



Instituto Universitário de Lisboa

Department of Information Science and Technology

Multifactor Authentication using Smartphone as Token

Pedro Fonseca das Neves

A Dissertation presented in partial fulfillment of the Requirements

for the Degree of

Master in Computer Engineering

Supervisor

Dr. Luís Miguel Martins Nunes, assistant professor,

ISCTE-IUL

Co-Supervisor

Dr. André Ribeiro Lourenço, assistant professor,

ISEL-IPL

Resumo

A biometria é uma área de estudo que observou desenvolvimentos relevantes na última década. Em específico, a biometria baseada no eletrocardiograma (ECG) é atualmente considerada uma fonte de identificação confiável. Um dos maiores avanços nesta tecnologia consiste na evolução da autenticação *off-the-person*, que permite realizar a aquisição de sinal de forma não intrusiva usando as mãos do utilizador. Contudo, a identificação através deste método ainda apresenta uma performance relativamente baixa quando usada uma base de dados de dimensão acima das dezenas.

Nesta dissertação sugerimos usar a autenticação ECG associada a um telemóvel a funcionar como *security token* com o objectivo de melhorar a performance e diminuir o tempo necessário para o reconhecimento. Para isso, desenvolvemos a nossa solução usando a tecnologia Bluetooth (BL) clássico, mas também Bluetooth Low Energy (BLE) para preservar a bateria do telemóvel; além disto, desenvolvemos as aplicações em Windows Phone e também Android, dadas as limitações que encontramos. Para criar um ambiente mais versátil e móvel, usámos a recente plataforma Intel Edison.

Os resultados obtidos provam que a nossa solução é viável. Executámos uma série de testes, nos quais observámos uma melhoria nos tempos associados à autenticação quando comparados com o cenário clássico de identificação por ECG. Adicionalmente, a performance do ECG no que diz respeito ao número de falsos-negativos e falsos-positivos apresentou também melhoria.

Abstract

Biometrics are a field of study with relevant developments in the last decade. Specifically, electrocardiogram (ECG) based biometrics are now deemed a reliable source of identification. One of the major advances in this technology was the improvements in off-the-person authentication, by requiring nothing more than dry electrodes or conductive fabrics to acquire an ECG signal in a non-intrusive way through the user's hands. However, identification still has a relatively poor performance when using large user databases.

In this dissertation we suggest using ECG authentication associated with a smartphone security token in order to improve performance and decrease the time required for the recognition. We develop this technique in a user authentication scenario for a Windows login. We developed our solution using both normal Bluetooth (BT) and Bluetooth Low Energy (BLE) technologies to preserve phone battery; also, we develop apps for Windows Phone and Android, due to limitations detected. Additionally, we took advantage of the Intel Edison's mobility features to create a more versatile environment.

Results proved our solution to be possible. We executed a series of tests, through which we observed an improvement in authentication times when compared to a simple ECG identification scenario. Also, ECG performance in terms of false-negatives and false-positives is also increased.

Contents

<i>Resumo</i>	iii
<i>Abstract</i>	iv
<i>Abbreviations</i>	vii
<i>List of Figures</i>	viii
<i>List of Tables</i>	x
1. Introduction	1
1.1. Objectives	1
1.2. Contribution	2
1.3. Structure of the Dissertation	3
2. State of the Art	4
2.1. Introduction	4
2.2. Bluetooth Low Energy	4
2.3. Authentication Methods	6
2.3.1 Biometrics	7
2.3.2 ECG-based authentication	9
2.3.3 Smartphones as Security Tokens	14
2.4. Custom Authentication Experience in Windows Environment	17
2.4.1. pGina	19
2.5. Intel Edison	21
2.6. Conclusion	22
3. Implementation	24
3.1. Introduction	24
3.2. Classic Bluetooth communication	25

3.2.1.	First Scenario – Bluetooth Discovery.....	25
3.2.2.	Second Scenario – Universal App.....	28
3.2.3.	Third Scenario – Windows Phone App.....	29
3.2.4.	Fourth Scenario – Windows Phone 8.1 Background Task.....	31
3.2.5.	Fifth Scenario – Windows Phone 8.0 App.....	32
3.2.6.	Sixth Scenario – Proximity-based token.....	36
3.2.7.	ECG Keyboard authentication.....	38
3.3.	Bluetooth Low Energy communication.....	39
3.3.1.	Proximity.....	41
3.3.2.	First scenario – Post Delayed Runnable.....	41
3.3.3.	Second Scenario – AlarmManager.....	43
3.3.4.	Third Scenario – Subscribing a GATT service.....	43
3.3.5.	Fourth Scenario – Using accelerometer and BLE.....	45
4.	Tests.....	46
4.4.1.	Signal strength reliability.....	46
4.4.2.	Bluetooth performance and battery impact.....	47
4.4.3.	Token/ECG vs ECG only comparison.....	52
5.	Conclusions and Future Work.....	58
Appendices.....		60
A.	Installation Procedures.....	60
Bibliography.....		62

Abbreviations

API – Application Programming Interface

BL - Bluetooth

BLE – Bluetooth Low Energy

CP – Credential Provider

ECG - Electrocardiogram

GATT – Generic Attribute Profile

GSM – Global System for Mobile Communications

IMSI – International Mobile Subscriber Identity

WP – Windows Phone

List of Figures

Figure 1 - LE Security Modes 1 and 2.	5
Figure 2 - Heartbeat wave.	9
Figure 3 - Common examples of heart rates.....	10
Figure 4 - An example of an ECG.....	10
Figure 5 - Electrolycra on the top, dry Ag/AgCl on the bottom.....	11
Figure 6 - ECG recorded at hand palms and fingers	13
Figure 7 - Example of a segmented and filtered ECG taken from the hand palms.	13
Figure 8 - Authentication scheme.....	14
Figure 9 - Authentication scheme.....	15
Figure 10 - Credential process for Windows 7 and later.	18
Figure 11 - Diagram showing the main pGina systems interacting in a common logon.	19
Figure 12 - Pipeline representing the three stages of authentication process.....	20
Figure 13 - Intel Edison.....	21
Figure 14 - Set of different implementations we worked in.....	25
Figure 15 - First scenario detection scheme.	27
Figure 16 - Logon button.....	28
Figure 17 - Universal app scheme.	29
Figure 18 - Windows phone app scheme.	30
Figure 19 - Background task scheme. No need for human interaction with WP.	31
Figure 20 - Communication scheme.....	33
Figure 21 - Hypothesis nr.1..	34
Figure 22 – Hypothesis nr.2..	35
Figure 23 - Hypothesis nr.3..	36
Figure 24 - Proximity scheme..	37
Figure 25 - A photo of the actual used keyboard.	38
Figure 26 - On the left, without a smartphone; on the right, with a smartphone.....	39
Figure 27 - Photo of the actual Edison board used.....	40
Figure 28 - First scenario.....	42
Figure 29 - Third scenario..	44
Figure 30 - Error in Identification (EID) based on the number of subjects.....	53

Figure 31 - False Acceptance Rate (FAR) and False Rejection Rate (FRR) depending on acceptance threshold in two individuals.....	54
Figure 32 - BLE database table.....	60
Figure 33 - Classic Bluetooth database table.....	61

List of Tables

Table 1 - RSSI values depending on distance.	46
Table 2 - Identification and authentication times using an 8 user ECG database.	55
Table 3 - Identification and authentication times using a 68 user database.	55
Table 4 - Identification and authentication times using an 8 user ECG database (in seconds), with the acquisition times subtracted.....	56
Table 5 - Identification and authentication times using a 68 user database (in seconds), with the acquisition times subtracted.	56

1. Introduction

The last twenty years lead the phone industry to a big growth in business volume, but also in the technology used. We went from mobile phones that were not that mobile, to smartphones with the same (or more) computational power as our computers. They replace cameras, clocks, GPS, books, mp3 players, and an extensive list of apparatus that became essential for everyday life. This extremely rapid evolution led to a number of possibilities in which these capabilities could be applied to in order to enhance our everyday activities or even create new automations.

A field of study that also presented great progress in the last years was biometrics. Examiners analyzing ink-stained fingerprints would be far from imagining computers would read fingerprints digitally, or that a person could be identified using unique characteristics such as iris or heartbeat (Jain et al., 2004).

These contexts led us to trying a solution for a specific problem. An electrocardiogram authentication using a keyboard presented a significant time delay associated with identification (comparing one ECG to N database entries, where N is the cardinality of the database) and authentication (comparing one ECG to one database entry), as well as a consequent decrease in identification performance with larger databases (Carreiras et al, 2011); we suggest using a smartphone, having a role of security token, to improve this process by sending the individual's identity previous to the ECG acquisition. With this, we aim to develop a multi-factor authentication scheme in Windows environment capable of joining both ECG and security token authentications in an everyday usage.

1.1. Objectives

Starting this project, we intended to create a multi-factor authentication scheme in Windows environment. This two-factor scheme would include a smartphone used as a security token and an electrocardiogram reader keyboard.

Part of objective was to improve the performance of the ECG keyboard. A typical authentication using this method would include identification or authentication,

depending on whether the username identity is previously known – if not, the identification process has worse performance than authentication, mainly when working with large databases. Also, authentication is preferable because it presents a better performance, mainly regarding Equal Error Rate (EER). This rate refers to the point from which a False Acceptance Rate (FAR) is equal to False Rejection Rate (FRR) (Carreiras et. al, 2011). Using a smartphone, we propose a previous identification based on its presence, thus eliminating part of the process. Also, this would deal with an additional problem regarding this method, which refers to the loss of performance when a bigger user database is used.

The Windows environment was chosen to prove the possibility of applying such security methods in an everyday context. The initial objective was to login a Windows computer using a Windows Phone and the ECG keyboard.

In order to guarantee a battery-friendly scheme (especially on the smartphone side), we propose and test a communication protocol between the smartphone and the computer using Bluetooth Low Energy.

Also, we aim to take advantage of the Intel Edison's mobility features to create a more versatile environment. This small board runs Linux and is capable of handling Wi-Fi and Bluetooth Low Energy connections. Additionally, it could be embedded in various everyday items, such as a driving wheel or a bicycle, allowing the possibility of storing a user profile and communicate with a variety of peripherals.

1.2. Contribution

The process of authenticating an individual using ECG may become more time-consuming when associated with large databases. Therefore, we suggest using an extra authentication factor – a smartphone – in order to send the identification of the user. This would allow the ECG authentication to avoid comparing one reading to all the elements of a database and, instead, compare it to a single element.

As intended, we provided a solution involving a smartphone in a multi-factor authentication scheme. Also, we provided an increase in performance of ECG authentication by not requiring the identification part and providing a better confidence

level to the process when executing an ECG authentication with previously known user identity.

1.3. Structure of the Dissertation

In Chapter 2, we introduce some of the most relevant concepts and research related to multi-factor authentication using a mobile phone, as well as a background on biometrics, specifically ECG authentication. In Chapter 3, we explain and justify our different approaches and choices through the development of the suggested solution; also, we test the most relevant cases and compare results. In Chapter 4, we conclude with a synthesis of our dissertation and suggest future work.

2. State of the Art

2.1. Introduction

The use of smartphones has been growing as our society tends to focus on communication and social networking. This revolution led to a point where it is common to possess a mobile phone capable of interacting with other devices using a number of different technologies, from Wi-Fi to Bluetooth. This allows researchers to think of new ways of integrating these devices in a complete technological experience.

One of the different usages given to smartphones is their integration in a multi-factor authentication context. This may occur by making a mobile phone act as a token to reinforce security. In this document, we expose a possibility of including such methodology in an Electrocardiogram-based (ECG) authentication scheme in Windows environments. Initially, we provide a brief introduction to Bluetooth Low Energy; afterwards, we talk about authentication methods, specifically biometrics (and the individual case of ECG) and the use of smartphones as security tokens; finally, we present the custom authentication experience in Windows, and refer to the specific case of pGina.

2.2. Bluetooth Low Energy

Bluetooth Low Energy (BLE), part of the Bluetooth 4.0 specification, is a technology designed for low-power consumption capable of, according to theoretical results, running on a device with a coin cell battery for a timeframe of 2 days to 14.1 years depending on its activity (Gomez et al, 2012). It is single-hop solution – meaning it does not require intermediary network points between both sides – and is typically used in areas as healthcare, consumer electronics and security. In fact, Apple uses this technology in iBeacon (“iOS: Understanding iBeacon”, 2014), which consists in system that alerts an app when leaving or approaching an iBeacon point.

BLE protocol is based on a controller-host scheme, which is the scheme generally used when connecting a host system (for example, a computer) to network or storage devices. The standard communication protocol between the host stack and the controller is Host Controller Interface (HCI). The controller is responsible for both Physical Layer and Link Layer and usually found as small System-on-Chip (SoC) along with an integrated radio; the host is executed on an application processor and implements upper layer functions such as Logical Link Control and Adaptation Protocol (L2CAP), Attribute Protocol (ATT) and Generic Attribute Profile (GATT) – these are some of the most common communication protocols used in Bluetooth connections.

The ATT protocol is run on a L2CAP channel and is meant for server-client schemes. However, the client/server role is independent from the master/slave role defined by the Bluetooth connection. A set of attributes is defined on server side, each one containing a data structure. This structure is managed by GATT, where a framework is defined and used in the discovery and communication processes. An example is a server running a *temperature sensor*, which provides a service that may include a characteristic *temperature* composed by a set of attributes, such as temperature unit and *temperature measures* (Gomez et al., 2012).

Security in BLE consists of two main modes: LE Security Mode 1 (when referring to security applied at the Link Layer, in the controller) and LE Security Mode 2 (applied at the ATT Layer, at the host). The first mode implements Cipher Block Chaining-Message Authentication Code (CCM), an algorithm designed to provide encryption and authentication; the second mode works in an authentication but unencrypted environment, allowing authenticated data to be send over an unencrypted Link Layer connection by adding a signature to data payload at the ATT Layer. Each mode requires specific types of pairing, as can be seen in Figure 1.

		Pairing	Encryption	Data Integrity	Layer
LE Security Mode 1	Level 1	No	No	No	Link Layer
	Level 2	Unauthenticated	Yes	Yes	
	Level 3	Authenticated	Yes	Yes	
LE Security Mode 2	Level 1	Unauthenticated	No	Yes	ATT layer
	Level 2	Authenticated	No	Yes	

Figure 1 - LE Security Modes 1 and 2, and their respective requirements in terms of pairing (Adapted from Gomez et al., 2012).

The pairing process consists of three phases: (i) the two devices inform each other about their possible inputs and outputs; (ii) a Temporary Key (TK) is agreed and used by both, this phase is where the Short-Term Key (STK) is generated, to be used in phase 3; (iii) the last phase is where the Long-Term Key (LTK, used to generate the 128-bit key used in Link Layer encryption and authentication), the Connection Signature Resolving Key (CSRK, used to sign data at ATT Layer) and Identity Resolving Key (IRK, which allows the generation of volatile private addresses) are defined.

Additionally, BLE provides the *privacy feature*, allowing the use of volatile private addresses by a device. These addresses may be resolved only by trusted devices.

Thus, bearing in mind the low consumption characteristics of BLE, as well as its availability, we considered the best solution for the communication between a mobile phone and a computer – Wi-Fi was ruled out as it would require both devices to be in the same network.

2.3. Authentication Methods

The first access point to many individual computers and networks often is user authentication. This situation exposes the crucial contribution of this process to the security of a whole system (Braz et al., 2006).

User authentication is based on three commonly used factors: *what you know* (passwords), *what you have* (security tokens) and *what you are* (biometrics) (Zheng, 2001).

Passwords (*what you know*) may be easily breakable if users ignore standard security rules (Zviran et.al, 1999; Bonneau, 2012), such as creating long passwords, avoiding trivial phrases and including numbers and a mix of upper and lower case letters or even writing them down. However, the adoption of these security advices often create a usability issue due to the human difficulty in remembering complex phrases. (Braz et al., 2006). Also, further vulnerability may come due to computing power rapid improvement, allowing hackers to test a great number of different keywords in a short

period of time - therefore, tokens provide a stronger security system by working with longer passwords (Zheng, 2001).

A security token (*what you have*) can be classified as *static token*, *synchronous dynamic token*, *asynchronous token* and *challenge response token*. *Static tokens* refer to any device which is only capable of storing authentication data (such as cryptographic keys, digital signature and biometric data); *Synchronous dynamic tokens* generate one-time passwords using a synchronous mechanism between the token and the server; *Asynchronous tokens* provide a challenge response to a number entered on the token's keypad; *Challenge response* performs authentication using the challenge response protocol (Zheng, 2001).

One of the possibilities for an authentication scheme is suggested as an RFID-based authentication solution (Radio-Frequency Identification) (Syta et al., 2010). This process requires two, preferably distinct, authentication methods which should be simple and easy in the eyes of the user, independent from user interaction, cost-effective and permanent (presence of the individual should login; absence should logout).

Biometrics (*what you are*) consists of authenticating or identifying a person using physical and/or behavioral information. This information is eligible to be used if it presents the following characteristics: (1) universality, meaning that every person has this characteristic; (2) distinctiveness, resulting in a difference of this characteristic between every individual; (3) permanence, which means this characteristic should not vary with time in a way that may interfere with the recognition; (4) collectability, meaning this characteristic is quantitatively measurable. Additionally, practical situation also reveal requirements as accuracy and speed, user-friendliness and fraud-resistance (Jain et al., 2004).

2.3.1 Biometrics

Biometric automated authentication methods have been growing in the last years, transferring security to factors which are difficult to replicate – such as fingerprint, retina scan or face pattern. This, nevertheless, also created ethical discussion around computer storage of an individual's characteristics (Zheng et al. 2011).

Some of the improvements in biometric authentication are related to the high accuracy and transparency of the process, as well as the distinctive nature of the authentication elements (Zheng 2011). These elements are mostly classified as biological or behavioral. Biological elements refer to an individual's physical characteristics – fingerprint or retina, for example; Behavioral elements, on the other hand, require learning and are not immediate – keystroke or signature dynamics are examples of such elements.

Biometrics can be applied in three main contexts: commercial context can adopt this method to enhance computer network logons, ATMs, medical records, etc.; Government applications may occur in national ID cards, passport control, among others; Forensic application to corpse identification or criminal identification, among others (Jain et al., 2004).

Some of the common biometrics elements used are: DNA, deoxyribonucleic acid, often used in forensic context, is easily contaminated and stolen; Ear, specifically its shape, are however not very distinctive; Face, which is easily obtained but has some limitations associated with surrounding illumination, among others; Infrared Thermogram, obtained from a face, hand and hand vein, is easy to obtain but is conditioned by surrounding heat; Fingerprint, which is an affordable solution and its accuracy is adequate to small/medium scale authentications, although some factors as fingerprint deterioration may oppose.

Some studies also suggest other easy-to-use methods, namely hand-based methods (Ramalho et. al, 2012). The authors suggest an identification system based on three hand characteristics – hand geometry (HG), palmprint (PP) and finger surfaces (FS). Hand geometry, contrarily to PP and FS, has not a consensus regarding its uniqueness; however, this fact was kept in mind by Ramalho et. al. In fact, in terms of processing, first the hand geometry is read to identify the best matches (creating a small list of probable candidates), and then the final decision is based on palmprint and finger surfaces.

Additionally, another method regarding the use of hands in an authentication process consists of reading electrocardiogram information from hand palms. The ECG reading consists of a difference of electrical potential between two limbs, and represents an external measure of the electrical activity of the heart.

2.3.2 ECG-based authentication

The field of Electrocardiographic research has been facing a problem of lacking large datasets to test the developed methods. Many solutions were proposed, namely the use of repositories containing clinical and laboratorial data. Silva et al. (2013) offer a different approach they call *off-the-person* – meaning collecting ECG data from the hand palms and fingers – through which they intend to facilitate the large scale acquisition of ECG signals.

The intrusiveness of ECG (Silva et al., 2013) data acquisition can be classified as: (1) *In-the-person*, which means implanting devices (e.g. pacemakers) and is highly intrusive, potentially resulting in an industrial application failure; (2) *On-the-person*, meaning the devices that often require the use of paste or gel and work by being attached to the person; (3) *Off-the-person*, which includes devices capable of acquiring ECG signals without any preparation, simply by maintaining contact for a few seconds with the acquisition device.

A regular heartbeat signal is composed of three elements, being T-wave, P-wave and QRS complex (Cardiology Explained, 2004).

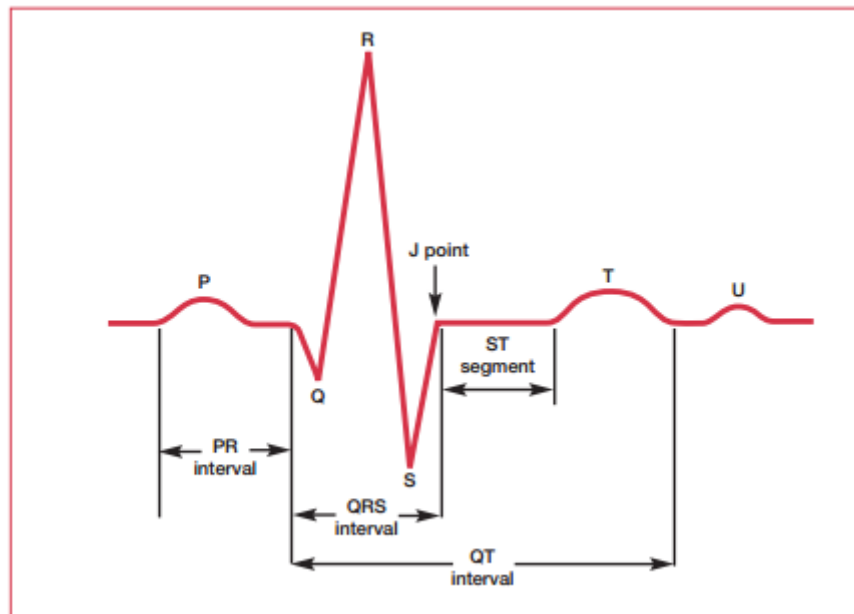


Figure 2 - Heartbeat wave (Adapted from Cardiology Explained, 2004).

As seen in Figure 2, the heartbeat shows the P-wave, a first small deflection; the QRS complex, consisting of three separate waves (Q, R and S) and separated from P-wave by an interval called PR-interval; and the T-wave, separated from QRS by an interval called ST segment.

Additionally, these waves and intervals may have different durations due to a person's state of mind at the reading moment. This may result in very distinct readings of the same subject, as depicted in figure 3 and 4.

Number of large squares between QRS complexes	Heart rate (bpm)
5	60
4	75
3	100
2	150

Figure 3 - Common examples of heart rates (Adapted from Cardiology Explained, 2004).

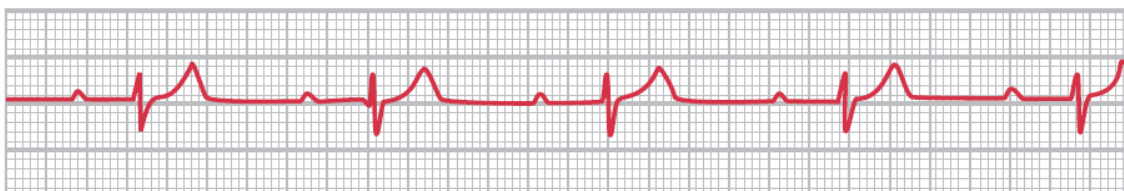


Figure 4 - An example of an ECG showing the squares (Adapted from Cardiology Explained, 2004).

Heartbeat signal recognition is used in a number of different IT applications, such as information security and user identification, due to its characteristics – measurability, uniqueness and universality, among others. This method of identification solves several issues related to other biometric identification processes, e.g. it cannot be faked using a deceased body (Islam et al., 2014). On the other hand, this method also takes advantage of the fact of being relatively easy to execute, as a good heartbeat signal can be acquired from fingers (Silva et al., 2013) and used in practical situations (authentication, etc.).

One major inconvenience can be, however, the acquisition of this element. Authors propose a solution for ECG acquisition by reading signals on hand palms with dry

Ag/AgCl (silver chloride) electrodes, which are types of reference electrodes known for their stability and electrode potential. Also, they are able to read at the fingers using Electrolycras (see Figure 5), and use both readings to compare each other's results. Additionally, the authors incorporate both neutral and emotion inducing tasks - using electrodermal (EDA) data -, motivated by the recent study of relation between ECG heartbeat waveform and emotional stimuli. They refer studies where people were submitted to a set of stimuli induced by a video game or visual elements and concluded that there are small changes in the form of the ECG heartbeat signal related with different emotional or affective situations. Therefore, this aspect embraces the intra-subject variability of acquired data.



Figure 5 - Electrolycras on the top, dry Ag/AgCl on the bottom (Adapted from Silva et al., 2013).

Silva et al. followed this line of research and performed two series of ECG data acquisition sessions – one long data-set and one short data-set. The short term sessions were executed with a group of participants who were submitted to the visualization of a video, along with background sound through headphones, lasting 1 minute and presenting a triggering stimulus in the last 5 seconds. With this setup they performed two distinct sequences of tests, to achieve both low and high arousal, using different videos and soundtracks. The entire procedure was intended to have a completion time of near 5 minutes, consisting of three phases: (1) initial briefing and informed consent, during which only ECG signals at the hand palms and fingers were recorded; (2) low

arousal video, where participants were also fitted with the EDA sensor and the headphones were used; (3) high arousal video, maintaining both EDA sensor and headphones. An optical synchronization method was applied with the intention of allowing time fitting in the order of milliseconds. An overall total of 65 participants' data was acquired, 49 males and 65 females, all reported as healthy.

The long term data-set was acquired during several days and using different settings. A single sensor was used, with dry Ag/AgCl electrodes, to acquire ECG data and the setup was prepared at the cardiopneumology laboratory at the Escola Superior de Saúde da Cruz Vermelha Portuguesa. There were two moments of data acquisition with a 3-month separation, in order to examine changes in the ECG morphology over time. In both times the participants were presented with the informed consent document, and ECG signals were recorded during 2 minutes with the subject sitting in resting position and with two dry A/AgCl electrodes in their fingers, one on the left one on the right hands. A total of 63 subjects were analyzed, 14 males and 49 females, all reported healthy.

This experiments led to several conclusions. Firstly, the time synchronization method allowed to compare independent subject data in the same time base. Also, they found the two different signal sources to be very similar, representing high detail heartbeat waveform data, as seen in Figure 6.

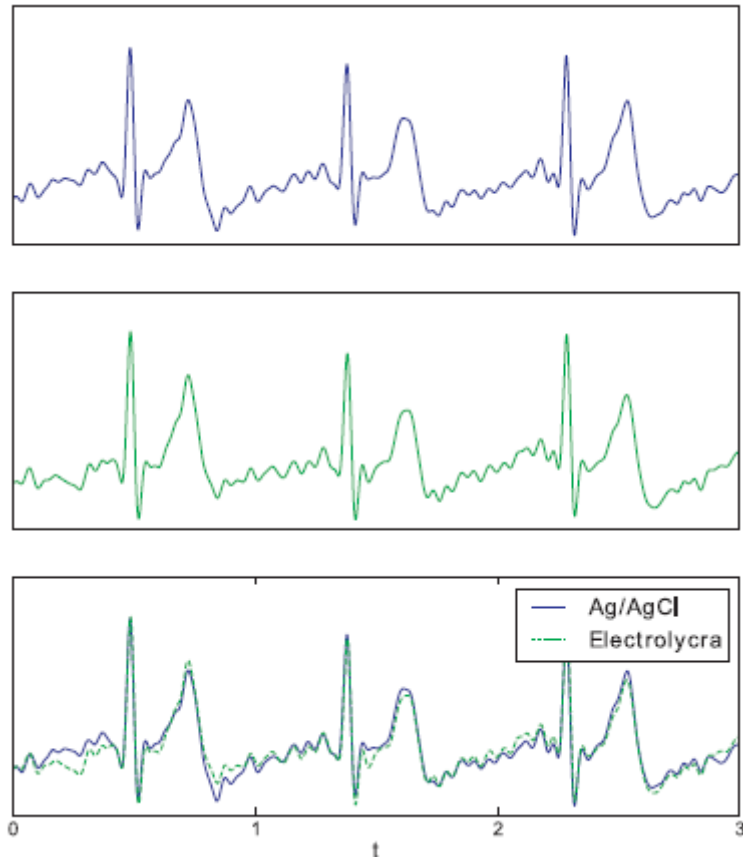


Figure 6 - ECG recorded at hand palms and fingers (Adapted from Silva et al., 2013).

Due to this level of detail, one can notice both the QRS complex and the T and P waves (Figure 7). This led the authors to the conclusion that dry Ag/AgCl electrodes and electrolycra may be used as an interface with the skin in a biometric system.

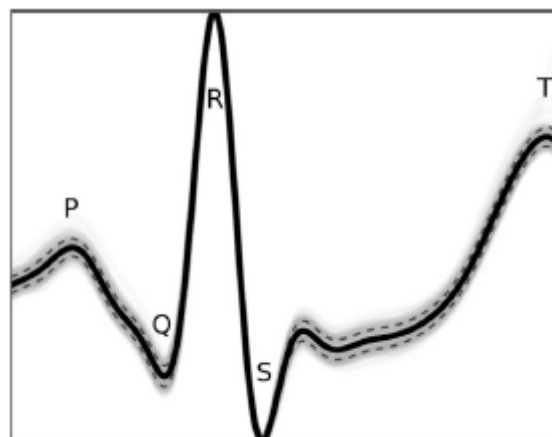


Figure 7 - Example of a segmented and filtered ECG taken from the hand palms. (Adapted from Silva et al., 2013).

2.3.3 Smartphones as Security Tokens

In order to improve security, one can consider using multi-factor authentication. This term refers to the use of more than one authentication form in an authentication system. As an example, many tokens require a password to generate private keys used to encrypt communication (chip and PIN authentication) (Zheng 2011). Such method, though, requires explicit action from the user.

Tanvi et al. (2011) suggested a solution based on a mobile phone. They referred hardware cost as a relevant drawback of the traditional approach to security tokens (which includes individual tokens and authentication server costs) and is proposed the use of phones as an alternative.

Therefore, Tanvi et al. presented a scheme consisting of: user, referring to the individual trying to gain access to the application; dynamic password generator, for each identification process; SMS server, responsible for sending dynamic passwords to the user; “Visitor password Register” (VPR), temporarily registering session’s username and password and password.

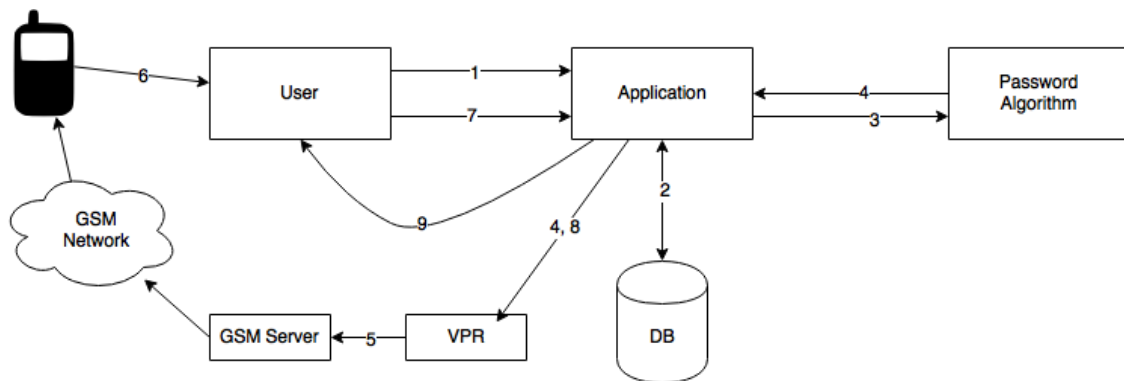


Figure 8 - Authentication scheme (based on Tanvi et al., 2011).

Figure 8 illustrates the token based solution suggested by Tanvi et al. The action starts when the user sends a request and its identity to the application (1); then, the application confirms user credentials in database; thereafter, the application requests a 8 digit dynamic password based on seed value, timestamp, date and static password (3); the

password is stored (4); the dynamic password is sent to the GSM server (5); user requests the dynamic password (6); identification request is sent, with username and dynamic password (7); credentials are verified (8); access granted if authentication is successful (9). The dynamic password has a short lifetime of 60 seconds.

Also, these authors suggest using the IMSI (International Mobile Subscriber Identity) number to strengthen the authentication. This number is a unique identification associated with the SIM card and is sent by the phone to the mobile network. Thus, they claim this hardware identification actually improves the authentication process security. However, there are multiple tools for SIM card cloning available to the general public on the internet nowadays, thus ruling out this hypothesis. Also, this solution would require SMS exchanging, resulting in additional costs and time consuming action.

Additionally, the authors refer to extending this approach to wireless connections, namely Bluetooth. Such approach was taken with the “Laptop-user Authentication Based Mobile phone” (LABM) solution (Abdelhameed et al., 2005), suggesting a java solution for laptop user authentication focused on proximity-based login. This work consists of a scheme for authenticating an individual by approaching the computer with a Bluetooth-enabled phone. This device acts as an authentication token, by continuously communicating with the laptop through a wireless link.

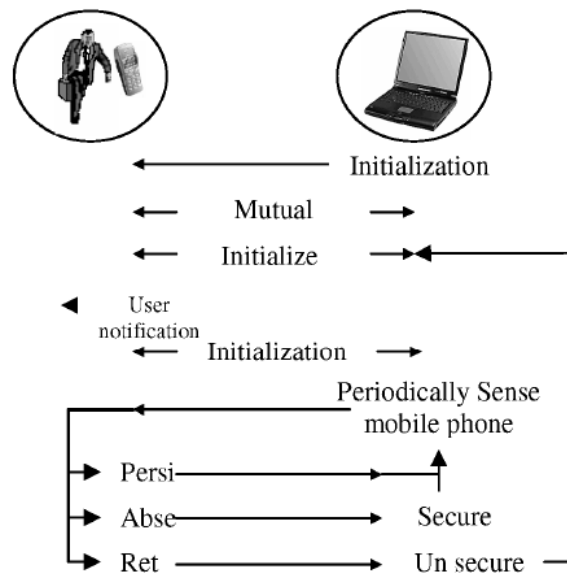


Figure 9 - Authentication scheme (adapted from Abdelhameed et al., 2005).

The system is composed of three main components (see Figure 9):

1. The communication link between a phone and a laptop, in which a public key based authentication of both parts is used, a user notification of connection establishment, a session key creation and a sequence of disconnection/reconnection depending on the presence or absence of the phone in the laptop's range;
2. The authentication system, a persistent authentication of the user-phone pair that will work until explicitly revoked;
3. The communication module, based on UDP transfer of encrypted messages through a Bluetooth connection between laptop and phone.

A user is considered absent – thus, triggering the laptop's lock – when three failed connection attempts occur. This translates in about 2 to 2.4 seconds. User reconnection times varied from 1.3 to 1.1 seconds.

Although the authors do not specify exactly when the credentials are revoked, this permanent authentication approach might result in loss of security in a multi-factor authentication context – by gaining access to one of the security factors (the mobile phone), an attacker would achieve a single-factor authentication situation.

An alternative approach application was presented by Lee et al. (2012) for computer authentication using a smartphone as token. Their solution includes audio challenges between both devices, justified by their claims that many desktop PCs do not have hardware for the most common researched means of tokens, such as WIFI, Bluetooth or RFID.

The method consists of transmitting the sound from the speakers of the computer to the microphone in the smartphone, thus requiring no action from the user. It uses high frequencies (15800Hz to 20000Hz) resulting in a hardly hearable sound to avoid disturbing the user and distorting the signal with ambient low frequency interference.

In order to allow accessing multiple systems with the same phone, the authors suggest a solution with three components: the smartphone representing the individual; the target system of which user wants to gain access; an authentication server, with which the phone shares a secret key and can access every system in the secured network.

However, this solution presents a drawback if the target system is not a regular use computer and does not have a microphone.

In a commercial approach, HealthySystems (“HealthySystems”, 2015) has already developed a system based on NFC (Near Field Communication) and smart cards. This solution, RFIDSmartAuth, enables access to a certain web application when an NFC enabled card is near the reader, and locks again when it is taken away. This scheme is similar to what we propose – a lock system based on a proximity token – although it does not combine with biometrics.

2.4. Custom Authentication Experience in Windows Environment

Windows 7 and later use Credential Providers (CP) to manage credential input. This architecture replaces the old GINA (Graphical Authentication and Identification) present in previous versions. These CPs result in alternative logon tiles that may or may not refer to the same user account, meaning that one user may have several distinct logon methods through a number of different logon tiles. The new approach taken by Microsoft, however, won’t allow a full replacement of logon UI (“Credentials Management in Windows Authentication”, 2013).

Tiles are processed by Logon UI. It requests every credential type provided by each CP and displays them. One CP may have more than one associated tile, and can specify one of them as default.

Credential data is processed, by default, through a service called WinLogon. This process includes confirming logon data with the Security Accounts Manager (SAM) database when referring to a local account, or with Active Directory if the computer is part of a domain. Credentials may be acquired by user input or by API.

Credential Providers can, therefore, extend Windows capabilities by allowing alternative authentication processes.

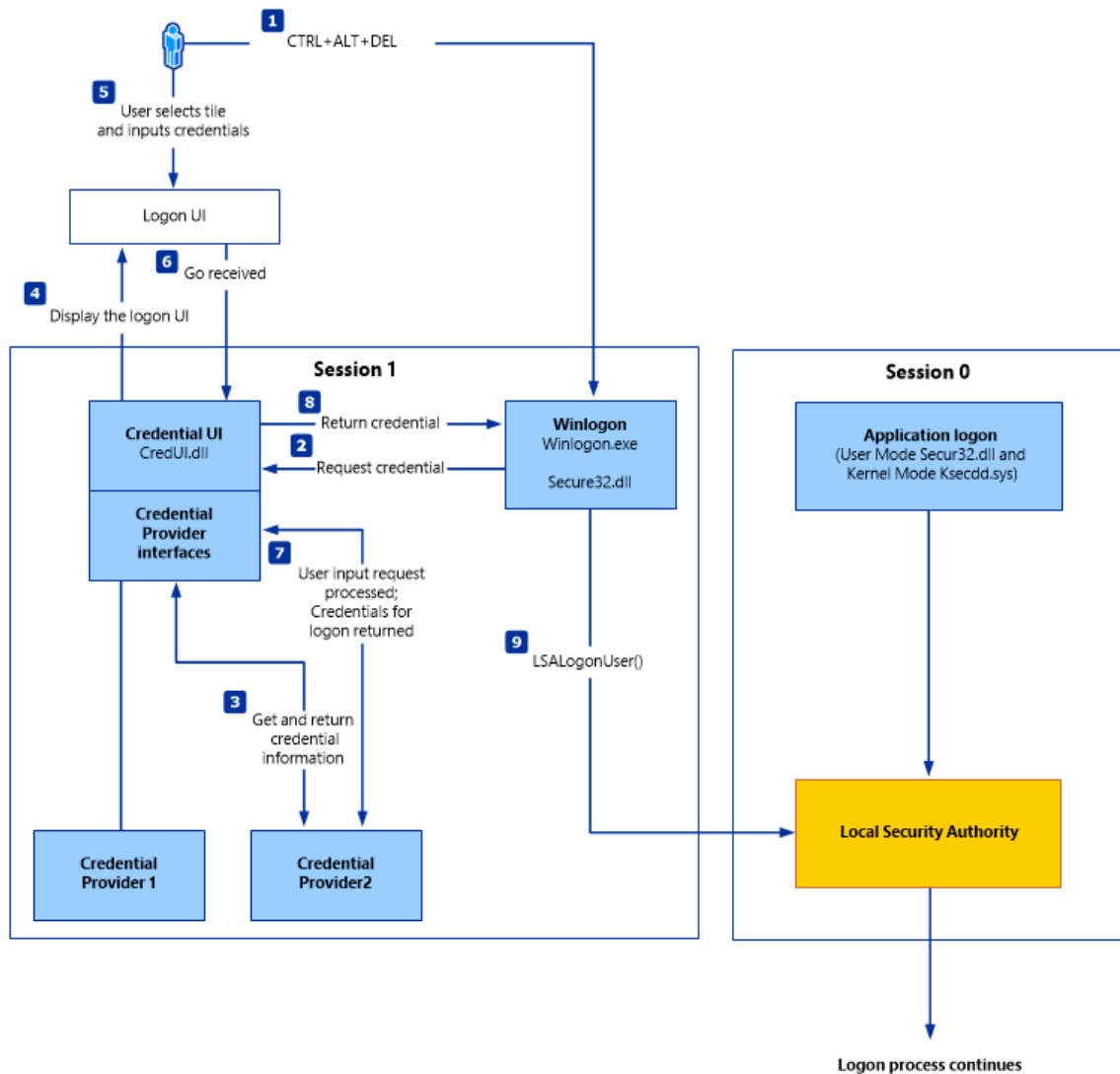


Figure 10 - Credential process for Windows 7 and later (Adapted from *Credentials Management in Windows Authentication*, 2013).

Figure 10 illustrates logon process in Windows. *Session 0* is where critical processes are executed; *Local Security Authority* is a system process responsible for authenticating and logging users on to the computer (using SAM), as well as maintaining information about security policies.

Microsoft allows Credential Provider development through Windows SDK. This development is C++ based and they share buildable examples (Griffin, 2007).

2.4.1. pGina

Windows authentication system can be managed through pGina. This tool replaces the default credential provider (CP) used by Windows Vista and later (XP and above used the GINA framework instead). It is a plugin-ready open source application that allows an end user to manage his own way of logging into Windows (see Figure 11).

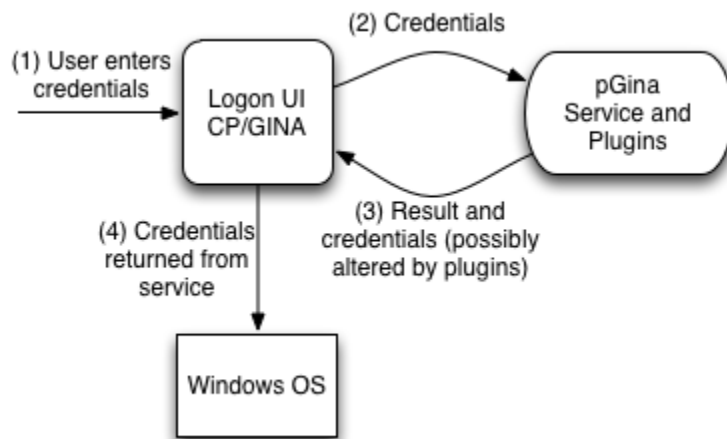


Figure 11 - Diagram showing the main pGina systems interacting in a common logon (Adapted from Wolff, 2014).

A typical logon involves the following process: (1) user providing credentials to pGina using its GUI or any other credential provider; (2) pGina's CP sends credentials to pGina service; (3) pGina service sends the logon attempt result (Boolean answer along with the credentials) to pGina's CP after being processed by the installed plugins; (4) if the logon attempt returns true, pGina sends credentials to Windows and the latter performs logon.

A full logon— since the creation of a new user to the actual logon process — can be described as follows: (1) a new user is created through the usual Windows GUI mechanism; (2) the associated credentials (username and password) and stored or retrievable somewhere; (3) the pGina plugin accesses the credentials, does its own processing (which was programmed by its writer), and sends the credentials to Windows in a logon attempt; (4) logon may succeed or fail, and pGina and its plugins

may do their clean-up or any other processing. The creation of a new user, however, can be replaced by a user-created plugin as well.

In this scenario, user authentication is in fact managed by the running plugins within pGina service. They are responsible for approving authentication and authorization, as well as doing their own clean-up and other custom actions associated with the process.

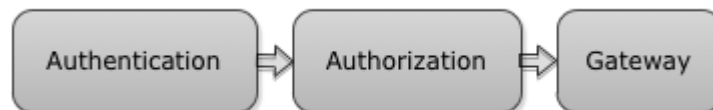


Figure 12 - Pipeline representing the three stages of authentication process (Adapted from Wolff, 2014).

These three stages of the process (Figure 12) may involve some or no plugins at all, and one plugin may provide services to one or all the stages. In the authentication phase, plugins may want to test whether the user is in fact who he says he is. This can be done by using any external database or other source and requires at least one plugin reporting success to proceed. The authorization phase refers to giving the user the permission to access certain resources or not. The gateway implements other actions related to the login process which may have direct interference with the overall success of the process. For example, it may be responsible for creating an environment in which the user is supposed to enter, and a failure here would result in a failure of the whole process, even if user is authenticated and authorized. In both authorization and gateway stages all plugins should register success in order to continue.

Along with the execution of the three stages, pGina keeps record of user-name, password, group membership and other data about the user. All this may be altered during any of the stages.

Additionally, plugins can receive notifications from Windows Terminal Services about events including logon, logoff, lock and unlock.

2.5. Intel Edison

Intel Edison consists of a small board (35.5x25.0x3.9 mm) capable of running a Linux environment (“Intel® Edison—One Tiny Platform, Endless Possibility”, 2015).



Figure 13 - Intel Edison. Adapted from Intel® Edison—One Tiny Platform, Endless Possibility, 2015.

In terms of hardware, its main characteristics are a dual-core CPU, as well as Wi-Fi and Bluetooth Low Energy adapters. Also, it features 1GB LPDDR3 POP memory and a 4GB eMMC. Regarding power consumption, it requires 13mW in standby and no wireless connections; 21.5mW with BL; 35mW with Wi-Fi.

This board opens a wide range of opportunities related with the Internet of Things. In this dissertation we develop a custom authentication system in Windows; however, Edison opens space for a number of alternatives. The dual-core CPU provides significant processing power, and the Wi-Fi and Bluetooth Low Energy adapters provide low power consuming connectivity as well as recent and growing technology. For example, one could develop an authentication system in a car powered by the board; we could implement a smart house door capable of adding safety, for example recognizing the person’s ECG while he touches the door handle; we could develop our own on-board computer on a car or a bicycle; in a more informal approach, one could even create his own small gaming console.

Edison was already used in several different applications. For example, the Smart Spider Dress is a dress powered by Edison that lights a set of LEDs depending on the

user's read biometric values, in order to reveal emotions ("Is That a Spider on Your Dress or Are You Happy to See Me?", 2015). Another application was Jimmy, a humanoid prototype based on Edison. It is an 18 inches tall robot, 3D printable ("My Robot is Cuter than Your Robot", 2015). Their creators, Trossen Robotics, intend to make it work in a similar way to smartphones – install an app, and make it capable of doing more activities. Also, a face recognition system powered by Edison was developed by an Intel demo team at CES 2015, proving its processing power and versatile capabilities ("Build Your Own Face-Recognition System with Intel Edison", 2015).

The Edison we used in this dissertation was provided by André Lourenço, therefore the opportunity of exploring this platform became possible.

2.6. Conclusion

The end-user authentication process can take great advantage of the rapid improvement in technology, namely that referring to smartphones. Being mobile phones one of the typical personal items of the average person, one can look at it as an identifier of its owner. Therefore, functionalities may be added to improve security associated to ordinary daily activities, such as logging on a personal computer. Authors proved to be possible to create an authentication with a mobile phone acting as an intermediary using unique identifiers such as IMSI or a third identifier capable of managing the security associated with both sides; also, they implemented their solutions in a multi-factor authentication scheme.

On the other hand, biometrics are also getting more accessible with user-friendly approaches, namely ECG-based authentication. Research shows that it is, in fact, a reliable authentication method and possible to implement in easy-to-use contexts.

Consequently, we propose associating both ECG and smartphone as token authentication solutions in a multi-factor security scheme. As a result, security is improved by requiring *something you have* and *something you are*.

As a conclusion, the communication between the computer and the mobile phone acting as a security token may be implemented using the recent Bluetooth Low Energy

technology, diminishing the impact on battery life – especially relevant in the smartphone side. This type of approach was not previously found in the related literature.

3. Implementation

3.1. Introduction

As we showed, the implementation of a security context using a smartphone as token is possible. Therefore, in this dissertation, we intended to implement a scheme where a Windows Phone would send its owner identification through Bluetooth Low Energy when within a certain range of a computer. This computer would advertise a GATT server, which would be accessed and wrote into by the phone, and would have an ECG keyboard plugged in.

We developed several different implementations along the process of refining the solution, as seen in Figure 14. We started with a version using Bluetooth, in order to easily build a prototype. In this context, we explored using Windows apps, capable of running on WP, Windows Tablet and Windows Desktop, as well as different APIs; afterwards, we adapted it for Bluetooth Low Energy, resorting to Android and Intel Edison due to the previously acknowledged limitations in Windows (explained in the following pages). Final versions are Scenario 6 in Bluetooth, and Scenario 3 in Bluetooth Low Energy.

Implementations

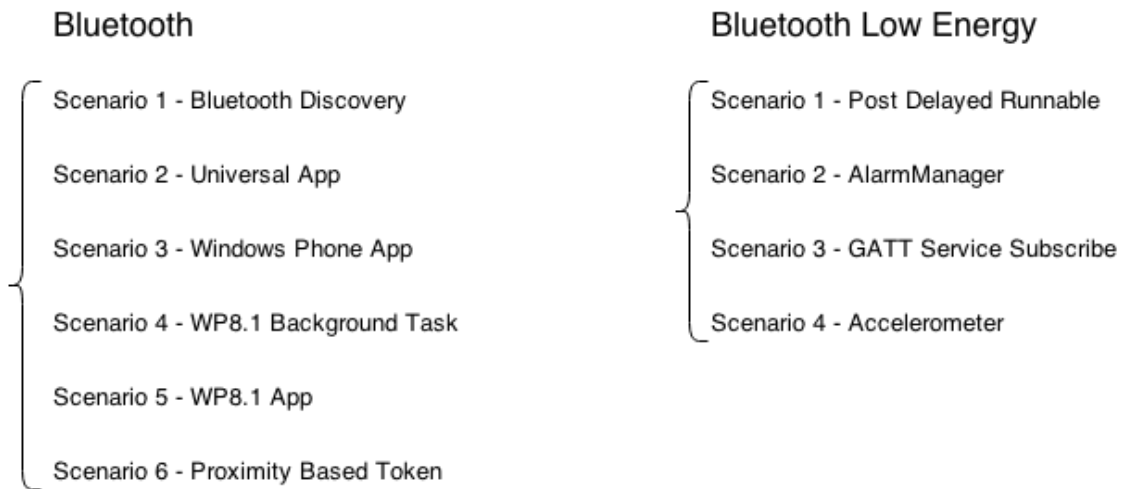


Figure 14 - Set of different implementations we worked in.

However, implementing a GATT advertisement in Windows has proven to be a limitation, as it only provides APIs to act as a GATT client (“GATT server role on Windows 8”, 2013). Given this fact, we decided to implement a similar scheme using classic Bluetooth. Afterwards, we also implemented a solution using BLE, with a Linux system acting as a connection enabler between a BLE device and a Windows computer.

3.2. Classic Bluetooth communication

3.2.1. First Scenario – Bluetooth Discovery

Initially, we created an example plugin for pGina. This plugin was written in C# and intended to enable a Bluetooth interaction resulting in a successful logon. We used a laptop with a Bluetooth 4.0 Dual Mode dongle and a Windows tablet with Bluetooth 4.0. This implementation was based on the 32feet.net Bluetooth (“32feet.NET - User's Guide / Tutorial / Examples”, 2012).

In this preliminary version, we implemented the authentication stage of pGina only, starting with a Bluetooth interaction. We experienced problems - also reported by other users with no known solutions by the moment of writing the thesis (“RFCOMM connection fails, Unhandled exception, [Bluetooth RFCOMM] SocketStream.ConnectAsync(HostName,String) – SystemException Element Not Found, RFCOMM connection works unstable”, 2015) - implementing a Bluetooth connection between 32feet and Windows Phone API, so we were constrained to use Bluetooth scan to get nearby MAC addresses, even without establishing a connection. In this version we verify user identity through the MAC address associated to the users’ mobile device. This verification only discovers devices, and has no other interaction. Other unique identifiers were used previously, such as IMSI (International Mobile Subscriber Identity), consisting of the ID associated to a SIM card (Tanvi et al., 2011).

It is relevant to mention that, although the security of such solution is low, this solution is fast as requires no specific operating system on the mobile phone. In fact, any discoverable Bluetooth device would work, and required no previous pairing. However, limitations would occur if we wanted to add additional functionality requiring data exchange with the mobile device, as well as when we think about the possibility of simultaneous presence of multiple logon-allowed mobile phones – which one would be prioritized?

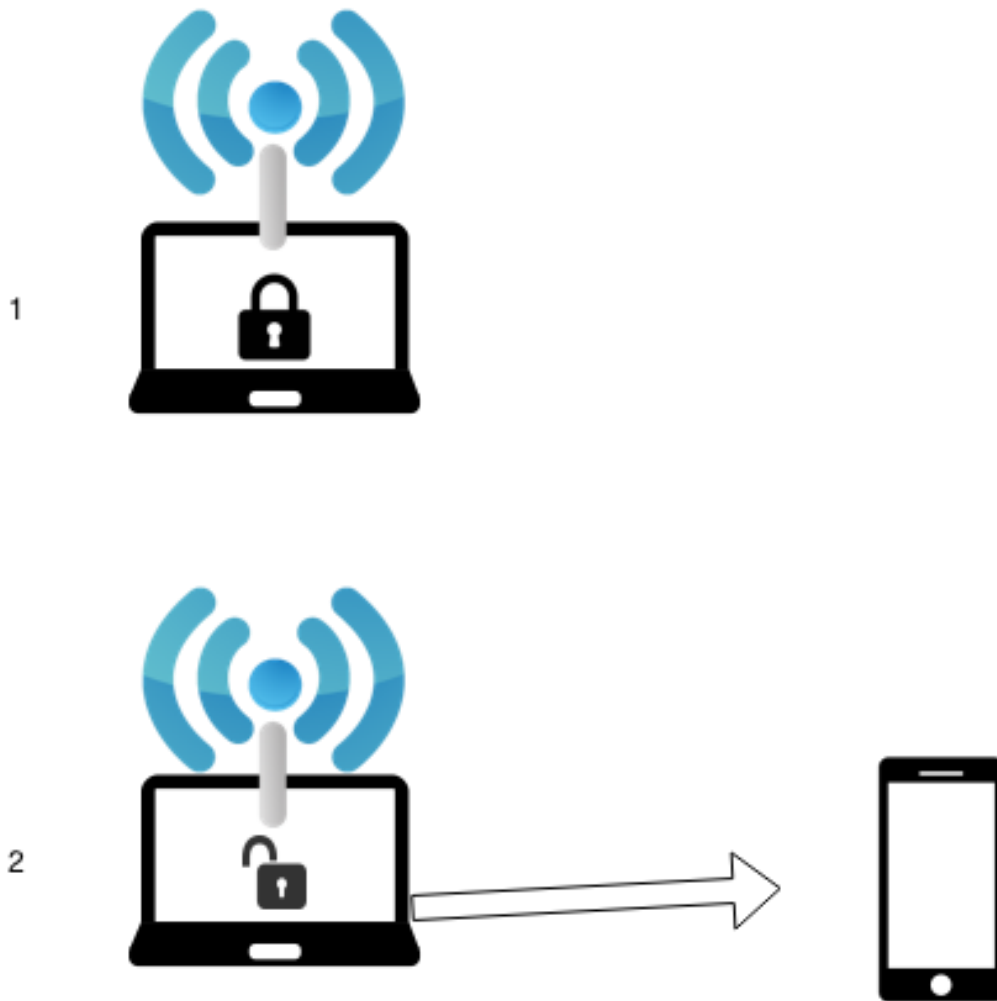


Figure 15 - First scenario detection scheme.

Initially, we implemented a solution that would trigger Bluetooth discovery when the user attempted login through regular Windows GUI (Figure 15). This attempt was done by clicking a button on logon interface (see Figure 16), with both username and password fields hidden (configured in pGina). pGina did not offer any solution to completely automate this process at the time, i.e., skip the button pressing.

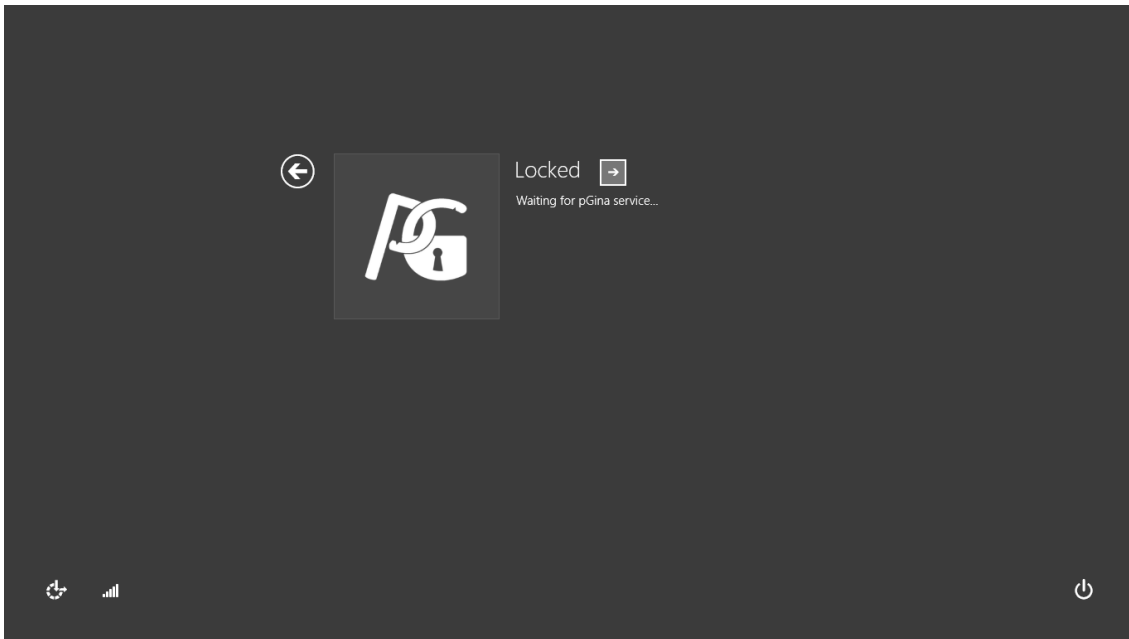


Figure 16 - Logon button.

This proved to slow down the process when comparing to the immediate response of a typical logon attempt, requiring around 10s to return the result in an environment containing three discoverable devices. A significantly higher number of devices could expand this time frame (Chakraborty et al., 2008). To attenuate this drawback, we tried a solution based on a Boolean variable stating if the mobile phone is present that is constantly updated through a loop discovery. This addition solved the problem concerning the delay of user interaction on a logon process, but the present/absent variable still took around 10s to be consistent with actual state – meaning the device could leave the room and someone could still log on for a few seconds, but any logon attempt would have instant feedback.

3.2.2. Second Scenario – Universal App

Afterwards, we started implementing a server-client solution between a computer and a Windows universal app (written in C# for Windows 8 and C# for Windows Desktop). This type of app is capable of easily be adapted between Windows Phone (WP) and Windows apps (“Building universal Windows apps for all Windows devices”, 2015).

This choice would allow people to run the app whether they use WP, a Windows 8 (or later) computer or a Windows 8 (or later) tablet (Figure 17).



Figure 17 - Universal app scheme.

To test this scenario, we used a Windows 8.1 tablet as client app, trying to connect with the server application in a Windows desktop. The result was not favorable – the class used to make a Bluetooth connection turned out to have distinct behavior when run in a tablet from what it would have when in a WP. In fact, our tests shown that if both the devices where in the same Wi-Fi network, they would use this technology instead of Bluetooth. Similar behavior is reported by other developers (“Proximity Sample”, 2015) Microsoft employee Vishal Mhatre explains “*FindAllPeersAsync (or PeerWatcher) on Windows uses Wi-Fi Direct to find nearby peers. On Phone, it uses Bluetooth. Cross-platform discovery of peers is currently not supported.*”. Given this situation, we gave up developing universal apps and focused on WP targeted apps.

3.2.3.Third Scenario – Windows Phone App

Developing for WP required using a physical phone, since the emulator offered in Visual Studio for Windows did not offer the possibility of using Bluetooth. Therefore, we started developing a WP8.1 app for Lumia 530. This new scenario was composed of a WP8.1 app connecting to a Windows desktop application server using 32feet.NET. The connection was established using classic Bluetooth instead of Bluetooth Low Energy (BLE) due to the lack of APIs to perform discovery and pairing with BLE in Windows environment (“Windows 8, Bluetooth LE and

BluetoothFindFirstDevice/BluetoothFindNextDevice/BluetoothFindDeviceClose”, 2015). This time we achieved the goal of connecting both devices and communicating strings between each other, allowing us to build a prototype of Bluetooth-based login.

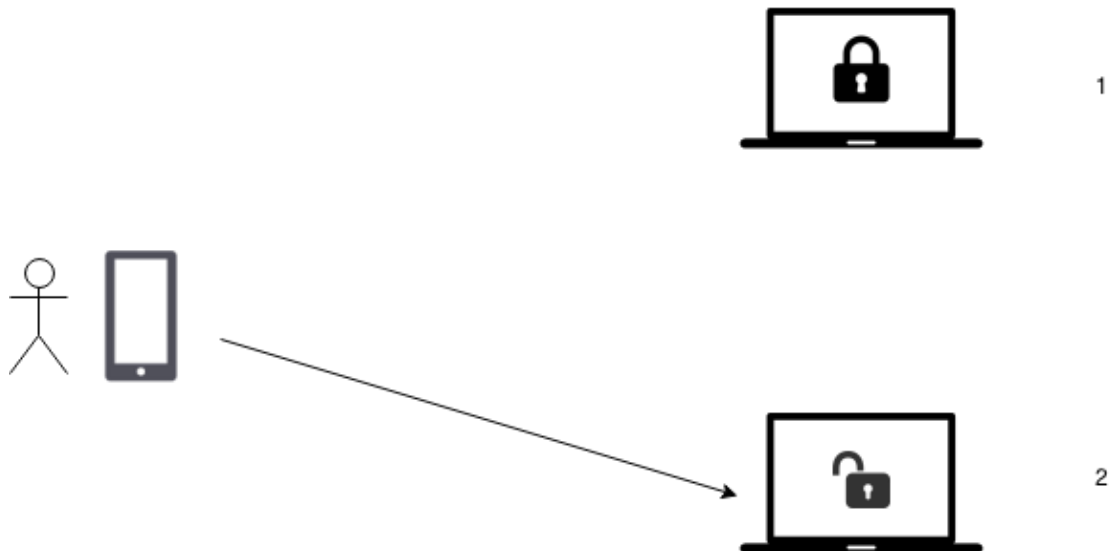


Figure 18 - Windows phone app scheme.

This prototype (Figure 18) consisted of a WP app that, by pressing a button, would send a string message (unencrypted) to the desktop server. The server side –which was a pGina plugin also managing a BL connection – would then release the hardcoded user credentials to Windows when asked for This prototype was significantly faster than the user detection implemented in the the first scenario, when using logon-triggered BL scan (which was the most reliable in terms of false negatives/false positives). However, this solution requires a software-specific device (a WP instead of any Bluetooth device) and user interaction. Also, the connection was not reliable – it would randomly connect, or throw an exception. Similar problems were reported by other developers on Microsoft Developers Network (MSDN) forums (“RFCOMM connection fails, Unhandled exception, [Bluetooth RFComm] SocketStream.ConnectAsync(HostName,String) – SystemException Element Not Found, RFCOMM connection works unstable”, 2015).

3.2.4. Fourth Scenario – Windows Phone 8.1 Background Task

In order to avoid requiring user interaction with mobile phone to perform login, we tried to execute the connection as a background task (Figure 19).

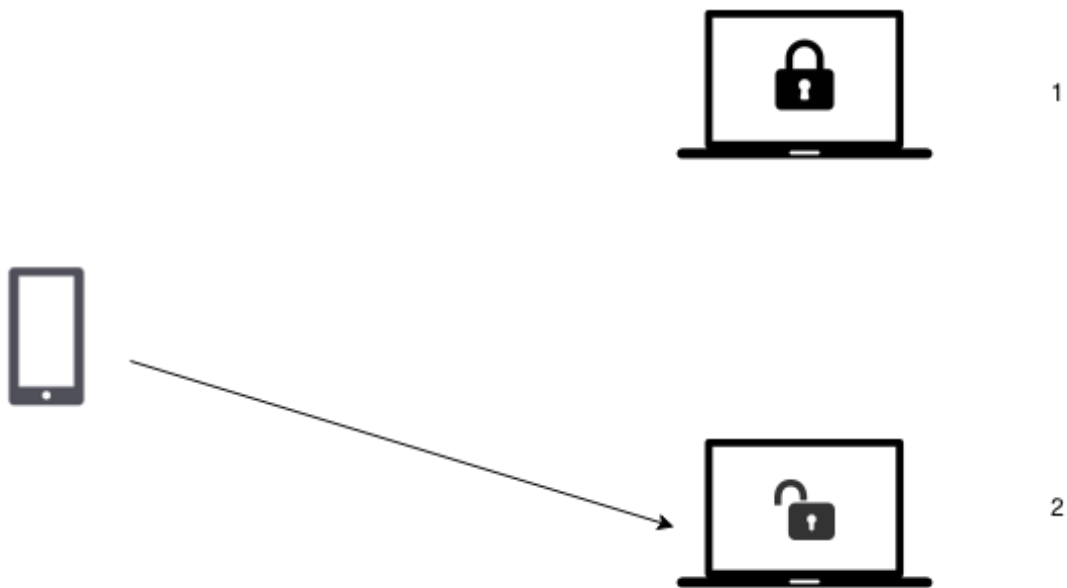


Figure 19 - Background task scheme. No need for human interaction with WP.

Additionally, while trying to solve the unstable connection, we concluded that the exact same code would work every time when compiled to WP8.0, but unstable when compiled to WP8.1, as other developers reported (“RFCOMM connection fails”, 2015). The use of an external Bluetooth trigger (for example, the presence of a Bluetooth server nearby) in Windows Phone was, however, only available in 8.1 version at the time. This conflict forced us to choose connection stability over functionality. Therefore, the use of a background task was not considered further and the need for user interaction was maintained.

3.2.5. Fifth Scenario – Windows Phone 8.0 App

Given these constraints, we backtracked and picked up the development of an interaction based app only now, compiling to WP8.0. This scenario was an obvious improvement comparing to the third scenario, as it would always connect with no errors. We also added a small database to store user data, created in SQL Server 2014.

Having a more stable connection to work with, we started improving the prototype. We created a better structured plugin, allowing the app to send both the username and password to be used in the logon process, as well as being able to use logon multiple times with the server always running.

In the following diagrams, we introduce the ECG keyboard. This hardware, as well as the corresponding software that allows interaction with the keyboard were provided by André Lourenço. The interaction between the pGina plugin and this system is made using its C# APIs, vitalidiAPI (VitalidiType, 2015).

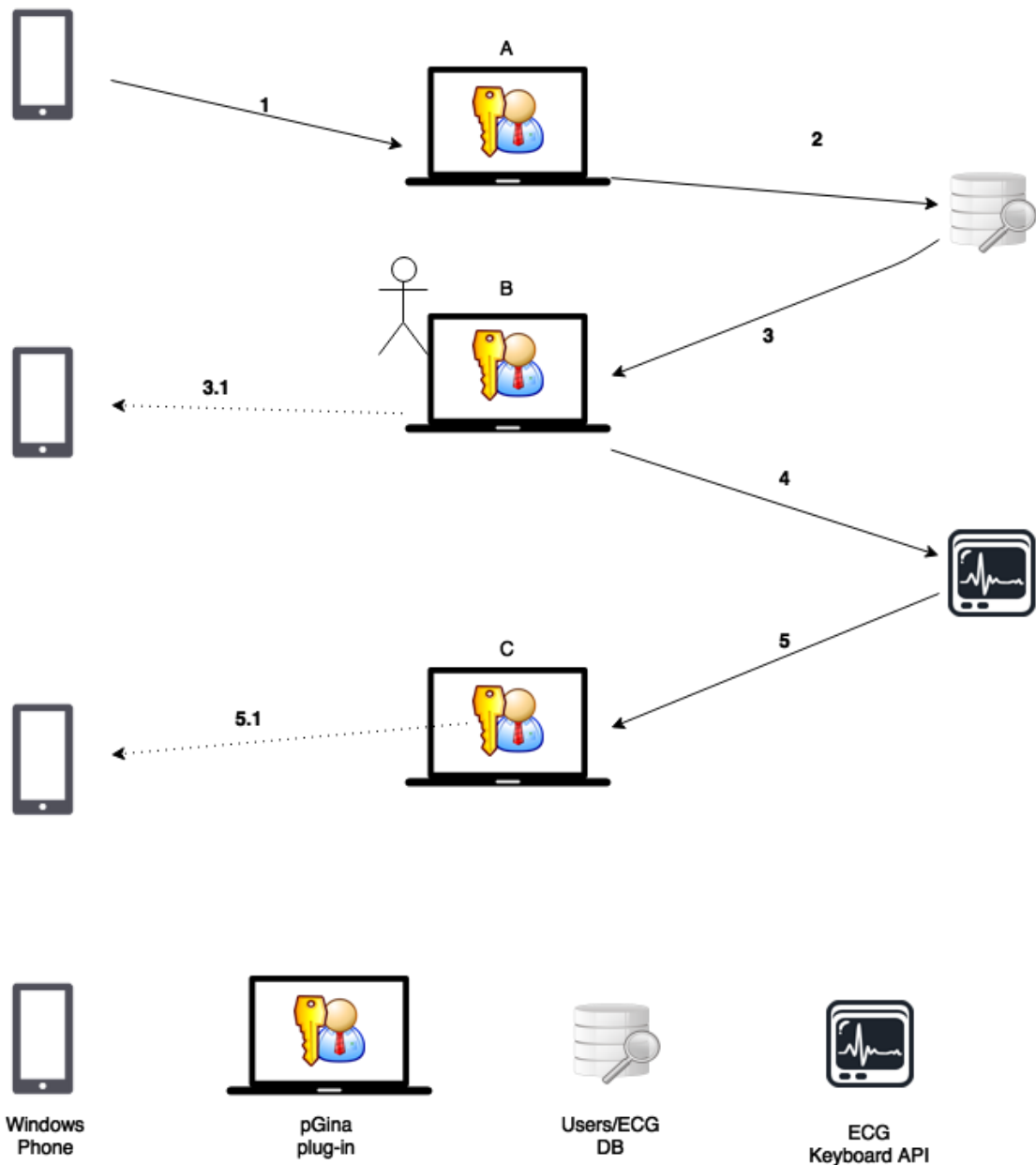


Figure 20 - Communication scheme (1) process is started by interacting with WP app; 2)pGina queries the database for credentials validation; 3) query response; 4) ECG is read; 5) logon is successful/unsuccessful).

The above diagram (Figure 20) refers to the communication scheme in which this scenario was based. The main actors are: Windows Phone, the pGina plugin working as BL connection point, a database storing credentials information and the ECG authentication accessed using VitalidiAPI. In 1, the WP app would send a message to our server containing, at least, the username; in the computer, on stage A, pGina confirms the username by querying the database (2) and retrieves the password if the username exists (this approach may vary, as we will see next); a feedback might be

implemented, although we considered it would unnecessarily increase the process complexity (3.1 and 5.1); action is suspended before 4 while waiting for Windows logon button press; afterwards, if ECG authentication is successful (5), pGina allows logon and, optionally, may return feedback to the WP.

We also considered sending Bluetooth MAC address from WP to pGina, in order to confirm the identity of the phone itself. Still, it is relevant to mention the fact that WP will only allow BL communication in-app with previously paired devices – meaning no unpaired device could connect with the server. These details were discussed based on the three hypotheses shown in Figures 21, 22 and 23.

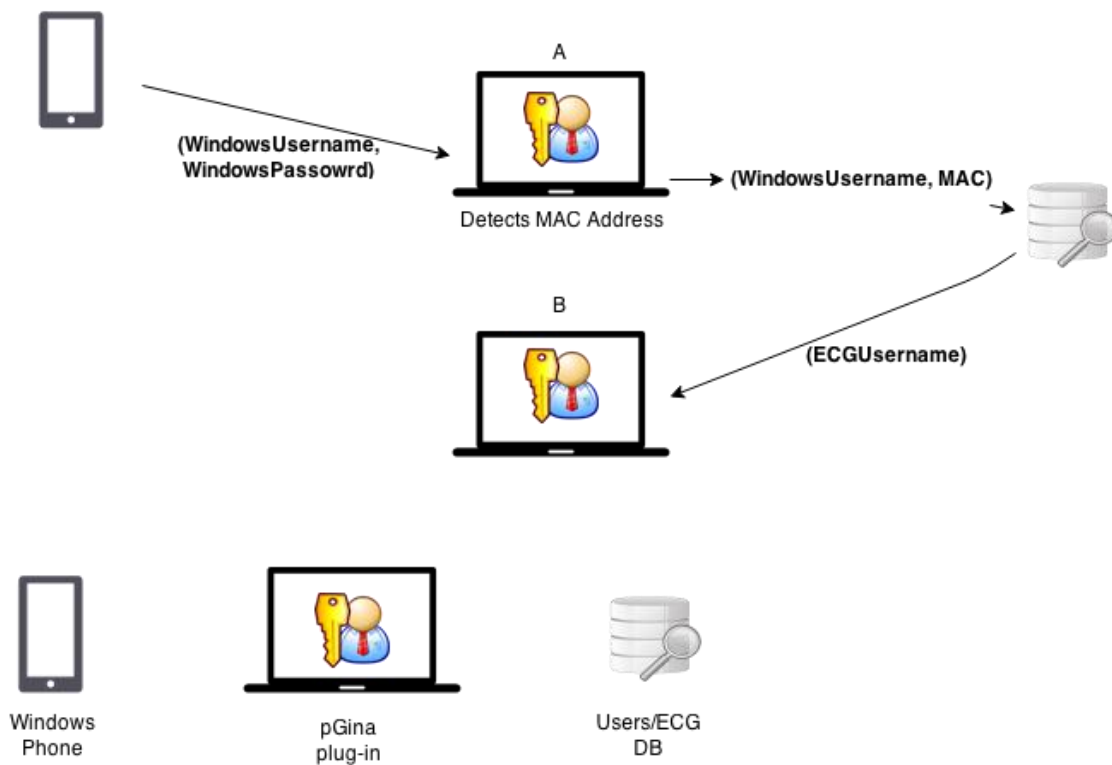


Figure 21 - Hypothesis nr.1.WP has the initiative. Windows password is stored in the mobile phone.

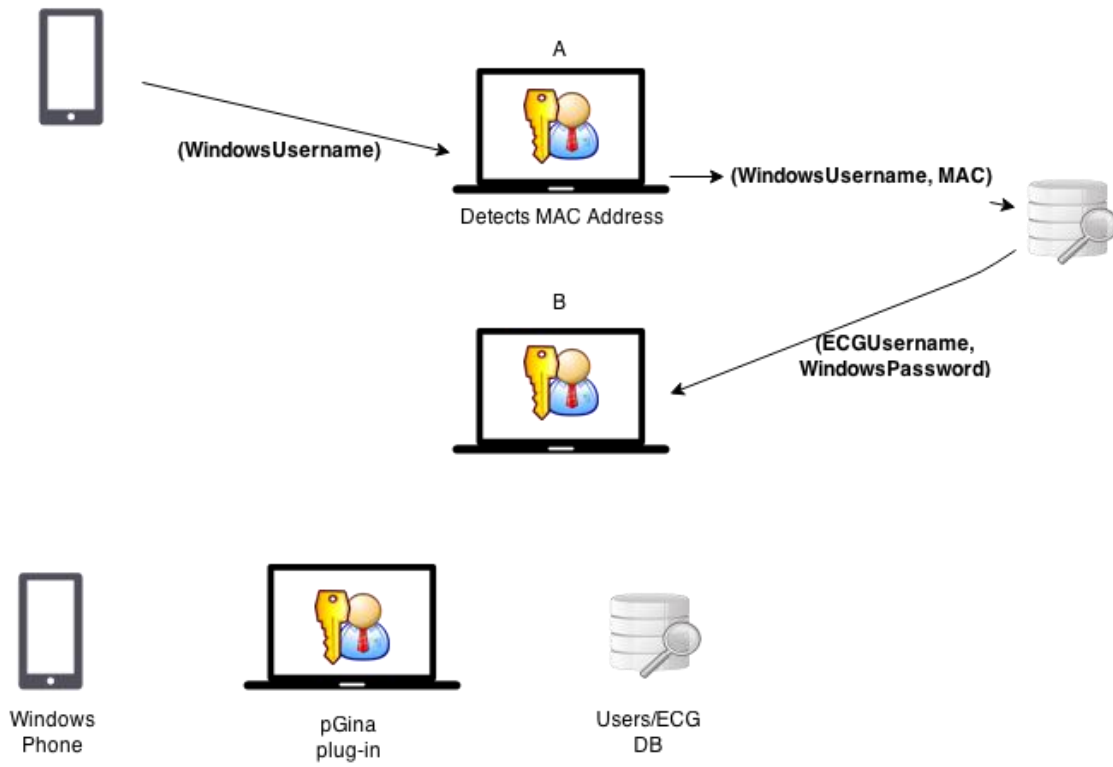


Figure 22 – Hypothesis nr.2. WP has the initiative. Windows password is stored in the computer.

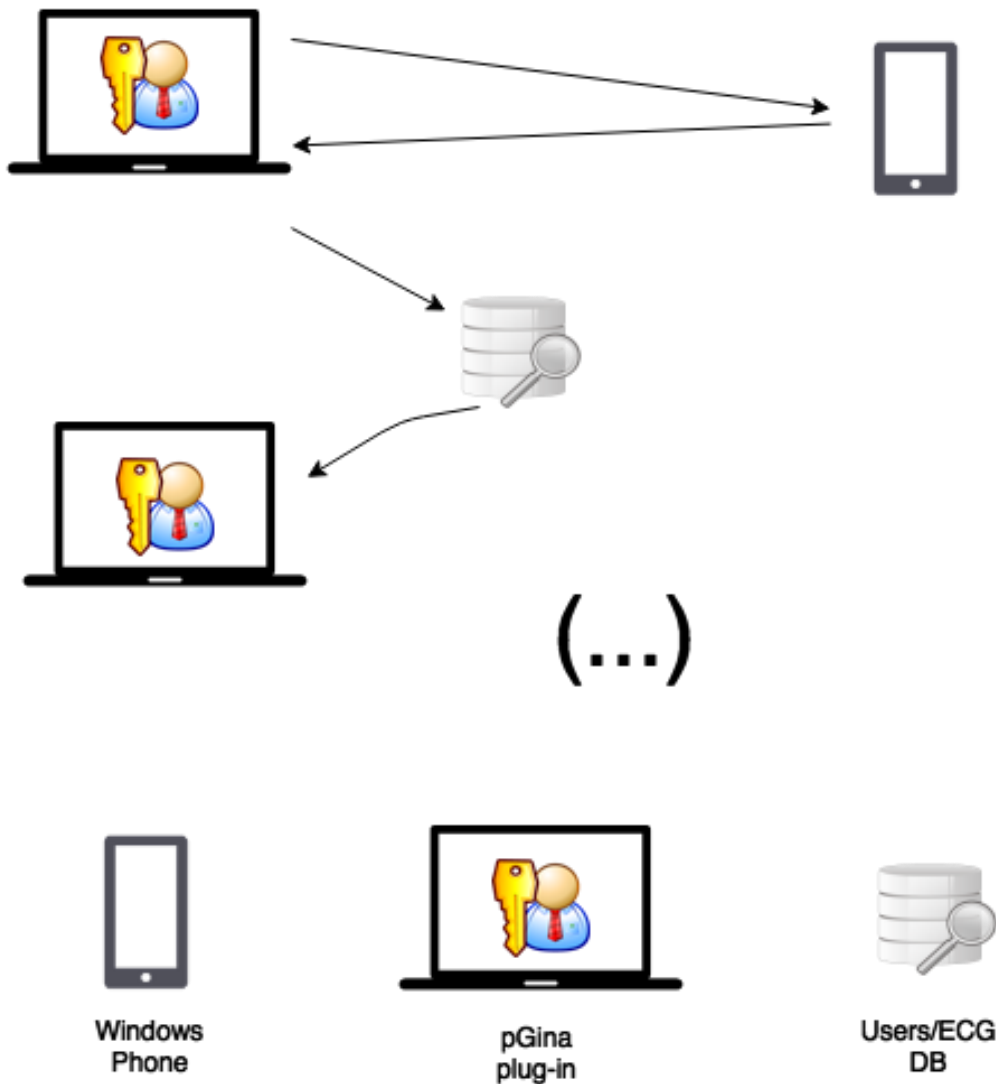


Figure 23 - Hypothesis nr.3. Computer has the initiative. The missing part is irrelevant - either of the previous message contents would apply.

3.2.6. Sixth Scenario – Proximity-based token

We chose hypothesis nr.2 due to two main reasons: (1) the fact that it was mandatory to request for an explicit user interaction - this is a WP limitation that would require constant phone wake state (“Automatic Bluetooth connection to a certain device”, 2015) - ruled out option nr. 3, where the initiative is on the server side; (2) in order to avoid sending full credentials (user and password) through Bluetooth we decided to store passwords on the server side, and then identify the smartphone based on username received and associated Bluetooth address, both previously stored as well.

Furthermore, we decided to implement an additional feature with the goal of avoiding requiring too much user interaction. This solution worked by receiving a one-time logon message from WP, which the server would register with a timestamp and use in future logons based on proximity. The following scheme explains the process (Figure 24).

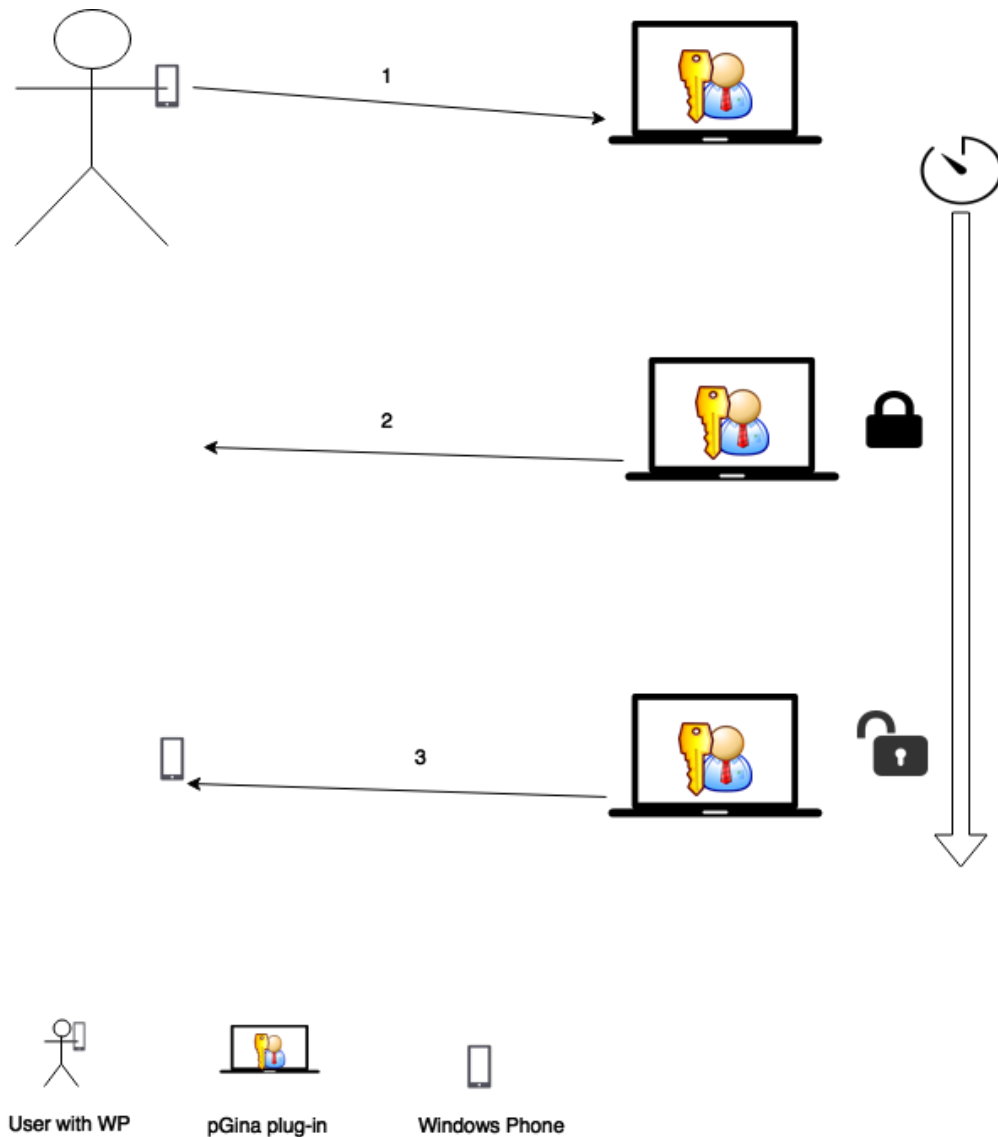


Figure 24 - Proximity scheme. 1) a logon message is sent; 2) the phone is not detected by the computer, logon is denied; 3) the phone is detected, a logon with the associated credentials is allowed.

In (1), user is approaching the computer and sending the logon message via Bluetooth. Then, the server recognizes the phone as one authorized to logon in the following hours (or any other predefined time range). So, in (2) when no phone can be found nearby,

logon is denied; in (3) when the phone is again present, a logon with the credentials associated to that user is allowed. This means that, starting from (1), the computer itself will search for the presence of the authorized phone; the search will not require establishing a connection.

In fact, this search is different from the one stated in the first scenario. Previously, we scanned for all devices nearby and then verified the identity of each one in search of a particular phone; with this new approach, we tested the presence of a specific device using a pre-stored address. In terms of time required, a loop testing for a list of one phone changed logon status in less than 2 seconds after turning off Bluetooth in WP.

3.2.7.ECG Keyboard authentication

After implementing the Bluetooth side, we approached the introduction of the electrocardiogram-reader keyboard on the existing scheme (Figure 25).

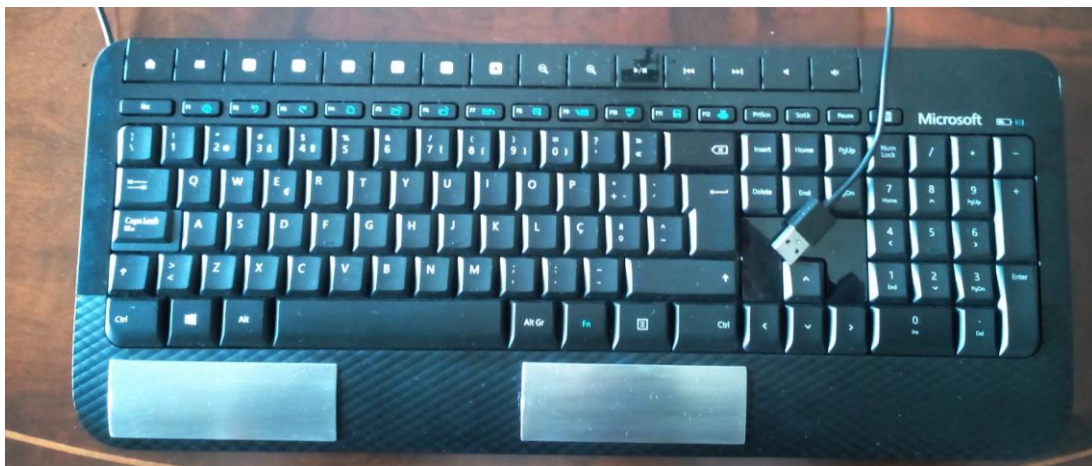


Figure 25 - A photo of the actual used keyboard. ECG is acquired through the metal bars.

To better understand the advantage of using a smartphone combined with an ECG authentication, one must explain the latter. This method works by first identifying an individual's heart beat and then authenticating the person. Both processes can take several seconds, thus leading to a delay in the logon process. Also, the authentication itself can be based on different confidence levels. These two characteristics bring a

relevant improvement related to the use of a smartphone as token – the computer is able to receive the identification of the individual beforehand, skipping the identification process and also being able to use a better authentication confidence level.

The keyboard does its own autonomous processing. It requires installing VitalidiType, the software that allows interaction with the keyboard. The interaction between the pGina plugin and this system is made using its C# APIs, vitalidiAPI (VitalidiType, 2015).

The ECG reader is the final step in our entire logon scheme, taking advantage of the previously sent information from the individual’s mobile phone.

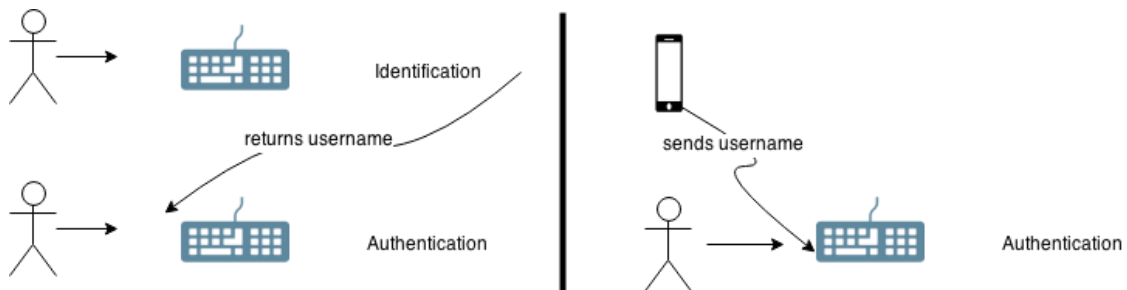


Figure 26 - On the left, without a smartphone; on the right, with a smartphone.

Explaining the whole sequence (Figure 26): (1), an individual gets in the area covered by a Bluetooth scanner in the computer; (2) this proximity triggers a process of getting the ECG authentication username associated with the detected mobile phone ready for authentication; (3) a person attempts logon and the keyboard uses the previously received username to start authentication, reading the heartbeat; (4) the user is granted access if both authentications are consistent with each other and succeeded.

3.3. Bluetooth Low Energy communication

As a solution to the GATT server role limitation in Windows, we decided to implement a “middle point” between the smartphone and Windows desktop. This system was running the Linux debian-based distro Ubilinux and, therefore, capable of advertising a

GATT service. In this scenario we also changed from Windows Phone to Android to avoid more limitations associated with their APIs.

The Linux system was capable of using BLE thanks to BlueZ (“BlueZ”, 2015), an open source API for Linux which provides support for this technology. Also, in order to keep this a mobile, more versatile and independent point, we used Intel Edison (Figure 27).



Figure 27 - Photo of the actual Edison board used.

This board allowed us to create a GATT server, listening for BLE messages from a smartphone, and redirect them to our adapted pGina plugin running in a Windows computer through Wi-Fi, while giving an “internet-of-things” tone to the approach. Still, we also tested this scheme using a Linux virtual machine in the pGina laptop itself, which proved to work.

All the following scenarios maintain the working scheme associated to the keyboard, i.e., ECG keyboard expects a contact (hands on keyboard) after pGina plugin acknowledges the presence of a valid smartphone token.

These scenarios were written in C# (pGina plugin), JavaScript (specifically Node.js in Linux) and Java (Android app). A small database was also created using SQL Server 2014 in order to store user credentials.

3.3.1. Proximity

These schemes are an improvement from the classic Bluetooth scheme as Android API 19 (KitKat) allowed to read the RSSI value associated with a BL signal and determine if the smartphone is within close range to the computer. This value, Received Signal Strength Indicator, refers to the strength of a radio signal. Although several studies indicate that RSSI (Parameswaran, 2009; Al Alawi, 2011) is not exact measure of a device proximity, it may suggest the distance if carefully handled. Also, our scheme is not dependent on an exact read of distance, but only in a definition of “proximity”. In order to understand the variation in RSSI values depending on distance, we executed a series of tests (see section Tests).

3.3.2. First scenario – Post Delayed Runnable

First, in order to solve the limitation concerning the creation of GATT servers in Windows, we started developing a GATT server in Edison’s Linux. We used Bleno (“Bleno”, 2015), a node.js module for BLE peripheral development, and created a Link-Loss service, and Battery-Level characteristic. However, none of these specifications are mandatory – any service with any writable characteristic would do. This scheme worked by advertising a GATT service, and the provided characteristic could be written by a BLE peripheral (an Android phone specifically).

The first issue we found when implementing a similar message system to the one previously implemented in our classic Bluetooth scheme was the message size limit of 20 bytes (“Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology”, 2012). This would prove impossible to send both username and password in most cases. So, we created a third username, associated with the pair *Windows username/ECG username*, represented by a smaller string. Also, we eliminated any overhead associated with the messages, being “login”, “logout” or “register”, because: (1) register is now a process started in the pGina plugin configuration dialog; (2) this scheme is proximity-based, thus requiring no “log out”, the device is either present or absent.

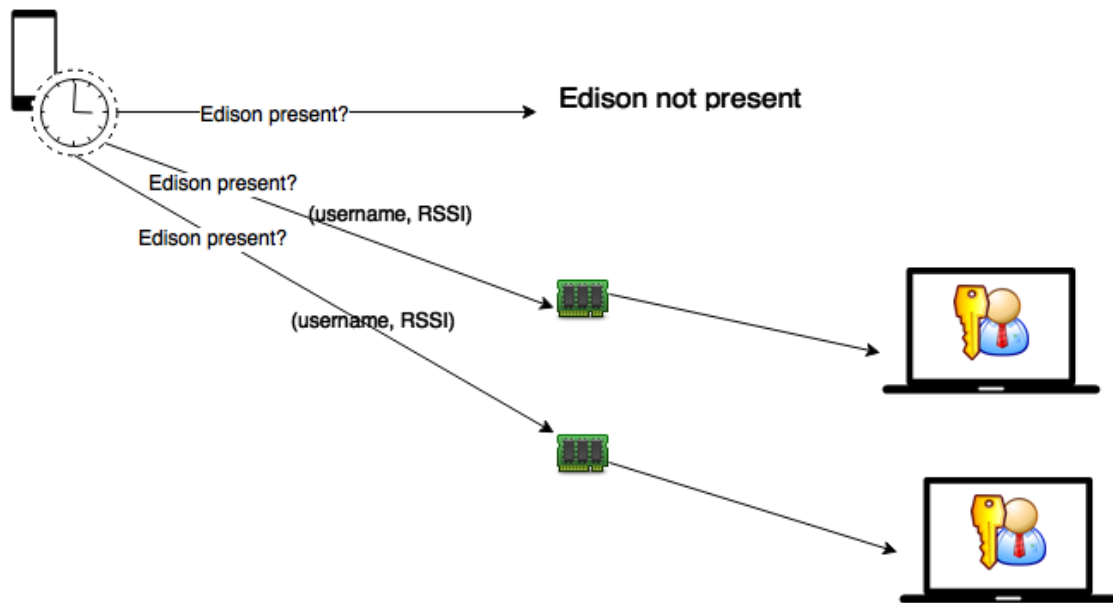


Figure 28 - First scenario. The phones scans for the GATT service advertised by Edison. When connected, writes a characteristic and it is forwarded to pGina. After this, the user should start ECG authentication by clicking the arrow button on Windows GUI.

After the message is received on Edison (meaning the GATT characteristic is written), it is forwarded to the pGina plugin via Wi-Fi, and from there it is treated as it would in the Bluetooth classic method. Notice that this message is only sent by the phone if when it is within the RSSI margin predefined, so this validation was already done by the phone itself. This approach is explained also by Figure 28.

Referring to the implementation on Android, we used post delayed runnable, which is a class in Android's API, with an interval of 1 second, to intermittently write to GATT server, and continuously attempt to reconnect after a disconnection. This initial prototype was costly to battery performance. Also, we implemented an RSSI determination that reads 10 RSSI values with an interval of 1 second and sets as the RSSI threshold for proximity to the maximum value of the set, allowing the user to set a preferred distance.

3.3.3.Second Scenario – AlarmManager

Given the issue of battery performance, we decided to implement the same scheme using the `AlarmManager` class, which is another class in Android's API with the purpose of executing a certain code from time to time. It works by trying to connect or write a characteristic every time the alarm sets off, and do nothing in the meantime. This implementation could avoid constantly wakeup of the phone, even if setting the alarm to the same interval, as it would completely stop the app between alarms. The performance of this approach is analyzed in the *Tests* section.

3.3.4.Third Scenario – Subscribing a GATT service

In this scenario (Figure 29), the most relevant difference is that the initiative is intended to be on the server side. Also, we changed the identification method from using a small username to a partial device MAC address (excluding the two characters; not total due to the limitation of 20 bytes per message – for example, AA-BB-CC-DD-EE instead of AA-BB-CC-DD-EE-FF) to avoid the need to create a third username. Keep in mind that the RSSI value is also sent in this message.

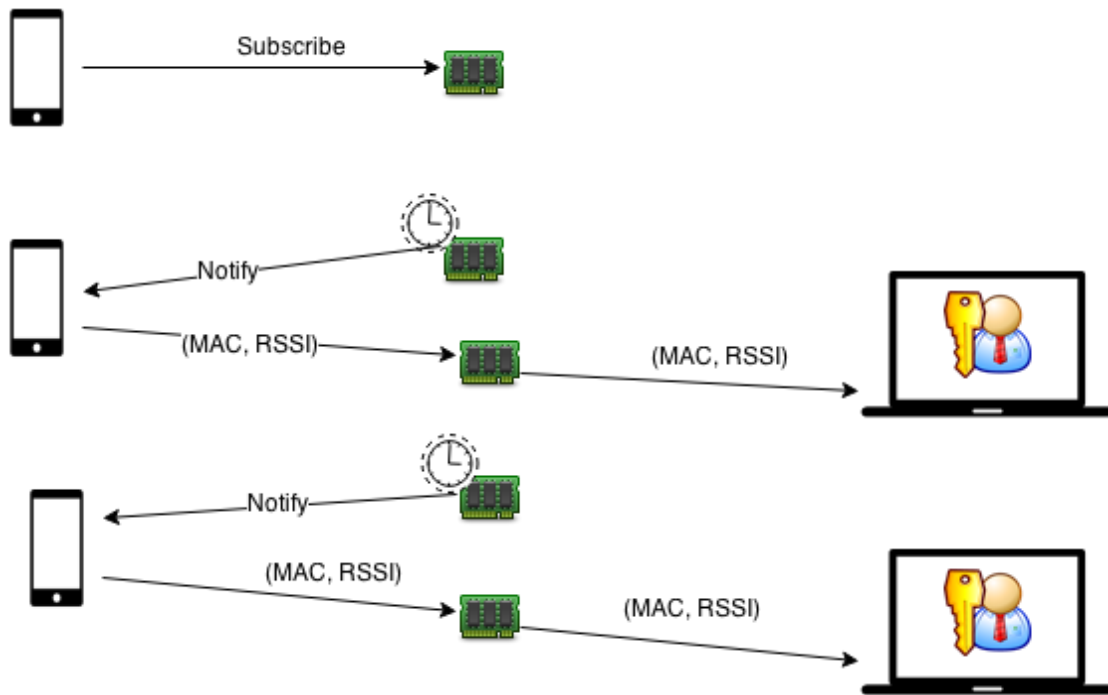


Figure 29 - Third scenario. The phone subscribes a GATT characteristic advertised by Edison and, when notified, writes a characteristic. After the message is forwarded to pGina, user should start ECG authentication by clicking the arrow button on Windows GUI.

The previous scenarios depended on the phone's initiative to write a GATT characteristic advertised by Edison. So, the GATT server advertised a service, which would be constantly scanned by a nearby phone. When connected, it would write a characteristic – meaning it would send its RSSI and small username – which would then be forwarded to pGina via Wi-Fi. However, this new scenario changes the scheme by requiring a one-time subscribe to a GATT characteristic. After this, the timer implemented on Edison associated to the characteristic notifies the subscribed phone within a certain time interval (2 seconds in this prototype). Each time a phone receives this notification, it replies with its RSSI and partial MAC address.

Therefore, this scenario is more battery friendly as the phone only responds to questions asked by the server when in range, and automatically reconnects to the server when possible by setting the connect method to auto connect – thus, leaving this process to the Android itself and not keeping the app in a reconnection loop.

3.3.5.Fourth Scenario – Using accelerometer and BLE

In this scenario we intended to avoid keeping the phone sending its position after it detects a stable position. The idea was to detect whether a phone is put down on a table next to the computer, and then sleep until its position changes. Therefore, we are assuming that in a computer-phone scheme, only the phone will move, and by knowing the phone is still we could conclude the distance between the two actors would never change.

A normal use associated with this scenario would be: (1) the phone approaches BLE server and it is transmitting its position; (2) the phone is put down next to the computer and the transmission stops; (3) a person takes the phone and it starts transmitting as it detects motion.

We decided to use accelerometer to detect motion. Other sensors could be used, like gyroscope or light change detection.

We implemented this scenario on Android to first test the accelerometer's sensitivity. However, this scenario was not successful – the sensor is so sensitive that even when standing still it would detect small variations. This issue, in addition to the impossibility of filtering motion detection with KitKat's API (i.e. waking the phone only when relevant motion is detected), lead to an app constantly being triggered by nonexistent movements.

When we tried the other two sensors, we concluded that all of them presented the same problem. It could be overpassed by filtering the motion after it is detected, but this would keep the phone always on a wake state processing motion and, therefore, would not be likely to improve battery usage over the previous scenarios.

4. Tests

In this Chapter, we explain a series of tests we performed. These tests refer to the reliability of RSSI as a measure of proximity, the improvements in smartphone battery performance associated to the use of BLE, the possibility of false negative readings of a smartphone's proximity and the actual benefit of using a token in association with the ECG authentication method.

4.1.1. Signal strength reliability

Given the uncertainty about how reliable could RSSI be for detecting proximity, we performed the following test shown in Table 1.

RSSI Values (dBm)	-33	-44	-52	-52	-59	-57
	-27	-49	-49	-54	-57	-60
	-32	-44	-49	-54	-55	-56
	-35	-42	-49	-54	-58	-55
	-43	-43	-48	-53	-57	-56
	-40	-43	-47	-54	-55	-55
	-35	-46	-48	-53	-54	-55
	-37	-44	-48	-54	-57	-60
	-34	-45	-49	-56	-55	-56
	-33	-44	-49	-56	-54	-55
Distance	30cm	1M	2M	3M	4M	5M
Best RSSI	-27	-42	-47	-52	-54	-55
Worst RSSI	-43	-49	-52	-56	-59	-60
Average	-34,9	-44,4	-48,8	-54	-56,1	-56,5

Table 1 - RSSI values depending on distance.

We performed these tests with the mobile phone as the single Bluetooth device in our environment; the phone was placed in the referred distances, with no obstacles between it and the Bluetooth scanner connected to a laptop. The adapter was a CSR8510

(“BlueCore® CSR8510™ A10 WLCSP”, 2015) and the phone was a Wiko Darkmoon (“Darkmoon”, 2015) with Android 4.4.2 . Higher RSSI values mean better signal.

We also considered adding obstacles, but considered the scope of these tests to be specific to the impact of distance in RSSI values.

Analyzing the results, we can see a decrease in average, minimum and maximum values of RSSI with the increase of distance between both devices. Also, the clearest difference in RSSI values is seen when the distance varies the least – when we step from 30 centimeters to 1 meter. However, 30 centimeters is the situation where the phone is actually in a similar distance to the one where a person carrying it approaches his computer. These results provide a more substantiated way of defining a threshold value for phone detection and logon authorization.

For implementation purposes, we defined a threshold of -49dBm (as a mere example). Even in a situation of random minimum peaks where a phone is incorrectly detected as being out of proximity, the consequence is likely to be obfuscated by the loop timer of RSSI reading – the value is so often retrieved that a single value is not likely to impact a common interaction with the system.

4.1.2. Bluetooth Performance and Battery Impact

We also conducted a series of tests to analyze the performance of both classic Bluetooth and Bluetooth Low Energy scenarios, in terms of battery and reliability. All tests were executed using a Wiko Darkmoon Android phone – the Windows Phone scenario was tested in Android by tweaking the server side implementation, in order not to require a login message and just detect the pre-stored MAC address associated to the phone (as mentioned before, this scenario did not require an app being in execution).

The phone was put in airplane mode, only with Bluetooth on. It was 100% charged for each test, and standing still about 30cm from the laptop.

For comparison, we tested the phone in airplane mode, with Bluetooth and apps off and concluded that in 1 hour, 0% battery is lost.

We considered used a more specific value than battery percentage – however, we could not access such parameter.

We tested for the following parameters:

- Phone battery performance when in-range;
- Phone battery performance when off-range;
- pGina false negatives (not detecting an in-range phone).

Regarding the classic Bluetooth scenario (sixth scenario – proximity based token), the results were the following:

- Phone battery performance when in-range: 6% drain in 1 hour.
- Phone battery performance when off-range: 0% drain in 1 hour.
- pGina false negatives: 0.

Regarding the Bluetooth Low Energy scenarios, the results were the following:

- Post delayed runnable (Scenario 1):
 - Phone battery performance in-range: 3% drain in 1 hour;
 - Phone battery performance off-range: 1% drain in 1 hour;
 - pGina false negatives: 20.
- AlarmManager (Scenario 2):
 - Phone battery performance in-range: 4% drain in 1 hour;
 - Phone battery performance off-range: 1% drain in 1 hour;
 - pGina false negatives: 16.
- GATT subscribe (Scenario 3):
 - Phone battery performance in-range: 3% drain in 1 hour;
 - Phone battery performance off-range: 0% drain in 1 hour;
 - pGina false negatives: 15.

In order to provide more conclusive visual data, we created the following charts:

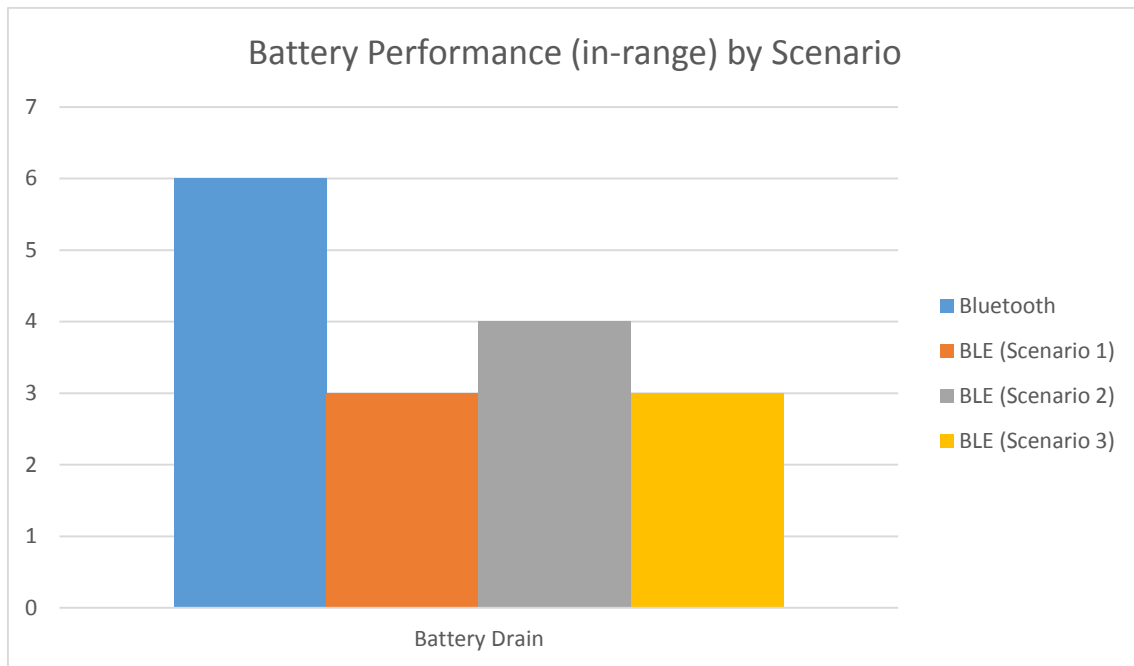


Chart 1 - Battery performance chart for each scenario when the phone is in-range.

The first thing we can conclude (Chart 1) is that the classic Bluetooth scenario was less battery efficient than all other scenarios, as expected. Although in this scenario the phone is supposedly passive - meaning no app is working and it is only detected by the server's BL scan – these results may indicate that the scan actually involves waking the scanned peripherals, and performing some work associated with the Bluetooth scan itself.

Also, we see there is no significant difference between all the other scenarios. This situation might derive from the fact that in all of them the phone is forced to do some form of processing on the background. The solution could be adding to Android's API the capability of triggering code by BLE detection, specifically with a pre-determined RSSI.

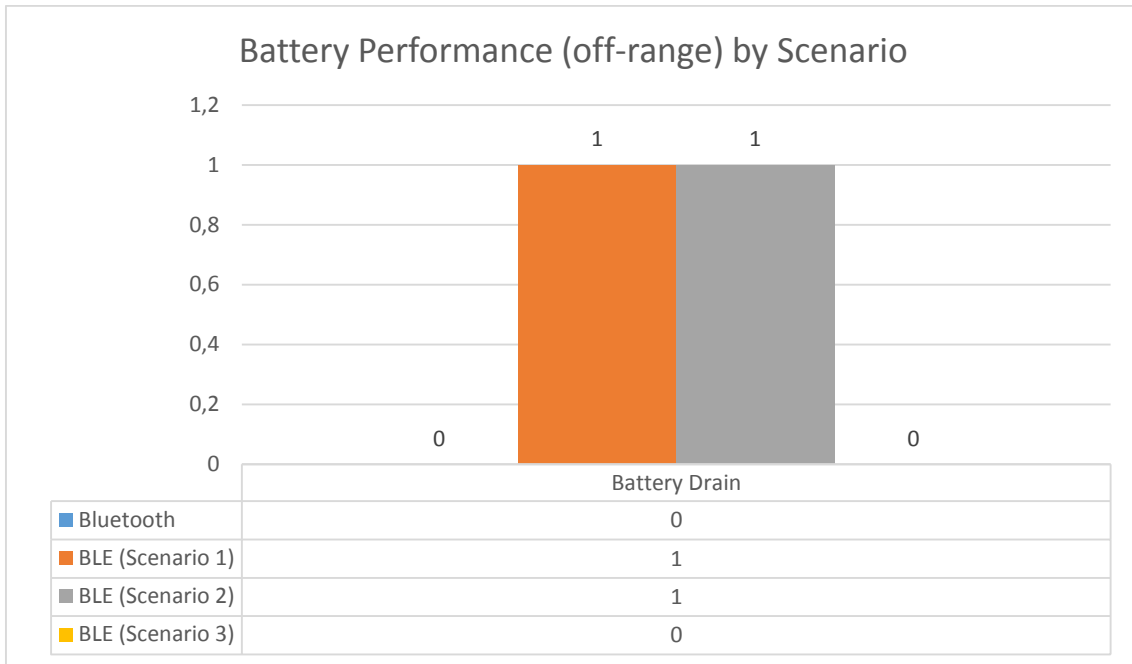


Chart 2 - Battery performance chart for each scenario when the phone is off-range.

In this off-range chart (Chart 2), we see that the battery performance is better in the classic Bluetooth and server based BLE scenario. This is explained by the fact that in both scenarios the phone is not required to be awake – in the BL scenario, the phone is passive and no server is scanning; in the BLE scenario, the phone only replies when contacted by server’s notifications. On the other two scenarios, the app is attempting connection in a loop, explaining battery usage.

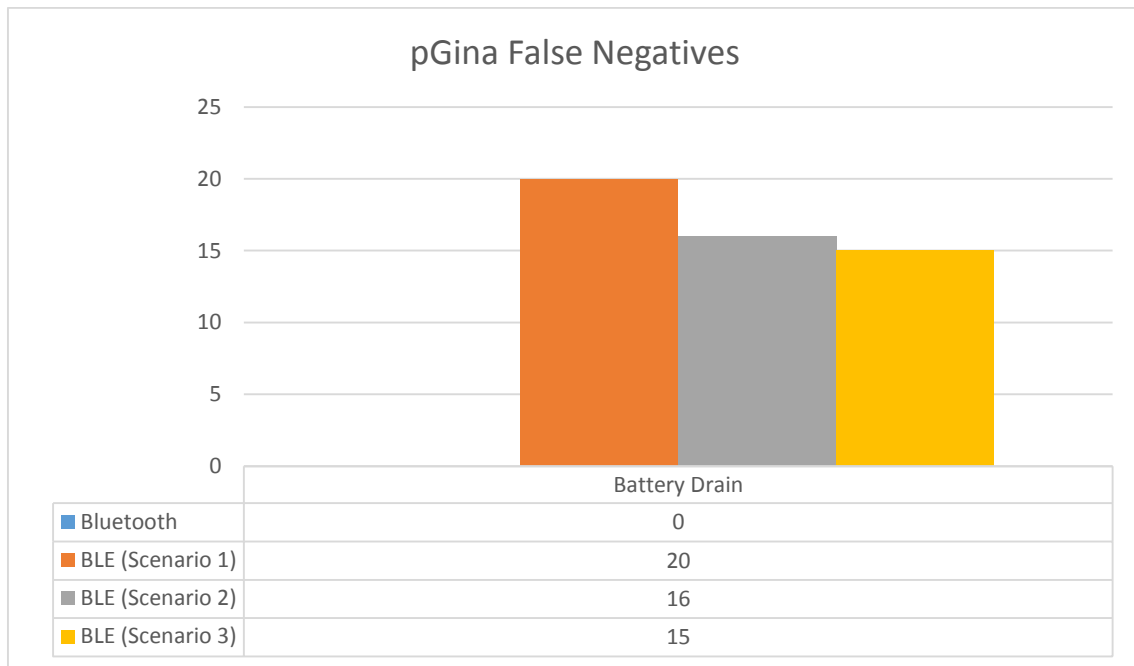


Chart 3 - pGina incorrectly reporting phone absence.

Firstly, the absence of false negatives in the BL scenario is because it is not using RSSI values (Chart 3). This positive outcome of the test is compensated by the fact that a phone is considered near even if several meters away. In BLE scenarios, on the other hand, a phone is only considered in-range if the RSSI reads -49dB or better. So, there is a tradeoff between distance detection and false negatives.

Although we could consider the number of false negatives excessive, one must consider the fact that the position is updated every second, sometimes more, due to connection inconstancies (both in Bluetooth and Wi-Fi). This means that these results are not likely to affect user interaction. Additionally, by analyzing the logs we can see that they are actually sorted in long moments of correct readings and short moments of incorrect readings. For example, the log for Scenario 2:

```

11:36:21 : Creating file
11:36:21 : Hello!
11:36:24 : Starting wi-fi server...
11:36:28 : New wi-fi client detected.
11:36:28 : Wi-fi connection established.
11:38:58 : No phone in range.
11:39:10 : No phone in range.

```

11:41:28 : No phone in range.
11:41:32 : No phone in range.
11:41:42 : No phone in range.
11:43:30 : No phone in range.
11:43:46 : No phone in range.
11:44:20 : No phone in range.
11:49:46 : No phone in range.
11:50:04 : No phone in range.
11:50:26 : No phone in range.
11:51:13 : No phone in range.
12:07:18 : No phone in range.
12:07:24 : No phone in range.
12:08:23 : No phone in range.
12:08:31 : No phone in range.

For identification purposes, we colored the occurrences that appear near in time. We can see only five major moments of incorrect readings, while maintaining stable information the rest of the hour. Also, keep in mind that we used a value for RSSI threshold (-49dB) that requires close proximity to the computer, leading to a more rigorous demand of computer-phone closeness (or less signal interference) than if a higher dB threshold was chosen.

4.1.3. Token/ECG vs ECG only comparison

In order to better understand the delay associated with identifying a person vs authenticating a person, we performed two series of tests.

Firstly, we should mention that we did not perform any tests on the keyboard capability of recognizing an individual, as it was already proven (Silva et. al, 2013). In the first test, we tested identification times; in the second test, we tested authentication times with a previously identified person (reflecting the scenario of our work). For both tests we used various databases with increasing amount of users. In these tests we aim to reflect the increase in delay associated with the identification comparing to authentication, specifically when larger databases are used. This behavior was already

referred by the team behind the ECG keyboard. Also, the same team - Carreiras et. al - used the following graphs to illustrate an identification performance with negative relation to the number of subjects in database (Figure 30). This test was performed in a context with a database of a total of 618 subjects (for more details on this, see Carreiras et al., 2011).

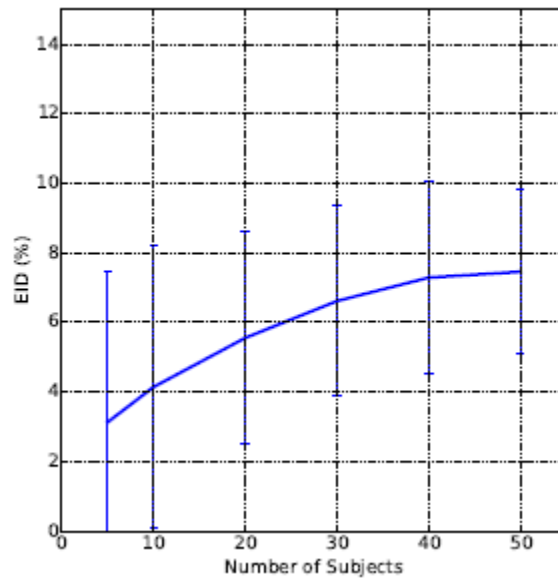


Figure 30 - Error in Identification (EID) based on the number of subjects (Adapted from Carreiras et. al, 2011).

The graph shows a clear increase in Error in Identification (EID) with the increase of the number of subjects. This proves the benefit of performing authentication only with a pre-known individual. Also, the fact that we have the identity of the person beforehand also increases the confidence level.

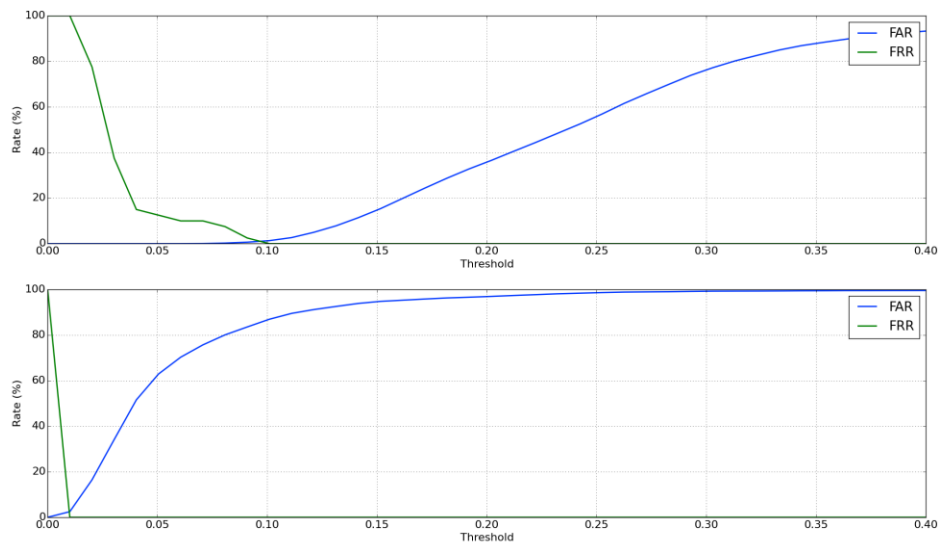


Figure 31 - False Acceptance Rate (FAR) and False Rejection Rate (FRR) depending on acceptance threshold in two individuals (Adapted from Carreiras et. al, 2011).

In a more safety demanding environment, false negatives would be preferable to false positives; on the other hand, a less demanding approach opening the door for more false positives, would increase the overall performance. With this logic in mind, the fact that the system knows the person's identity before ECG authentication allows it to have a better confidence level, possibly allowing more false positives to increase overall performance, or even a dynamic decision of when to stop collecting evidence (acquire a smaller number of ECG cycles if match was already undoubtful) (Figure 31). In other words, our solution offers the possibility of regulating the threshold point (Carreiras et. al, 2011).

Regarding the tests, we executed two series of tests. In the first series, we tested ECG identification (the classical ECG-based authentication scenario) and ECG authentication (the scenario we suggest, where the individual's identity is previously known and the ECG confirms it) times with an 8 user ECG database (see Table 2); in the second series, we tested the same with a 68 user database (see Table 3). Both tests were performed with a group of five young males (an arbitrary small number, to simplify the tests). The lack of time, as well as experience and people willing to collaborate made a higher number of individuals in the database impossible to achieve. The smaller database was

generated by asking our colleagues to collaborate; the bigger database was given by André Lourenço, and added the five testing individuals.

Individual	Identification	Authentication
1	9,51	9,33
2	9,76	9,59
3	9,32	9,23
4	9,75	9,60
5	9,80	9,49

Table 2 - Identification and authentication times using an 8 user ECG database (in seconds).

Individual	Identification	Authentication
1	12,83	9,12
2	12,54	9,86
3	11,33	9,54
4	12,87	9,89
5	12,20	9,14

Table 3 - Identification and authentication times using a 68 user database (in seconds).

It is relevant to mention that all times include the ECG acquisition. Vitalidi collects eight heartbeats, therefore an increased heartbeat will result in a faster read, and a slow heartbeat will originate longer times. For the tested individual – five young healthy males – a 60 beats per minute heart rate is considered the average (Growth, Maturation, and Physical Activity, 2004). Consequently, we assumed an average value of 8 seconds in acquisition time, and subtracted the original values (see Table 4 and Table 5).

Individual	Identification	Authentication
1	1,51	1,33
2	1,76	1,59
3	1,32	1,23
4	1,75	1,60
5	1,80	1,49
Avg.	1,628	1,448
Std. Dev.	0,185	0,146

Table 4 - Identification and authentication times using an 8 user ECG database (in seconds), with the acquisition times subtracted.

Individual	Identification	Authentication
1	4,83	1,12
2	4,54	1,86
3	3,33	1,54
4	4,87	1,89
5	4,20	1,14
Avg.	4,354	1,510
Std. Dev.	0,566	0,333

Table 5 - Identification and authentication times using a 68 user database (in seconds), with the acquisition times subtracted.

As the chart shows, a clear improvement is noticed when comparing identification and authentication times. This difference is increased when using a larger database (see Chart 4). One could expect the times to be N-times better, being N the number of total individuals in the database. Although this does not happen – and is related to the implementation of the ECG keyboard itself – the improvement is still clear.

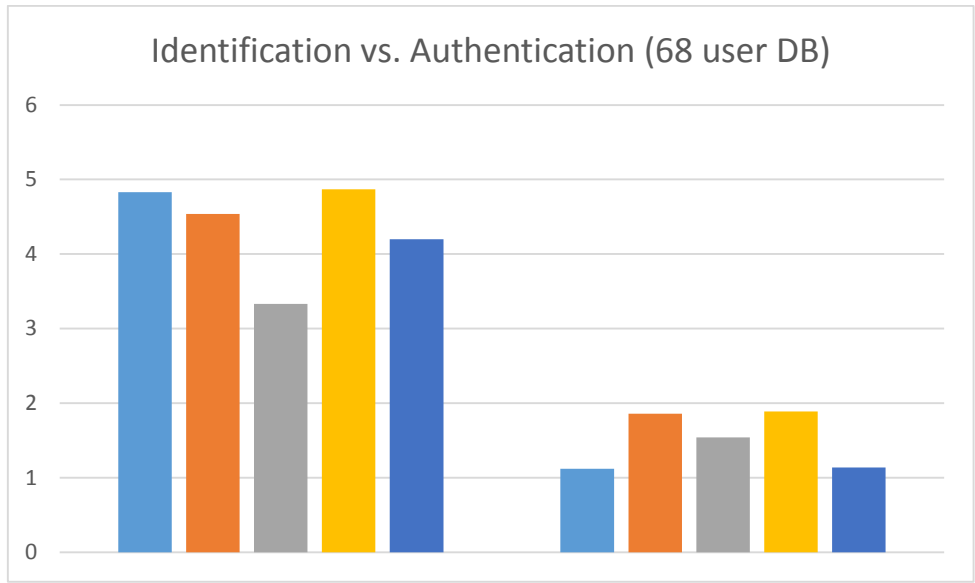


Chart 4 - Identification vs. Authentication times, in seconds, referring to the five tested individuals, using a 68 user DB. This chart translates the improvement in times comparing a typical ECG authentication with our solution, using a smartphone as token.

Results confirm the advantage of using a smartphone as a security token in an ECG authentication context. Identification is unnecessary, thus comparing the readings with every user in a large database is avoided. Instead, the readings can be compared to a single, previously known, individual's registered ECG.

5. Conclusions and Future Work

In this dissertation, we proposed the use of a second element in ECG-based authentication. We tested a total of eleven different scenarios in a Windows login context, introducing an unconventional multi-factor authentication in an everyday system. Also, we developed in three different languages (Javascript, C# and Java) and used two different mobile OS's (Android and Windows Phone), interacting with seven distinct APIs and two different computer OS's (Windows 8.1 and Linux).

Our suggestion consists mainly of an approach using an Android phone and an ECG reader keyboard, VitalidiType. In this scheme, we used Bluetooth Low Energy technology, now emerging in the commercial scene mostly with peripherals, in order to take a more battery-friendly approach. As a part of our work, we also suggested the use of classic Bluetooth in a Windows-Windows Phone scheme. In both schemes, our system is capable of working in a single-factor mode (using the phone only), or in a multi-factor mode (using the phone and the ECG keyboard). The numerous approaches presented progressively more complex situations, allowing us to approach step by step the initial objective – each approach presented a better performance than the previous, as one of the main concerns in this work was battery-life.

In the classic Bluetooth approach, we tested six different scenarios. We struggled with a number of limitations associated with Windows environments – both in desktop Windows and Windows Phone. Although we created a working prototype in this context, it did not correspond to our expectations, leading us to a different approach capable of meeting our goals.

In the Bluetooth Low Energy approach, we tested four different scenarios. We used mostly Linux on an Edison board which allowed us to create a BLE server and test the scenario we initially intended. Also, the use of Edison proves that this approach is feasible for small embedded devices, allowing it to be integrated in the Internet-of-Things. This board functioned as a middle worker, forwarding BLE messages from the phone through Wi-Fi to a computer running pGina.

Android allowed us to work with more stable Low Energy connections. Also, the BLE approach allowed us to read RSSI, giving us a hint of how close the phone actually was

from the computer. This gave us the opportunity to be stricter concerning the individual's proximity, thus denying the identification if the person is in the room but not in front of the computer. Also, Android allowed us to work with more stable Low Energy connections. Even though we used an Edison board, we also tested this scheme with a Linux virtual machine – allowing us to run both pGina (in the host Windows machine) and the GATT server (in the client virtual machine, running Linux) in the same computer.

We developed different versions using Bluetooth Low Energy, with the intention of achieving the best performance possible related to the use of CPU time in Android. Testing our solution, we conclude that our four different programmatical approaches presented little or no impact on battery life. On the other hand, we saw a more clear difference between the classic Bluetooth and BLE approaches.

Also, we saw a clear improvement in authentication times comparing to the identification and authentication based on ECG readings alone. We went from a situation where one had to be identified to a situation where the system is previously informed about user identity, requiring the authentication delay only. This translated in faster authentication, as well as decreased EER.

We submitted a paper to the 13th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC2015), for which we are awaiting a response to this date (“EUC 2015”, 2015).

For future work, we would suggest the improvement of the Android app in order not to require constant wake time of the CPU. This could also be an improvement in Android API, which could allow the registration of a listener indicating whether a specific GATT server is within a certain dB signal strength.

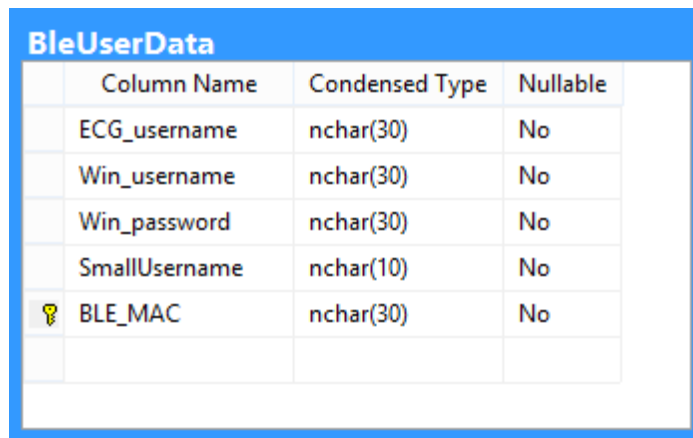
Also, the credential provider could add the possibility of authentication by detecting one's hand palms on the keyboard, thus not requiring pressing the logon button in Windows logon UI. pGina did not supported a completely autonomous logon at the time, but the development of a custom Credential Provider directly with Windows SDK in C++ is a possibility.

Appendices

A. Installation Procedures

This section is intended to help anyone install the whole system from scratch. Obviously, an ECG keyboard is needed if it is intended to use it; however, every scenario of the system is capable of running without it, making it a single-factor authentication.

1. Install pGina;
 - a. The version used in this project was 3.2.1.0.
2. Create a database;
 - a. The database connection settings are located in the method `openDatabaseConnection()`, class `Tools`;
 - b. If the scenario tested is Bluetooth Low Energy, create the following table:



	Column Name	Condensed Type	Nullable
	ECG_username	nchar(30)	No
	Win_username	nchar(30)	No
	Win_password	nchar(30)	No
	SmallUsername	nchar(10)	No
🔑	BLE_MAC	nchar(30)	No

Figure 32 - BLE database table.

- c. If the scenario tested is classic Bluetooth, create the following table:

UserData			
	Column Name	Condensed Type	Nullable
	ECG_username	nchar(30)	Yes
	Win_username	nchar(30)	Yes
	Win_password	nchar(30)	Yes
🔑	BL_MAC	nchar(20)	No

Figure 33 - Classic Bluetooth database table.

- d. Add an entry for your user. BL_MAC refers to the MAC address of the phone. If the tested scenario is the BLE third scenario, exclude the last two characters due to the previously explained constraints.
3. Compile the scenario intended to test with Visual Studio;
 - a. The output .dll file should be located in pGina/Plugins/Contrib;
 - b. Configure the plugin:
 - i. If the keyboard is intended to use, the ECG keyboard software must be installed (VitalidiType, 2015), and username must be registered. Use the provided register user button; otherwise, it is only needed to manually insert your user details to DB.
4. If using BLE, do the following in the Linux system:
 - a. Install node.js;
 - b. Install Bleno;
 - c. Compile the server code;
5. Plug in the keyboard, if it is intended to be used.
6. Start pGina service using pGina UI.
7. Run the GATT server, using `node main 123.0.0.1` (replace with the Wi-fi ip address of the computer running pGina).

Bibliography

- [Bluetooth RFComm] SocketStream.ConnectAsync(HostName,String) – SystemException element not found. (n.d.). Retrieved May 8, 2015, from <https://social.msdn.microsoft.com/Forums/windowsapps/en-US/4760a2a4-9133-4658-9f18-c5aae2b39341/bluetooth-rfcomm-socketstreamconnectasynchostring-systemexception-element-not-found?forum=winappswithsharp>
- 32feet.NET. (2012). 32feet.NET - user's guide / tutorial / examples. Retrieved January 13, 2015, from <http://32feet.codeplex.com/documentation>
- Abdelhameed, R., Khatun, S., Ali, M. B., & Ramli, A. R. (2005). Authentication model based bluetooth-enabled mobile phone. *Journal of Computer Science*.
- Al Alawi, R. (2011). RSSI based location estimation in wireless sensors networks. *2011 17th IEEE International Conference on Networks*, 118–122. <http://doi.org/10.1109/ICON.2011.6168517>
- Apple. (2014). iOS: Understanding iBeacon. Retrieved December 21, 2014, from <http://support.apple.com/en-us/HT202880>
- Ashley, E. A., & Niebauer, J. (2004). *Cardiology explained*. Andrew Ward.
- BlueCore® CSR8510™ A10 WLCSP. (n.d.). Retrieved May 28, 2015, from <http://www.csr.com/products/bluecore-csr8510-a10-wlcsp>
- Bonneau, J. (2012). The science of guessing : analyzing an anonymized corpus of 70 million passwords.
- Braz, C., & Robert, J. (2006). Security and usability : the case of the user authentication methods. *IHM*, 199–203.
- Carreiras, C., Lourenço, A., Fred, A., & Ferreira, R. (2011). ECG signals for biometric applications are we there yet ?
- Chakraborty, G., Naik, K., Chakraborty, D., Shiratori, N., & Wei, D. (2008). Analysis of the bluetooth device discovery protocol. *Wireless Networks*, 16(2), 421–436. <http://doi.org/10.1007/s11276-008-0142-1>
- Darkmoon. (n.d.). Retrieved May 28, 2015, from <http://pt.wikomobile.com/m131-DARKMOON>
- EUC 2015. (n.d.). Retrieved June 21, 2015, from <http://paginas.fe.up.pt/~specs/events/euc2015/>

- GATT server role on Windows 8. (2013). Retrieved April 6, 2015, from <https://social.msdn.microsoft.com/Forums/windowsapps/en-US/b89e673b-38b5-4ac4-b9e9-47e634d668fc/gatt-server-role-on-windows-8?forum=wdk>
- Gomez, C., Oller, J., & Paradells, J. (2012). Overview and evaluation of bluetooth low energy: an emerging low-power wireless technology. *Sensors, 12*(12), 11734–11753. <http://doi.org/10.3390/s120911734>
- Griffin, D. (2007). Create custom login experiences with credential providers for windows vista. Retrieved December 20, 2014, from <http://msdn.microsoft.com/en-us/magazine/cc163489.aspx#edupdate>
- HealthySystems. (n.d.). Retrieved April 29, 2015, from <http://www.healthysystems.pt/index.php>
- Intel® edison—one tiny platform, endless possibility. (n.d.). Retrieved April 6, 2015, from <https://www-ssl.intel.com/content/www/us/en/do-it-yourself/edison.html>
- Islam, M. S., & Alajlan, N. (2014). Model-based alignment of heartbeat morphology for enhancing human recognition capability. *The Computer Journal*.
- Jain, a. K., Ross, a., & Prabhakar, S. (2004). An introduction to biometric recognition. *IEEE Transactions on Circuits and Systems for Video Technology, 14*(1), 4–20. <http://doi.org/10.1109/TCSVT.2003.818349>
- Kaplan, K. (n.d.). Is that a spider on your dress or are you happy to see me? Retrieved May 18, 2015, from <http://iq.intel.com/smart-spider-dress-by-dutch-designer-anouk-wipprecht/>
- Landau, D. M. (n.d.). My robot is cuter than your robot. Retrieved May 18, 2015, from <http://iq.intel.com/my-robot-is-cuter-than-your-robot/>
- Lee, M.-K., Kim, J. B., & Song, J. E. (2012). Smart phone user authentication using audio channels. *2012 IEEE International Conference on Consumer Electronics (ICCE)*, 735–736. <http://doi.org/10.1109/ICCE.2012.6162060>
- Liu, J., Chen, C., & Ma, Y. (2012). Modeling neighbor discovery in bluetooth low energy networks. *IEEE Communications Letters, 16*(9), 1439–1441. <http://doi.org/10.1109/LCOMM.2012.073112.120877>
- Liu, J., Chen, C., Ma, Y., & Xu, Y. (2013). Energy analysis of device discovery for bluetooth low energy. *2013 IEEE 78th Vehicular Technology Conference (VTC Fall)*, 1–5. <http://doi.org/10.1109/VTCFall.2013.6692181>
- Microsoft. (n.d.-a). Building universal windows apps for all windows devices. Retrieved March 6, 2015, from <https://dev.windows.com/en-us/develop/building-universal-windows-apps>

- Microsoft. (n.d.-b). Proximity sample. Retrieved March 6, 2015, from <https://code.msdn.microsoft.com/windowsapps/Proximity-Sample-88129731/view/Discussions>
- Microsoft. (n.d.-c). What's new in windows phone 8.1. Retrieved March 6, 2015, from <https://msdn.microsoft.com/en-us/library/windows/apps/dn632424.aspx>
- Microsoft. (2013). Credentials management in windows authentication. Retrieved December 19, 2014, from [http://technet.microsoft.com/en-us/library/dn169014\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/dn169014(v=ws.10).aspx)
- Moyerman, S. (n.d.). Build your own face-recognition system with intel edison. Retrieved May 20, 2015, from <http://makezine.com/projects/make-43/photographic-memory/>
- Parameswaran, A. T., Husain, M. I., & Upadhyaya, S. (2009). Is RSSI a reliable parameter in sensor localization algorithms – an experimental study. *Field Failure Data Analysis Workshop (F2DA09)*, 5.
- Phan, R. C.-W., & Mingard, P. (2010). Analyzing the secure simple pairing in bluetooth v4.0. *Wireless Personal Communications*, 64(4), 719–737. <http://doi.org/10.1007/s11277-010-0215-1>
- Ramalho, M. B., Correia, P. L., & Soares, L. D. (2012). Hand-based multimodal identification system with secure biometric template storage. *IET Computer Vision*, 6(3), 165. <http://doi.org/10.1049/iet-cvi.2011.0095>
- RFCOMM connection fails. (n.d.). Retrieved March 6, 2015, from <https://social.msdn.microsoft.com/Forums/en-US/62de78de-b6f1-4c8a-bc2d-b88c77c6dd4d/rfcomm-connection-fails?forum=winappswithcsharp>
- RFCOMM connection works unstable. (n.d.). Retrieved March 6, 2015, from <http://stackoverflow.com/questions/27122484/rfcomm-connection-works-unstable/>
- Robert M. Malina, Claude Bouchard, O. B.-O. (2004). *Growth, Maturation, and Physical Activity*. (J. P. Wright, Ed.) (2nd ed.). Human Kinetics.
- Sandeepmistry. (n.d.). Bleno. Retrieved April 25, 2015, from <https://github.com/sandeepmistry/bleno>
- Silva, H. P. da;, Lourenço, A., Fred, A., Raposo, N., & Aires-de-Sousa, M. (2013). Check your biosignals here : a new dataset for off-the-person ECG biometrics. *Computer Methods and Programs in Biomedicine*.
- Suri, P. R., & Rani, S. (2006). Avoidance of intruder attack with changed bluetooth authentication procedure. *Information Technology Journal*.
- Syta, E., Kurkovsky, S., & Casano, B. (2010). RFID-based authentication middleware for mobile devices. *2010 43rd Hawaii International Conference on System Sciences*, 1–10. <http://doi.org/10.1109/HICSS.2010.320>

- Tanvi, P., Sonal, G., & Kumar, S. M. (2011). Token based authentication using mobile phone. *2011 International Conference on Communication Systems and Network Technologies*, 85–88. <http://doi.org/10.1109/CSNT.2011.24>
- Technologies, C. (2015). VitalidiType. Retrieved from https://drive.google.com/folderview?id=0B4E-dmCbJgp8UU9pQmM5NngtUEk&usp=sharing_eid&tid=0BymVvbGuqTtWeWttZG51a3h5Nm8
- Unhandled exception. (n.d.). Retrieved June 9, 2015, from <https://github.com/Keboo/PebbleSharp/issues/1>
- Windows 8, bluetooth LE and bluetoothFindFirstDevice/BluetoothFindNextDevice/BluetoothFindDeviceClose. (n.d.). Retrieved March 6, 2015, from <https://social.msdn.microsoft.com/Forums/windowsdesktop/en-US/3b62bdbf-9a55-4c0f-becf-f4e91d4bc027/windows-8-bluetooth-le-and?forum=wdk>
- Wolff, D. (2014). How pGina works. Retrieved December 20, 2014, from <https://github.com/pgina/pgina/wiki/How-pGina-Works>
- Zheng, J. De. (2011). A framework for token and biometrics based authentication in computer systems. *Journal of Computers*, 6(6), 1206–1212. <http://doi.org/10.4304/jcp.6.6.1206-1212>
- Zviran, M., & Haga, W. J. (1999). Password security: an empirical study. *Journal of Management Information Systems*, 15(4), 161–185. Retrieved from <http://dl.acm.org/citation.cfm?id=1189470>