Multi-factor Authentication for Improved Efficiency in ECG - Based Login

Pedro Neves[1,3], Luís Nunes[1,3] and André Lourenço[2,3]
[1]*Instituto Universitário de Lisboa (ISCTE), Lisboa, Portugal*
[2]*Instituto Superior de Engenharia de Lisboa (ISEL), Lisboa, Portugal*
[3]*Instituto de Telecomunicações, Lisboa, Portugal*

Keywords: Biometrics, Authentication, Bluetooth, Internet of Things.

Abstract: Electrocardiogram (ECG) based biometrics have proven to be a reliable source of identification. ECG can now be measured off-the-person, requiring nothing more than dry electrodes or conductive fabrics to acquire a usable ECG signal. However, identification still has a relatively poor performance when using large user databases. In this paper we suggest using ECG authentication associated with a smartphone security token in order to improve performance and decrease the time required for the recognition. This paper reports the implementation of this technique in a user authentication scenario for a Windows login using normal Bluetooth (BT) and Bluetooth Low Energy (BLE). This paper also uses Intel Edison's mobility features to create a more versatile environment. Results proved our solution to be feasible and present improvements in authentication times when compared to a simple ECG identification.

1 INTRODUCTION

This paper addresses a problem reported in the literature concerning electrocardiogram (ECG) authentication. The problem is that it presents a significant time delay associated with identification performance when using larger databases (Carreiras et al., 2014). We suggest using a smartphone, as security token, to improve the performance of the process.

In this paper we describe the experiments leading to the implementation of a multi-factor authentication scheme, in a Windows environment, capable of joining ECG and mobile-phone security token authentications to improve identification efficiency. Also, in order to guarantee a battery-friendly scheme we propose and test a communication protocol between the smartphone and the computer using Bluetooth Low Energy (BLE).

2 BACKGROUND

2.1 Authentication Methods

User authentication is based on three commonly used factors: what you know (passwords), what you have (security tokens) and what you are (biometrics) (De Zheng, 2011). 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.

2.2 Electrocardiogram Authenticaton

ECG information can be used as a biometric method. 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. The intrusiveness of ECG data acquisition can be classified as: (1) In-the-person, which means implanting devices (e.g. pacemakers) and is highly intrusive,; (2) On-the-person, meaning the devices that 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.

One approach to this method is an off-the-person approach (Silva et al., 2013), that collects ECG data from the hand (palms and fingers), facilitating large scale acquisition of ECG signals. Authors propose a solution for ECG acquisition by reading signals on

Neves, P., Nunes, L. and Lourenço, A.
Multi-factor Authentication for Improved Efficiency in ECG: Based Login.
DOI: 10.5220/0005936500670074
In *Proceedings of the 3rd International Conference on Physiological Computing Systems (PhyCS 2016)*, pages 67-74
ISBN: 978-989-758-197-7

hand palms with dry Ag/AgCL (silver chloride) electrodes, which are types of reference electrodes known for their stability and electrode potential.

2.3 Smartphone as Token

Smartphones have been used in authentication contexts before. A solution based on a mobile phone is suggested in (Tanvi et al., 2011). This source refers hardware cost as a relevant drawback of the traditional approach to security tokens (which includes individual tokens and authentication server costs) and proposes the use of phones as an alternative.

The authors presented a scheme consisting of: a user (trying to gain access to the application), a dynamic password generator for each identification process, a SMS server, responsible for sending dynamic passwords to the user, and a “Visitor password Register” (VPR), temporarily registering session’s username and password. Also, these authors suggest using the IMSI (International Mobile Subscriber Identity) number to strengthen the authentication.

This paper also refers an extension of this approach to Bluetooth, labeled “Laptop-user Authentication Based Mobile phone” (LABM) (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.

2.4 Custom Authentication in Windows

Windows 7 and later use Credential Providers (CP) to manage credentials 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 (Microsoft, 2013). Tiles are processed by the logon UI. It requests all credential types provided by each CP and displays them. One CP may have more than one associated tile, and can specify one of them as default.

Windows authentication system can also be managed through pGina (Wolff, 2014). 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.

A typical logon (Figure 1) 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.

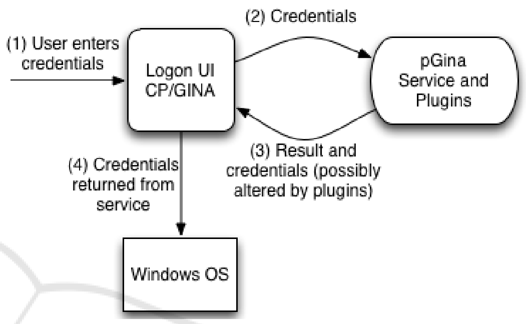
(2) Credentials
(1) User enters credentials
Logon UI CP/GINA
pGina Service and Plugins
(3) Result and credentials (possibly altered by plugins)
(4) Credentials returned from service
Windows OS

Figure 1: Diagram showing the main pGina systems interacting in a common logon, adapted from (Wolff, 2014).

2.5 Intel Edison

Intel Edison consists of a small board (35.5x25.0x3.9 mm) capable of running a Linux environment (Intel, 2015).

This board opens a wide range of opportunities related with the Internet of Things. In this paper we develop a custom authentication system in Windows; however, Edison opens 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.

Edison was already used in applications such as Jimmy, a humanoid prototype. It is an 18 inches tall robot, 3D printable (Landau, 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.

3 IMPLEMENTATION

We intended to implement a scheme where a Windows Phone would send its owner’s identification through Bluetooth Low Energy when within a certain range of a computer. This computer would advertise a

Generic Attribute Profile (GATT) server, which would be accessed and wrote into by the phone, and would have an ECG keyboard plugged 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 (Social.msdn, 2013). Given this hurdle, 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.

3.1 Classic Bluetooth Communication

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, 2012).

In this preliminary version, we implemented the authentication stage of pGina only, starting with a Bluetooth interaction. We experienced problems 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 requires no previous pairing.

Initially, we implemented a solution that would trigger Bluetooth discovery when the user attempted logon through regular Windows GUI by clicking a button on logon interface, 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. This proved to slow down the process, 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).

Afterwards, we started developing a Windows Phone (WP) 8.1 app for Lumia 530 that connected 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, previously mentioned, lack of APIs to perform discovery and pairing with BLE in Windows environment (Social.msdn, 2015). This time we achieved the goal of connecting both devices and communicating strings between them, allowing us to build a prototype of Bluetooth-based login. This prototype 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 a significant increase in performance when compared to the time it would take to detect user presence/absence in the first scenario. However, this solution requires user interaction. Also, the connection was not reliable – it would randomly connect, or throw an exception. Similar problems were reported by other developers (Social.msdn, 2015).

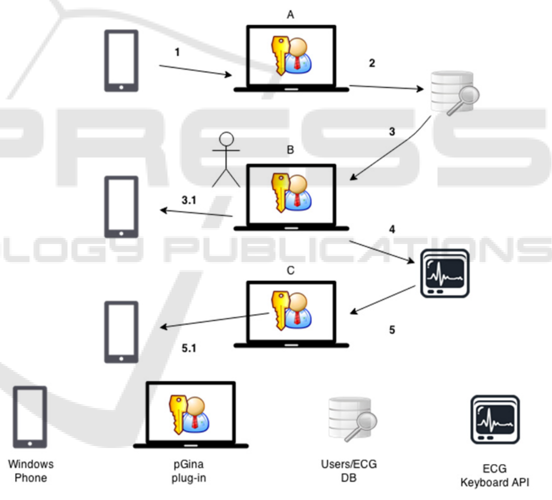

Figure 2: Communication scheme: 1) process is started by interacting with WP app; 2) pGina processes username; 4) ECG is read; 5) logon is successful/unsuccessful).

In order to avoid requiring user interaction with mobile phone to perform login, we tried to execute the connection as a background task. 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 was unstable when compiled to WP8.1, as was also reported in (Social.msdn, 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 further considered and we backtracked to WP8.0.

Figure 2 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 API. In 1, the WP app would send a message to our server containing the username; in A, pGina confirms the username with a database (2). A feedback might be implemented, although we considered at this point it would unnecessarily increase the process complexity (3.1 and 5.1). Action is suspended before 4, while waiting for Windows logon; then, if ECG authentication is successful (5), pGina allows logon and, possibly, returns feedback to WP.

We also considered sending Bluetooth MAC address from WP to pGina, in order to confirm the identity of the phone itself – but WP will only allow BL communication in-app with previously paired devices, meaning no unpaired device could connect with the server.

We chose to give the initiative of the communication to WP and store all passwords in a computer located database because: *1)* the necessity of asking for an explicit user interaction ruled out the initiative being on 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.

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 process is the following: *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); *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.

After implementing the Bluetooth side, we approached the introduction of the ECG-reader keyboard on the existing scheme. 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 a heart beat and then authenticating the individual. Both processes can take several seconds, thus leading to a delay in 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 confidence level. The keyboard does its own autonomous processing. It requires installing VitalidiType (Technologies, 2015), the software that allows interaction with the keyboard. The interaction between the pGina plugin and this system is made using its C# APIs, vitalidiAPI.

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.

In summary, the whole sequence is: *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 users's heartbeat; *4)* the user is granted access if both authentications are consistent with each other and succeeded.

3.2 Bluetooth Low Energy

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 Linux 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 through BlueZ, which provides support for this technology. Also, in order to keep this a mobile, more versatile and independent point, we used Intel Edison. 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.

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. They are an improvement from the classic Bluetooth scheme as Android API 19 (KitKat) allowed to read the RSSI (Received Signal Strength Indication) value associated with a BL signal. Although several studies indicate that RSSI is not an exact measure of device proximity, it may suggest the distance if carefully handled (Parameswaran et al., 2009), (Al Alawi, 2011). 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).

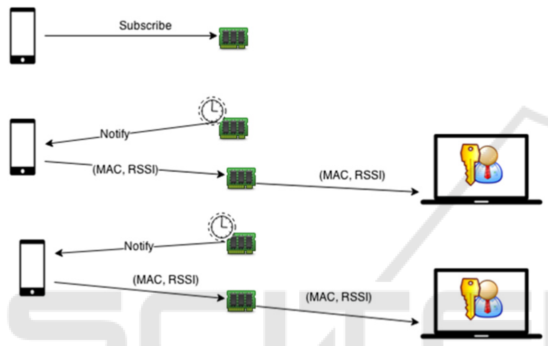

Figure 3: The phone subscribes a GATT characteristic advertised by Edison and, when notified, writes a characteristic. After the message is forwarded to pGina, user should start ECG authentication by clicking the arrow button on Windows GUI.

First, we started developing a GATT server in Edison's Linux. 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 (Gomez et al., 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.

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 it is within the RSSI margin predefined, so this validation was already done by the phone itself.Referring to the implementation on Android, we changed the initiative of the communication from the phone to the server side. Also, we used a partial device MAC address (excluding two characters due to the limitation of 20 bytes per message. This new scenario requires 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. An attempt to use the accelerometer to detect user's (in)activity was not.

4 TESTS

4.1 Signal Strength Reliability

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

Table 1: RSSI values depending on distance.

RSSI (dB)	**Distances**					
	30cm	***1M***	***2M***	***3M***	***4M***	***5M***
Best	-27	-42	-47	-52	-54	-55
Worst	-43	-49	-52	-56	-59	-60
Avg.	-34,9	-44,4	-48,8	-54	-56,1	-56,5

The 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 CSR851 (CSR, 2015) and the phone was a Wiko Darkmoon (Wiko, 2015). Higher RSSI values mean better signal. We obtained 10 readings with 1s interval for each distance. 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 -49dB. 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

4.2 Bluetooth Performance

We also conducted a series of tests aiming 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 tested for the following parameters (Figure 4): Phone battery drain in-range; phone battery drain off-range; pGina false negative (not detecting an in-range phone). Regarding in-range performances, we can conclude is that the classic Bluetooth scenario was less battery efficient than all other scenarios. 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 doing some work associated with the Bluetooth scan itself.

Off-range results show us 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 scen7ario, 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.

In another test we measure the difficulty in detecting the phone when in range (false negatives). In the BL scenario there are no false negatives because it is not using RSSI values. This has a down-side, the fact that a phone is considered near even if its 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. Tests showed a minor (0,416%) false negative rate.

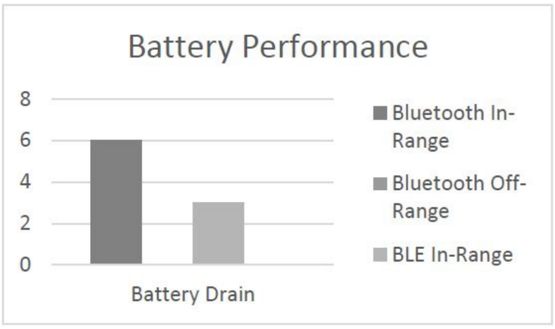

Figure 4: Percentage of battery loss in each scenario.

Although we could consider the number of false negatives excessive, one must consider the fact that the position is updated every second, sometimes more (or even less) due to connection inconstancies (both in Bluetooth and Wi-fi). This means that these results are not likely to affect user interaction. Also, by analyzing the logs we concluded that they are sorted in long moments of correct readings and short moments of incorrect readings (thus showing peaks of bad readings and, most of the time, stable correct readings). Additionally, we used a value for RSSI threshold (-49db) that requires close proximity to the computer, leading to the possibility of more false negatives.

4.3 Token/ECG Vs ECG Comparison

In order to better understand the delay associated with identifying vs authenticating a person we ran a (necessarily) limited number of tests to confirm the conclusions of previous studies published in (Carreiras et al., 2014). This study shows an increase in identification delay and decrease in accuracy when the number of subjects in database increases.

We executed two series of tests. In the first series (Table 2), 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; in the second series (Table 3), we tested the same with a 68 user database. Both tests were performed with a group of five persons (an arbitrary small number, to simplify the tests). The smaller database was generated by measuring and registering the volunteers ECG; the bigger database was provided by the authors' of (Carreiras et al., 2014) complemented by the five testing individuals of the smaller data-base.

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. For the tested individual – five young healthy males – a 60 beats per minute heart rate is considered the average (Malina et al., 2004). Consequently, we assumed an average value of 8 seconds in acquisition time, which were subtracted to the results.

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.

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

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

In a more safety demanding environment, false negatives would be preferable to false positives; on the other hand, a less demanding approach opens the door for more false positives if that 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. In other words, our solution offers the possibility of regulating the threshold point.

5 CONCLUSION

In this paper, we proposed the use of a second element in ECG-based authentication. We tested the techniques in a Windows login scenario, introducing an unconventional multi-factor authentication in an everyday system. Our suggestion consists mainly of an approach using an Android phone and an ECG reader keyboard. In this scheme, we took advantage of 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).

Table 3: Identification and authentication times using a 68 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

In the classic Bluetooth approach, 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.

In the Bluetooth Low Energy approach, 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. In this case Edison functioned as a middle worker, forwarding BLE messages from the phone through Wi-Fi to a computer running pGina. Also, Android allowed us to work with more stable Low Energy connections. 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.

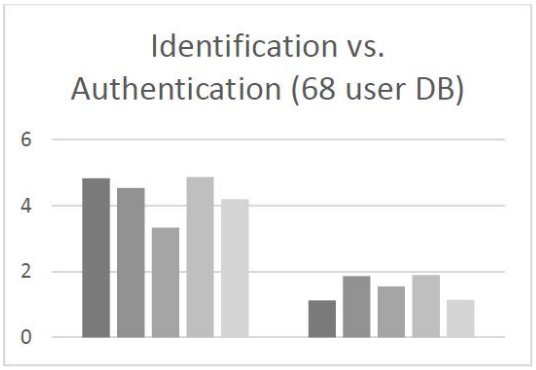

Figure 5: Identification vs. Authentication times, in seconds, referring to the five tested individuals, using a 68 user DB. Comparison between a typical ECG authentication with our multi-factor approach.

We also analyzed, although limited by the number of tests performed, the improvement in authentication performance comparing to identification based on ECG readings alone. We went from a situation where one had to be identified and authenticated (adding the delay associated to each), to a situation where the system is previously informed about user's foressen identity, requiring the authentication delay only. This translated in faster authentication, as well as decreased identification error rates.

REFERENCES

32feet.NET, 2012. 32feet.NET - user's guide / tutorial / examples. [Online] Available at: http://32feet.codeplex.com/documentation. [Accessed 13 Jan 2015].

Abdelhameed, R., Khatun, S., Ali, M. B. & Ramli, A. R., 2005. Authentication model based bluetooth-enabled mobile phone. *J. Comput. Sci.*.

Al Alawi, R., 2011. *RSSI based location estimation in wireless sensors networks.* s.l., s.n., p. 118–122.

Carreiras, C., Lourenço, A., Fred, A. & Ferreira, R., 2014. *ECG signals for biometric applications are we there yet?*. s.l., s.n.

Chakraborty, G. et al., 2008. Analysis of the bluetooth device discovery protocol. *Wirel. Networks,* 16(2), p. 421–436.

CSR, 2015. *BlueCore® CSR8510TM A10 WLCSP.* [Online] Available at: http://www.csr.com/products/bluecore-csr8510-a10-wlcsp [Accessed 28 May 2015].

De Zheng, J., 2011. A framework for token and biometrics based authentication in computer systems. *J. Comput.,* 6(6), p. 1206–1212.

Gomez, C., Oller, J. & Paradells, J., 2012. Overview and evaluation of bluetooth low energy: an emerging low-power wireless technology. *Sensors,* 12(12), p. 11734–11753.

Intel, 2015. *Intel® Edison—one tiny platform, endless possibility.* [Online] Available at: https://www-ssl.intel.com/content/www/us/en/do-it-yourself/edison.html [Accessed 6 Apr 2015].

Landau, D. M., 2015. *My robot is cuter than your robot.* [Online] Available at: http://iq.intel.com/my-robot-is-cuter-than-your-robot [Accessed 18 May 2015].

Malina, R. M., Bouchard, C. & Oded, B.-O., 2004. Growth, Maturation, and Physical Activity. In: J. P. Wright, ed. *Human Kinetics.* s.l.:s.n.

Microsoft, 2013. *Credentials management in windows authentication.* [Online] Available at: http://technet.microsoft.com/en/library/dn169014(v=ws.10).aspx [Accessed 19 Dec 2014].

Parameswaran, A. T., Husain, M. I. & Upadhyaya, S., 2009. Is RSSI a reliable parameter in sensor localization algorithms – an experimental study. *F. Fail. Data Anal. Work.,* p. 5.

Sandeepmistry, 2015. *Bleno.* [Online] Available at: https://github.com/sandeepmistry/bleno [Accessed 25 Apr 2015].

Silva, A. P. et al., 2013. Check your biosignals here : a new dataset for off-the-person ECG biometrics. *Comput. Methods Programs Biomed.*

Social.msdn, 2013. *GATT server role on Windows 8.* [Online] Available at: https://social.msdn.microsoft.com/Forums/windowsapps/en-US/b89e673b-38b5-4ac4-b9e9-47e634d668fc/gatt-server-role-on-windows-8?forum=wdk [Accessed 6 Apr 2015].

Social.msdn, 2015. *RFCOMM connection fails.* [Online] Available at: https://social.msdn.microsoft.com/Forums/en-US/62de78de-b6f1-4c8a-bc2d-b88c77c6dd4d/rfcomm-connection-fails?forum=winappswithcsharp [Accessed 6 Mar 2015].

Stackoverflow, 2015. *RFCOMM connection works unstable.* [Online] Available at: http://stackoverflow.com/questions/27122484/rfcomm-connection-works-unstable [Accessed 6 Mar 2015].

Tanvi, P., Sonal, G. & Kumar, S. M., 2011. *Token based authentication using mobile phone.* s.l., s.n., pp. 85-88.

Technologies, C., 2015. *VitalidiType,* s.l.: s.n.

Wiko, 2015. *Darkmoon.* [Online] Available at: http://pt.wikomobile.com/m131-DARKMOON [Accessed 28 May 2015].

Wolff, D., 2014. *How pGina works.* [Online] Available at: https://github.com/pgina/pgina/wiki/How-pGina-Works [Accessed 20 Dec 2014].