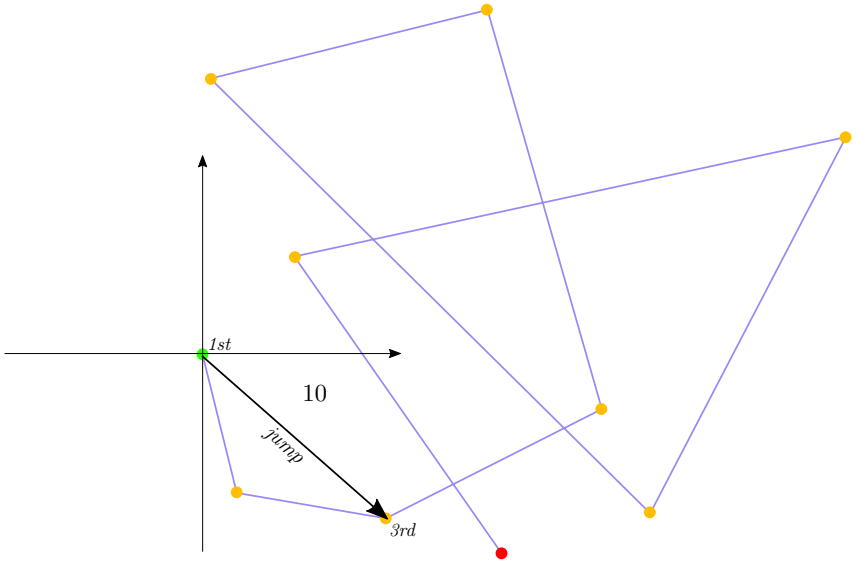


Figure 26: Extraction of the first two bits with $jump = 2$ from 1^{st} to 3^{rd} data-point.

This encoding method is also convenient since there is no loss of information from the original signal. In addition, the resulting code includes a 50% of signal overlap as approximately every data-point is overcome by the *jump* vector once.

Furthermore, this method makes possible to extend code generation using different mapping between the coordinate system and bit sequences. For instance, as shown in Figure 28, it is possible to expand the resulting code by dividing the Cartesian coordinate system into eight portions and assigning a 3-bit code to each of them. In fact, by following the same procedure described above and assigning 3 bits per step the resulting code is "100 000 001 010 101 100 010 111". The first two steps of the procedure are depicted by Figure 29 and Figure 30.

Finally, by using this method it is possible to also increase or decrease the value of the *jump* vector, resulting in a different signal overlap (e.g., with a $jump = 3$ the resulting code includes a 66% of signal overlap as approximately every data-point is overcome by the *jump* vector twice).

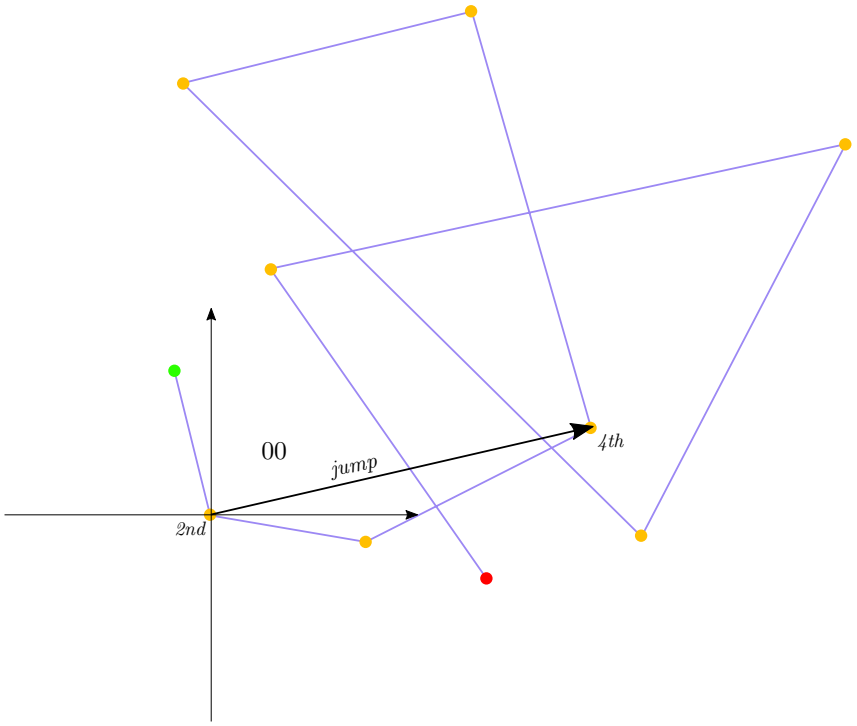


Figure 27: Extraction of the second two bits with $jump = 2$ from 2^{nd} to 4^{th} data-point.

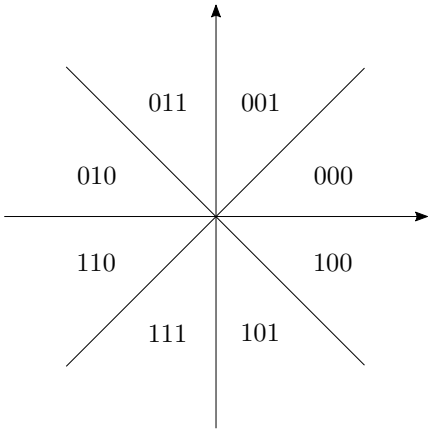


Figure 28: 3-bit Grey-code ordering for signal encoding.

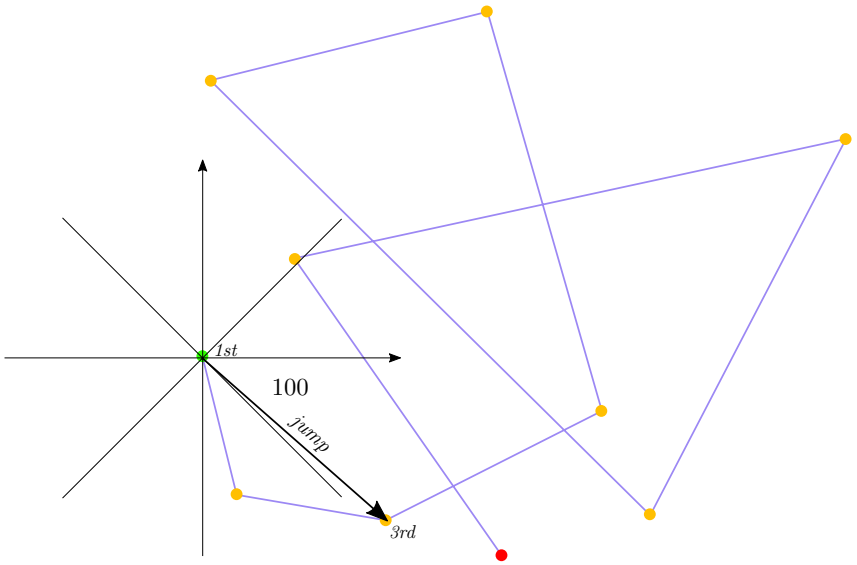


Figure 29: Extraction of the first three bits with $jump = 2$ from 1^{st} to 3^{rd} data-point.

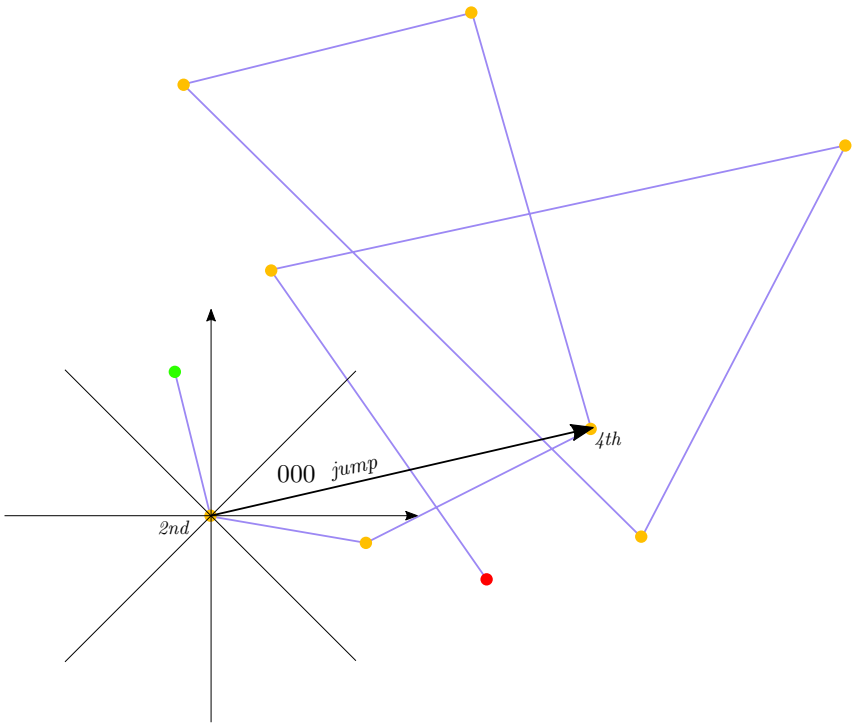


Figure 30: Extraction of the second three bits with $jump = 2$ from 2^{nd} to 4^{th} data-point.

4 Evaluation

In the following, we describe implementation details about our approach to sample, pre-process and encode data explained in Chapter 3. Moreover, we evaluate its effectiveness to distinguish matching movements from non-matching ones by analyzing obtained results.

In particular, Section 4.1 depicts the experimental setup and methodology. Section 4.2 illustrates the empirical characterization of signals after the pre-processing pipeline described in Section 3.5. Section 4.3 introduces a function metric for a precise comparison between signals. Moreover, Section 4.4 studies the existence of a suitable comparison threshold that allows to detect matching movements and discard non-matching ones. Eventually, Section 4.5 evaluates the obtained results.

4.1 Experimental setup

4.1.1 Equipment

We employ one LG Nexus 5X smart-phone for tracking finger drawings and one Huawei Watch 2 2358 smart-watch for tracking wrist movements. Table 1 and Table 2 summarize the relevant characteristics of the two devices for our study. Both devices run the latest versions of Wear OS and Android respectively[1]. We decided to use these devices since they are easily available on the market and they support the latest APIs. In addition, we do not have any constraints in using specific devices. In fact, we basically need one device equipped with an accelerometer and the second one equipped with a touch-sensitive surface.

As already discussed in Section 3.4 and shown in Figure 17, to minimize the impact of roll, pitch and yaw [49] on acceleration data, we perform our experiments keeping the smart-watch as parallel as possible to the ground and perpendicular to the smart-phone while drawing. However, as better discussed later in Section 5, for future follow-ups and a proper user study the accurate removal of roll, pitch and yaw must be considered as a core objective. Moreover, similar to Figure 14a we decide to draw jerkily on the screen to facilitate the visual detection of similarities among signals.

4.1.2 Wear OS smart-watch application

The aim of the Wear OS application is to sample raw acceleration in the time domain along x, y and z axes from smart-watch.

In Wear OS it is possible to use any standard Android API. Thus, we implement the application [55] and gather acceleration data along with timestamps by using the SensorManager API [56] and listening to the sensor `Sensor.TYPE_ACCELEROMETER` [57].

[1] At the time of this study, the two devices were running Wear OS App version 2.15 and Android 8.1.0 (Oreo) respectively.

Huawei Watch 2 2358		
Processing	CPU	Qualcomm® Snapdragon Wear™ 2100 processor, 1.2 GHz quad-core 32-bit
	Memory	4 GB, 768 MB RAM
Display	Resolution	390x390 pixels
	Size	1.2 inches
	Pixels per inch (PPI)	326
Connectivity	Wi-Fi	802.11 b/g/n 2.4 GHz
	Bluetooth	4.1
Sensors	Accelerometer + Gyroscope	6-axis
	Compass	3-axis

Table 1: Huawei Watch 2 2358 - Device specifications [53].

LG Nexus 5X		
Processing	CPU	Qualcomm® Snapdragon™ 808 processor, 1.8 GHz hexa-core 64-bit
	Memory	32 GB, 2 GB MB RAM
Display	Resolution	1920x1080 pixels
	Size	5.2 inches
	Pixels per inch (PPI)	423
Connectivity	Wi-Fi	802.11 a/b/g/n/ac dual-band (2.4 GHz, 5.0 GHz)
	Bluetooth	4.2

Table 2: LG Nexus 5X - Device specifications [54].

Conversely, we could also listen to the synthetic sensor `Sensor.TYPE_LINEAR_ACCELERATION` [57] to directly dispense with the gravity contribution. On the other hand, the official documentation lacks details about its actual implementation. For this reason, we decide not to use this sensor and instead to delegate the gravity removal to the data pre-processing (see Section 3.4).

Collected data are written to the smart-watch's internal storage then exported for subsequent manipulation and analysis of data.

4.1.3 Android smart-phone application

We develop the Android smart-phone application [58] to gather x and y positions on the screen in the time domain by using the MotionEvent API [59].

Furthermore, since the VelocityTracker API [60] helps for tracking the velocity of touch events and its documentation is exhaustive, we utilize this API for obtaining the velocity of finger strokes on the screen instead of assigning the calculation to the differentiation and filtering phase described in Section 3.5.2.

Similarly to the Wear OS application (Section 4.1.2), we decide to write the gathered data (i.e., timestamps, x and y screen coordinates, x and y velocities) to the smart-phone's internal storage then exported for subsequent pre-processing.

4.1.4 Data processing

We develop a script [61] which implements the pre-processing pipeline depicted in Figure 19, offers an empirical characterization of data (Section 4.2), and collects experimental results (Section 4.5).

The script is developed using Python 3 language. We choose Python since it provides libraries for simple manipulation, analysis and plotting of data. In addition to these, it offers several powerful and well-documented libraries for mathematics and science.

To easily manipulate raw data collected by the Android and Wear OS applications (gathered into *.csv* files) we use the Pandas library version 0.22.0 [62], which offers its own data structures, namely Series (1-dimensional) and DataFrame (2-dimensional).

Furthermore, for mathematical and scientific computations we adopt NumPy version 1.14.0 [63] and SciPy version 1.0.0 [64] libraries. We also use an external Python implementation of the Smoothed z-score algorithm [47] (see Section 3.3) to extract acceleration magnitude peaks.

In addition, we use the Matplotlib library version 2.1.2 [65] for plotting data and easing the analysis of experimental results (Section 4.5).

4.2 Empirical characterization

Once the experimental setup is ready and the pre-processing pipeline is implemented, we compare filtered linear accelerations and filtered velocities before the encoding step to visualize possible similarities among signals, as depicted by Figure 31.

Then again, we discard the comparison among screen coordinates since we expect a high variance among the two devices. Indeed, the integration steps in the smart-watch pre-processing include additive constants to velocities and, mainly, to screen coordinates which compromise their eventual comparison.

Hence, to visualize possible signal similarities from the example in Figure 18, we plot x velocities, y velocities, x accelerations and y accelerations of the two devices in single plots, as shown by Figures 32a, 32b, 33a and 33b. In

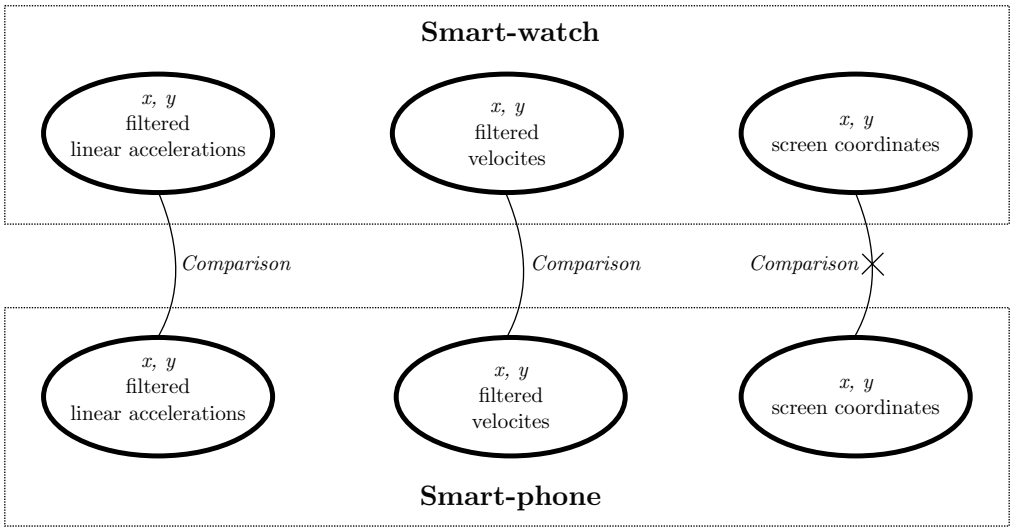
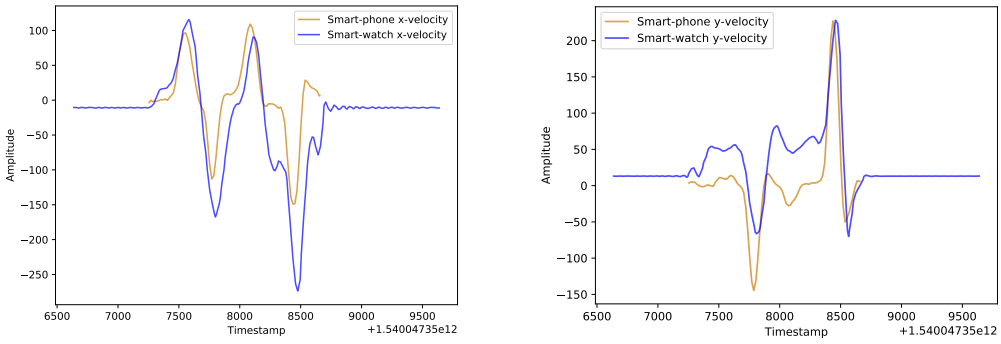


Figure 31: Relation of signals after the pre-processing phases.

these plots we also include non-meaningful information that has been recorded by the accelerometer before the actual tap on the screen to demonstrate the effectiveness of the the data synchronization step (see Section 3.3).

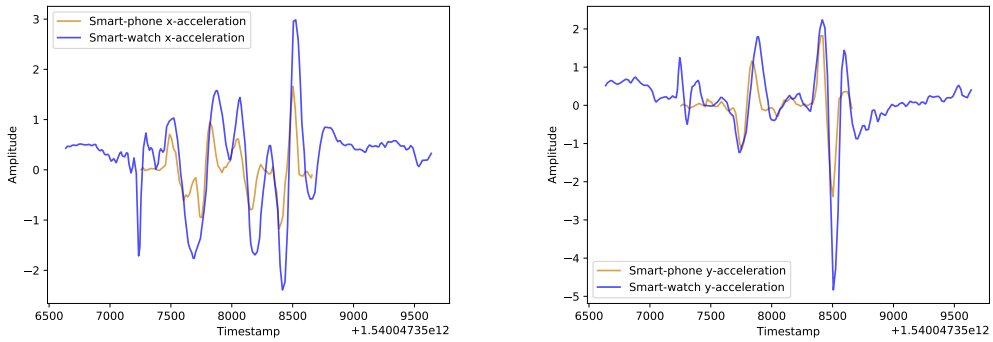


(a) (b)

Figure 32: Filtered velocities along the x axis (a) and the y axis (b) from the two devices.

However, Figure 32a shows some discrepancies in terms of amplitude of signals in the time interval 8000-8500 ms, probably due to the high accelerometer sensitivity. Moreover, Figure 32b depicts some signal deviation. The reason is most likely due to the subtle contribution of wrist pitch and yaw [49] at the smart-watch. Furthermore, in Figure 32b some peaks present clear differences (e.g., in the time interval 7500-7800 ms and 8300-8400 ms) due to the fact that finger friction on the screen (i.e., scatters while drawing) may not that accurately reflect on the wrist.

Then again, the highly similar trend of signals in these plots demonstrates the actual presence of similarities we are aiming to find. In other words, the



(a) (b)

Figure 33: Filtered linear accelerations along the x axis (a) and the y axis (b) from the two devices.

pre-processing phases at both devices are effective and successful. On the other hand, based on the plots in Figure 32 and 33 only, without considering the output of the encoding step described in Section 3.6, we cannot predict whether velocities or accelerations provide better result to spot a clear similarity threshold.

4.3 Comparison metric

After performing the encoding of signals (illustrated in Section 3.6), a function metric for a formal comparison between resulting codes needs to be found. That is, we need a method for computing the *similarity ratio* between smart-watch and smart-phone encoded signals that complies with the key agreement in Figure 13.

To this purpose, we decide to firstly use the bit-wise XNOR gate [66] between the two encoded signals (Table 3 shows the XNOR Truth Table).

XNOR - Truth Table		
$\boldsymbol{bit_A}$	$\boldsymbol{bit_B}$	$\boldsymbol{\overline{bit_A \oplus bit_B}}$
0	0	1
0	1	0
1	0	0
1	1	1

Table 3: Truth table of the digital logic XNOR gate [66].

Then, let the length of the XNOR resulting array R be N. The metric function is defined as follows:

$$S_r = \frac{\sum_{n=0}^{N-1} (R_i = 1)}{N}$$

Figure 34 describes a simplified comparison example between the encoded signals S_A and S_B. The resulting similarity ratio of 70% is calculated dividing the number of bits equal to 1 in the array R (i.e., 7 bits) by the length of R (i.e., 10 bits).

Moreover, we use a *sliding window* that describes the maximum bit shift between signals to obtain a more accurate comparison. In practice, the sliding window helps tolerate possible subtle inaccuracies still present after performing the data synchronization phase (see Section 3.3). In other words, we re-calculate the similarity ratio by shifting the signal S_A bit-by-bit left-ward and right-ward from the basic position depicted in Figure 34 within a maximum *sliding window* shift. Figures 35b, 35a, 35c and 35d show the additional calculations of the similarity ratio.

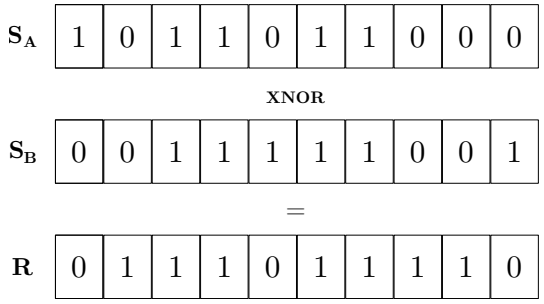


Figure 34: Similarity ratio of 70% between signals S_A and S_B.

Consequently, the final similarity ratio is picked as the maximum among the all computed ratios, namely 70% in the example.

4.4 Similarity threshold

Once a method for encoding signals (Section 3.6) and a comparison metric (Section 4.3) are found, the final step is to introduce a suitable *similarity threshold* which permits to detect matching signals from the ones that do not belong to the same pairing process.

Our final goal is to comply with the key agreement shown in Figure 13. In particular, since the two devices accept the pairing if and only if $D(v_A, v_B) \leq d$, where v_A and v_B are the encoded signals and D is the comparison metric, threshold d is the only missing parameter that the key agreement requires.

In other words, we need to study at which *similarity ratio* we have to set a *threshold* for minimizing the number of false-positives (i.e., signals that are

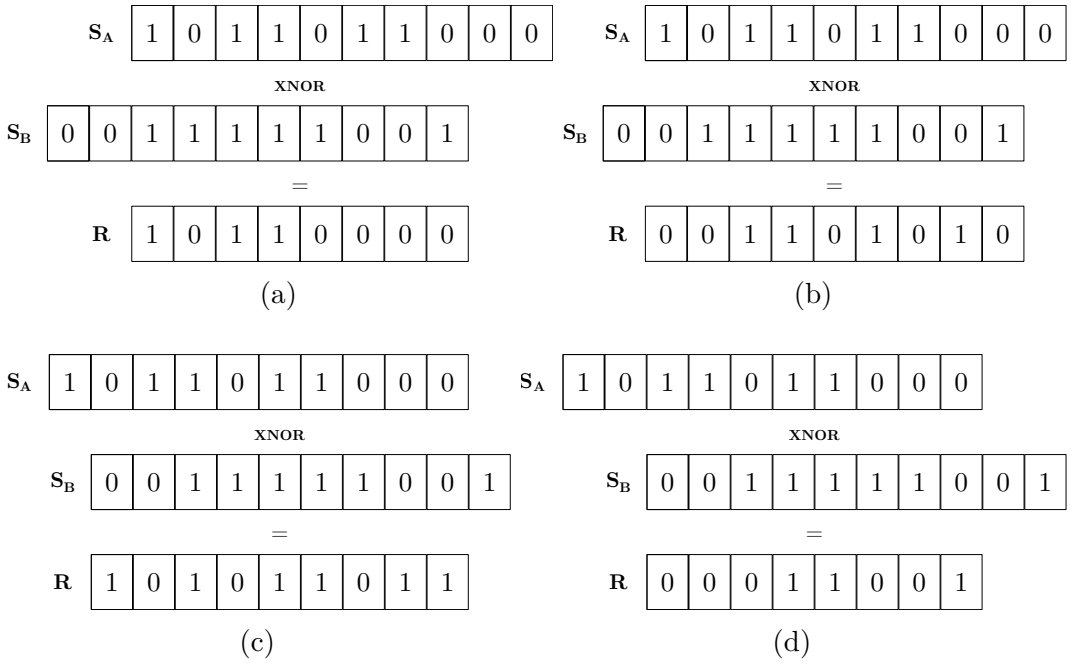


Figure 35: Similarity ratio between signals S_A and S_B with different shift: (a) 37.5% with a 2 bit shift right-ward; (b) 44.4% with a 1 bit shift right-ward; (c) 66.6% with a 1 bit shift left-ward; (d) 37.5% with a 2 bit shift left-ward.

accepted as matching but not belonging to the same paring process) and false-negatives (i.e., signals that are rejected as non-matching but instead belonging to the same paring process).

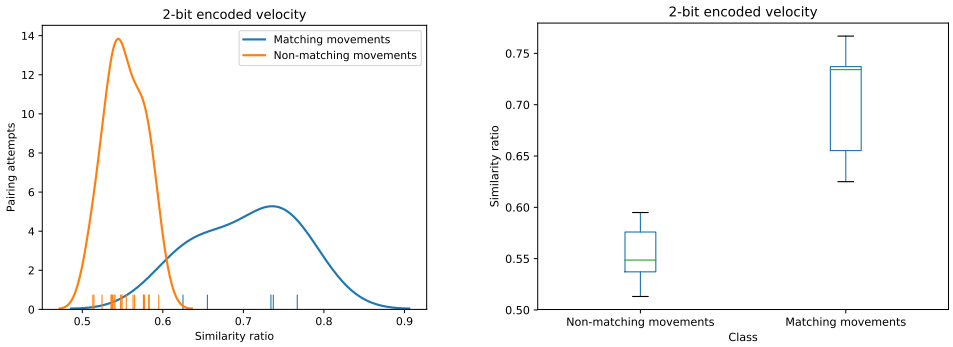
Consequently, finding a proper threshold (if any exists) is fundamental for understanding whether pairing a smart-watch and a mobile device through multi-modal gestures can be considered secure. To this purpose, Section 4.5 describes the test-bed and shows experimental results.

4.5 Experimental results

To find a similarity threshold that clearly separates matching movements from non-matching ones in terms of similarity ratio, we perform 10 different pairings and collect samples at the two devices.

We start all pairings by performing a strong tap on the smart-phone screen to ease data synchronization, as detailed in Section 3.3. In addition, we draw for 4 seconds approximately to collect adequate meaningful information from the two devices. Moreover, we use trajectories that contain vertices, avoiding spiral-like drawings, to provoke definite peaks of acceleration and velocity, similar to the example in Figure 18.

Consequently, we obtain 10 samples at smart-watch and 10 at smart-phone. After pre-processing and encoding sampled data, as depicted by Figure 19, for each smart-watch sample we calculate the similarity ratio in comparison with each smart-phone sample, resulting in 10^2 hypothetical pairings.



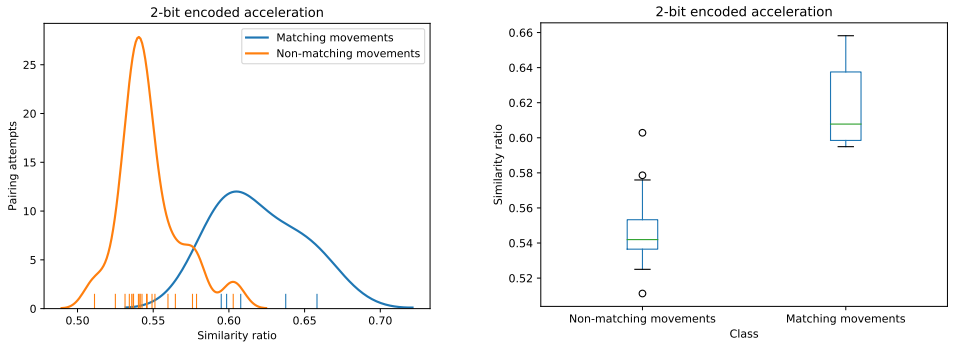
(a) (b)

Figure 36: Distribution plot (a) and box plot (b) of 2-bit encoded velocity with *jump vector* = 2 and a *sliding window* set to 5% of the number of data-points in the samples. The two boxes in (b) are well divided and their medians are adequately distant.

Among these, the 10% are matching movements, namely when calculating the similarity ratio between samples belonging to the above-mentioned real pairings we performed. The other 90% are non-matching, that is when calculating the similarity ratio between samples whose related drawing on the screen should not correspond with the wrist movements.

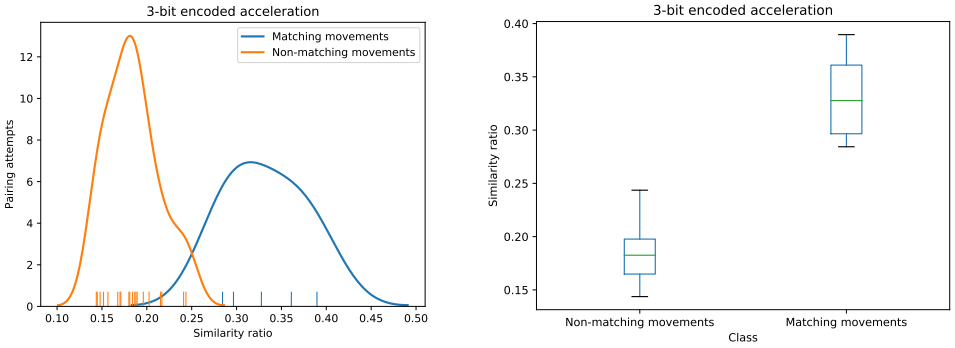
Furthermore, for each couple of samples, we compare velocities and accelerations using both 2-bit and 3-bit encoding with *jump vector* = 2, as explained in Section 3.6. In addition, we use a *sliding window* equal to 5% of the number of data-points in the samples.

Figures 37 and 36 show distribution and box plots about similarity ratio between 2-bit encoded accelerations and velocities, respectively. Similarly,



(a) (b)

Figure 37: Distribution plot (a) and box plot (b) of 2-bit encoded acceleration with *jump vector* = 2 and a *sliding window* set to 5% of the number of data-points in the samples. One outlier belonging to non-matching movements in (b) overlaps the similarity ratio of matching movements.



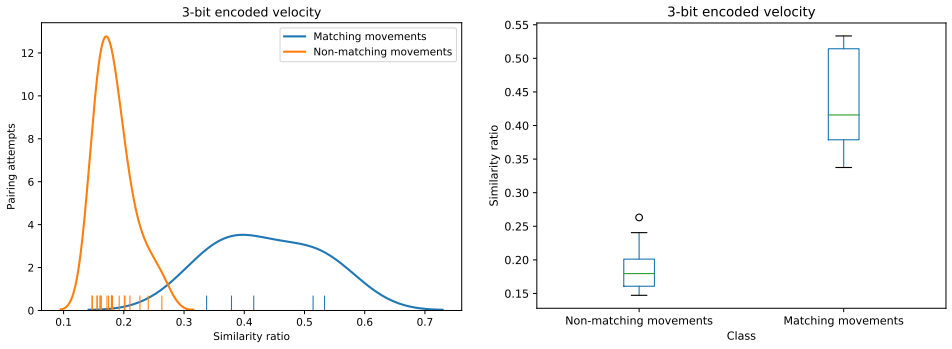
(a) (b)

Figure 38: Distribution plot (a) and box plot (b) of 3-bit encoded acceleration with *jump vector* = 2 and a *sliding window* set to 5% of the number of data-points in the samples. The two boxes in (b) are well divided, although their difference in median is lower than in Figure 36b.

Figures 38 and 39 show distribution and box plots about similarity ratio between 3-bit encoded accelerations and velocities, respectively.

Ideally, we are seeking for a vertical line that distinctly separates the distribution of non-matching similarity ratios from the matching ones, as shown in Figures 16a, 16c and 16d.

Distribution plots in Figures 37a, 36a, 38a and 39a are fit using a KDE (Kernel Density Estimate) [67] as non-parametric estimation of the probability density function of the collected samples. Hence, although all distribution plots show an overlap of the curves, box plots help to verify that in the vast majority



(a) (b)

Figure 39: Distribution plot (a) and box plot (b) of 3-bit encoded velocity with *jump vector* = 2 and a *sliding window* set to 5% of the number of data-points in the samples. The two boxes in (b) are well divided. Moreover, one outlier belonging to non-matching movements does not overlap with the similarity ratio of matching movements.

of the cases our non-matching samples have a distinct lower similarity ratio than the matching ones.

Then again, Figure 37b highlights the presence of one outlier that overlaps the similarity ratio of approximately 50% of matching samples. In other words, the non-matching outlier has a higher similarity ratio than the first quartile of matching movements, close to their median.

On the contrary, box plots in Figure 36b, 38b and 39b show a clear distinction between non-matching and matching distribution of similarity ratios.

In particular, Figure 36b shows a tangible difference between the two medians while considering the 2-bit encoded velocity. Furthermore, the 50% of the matching movements has a similarity ratio higher than 0.73. Moreover, the 3-bit encoded velocity (Figure 39b) presents only one outlier in the non-matching movement distribution which is adequately distant from the minimum value of the distribution of the matching movements.

Possible reasons, implications and potential improvements of these results are discussed in Section 5.

5 Conclusion

In this thesis, we scrutinized existing key establishment methods for secure device pairing. Then, we inspected secure paring protocols using *fuzzy data*, namely protocols using information that devices involved in the pairing process know only approximately.

Based on these, we studied the feasibility of a novel usable scheme that uses fuzzy data to securely pair a smart-watch and a mobile device, say a smart-phone, through gestures involving drawing on the screen supported by accelerometer data.

To this purpose, we developed mobile applications for smart-watch and smart-phone to sample signals. Moreover, we pre-processed data through a pipeline of steps, which aimed to synchronize data, remove the impact of the gravity and transform data to better compare them. In addition, after encoding data, we computed their similarity ratio.

We performed experiments aimed to find a similarity threshold that clearly separates the distribution of matching movements from non-matching ones in terms of similarity ratio. In other words, to goal was to verify whether encoded matching-movements have a clear, distinct, higher similarity ratio compared to non-matching movements.

Ensuring the presence of such separation is fundamental for verifying the effectiveness of our encoding method and metric function to compare signals. More importantly, a clear separation of distributions represents the evidence that pairing a smartwatch and a mobile device through multi-modal gestures is feasible. In this chapter, we discuss findings and potential improvements to our results.

The experiments were conducted by performing pairings for approximately 4 seconds each, in a simplified test-bed to gather adequate meaningful information and to avoid the impact of external factors. In fact, to minimize the impact of roll, pitch and yaw on acceleration data, we performed the drawings by keeping the smart-watch as parallel as possible to the ground and the finger perpendicular to the smart-phone. Furthermore, we used trajectories containing vertices, avoiding spiral-like drawings, to provoke definite peaks of acceleration and velocity, and to ease the subsequent analysis.

We analyzed the distribution of 100 different pairings, whose 10% were supposed to result as matching (i.e., pairings where finger drawing matched with wrist movements during the experiments). Signals were firstly sampled at a frequency of approximately 55 Hz at smart-phone and 110 Hz at smart-watch, namely the maximum possible sample frequency at the two devices in order to reproduce the original signals with no loss of information in the range of 0-20 Hz. Indeed, this range better describes movements of human limbs. Consequently, sampled signals were filtered applying a low-pass filter with a cut-off frequency of 20 Hz. Then, signals were encoded by using 2-bit and 3-bit Grey-code-like expansions with *jump* vector set to 2. This strategy was based on the fact that we sought to have minimal overlap of information while encoding signals (i.e.,

50% overlap with $jump = 2$). In addition, we aimed to study the impact on similarity ratio and threshold using a 2-bit and 3-bit code expansion.

According to our initial goals, in most of the cases our results showed that using our strategy, encoded matching signals have a clear, distinct, higher similarity ratio compared to non-matching ones, regardless of encoding with 2-bit or 3-bit code expansion, or considering their accelerations or velocities.

Moreover, the 2-bit encoding entails that matching movements result in a higher similarity ratio compared to 3-bit expansion. This trend is the evidence that 2-bit expansion better encodes original signals, minimizing possible discrepancies. On the other hand, the 3-bit encoding results in a higher difference in similarity ratio between matching and non-matching movements. Hence, although the 2-bit encoding better reproduces original signals, the 3-bit encoding better separates similarity ratios among matching and non-matching movements.

Then again, the distribution of 2-bit encoded accelerations presented one outlier whose similarity ratio overlapped the distribution of the encoded matching movements. Although the outlier impairs our goal to zero the number of false-positives (i.e., signals that are accepted as matching but not belonging to the same paring process) and false-negatives (i.e., signals that are rejected as non-matching but instead belonging to the same paring process), additional experiments taking into consideration a much larger number of pairings (e.g., at least one thousand different pairings) should be conducted to verify the presence of other outliers. However, the presence of only one outlier may be due to the fact that, while pre-processing and encoding signals, one sample coincided with another, resulting in a matching pairing. Hence, with the gathered results we cannot exclude the 2-bit encoded acceleration as a reliable method for accepting or rejecting pairings.

In any case, according to our initial goals, clear separation of distributions demonstrates that pairing devices through multi-modal gestures is feasible. As a consequence, it demonstrates the effectiveness of our pre-processing pipeline, encoding method and comparison metric function. Moreover, our study can benefit from future follow-ups. In the rest of this section we provide ideas for possible improvements.

As mentioned above, our study obtained satisfactory results in a simplified test-bed. In fact, we avoided spiral-like drawings to ease the detection of peaks in signals. However, future studies should consider the impact of this kind of trajectories on the similarity ratio. In case separation of distributions downgrades, we suggest implementing a function in the smart-phone mobile app that randomly generates patterns. These patterns should guide users to draw by connecting specific points on the smart-phone screen, similar to Fruit ninja [68] gaming app. The function should also avoid excessively fast changes of direction, as they would uselessly complicate the pre-processing steps.

Furthermore, to ensure sufficient entropy of signals, the system should use a timer that forces users to draw for a few seconds at least. Indeed, by using short signals the protocol would become insecure and prone to MITM attacks. In

this respect, follow-ups should simulate different kinds of attacks (e.g., MITM and shoulder-surfing attacks) and study their impact on the protocol.

Moreover, a comprehensive user-study would help to examine which gestures users actually prefer to use for performing pairings. It would also help to explore additional strategies to remove the impact of rotation, pitch and yaw on both devices, depending on the way users grab the smart-phone and draw with fingers. To this purpose, de-rotation using quaternions [69] may be helpful in this context.

In our study, the data synchronization step at the smart-watch reads the smart-phone sample to inspect the time duration of the drawing for eliminating clock skews and discarding information gathered by the accelerometer before the actual tap on the screen and after the finger lifting from the screen. In a real scenario, smart-watch should stop recording data as soon as the variance of the acceleration magnitude drops under a preset threshold for a specific time interval. Thus, when implementing the protocol in a real scenario, this improvement needs to be accomplished.

In addition, follow-ups should perform a fine-grained gravity removal in the preliminary standardization step. In our study, we compute the mean of the raw acceleration considering all data-points. Instead, the system would benefit using a moving mean over a small number of consecutive data-points only.

Future studies should also inspect the benefits of using additional filters (e.g., Savitzky-Golay filter [70]) in the transformation step, possibly leading to a higher similarity ratio for matching movements.

Eventually, follow-ups should inspect whether it is feasible to use multi-modal gestures for direct extraction of the session key instead of authenticating a previously exchanged secret through a Diffie-Hellman agreement.

References

[1] IDC, "IDC Forecasts Sustained Double-Digit Growth for Wearable Devices Led by Steady Adoption of Smartwatches." https://www.idc.com/getdoc.jsp?containerId=prUS44553518, 2019. Accessed : Mar 29, 2019.

[2] Google, "Pay with your smartwatch." https://support.google.com/pay/answer/7643998?co=GENIE.Platform%3DAndroid&hl=en. Accessed : Mar 29, 2019.

[3] Samsung, "Samsung Pay on Gear." https://www.samsung.com/us/support/owners/app/samsung-pay-gear. Accessed : Mar 29, 2019.

[4] Apple, "Set up Apple Pay." https://support.apple.com/en-gb/HT204506#applewatch. Accessed : Mar 29, 2019.

[5] Apple, "iOS Health suite." https://www.apple.com/lae/ios/health/. Accessed : Apr 1, 2019.

[6] iMore, "Apple Watch and activity tracking: Everything you need to know!." https://www.imore.com/apple-watch-and-activity-tracking-what-you-need-know. Accessed : Apr 1, 2019.

[7] Google, "Get Healthy with Google Fit." https://wearos.google.com/#stay-healthy. Accessed : Apr 1, 2019.

[8] Samsung, "Samsung Health." https://www.samsung.com/global/galaxy/apps/samsung-health/. Accessed : Apr 1, 2019.

[9] Apple, "Use your Apple Watch as a remote control for your Apple TV or iTunes." https://support.apple.com/en-us/HT205549. Accessed : Apr 1, 2019.

[10] phoneArena, "How to use your Android Wear smartwatch as a remote shutter for a smartphone camera." https://www.phonearena.com/news/How-to-use-your-Android-Wear-smartwatch-as-a-remote-shutter-for-a-smartphone-camera_id67946. Accessed : Apr 1, 2019.

[11] U. Remote, "Turn your smartphone into a universal remote control.." https://www.unifiedremote.com/. Accessed : Apr 1, 2019.

[12] C. Xu, P. H. Pathak, and P. Mohapatra, "Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch," in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, pp. 9–14, ACM, 2015.

[13] S. Sen, V. Subbaraju, A. Misra, R. K. Balan, and Y. Lee, "The case for smartwatch-based diet monitoring," in *2015 IEEE international conference on pervasive computing and communication workshops (PerCom workshops)*, pp. 585–590, IEEE, 2015.

[14] E. Uzun, K. Karvonen, and N. Asokan, "Usability analysis of secure pairing methods," in *International Conference on Financial Cryptography and Data Security*, pp. 307–324, Springer, 2007.

[15] W. Albert and T. Tullis, *Measuring the user experience: collecting, analyzing, and presenting usability metrics*. Newnes, 2013.

[16] Y. S. Kim, S. H. Kim, and S. H. Jin, "Srs-based automatic secure device pairing on audio channels," in *2010 International Conference for Internet Technology and Secured Transactions*, pp. 1–6, IEEE, 2010.

[17] W. R. Claycomb and D. Shin, "Secure device pairing using audio," in *43rd Annual 2009 International Carnahan Conference on Security Technology*, pp. 77–84, IEEE, 2009.

[18] C. Soriente, G. Tsudik, and E. Uzun, "Hapadep: human-assisted pure audio device pairing," in *International Conference on Information Security*, pp. 385–400, Springer, 2008.

[19] S. Sigg, Y. Ji, N. Nguyen, and A. Huynh, "Adhocpairing: Spontaneous audio based secure device pairing for android mobile devices," in *Proceedings of the 4th International Workshop on Security and Privacy in Spontaneous Interaction and Mobile Phone Use, IWSSI/SPMU*, vol. 12, 2012.

[20] N. Nguyen, S. Sigg, A. Huynh, and Y. Ji, "Using ambient audio in secure mobile phone communication," in *2012 IEEE International Conference on Pervasive Computing and Communications Workshops*, pp. 431–434, IEEE, 2012.

[21] M. T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun, "Loud and clear: Human-verifiable authentication based on audio," in *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, pp. 10–10, IEEE, 2006.

[22] A. Varshavsky, A. Scannell, A. LaMarca, and E. De Lara, "Amigo: Proximity-based authentication of mobile devices," in *International Conference on Ubiquitous Computing*, pp. 253–270, Springer, 2007.

[23] S. Mathur, R. Miller, A. Varshavsky, W. Trappe, and N. Mandayam, "Proximate: proximity-based secure pairing using ambient wireless signals," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pp. 211–224, ACM, 2011.

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

[25] B. Groza and R. Mayrhofer, "Saphe: simple accelerometer based wireless pairing with heuristic trees," in *Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia*, pp. 161–168, ACM, 2012.

[26] D. Kirovski, M. Sinclair, and D. Wilson, "The martini synch," *Technical Report MSR-TR-2007-123, Microsoft Research*, 2007.

[27] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.-W. Gellersen, "Smart-its friends: A technique for users to easily establish connections between smart artefacts," in *international conference on Ubiquitous Computing*, pp. 116–122, Springer, 2001.

[28] M. Sethi, M. Antikainen, and T. Aura, "Commitment-based device pairing with synchronized drawing," in *2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 181–189, IEEE, 2014.

[29] A. Håkansson, "Portal of research methods and methodologies for research projects and degree projects," in *The 2013 World Congress in Computer Science, Computer Engineering, and Applied Computing WORLDCOMP 2013; Las Vegas, Nevada, USA, 22-25 July*, pp. 67–73, CSREA Press USA, 2013.

[30] J. Suomalainen, J. Valkonen, and N. Asokan, "Security associations in personal networks: A comparative analysis," in *European Workshop on Security in Ad-hoc and Sensor Networks*, pp. 43–57, Springer, 2007.

[31] S. Gangan, "A review of man-in-the-middle attacks," *arXiv preprint arXiv:1504.02115*, 2015.

[32] K. H. Afkhamie, S. Katar, L. Yonge, and R. Newman, "An overview of the upcoming homeplug av standard," in *Power Line Communications and Its Applications, 2005 International Symposium on*, pp. 400–404, IEEE, 2005.

[33] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[34] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on information theory*, vol. 29, no. 2, pp. 198–208, 1983.

[35] J. Padgette, K. Scarfone, and L. Chen, "Guide to bluetooth security," *NIST Special Publication*, vol. 800, no. 121, p. 25, 2012.

[36] F. PUB, "Secure hash standard (shs)," *FIPS PUB*, vol. 180, no. 4, 2012.

[37] C. Gehrmann, C. J. Mitchell, and K. Nyberg, "Manual authentication for wireless devices," *RSA Cryptobytes*, vol. 7, no. 1, pp. 29–37, 2004.

[38] B. P. Crow, I. Widjaja, J. G. Kim, and P. T. Sakai, "Ieee 802.11 wireless local area networks," *IEEE Communications magazine*, vol. 35, no. 9, pp. 116–126, 1997.

[39] S. Antifakos, B. Schiele, and L. E. Holmquist, "Grouping mechanisms for smart objects based on implicit interaction and context proximity," in *Adjunct Proceedings of International Conference on Ubiquitous Computing (Ubicomp), Seattle, USA*, Citeseer, 2003.

[40] K. Hinckley, "Synchronous gestures for multiple persons and computers," in *Proceedings of the 16th annual ACM symposium on User interface software and technology*, pp. 149–158, ACM, 2003.

[41] R. D. Findling, M. Muaaz, D. Hintze, and R. Mayrhofer, "Shakeunlock: Securely unlock mobile devices by shaking them together," in *Proceedings of the 12th International Conference on Advances in Mobile Computing and Multimedia*, pp. 165–174, ACM, 2014.

[42] D. Bichler, G. Stromberg, M. Huemer, and M. Löw, "Key generation based on acceleration data of shaking processes," in *International Conference on Ubiquitous Computing*, pp. 304–317, Springer, 2007.

[43] M. Maes, "Polygonal shape recognition using string-matching techniques," *Pattern Recognition*, vol. 24, no. 5, pp. 433–440, 1991.

[44] H. Nyquist, "Certain topics in telegraph transmission theory," *Transactions of the American Institute of Electrical Engineers*, vol. 47, no. 2, pp. 617–644, 1928.

[45] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[46] J. Burbank, D. Mills, and W. Kasch, "Network time protocol version 4: Protocol and algorithms specification," *Network*, 2010.

[47] J.-P. van Brakel, "Python implementation of the smoothed z-score algorithm." https://stackoverflow.com/questions/22583391/peak-signal-detection-in-realtime-timeseries-data/43512887#43512887. Accessed : Apr 5, 2019.

[48] S. Glen, "Z-Score: Definition, Formula and Calculation." https://www.statisticshowto.datasciencecentral.com/probability-and-statistics/z-score/. Accessed : Apr 22, 2019.

[49] NASA, "Aircraft Rotations. Body Axes.." https://www.grc.nasa.gov/WWW/K-12/airplane/rotations.html. Accessed : Apr 2, 2019.

[50] D. A. Winter, *Biomechanics and motor control of human movement*. John Wiley & Sons, 2009.

[51] C. V. Bouten, K. T. Koekkoek, M. Verduin, R. Kodde, and J. D. Janssen, "A triaxial accelerometer and portable data processing unit for the assessment of daily physical activity," *IEEE transactions on biomedical engineering*, vol. 44, no. 3, pp. 136–147, 1997.

[52] G. Frank, "Pulse code communication," Mar. 17 1953. US Patent 2,632,058.

[53] Huawei, "Huawei Watch 2 native specifications." https://consumer.huawei.com/en/wearables/watch2/specs/. Accessed : Apr 3, 2019.

[54] LG, "Nexus 5X native specifications." https://www.lg.com/uk/mobile-phones/lg-H791. Accessed : Apr 3, 2019.

[55] D. Bernardi, "Source code of smart-watch application for Multi-Modal Gestures thesis project." https://github.com/Chiollodario/MMGPairing/tree/master/WristWatch. Accessed : Apr 27, 2019.

[56] Google, "Android SensorManager API." https://developer.android.com/reference/android/hardware/SensorManager. Accessed : Apr 4, 2019.

[57] Google, "Android Sensor API list." https://developer.android.com/reference/android/hardware/Sensor. Accessed : Apr 4, 2019.

[58] D. Bernardi, "Source code of smart-phone application for Multi-Modal Gestures thesis project." https://github.com/Chiollodario/MMGPairing/tree/master/Smartphone. Accessed : Apr 27, 2019.

[59] Google, "Android MotionEvent API." https://developer.android.com/reference/android/view/MotionEvent. Accessed : Apr 4, 2019.

[60] Google, "Android VelocityTracker API." https://developer.android.com/reference/android/view/VelocityTracker. Accessed : Apr 4, 2019.

[61] D. Bernardi, "Main Python script for Multi-Modal Gestures thesis project." https://github.com/Chiollodario/MMGPairing/blob/master/paircoding.py. Accessed : Apr 27, 2019.

[62] C. McKinney, "Python Data Analysis Library." https://pandas.pydata.org/. Accessed : Apr 4, 2019.

[63] NumPy developers, "NumPy." http://www.numpy.org/. Accessed : Apr 4, 2019.

[64] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python." https://www.scipy.org/. Accessed : Apr 5, 2019.

[65] C. Hunter, Droettboom, "Matplotlib." https://matplotlib.org/. Accessed : Apr 4, 2019.

[66] Electronics Tutorials, "Exclusive-NOR Gate Tutorial." https://www.electronics-tutorials.ws/logic/logic_8.html. Accessed : May 8, 2019.

[67] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," *The Annals of Mathematical Statistics*, pp. 832–837, 1956.

[68] Halfbrick Studios, "Fruit Ninja - The greatest fruit-slicing game in the world." https://fruitninja.com/. Accessed : May 22, 2019.

[69] J. B. Kuipers *et al.*, *Quaternions and rotation sequences*, vol. 66. Princeton university press Princeton, 1999.

[70] R. W. Schafer *et al.*, "What is a Savitzky-Golay filter," *IEEE Signal processing magazine*, vol. 28, no. 4, pp. 111–117, 2011.