University College London

Department of Computer Sciences
MSc Information Security

A Privacy Enhancing Architecture for Secure Wearable Devices

Author
Sonnino Alberto

Supervisor
Dr. Meiklejohn Sarah

Master Thesis

September 2016

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Dr. Sarah Meiklejohn for her continuous support, patience, motivation, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis.

I also place on record, my sense of gratitude to one and all, who directly or indirectly, have lent their hand in this venture.

Abstract

The impressive growth of the field of communication technology leads to the expansion of wearable devices and embedded systems. This upgrowth has turned the focus towards low power consumptions, small sizes and extremely embedded designs. Though higher efforts in these fields grant the boon of better usability, performances and user's experience, the price is often paid in terms of security and user's privacy.

While wearable devices, often deprived of operating systems, possess only a small embedded processor with minimal computational power, the computer server has the capability to execute advanced and expensive computations. In this paper, we describe an architecture of a privacy enhancing technology, exploiting an anonymous credential system based on the protocols developed by [13]. This architecture involves a server issuing and verifying the anonymous credentials, a user's wearable, and RF beacons for indoor positioning. The design of each entity remains flexible, generic and scalable in order to encourage further enhancements and easy integration into real-world applications. This project also presents an example of the architecture's hardware prototype and provides directions towards the realisation of a final industrial system.

Keywords: Secure Wearable Devices, Secure Embedded Systems, Anonymous Credentials, Privacy Enhancing Technologies, Private Indoor Positioning.

Contents

Introduction and Motivations

Why do we need security on wearable devices ? The primary reason comes from the fact that, being in direct contact with the user, wearable devices have access to very private and sensitive user's information more often than traditional technologies. The huge and increasing diversity of wearable technologies makes almost any kind of information at risk, going from medical records to personal habits and lifestyle. For that reason, when considering wearables, it is particularly important to introduce appropriate technologies to protect these data, and it is primary that both the user and the engineer are aware of the exact amount of collected information as well as the potential threats pending on the user's privacy. Moreover, it has also to be considered that the privacy of the wearable's user is not the only one at risk. In fact, more and more devices are not limited to record the user's activity, but can also gather information about people standing around [33, 23, 12].

For the above reasons, one of the most today's expanding trends concerns the realisation of privacy enhancing technologies (PET), which allows to protect personally identifiable information and preserve the user's privacy while benefitting of many advantages of the most modern technologies. In that purpose, constructions like anonymous credentials [10, 11, 14, 15, 27] have been developed. These technologies allow authenticated transactions between users and servers to remain anonymous, and for that reason, consist in powerful tools for the user's privacy protection.

Finally, the large diversity of today's wearable devices includes applications going from automatic doors opening to as far as the field of health monitoring [31]. For that reason, it is particularly complicated to rely on a common standard architecture ensuring security and efficiency at the same time. It is therefore necessary to possess a set of reliable constructions in order to select the most appropriate depending on the application. As described in section 1.1, many new challenges appeared with the introduction of wearable and embedded technologies that are just waiting to be overcome.

Contribution

This paper presents a flexible privacy enhancing system from its architecture to the prototyping level. The system takes advantage from anonymous credentials and is based on the protocols developed by [13]. Three main entities are involved in the system: a main server, wearable devices and localisation beacons. In this multi-purpose architecture, the server firstly issue some anonymous credentials to the wearables. Then, each time a wearable reach a particular physical location (gets close to a localisation beacon) where it desires to perform an action, it starts presenting its credentials in order to ask the server the execution of that a particular action.

Both the design of the wearable and the server remain generic and scalable in order to encourage further enhancements and easy integration into real-world applications; i.e., the central server can manage an arbitrary number of devices, each device can posses an arbitrary number of credentials and the coverage area of the localisation system is arbitrarily extendable.

Paper's Structure

This paper is divided in three main parts. The first, the literature review, starts by describing the main new challenges brought by wearable devices and embedded technologies, and by exposing some good examples of today's privacy enhancing embedded devices. Then, the fundamental concepts needed for a good understanding of this thesis are derived.

The second part focusses on the realised architecture and explains in dept its functionalities as well as its security characteristics and the potential threats pending on the system.

Finally, the third part is completely devoted to the hardware prototype. This part starts by developing the system's conceptual model and then explains in details the hardware and software implementation. This part ends by presenting the prototype's performances and limitations, and by showing the direction to transform the prototype into a fully industrial system.

Appendix A illustrates the design's steps and the strategy adopted during the realisation of this work, the main challenges that had to be overcome, and justifies the choice the hardware components.

Chapter 1

Literature Review

Intuitively, wearable devices can be seen as objects embedded with electronic circuitry and software, that can be incorporated into clothing and accessories, and capable of connectivity with a remote host without the need of humans. Wearable computing is becoming more and more popular and certainly constitutes one of the biggest improvements of this generation. Nevertheless, as any emerging field, it comes with new challenges, which are developed below [24, 12]. Then, some examples of today's privacy enhancing embedded devices are presented in section 1.2, and finally, the fundamental concepts needed for a good understanding of this thesis are derived.

1.1 New Challenges Brought by Wearable Technologies

This sections aims to present the main problematics brought by wearable and embedded technologies that were not significant in traditional computer systems. In fact, the following considerations raise concerns around the whole security community and present excellent opportunities for today's security engineers.

Challenge 1 - Very Limited Resources

When considering wearable devices, the designer should pay a special attention to the limit of the device's resources. Indeed, common wearable devices' requirements consist in predefined shapes, long battery life and small sizes, which lead to the construction of technologies with extremely low resources [8]. Getting short in them may lead to the catastrophic choice of having to redesign the system from the beginning or cutting off on security.

Commonly, moving heavy computations from the wearable to a remote server is an interesting way of mitigating that above problem. Nevertheless, this delegation is not without risks since the server needs to be thrusted or security countermeasures have to be implemented. Moreover, the wearable-to-server communication needs protection as well. To that purposes, many researchers developed secure-friendly architecture that, depending on the application, might mitigate this situation [31, 8]. However, most of these architectures do not focus on data privacy at all. Note that identifying which parts of the system could be better optimized in hardware instead of in software and taking advantage as much as possible from all cryptographic CPU instructions are also part of the key features of modern embedded systems.

Challenge 2 - Huge Diversity Among Devices

One of the key particularities of wearable devices distinguishing them from other technologies is their very wide diversification. Indeed, wearable devices can differ from each others in terms of application, size, shape, material, utility, and many other characteristics. This huge diversification introduces new challenges never seen before. For instance, wearables do not always have a screen and keyboard to enter a password and authenticate the user or retrieve sensitive data. Engineers need therefore to innovate and design new appropriate security mechanisms withstanding the required level of security [12].

Moreover, diversity and new mechanisms always introduce usability problems. First of all, they must enhance universal access; i.e., any kind of users, regardless their abilities, origins or characteristics should be able to use it. Secondly, if a mechanism is new and changes from device to device, the user might potentially never have seen it before. Hence, it is again, more than ever, important to design intuitive systems [12, 37]. This latter consideration should be taken into account even more seriously when designing security. In fact, the benefits of security are an avoided negative instead of a gained positive, and being secure is rarely the primary goal of the system. Therefore, people are not motivated by security and if it requires an excessive workload, users will simply try to circumvent it, which might lead to catastrophic consequences [3].

As last comment, the large variety of wearables and their huge range of possible applications make difficult to rely on a common standard architecture. Therefore, an additional challenge involves the design of a secure and efficient architecture, taking into account all the limitations and needs specific to the device. Indeed, no matter how secure is the underlying implementation, if the architecture is vulnerable, the entire system could be broken.

Challenge 3 - Unprecedented Connectivity

The next particularity of wearable devices pointed out in this section concerns their unprecedented connectivity. When systems with high connectivity are considered, security automatically becomes a key problem. Devices must only allow access to authorized users and must also keep the communication secure when transmitting or receiving personal or private information [31]. Therefore, a complete access control cannot be avoided and communications should be protected.

Challenge 4 - Emerging Field

The last point of this section aims to quickly recall the hidden dangers of new research fields. Indeed, due to their novelty, emerging fields lack in standards and, trying to solve new problems using well-known grounded concepts is a typical ways to proceed. Nevertheless, adapting these concepts to a new domain is not a trivial task, and even if the mathematical primitives or the architecture on top are reliable, a bad implementation can completely break the system [24]. The above problematic also raises the concern of the lack of libraries for embedded and OS-deprived devices. Indeed, most of the known cryptographic library have been implemented and widely tested on standard computer; i.e, devices with a reliable timer, a high CPU speed and talking on various abstraction layers. Therefore, it is not always possible to import these libraries as-they-are into wearables and embedded technologies due to the fact that most of them do not posses an OS, a proper timer and need specific hardware optimisations.

Moreover, an additional danger of emerging fields, more specific to wearable devices, concerns the risk's awareness. The continuous use of wearables involves a variety of privacy concerns that might not be understood yet and, since the usage of these devices is relatively recent, users are not aware of the potential privacy implications of their wearables [33].

1.2 Examples of Privacy Enhancing Embedded Devices

This section continues the literature review by presenting some examples of embedded devices focussing on privacy technologies.

One of the most cited paper concerning anonymity on embedded devices is referenced in [36]. The authors present a Java implementation of the DAA anonymous authentication protocols for smart cards [7] . One of the main aspects of this paper is that it understands and high-lines the problematic of the very limited resources of embedded devices, even more applicable when considering the inherently inefficiency execution of Java Card. Therefore, mathematically complex cryptographic protocols like the DAA scheme would be far too slow for practical use. However, the paper managed to identify performance bottlenecks and comes with workarounds that allow to obtain a reasonably fast implementation. The main point of this paper is its approach: it consists in taking advantage of the embedded functionalities of a simplified Java virtual machine and of the tamper resistance of the card's code in order to speed up the system without cutting off on security. In conclusion, their smart cards needs about 4.2 seconds to achieve a complete authentication (on a card running the version 2.2.1 of the Java Card standard). However, since their bottleneck is due to modular multiplications and exponentiations, their system could be further improved by using Elliptic Curve Cryptography (ECC), as discussed in section **??** of this thesis.

Around the same topic, some researchers from Brown University discuss how to bring Electronic cash schemes (e-cash) with all their privacy and physical benefits on constrained devices such as a MetroCard by investigating the well-known Pay-As-You-Go (PAYG) system [2, 20, 60]. Therefore, their biggest challenge comes from the very thigh processing time constraints of transportation payments as well as from the necessity to keep the hardware as cheap as possible due to the high system's volume and the need to replace cards frequently. The PAYG system has been of great inspiration for this master project since the scope of the authors is similar to this thesis; i.e., bridge the gap between theoretical cryptographic constructions and practical implementations on embedded devices. Many version of the PAYG system are available today [60] and combines ECC with efficient and provably secure constructions of blind signatures with attributes, in order to achieve authentications in a few seconds on relatively cheap hardware.

1.3 Fundamental Concepts

In this section, the fundamentals of the realised system are inspected. The scope of this section is to provide the reader with all the background concepts needed for a good understanding of this thesis.

1.3.1 Fundamental Concepts - Elliptic Curve Cryptography

Elliptic curves are more and more used in cryptography [22, 30]. Indeed, for a given level of security, it requires shorter encryption keys than traditional systems. This allows a great spare of resources [25]. The points on the elliptic curves are all those who satisfy Eq.(1.1).

$$\begin{aligned} \mathcal{E} \;&=\; \{(x,y) \mid y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\} \\ &\quad \text{with} \quad a,b \in \mathbb{F}_p \mid a^3 + 27b^2 \neq 0 \end{aligned} \tag{1.1}$$

where $\mathbb{F}_p$ is a finite field of prime order p and $\mathcal{O}$ is the point at infinity [16, 29]. Three main operations are defined for elliptic curves' points. The first, the *point addition*, defines how to add two points. The second, the *point doubling*, defines how to double a point (add a point to itself), and the third, the *point multiplication*, defines how to multiply a point by a scalar [22, 16, 25].

The security of ECC depends on the fact that, given two points P and Q on the curve $\mathcal{E}$ such that $Q = kP$ ($\forall k$ scalar), it is computationally infeasible to find k (if it is sufficiently large) [35, 16]. The main concept behind this is the use of the so-called *one-way functions*. A one-way function is a function for which it is relatively easy to compute the image of some elements in the domain but it is extremely difficult to reverse this process and determine the original element solely based on the given image [21]. In the case of ECC, the one-way function is the fact that multiplying k and P is easy, but recomputing k from P and $Q = kP$ is computationally infeasible.

1.3.2 Fundamental Concepts - Anonymous Credentials Protocol

This subsection aims to quickly summarise the essentials of the anonymous credentials protocol, called *Algebraic MACs and Keyed-Verification Anonymous Credentials*, described in [13]. This protocol focusses on the problem of constructing anonymous credentials in situations where the issuer of the credentials is also the verifier.

The goal of this protocol is to set up a system where users can be known in different environments under different names, but their behaviour under these names should remain unlinkable. In other words, no one should be able to recognise that two different names were actually used by the same person, yet the users should be able to prove possession of a credential issued to one given name to any other user, without revealing that name.

The following equations rewrites the original algorithms using ECC, as introduced in the previous paragraph. For the rest of this thesis, vectors are written in bold and ECC points are uppercase letters.

The $\mathbf{MAC_{GGM}}$ Message Authentication Code

The papers introduces a new Message Authentication Code (MAC) in prime-order groups[1], called MAC_{GGM}. First of all, this MAC considers a list of n messages $\boldsymbol{m} = (m_1, \ldots, m_n) \in \mathbb{F}_p^n$ known to both the issuer and the user. Then, the MAC can be defined with the following four algorithms.

- $\mathbf{Setup_{GGM}(1^k)}$ **:**
  1. choose a cryptographically secure k-bit prime number p
  2. select a cryptographically secure elliptic curve $\mathcal{E}$
  3. choose two points H and G from $\mathcal{E}\backslash\{\mathcal{O}\}$
  4. output $(\mathcal{E}, p, G, H)$
- $\mathbf{KeyGen_{GGM}}(params)$ **:**
  1. choose a secret key $s_k = \boldsymbol{x} \in \mathbb{F}_p^{n+1}$
  2. compute $X_1 = x_1 H, \ldots, X_n = x_n H$
  3. output the list $(X_1, \ldots, X_n)$
- $\mathbf{MAC_{GGM}}(sk, \boldsymbol{m})$ **:**
  1. choose a point $U \in \mathcal{E}\backslash\{\mathcal{O}\}$

[1]Actually, the paper presents two MACs, but only the MAC used in the rest of this work is presented here.

2. compute $U' = \mathfrak{H}_x(\boldsymbol{m})U$, where $\mathfrak{H}_x(\boldsymbol{m}) = x_0 + \sum_i^n x_i m_i$
3. output the tag $\sigma = (U, U')$

- **Verify$_{\textbf{GGM}}$**$(s_k, \boldsymbol{m}, \sigma)$ **:**

1. parse $\sigma = (U, U')$ as $U \in \mathcal{E}$ and $U' \in \mathcal{E}$
2. accept if $U \neq \mathcal{O}$ and $U' = \mathfrak{H}_x(\boldsymbol{m})U$

This MAC is the fundamental on which the credentials' issuance and the credentials' presentation algorithms are based.

Credential's Issuance

The credentials' issuance over the attributes $\boldsymbol{m}$ is based on the following two algorithms, given the construction of a MAC_{GGM} as described above.

- **Setup(1^k) :**

1. output $(\mathcal{E}, p, G, H) \leftarrow \mathsf{Setup}_{\mathsf{GGM}}(1^k)$

- **CredKeyGen$(params)$:**

1. parse $params$ as $(\mathcal{E}, p, G, H)$
2. compute the MAC keys as $(\boldsymbol{X}, \boldsymbol{x}) \leftarrow \mathsf{KeyGen}_{\mathsf{GGM}}(params)$
3. pick $\widetilde{x_0} \in \mathbb{F}_p$
4. commit to the secret x_0 by forming $C_{x_0} = x_0 G + \widetilde{x_0} H$
5. output $(C_{x_0}, \boldsymbol{X})$ and $s_k = (\boldsymbol{x}, \widetilde{x_0})$

After this two algorithms have been successfully run, the issuance finishes by outputting

$$(U, U') \leftarrow \mathsf{MAC}_{\mathsf{GGM}}(s_k, \boldsymbol{m})$$

along with the proof π_1 that certifies that (U, U') is a valid MAC with respect to the system's and issuer's parameters.

$$\pi_1 \;:=\; PK\left\{(\boldsymbol{x}, \widetilde{x_0}) : U' = x_0 U + \sum_{i=1}^{n} x_i(m_i U) \quad \wedge \quad C_{x_0} = x_0 G + \widetilde{x_0} H \quad \wedge \quad X_i = x_i H\right\}$$
$$\forall i \in \{1, \dots, n\}$$

The complete development and details of the proof of knowledge π_1 can be found in appendix B.1.

Credential's Presentation

Similarly to the previous paragraph, the credentials' presentation is based on the two following algorithms:

- **Show**$(params, C_{x_0}, \boldsymbol{X}, \phi, cred, \boldsymbol{m})$ **:**

  1. choose $(r, z_1, \dots, z_n) \in \mathbb{F}_p^{n+1}$
  2. parse cred $= (U, U')$
  3. compute $C_{U'} = U' + rG$ and $C_{m_i} = m_i U + z_i H \quad \forall i \in \{1, \dots, n\}$
  4. send $\sigma = (\boldsymbol{C_m}, C_{U'})$ and a proof of knowledge π_2 as follow:

$$\pi_2 \;:=\; PK\left\{(\boldsymbol{m}, \boldsymbol{z}, -r) : \phi(\boldsymbol{m}) = 1 \quad \wedge \quad C_{m_i} = m_i U + z_i H \quad \wedge \quad V = \left(\sum_{i=1}^{n} z_i X_i\right) - rG\right\}$$
$$\forall i \in \{1, \dots, n\}$$

- **ShowVerify**$(params, \boldsymbol{X}, C_{x_0}, \phi, \boldsymbol{x}, \sigma, \pi_2)$ **:**

  1. parse $\sigma = (\boldsymbol{C_m}, C_{U'})$
  2. compute $V = (x_0 U + \sum_{i=1}^{n} x_i C_{m_i}) \,/\, (C_{U'})$
  3. verify π_2 using V (see appendix B.2 for details)
  4. output $\boldsymbol{C_m}$ if the proof is valid, $\perp$ otherwise

The complete development and details of π_2 can be found in appendix B.2. Note that this protocol loses its anonymity properties if a dishonest issuer delivers to each users credentials associated to a different MAC key. Indeed, the issuer (who is also the verifier) will then be able to distinguish users during the credentials' showing process by observing which key verifies the associated MAC.

1.3.3 Fundamental Concepts - Set Membership and Proof of Range

Researchers from IBM and EPFL published a paper considering the problem of proving in zero-knowledge that a value σ belongs to a set Φ, given a commitment on σ. This problem is also denoted by *the set membership proof* [9]. The proof is based on Boneh-Boyen signatures [5] and works under the q-Strong Diffie-Helmann assumption [6], where q is the number of elements in Φ. The communication complexity of this protocol depends on on q. The following algorithms summarise the set membership and range proofs protocol in ECC.

Firstly, the algorithms $\mathsf{Param}_{\mathsf{public}}$ and $\mathsf{Param}_{\mathsf{prover}}$ generate respectively the public and the prover's parameters.

- **$\mathbf{Param_{public}}(\mathbf{1^k})$:**

  1. select $(p, \mathcal{E}, H, G)$ as described in the first paragraph of section 1.3.2
  2. consider a set Φ of q elements
  3. output $(\mathcal{E}, p, G, H, \Phi)$

- **$\mathbf{Param_{prover}}(\mathcal{E}, p, G, H, \Phi, \sigma)$:**

  1. parse $(\mathcal{E}, p, G, H, \Phi, \sigma)$ and verify $\sigma \in \Phi$
  2. pick $r \in \mathbb{F}_p$
  3. commit to the attribute σ by forming $C = \sigma G + rH$
  4. output C

Then, the exchanges below between the verifier $\mathcal{V}$ and the prover $\mathcal{P}$ complete the proof.

- **Exchange 1. $\mathcal{V} \to \mathcal{P}$:**

  1. pick $x \in \mathbb{F}_p$
  2. compute $Y = xG$ and $A_i = (x+i)^{-1}G \quad \forall i \in \Phi$
  3. send $(Y, \boldsymbol{A})$

- **Exchange 2. $\mathcal{P} \to \mathcal{V}$:**

1. pick $v \in \mathbb{F}_p$
2. compute $V = (A_\sigma)^v$
3. send V

At that point, the prover and the verifier run the proof π below (the complete development of this proof can be found in [9]).

$$\pi := PK\left\{(\sigma, r, v) : C = \sigma G + rH \quad \wedge \quad V = v(x+\sigma)^{-1}G\right\}$$

1.3.4 Fundamental Concepts - RF Positioning System

This last paragraph discusses the principles of RF positioning systems. As illustrated by Fig.(1.1), RF positioning systems use beacons as set-points and the users localise themselves with respect to them. Basically, each beacon emits a short-range signal with its name and when the user detects it, he knows he is close to that beacon and localises himself within an internal map. Beacon-based systems have many advantages: cost-effectiveness, unremarkable hardware, flexibility to integration into existing infrastructures, work where other positioning techniques do not have signals (e.g., in the basements), do not require much user-side computational power to determine the location, and finally, their precision is arbitrarily increasable by adding additional beacons [88, 32].

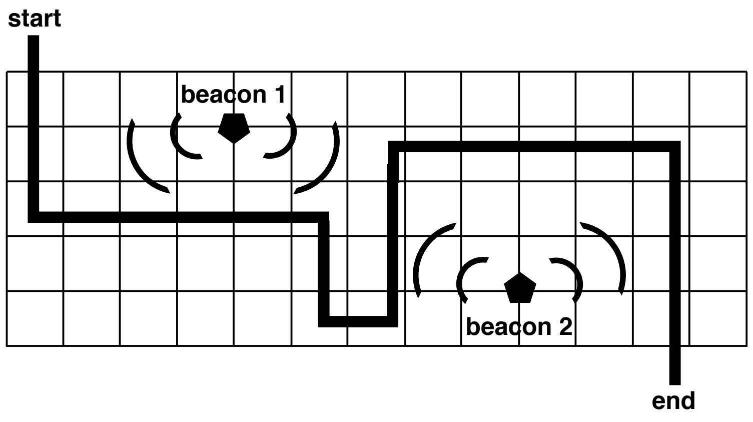


Figure 1.1: Sketch of RF positioning system

However, this very simple design comes with many security flaws. For instance, the system should incorporate a protection against an adversary trying to replicate a beacon's signal to confuse a user. For that reason, simply emitting the beacon's name is not enough and more advanced schemes based on digital signatures as discussed in sections 3.1 and 3.2 should be used.

Chapter 2

System's Architecture

This chapter starts by explaining the system's architecture by focussing on the role of each entity and the kind of connections between them. Then, it ends with the system's security analysis.

2.1 Architecture's Presentation

The complete system's architecture can be modelled as depicted in Fig.(2.1). Roughly speaking, the server starts by issuing some anonymous credentials granting a privilege level j to a wearable.

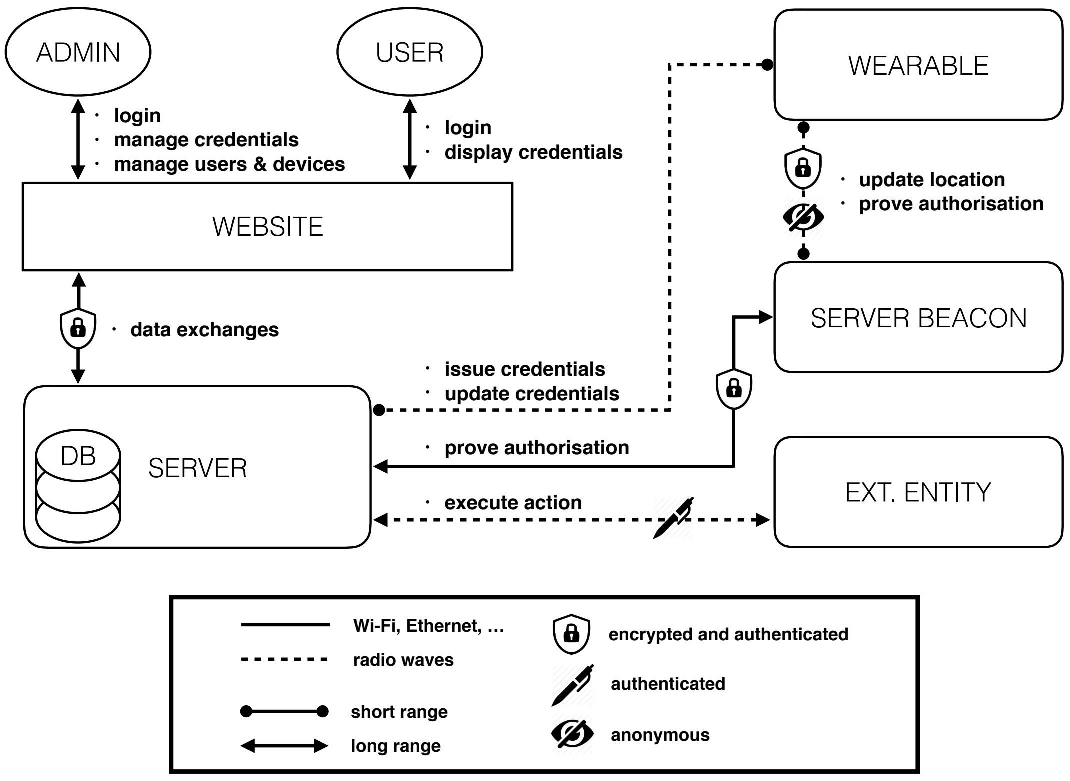


Figure 2.1: Overview of the complete system's architecture

Then, each time this wearable reaches some particular physical location (gets close to a localisation beacon) where it desires to perform an action requiring a privilege level of j, it starts presenting its credentials in order to ask the server the execution of that particular action. In fact, the attributes of the credentials represent the privilege level of the wearable's user. More specifically, a privilege level j is determined by a set Φ_j, such that if the user possesses an attributes $m_i \in \Phi_j$, then he has a privilege level of j. Therefore, this user can ask the server to execute an action requiring such privileges.

The credentials' showing protocol comes from [13] and operates as described in section 1.3.2. In general, the aMAC's function $\phi(\boldsymbol{m})$ is devoted to prove some statements over the attributes and is here used to perform a set of membership proof as described in section 1.3.3. In other words, the wearable proves to the server that it possesses credentials over some attributes included in the set Φ_j (without revealing them), and that these credentials have been previously issued by the server itself. Note that the system preserves anonymity only if many users are involved (for each privilege level), but this is a classic requirement of anonymous systems. As a supplement, appendix H presents a discussion around possible revocation mechanisms for the currently implemented design.

A concrete example of utilisation of this systems could be associating the external entity to an automatic office door. Therefore, each employee working on that office possesses a wearable with the credentials to access it. Subsequently, when an authorised employee steps close to the door, the device automatically detects its position, knows it is close to the door, and correctly presents the credentials to the server. Then, after credentials' verification, the server opens the door. Note the usability advantage of having such common operations executed automatically and based on the location; i.e, the user does not even need to present a smart card since the wearable reacts by its own.

2.1.1 Architecture's Presentation - Web Administration

A web interface is used to manage and display the device's functions. From that interface, each user and admin access the system. The admin's role consists in granting credentials to the users and in managing the devices' assignments, while standard users are only able to observe the credentials they own. The website is securely connected to the main server (e.g., using SSL) which possesses a database storing all the needed information; e.g., the list of users and their associated device, the device's credentials, etc. A presentation of this web interface can be found in appendix C.

2.1.2 Architecture's Presentation - Credentials' Issuance

During the setup phase, the server issues the credentials to a selected device according to the algorithms presented in section 1.3.2. The credentials' issuance is a short-range process. In fact, the wearable needs to be physically close to the server to allow the admins to physically verify, once and for all, the identity of the wearables' users. In order to improve security and battery life, the wearable only communicates using extremely low-power and short-range radio waves (dotted line on Fig.(2.1)).

2.1.3 Architecture's Presentation - Credentials' Showing

The server beacons can be seen as continuities of the main server and have essentially two roles: the first is to operate as an interface between the wearables and the server, and the second is to act as the RF localisation system explained in section 1.3.4.

When approaching these beacons, the wearable receives a location's update. To that purpose, it starts an unilateral authentication mechanism defined by the ISO 9798-3 standard where only the beacon is authenticated [38]. This process allows the wearable to accepts only authenticated location's code and to negotiate a symmetric key with the beacon without compromising its anonymity. At that point, if the user possesses some credentials allowing him to perform a particular action at that location, the device automatically presents these credentials in order to anonymously prove to the server that it has the required privilege level. The wearable does not talk directly to the server but uses the beacon as interface instead. Then, it starts the credentials' showing protocol by sending all the needed information encrypted under the derived shared key (for instance, using classic block ciphers like AES in CBC mode [18]). The beacon receives these information and decrypts them.

Since both the server and the beacon are under the admins' control, they can easily share a symmetric key. The beacon uses this key to forward the received credentials' data directly to the server using an Authenticated Encryption with Associated Data (AEAD) scheme, like GCM-AES [28]. In fact, these AEAD schemes can be implemented extremely efficiently on industrial hardware [26]. This communication does not need to be performed on radio frequencies. Moreover, this beacon-server symmetric key can also be useful for completely different purposes like beacon's code updates.

2.1.4 Architecture's Presentation - Action's Execution

Finally, once the credentials have been successfully verified by the server, the server issues a signed request to an external entity (which can be, for instance, an automatic door, an alarm system or any compatible IoT entity) to perform the requested action. This last interaction between the server and the external entity is signed along with a fresh random nonce with the server's public key.

2.2 Security Analysis

This section is firstly devoted to the Security Policy and Threat Model, which are key elements in the statement of security. Then, the security arguments put in place to ensure the policy remains unviolated are summarised along the list of the elements in the Trusted Computing Base (TCB).

2.2.1 Security Analysis - Security Policy

When considering building any secure system, one of the most important questions is about what should be protected. The concept of Security Policy can be summarised as a statement setting up the security objectives of a system and specifying the assets [1, 19]. Nevertheless, due to its vast versatility, writing down the Security Policy is a step requiring particular attention and should be specifically tailored for each system. Tab.(2.1) defines the Security Policy of this system.

Principals	• The wearable's users; • The admins of the main server.
Assets	• The wearable device;

	• The main server; • The localisation beacons (composing the positioning system); • The credentials embedded in the wearables; • The external entity performing the commanded actions; • The communication channels (frequency bands).
Policy	• No one can tamper with the wearable's code; • Only authorised admins access the server; • External entities accept only authorised commands from the server; • Wearables accept location's updates only from authentic beacons; • Users anonymity (credentials showing unlinkability) is ensured with respect to the server, as intended in section 1.3.2.

Table 2.1: System's Security Policy

2.2.2 Security Analysis - Threat Model

The Threat Model defines all the resources and capabilities the adversary can have; i.e., the parts of the system he can observe, tamper, as well as the parties he can corrupt. To make a long story short, the Threat Model describes what the adversary can or cannot do. More specifically, applying this concept to the current system, the Threat Model is depicted in Tab.(2.2) below.

Adversaries	• Insiders (or intruders) trying to make the external entity to perform an action, without having the suitable credentials; • Server's admins trying to invade user's privacy by observing their position or the actions their wearable ask to perform.

Capabilities	• The adversary could try to tamper with the wearable's code; • The adversary could try to access the server without authorisation; • The adversary could try to impersonate the server to activate and external entity without authorisations; • The adversary could try to move the localisation beacons or to impersonate it; • The server's admins may identify which user is presenting the credentials (privacy invasion); • The adversary could try to tamper or copy the beacon-server communication; • The adversary could try to tamper or copy the communication between the beacon and the wearable.
Threats to Assets	• Unauthorised activation of an external entity; • The server's admins could try to violate user's privacy (as intended in section 1.3.2).

Table 2.2: System's Threat Model

In addition to the above elements, some examples of more sophisticated attacks against the system are described in appendix G along with some possible mitigations.

2.2.3 Security Analysis - Security Arguments

The Security Arguments are rigorous arguments that the mechanisms put in place to ensure the Security Policy is not violated are indeed effective [19]. Here, these mechanisms are listed below:

- The little size of the embedded device's memories prevents attackers from extracting its source code and to tampering with it;

- A proper login system with secured connection and database as depicted in Fig.(2.1) of section 2.1 is realised to ensure only authorised persons access the server;
- The external entities accept only requests signed by the main server in order to prevent server's impersonation;
- To avoid beacon's impersonation, wearables accepts only signed location updates and the beacons are strongly anchored in the walls to prevent attackers from moving them;
- The implementation of the protocol depicted in [13] ensures the privacy requirements of the system;
- The beacon-server communication is authenticated and encrypted to prevent sniffing and tampering;
- The very short range of the wearable-beacon communication make difficult for an attacker to tamper or sniff it. However, to enforce even more this statement, the beacon-wearable communication is encrypted.

2.2.4 Security Analysis - Trusted Computing Base

The elements of the Trusted Computing Base (TCB) are the foundations on which the Security Policy relies on. If something goes wrong in the TCB, the policy may be violated [19]. In the case, these elements are listed here below.

- The wearable's user, in the sense that he is trusted to no give his wearable to someone else;
- The physical position of the localisation beacons, in the sense that they are placed at the correct locations;
- The server's admins during the credentials' issuance, in the sense that they are trusted to not issue credentials to non-authorised wearables and to use the same aMAC key for each wearable (see section 1.3.2).

Chapter 3

Prototyped System

Despite all the blocks and interactions described in section 2.1 are essential to ensure a fully industrial system, the prototype had to be realised with some simplifications due to hardware limitations. More specifically, the prototype's architecture becomes as shown in Fig.(3.1) below, instead of the original one pictured in Fig.(2.1) of section 2.1.

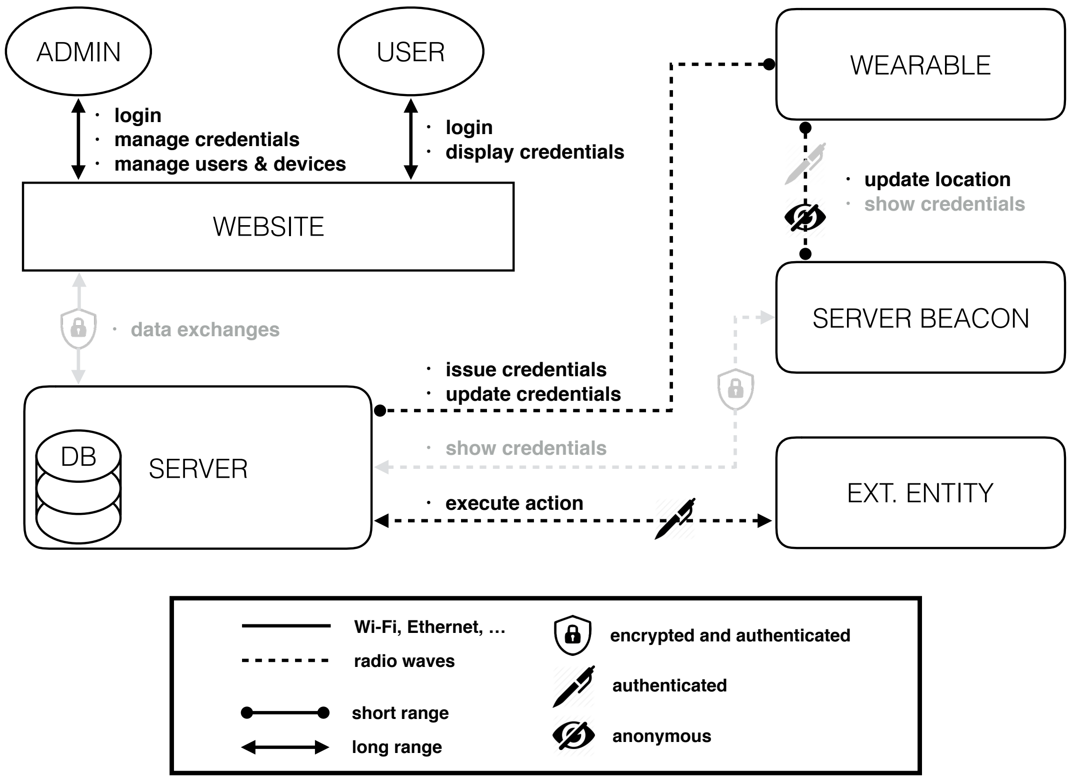


Figure 3.1: Overview of the prototyped system's architecture

The first difference concerns the wearable-beacon communication. Indeed, it is computationally to heavy to perform all the protocols described in section 2.1. Therefore, instead of deriving a shared key and encrypting the credentials' showing data, the wearable sends clear data but still accepts only signed location's updates. Moreover, the server-beacon is also left unencrypted and the web interface could not be tested on the system's server. Finally, the prototype communicates entirely through 2.4GHz radio waves.

3.1 Prototype's Conceptual Model

The prototype's architecture is firstly explain in general in order to provide the reader with a good overview of all the data transfers between each entities. Then, a more deep explanation through complete block diagrams illustrates the ideal prototype's behaviour.

3.1.1 Prototype's Conceptual Model - Communication's Overview

The realised prototype's interactions can be depicted as shown in Fig.(3.2). It is composed of

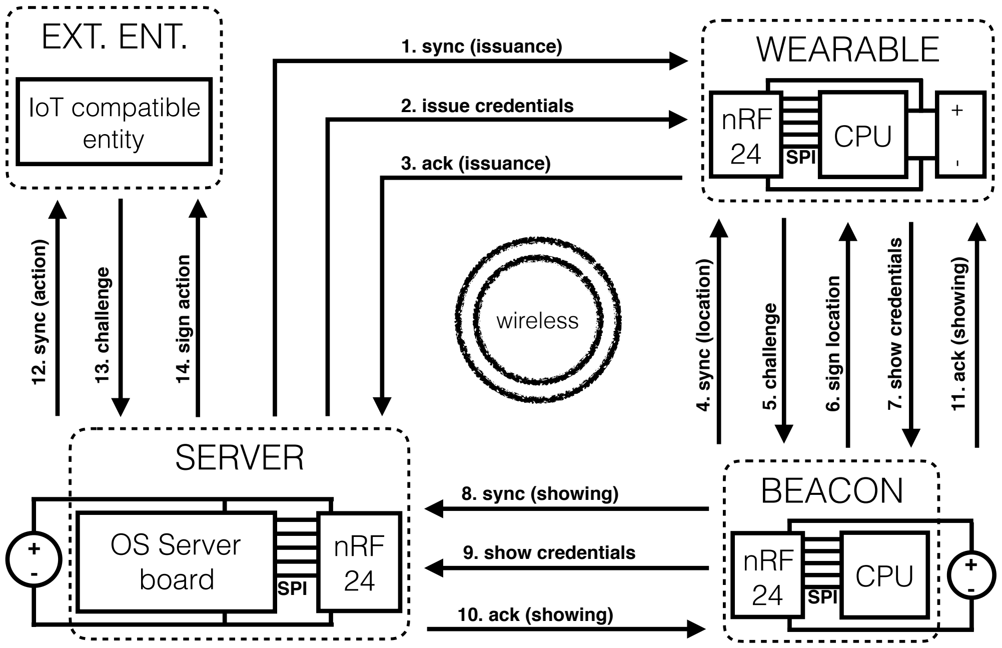


Figure 3.2: Prototype's communications overview

four main parts: the main server, the wearable devices, the positioning system (beacons), and the external entities. First of all, the server sends a sync packet to the suitable wearable in order to start the credentials' issuance phase. Therefore, the wearable gets automatically ready for receiving the new credentials. At that point, the server proceeds with the credential's issuance and the wearable verifies the issuance process according to the algorithm described in section 1.3.2. The wearable sends then a final packet acknowledging whether the new credentials have been accepted or not.

Afterwards, once the wearable approaches a localisation beacon, it receives a *location-sync packet* and the acquisition of a new location is negotiated: the wearable challenges the beacon with a short-range fresh random number and waits for the reception of a signed string made of the hash of the concatenation of the challenge and the beacon's location code area (which can be seen as the beacon's name). If the signature is valid, the new location is accepted and the device updates its current position.

If the wearable possesses any credentials that could be used at the new location, it enters in the credentials' showing phase and provides the beacon with all information needed to present the credentials to the server. As explained before, the beacon acts like a bridge between the server and the wearable. The first step of this phase is to send a *showing-sync packet* to the server. Then, the wearable applies the credentials' showing protocol, and finally, the server sends back an acknowledgement packet, through the beacon, telling the wearable whether or not the credentials have been correctly verified.

The last interaction is then performed between the server and the external entity. If the credentials shown by the wearable where successfully verified, the server sends a signed request to the entity demanding it to perform a specific action (this last step is very similar to the beacon-wearable location update).

The following paragraphs explain in details the behaviour of each block by focussing on the communications between them.

3.1.2 Prototype's Conceptual Model - Server and Wearable Communication

The communication between the server and the wearable device is illustrated in Fig.(3.3). As explained above, the server-wearable communication has essentially two roles: the credentials' issuance and the credentials' showing. However the second operation is performed though the beacon and is explained in the next paragraph.

For the credentials' issuance, the server starts by transmitting a sync packet made of seven bytes. The first byte indicates the kind of sync (in this case, it indicates the start of the credentials' issuance) and contains the hexadecimal value 0x10, while the other six bytes contain the server's address (arrow 1.).

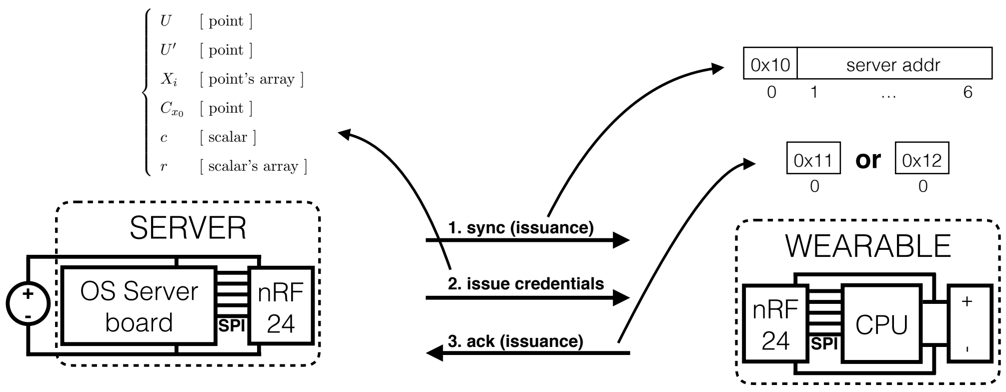


Figure 3.3: Server and wearable communication

Then, the server sends all the information needed to communicate the credentials in zero knowledge, as shown by Eq.(3.1) below (arrow 2.).

$$\begin{cases} U & [\text{ point }] \\ U' & [\text{ point }] \\ X_i & [\text{ point's array }] \\ C_{x_0} & [\text{ point }] \\ c & [\text{ scalar }] \\ r & [\text{ scalar's array }] \end{cases} \tag{3.1}$$

The parameters U, U', X_i and C_{x_0} are exactly those referenced in section 1.3.2 and come from [13], while c and r represent respectively the challenge of the responses of the associated Non-Interactive Zero-Knowledge (NIZK) proof. Finally, the wearable verifies the received proof, and if the verification is successful it acknowledges the server by sending back a 1-byte packet containing the value 0x11. On the contrary, if the verification is unsuccessful, an ack packet containing 0x12 is sent back (arrow 3.).

3.1.3 Prototype's Conceptual Model - Wearable and Beacon Communication

This paragraph is devoted to the communication between the wearable device and the localisation beacons composing the RF positioning system. As explained in section 1.3.4, the localisation beacons are continuously transmitting sync packets. Fig.(3.4) shows these 7-byte packets are composed of an indicator packet (0x16 indicates the location-sync) followed by the beacon's address (arrow 4.).

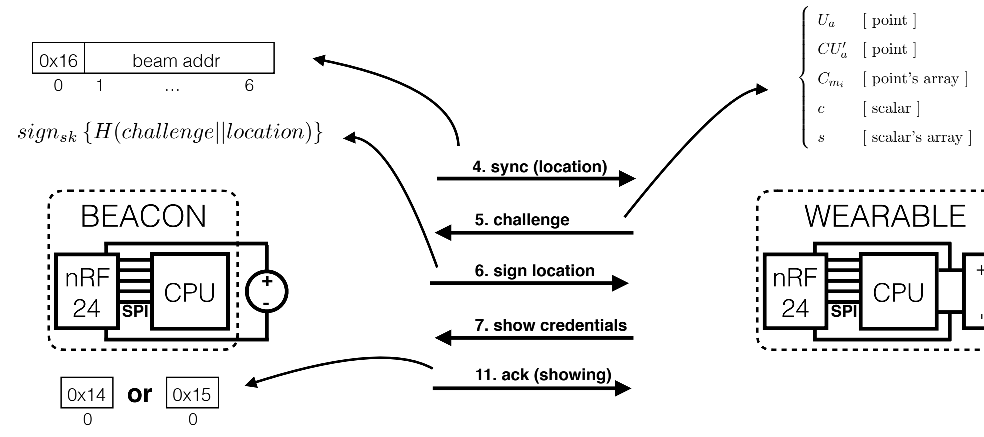


Figure 3.4: Wearable and beacon communication

When the wearable gets one of these packets[1], it answers with a fresh low-power random challenge (arrow 5.). Finally, when the beacon receives the challenge it responds with the Digital Signature Algorithm (DSA) signature represented in Eq.(3.2).

$$sign_{sk}\{\mathcal{H}(challenge \mid\mid location)\} \tag{3.2}$$

This signed string results from the hash (SHA-256) of the concatenation of the wearable's challenge and the beacon's location code, which is used by the wearable to determine its current location (arrow 6.).

If the signature is successfully verified and the wearable possesses any credentials applicable at this specific location, it starts the credentials' showing phase and provides the beacon with all the needed information to transmit to the server (arrow 7.). Afterward, the beacon forwards an ack packet from the server to wearable (arrow 11.). These lasts steps are further explained in the next paragraph.

[1]Note that each wearable shares the same RX address in order to preserve their anonymity.

3.1.4 Prototype's Conceptual Model - Server and Beacon Communication

This operation is analogue to the credentials's issuance but is initiated by the wearable when it decides to show its credentials. Indeed, this process also starts with a 7-byte sync packet, where the first byte is 0x13 (indicating the beginning of the credentials' showing operation) and the other six are the wearable's address (arrow 8.).

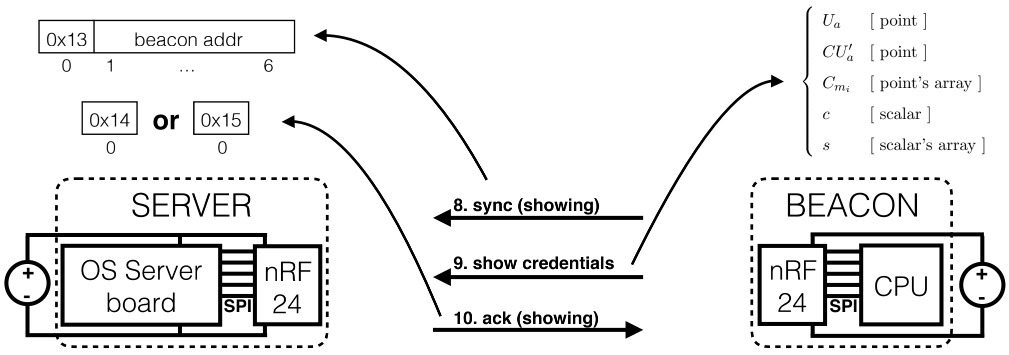


Figure 3.5: Server and beacon communication

Then, the following data are transmitted to the server, through the beacon (arrow 9.):

$$\begin{cases} U_a & [\text{ point }] \\ CU'_a & [\text{ point }] \\ C_{m_i} & [\text{ point's array }] \\ c & [\text{ scalar }] \\ s & [\text{ scalar's array }] \end{cases} \tag{3.3}$$

The parameters U_a, CU'_a and C_{m_i} are exactly those referenced in section 1.3.2 and come from [13], while c and s represent the challenge and the responses of the associated NIZK proof. Finally, the server acknowledges the reception by sending back respectively the byte 0x14 or 0x15, depending on the credentials' validity (arrow 10.). This ack packet is directly forwarded to the wearable.

3.1.5 Prototype's Conceptual Model - Server and Ext. Entity Communication

The last communication to analyse is the one between the server and the external entity. Once the wearable has successfully presented his credentials to the server in order to ask him to perform a specific action, the server is in charge of ordering its execution. Indeed, all the external entities accept only signed orders coming from the main server. In that purpose, a communication scheme similar to the beacon-wearable dialog has been set up (see Fig.(3.6) below).

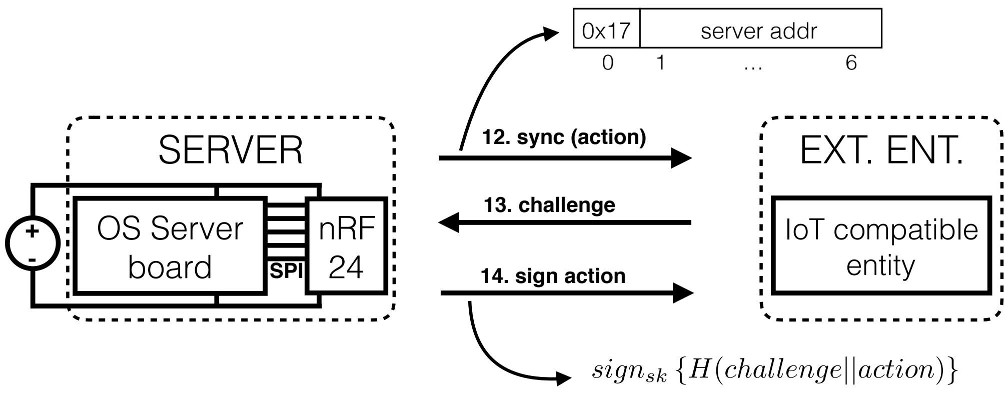


Figure 3.6: Server and external entity communicationl

First of all, the server issues the classic 7-byte sync packet with the first byte equal to 0x17 (arrow 12.). Then, the external entity answers with a fresh new random challenge (arrow 13.), and similarly to the previous case, the server transmits a signed request as described by Eq.(3.4) below.

$$sign_{sk}\{\mathcal{H}(challenge \mid\mid action)\} \tag{3.4}$$

The signed request if composed of the hash of the concatenation of the server's challenge and the action's identifier (arrow 14.).

3.2 Prototype's Implementation

This section is completely dedicated to the software and hardware implementation of the prototype. From the software perspective, this system is based on many different building blocks, some coming from standard and well-tested libraries while others had to be completely built from scratch. As indicated in the code files, the libraries come from [49, 48, 47]. Fig.(3.7) below provides an overview of the blocks composing the system. Each module considers 32-bit data type for prototyping purpose (which is not cryptographically secure), but section 3.4 explains how to address this problem.

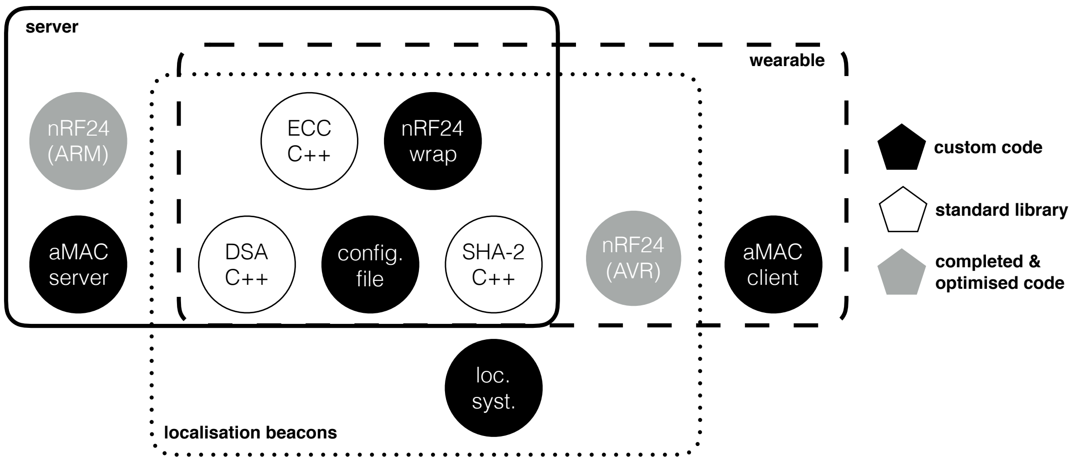


Figure 3.7: System's software building blocks

Moreover, each time randomness is needed, the software refers to a common function `get_random`, which has been implemented for testing purposes and is not cryptographically secure. The problematic of randomness is not part of this work but is further discussed in section 3.4.

First of all, the underlying drivers used by all the entities to communicate are presented, and then the block server, wearable and beacon are detailed. Appendix D presents an example of external entity's implementation. To facilitate the reading, Finite State Machines (FSM) are used to explain the software instead of displaying raw programming code.

3.2.1 Prototype's Implementation - Underlying Hardware Drivers

The server, the wearables and the localisation beacons possess a nRF24 module allowing them to communicate with other entities using 2.4GHz radio waves.

Hardware Design

The nRF24 module is based on the nRF24l01+ chip [43] which is a great trade off between performance, range, low power and cost (see section 3.3 and appendix A for further details). Fig.(3.8) displays a view of the module's hardware reference design. Ideally, the chip has to be powered at 3.3V, but it tolerates an input voltage from 1.9V to 3.6V. In order to communicate with the CPU, the module understands the standard Serial Peripheral Interface (SPI) and possesses an internal

FIFO to store data waiting to be transmitted. The SPI master output (slave input) and master input (slave output) respectively correspond to the PINs `MOSI` and `MISO`. Since the nRF24 module is a RF transceiver, it is able to either send data or receive them. To that purpose, the PIN `CE` is used to set the module in transmitter or receiver mode. The last two PINs, `CSN` and `SCK`, are respectively the standard chip-select-not and the synchronous clock input. The nRF24 module comes with four possible modes of operations, called 0x00, 0x01, 0x02 and 0x03. Tab.(3.1) summarises the main differences between these modes[2].

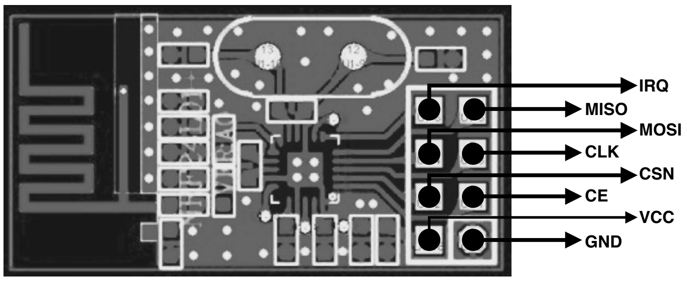


Figure 3.8: nRF24 module hardware reference design [52]

Modes	DC current	Data Rate	BER
0x03	11.3 mA	2 Mbps	$\leq 0.1\%$
0x02	9 mA	2 Mbps	1%
0x01	7.5 mA	1 Mbps	1%
0x00	7 mA	250 kbps	1%

Table 3.1: nRF24 module's characteristics summary [43]

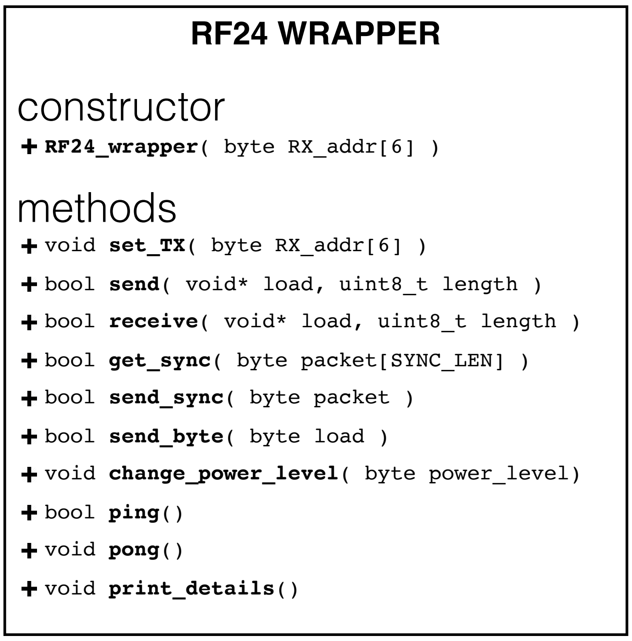
The second column indicates the chip's current consumption, the third indicates the theoretical data rate, and the last column provides the Bit Error Rate (BER). However, these characteristics

[2]Actually, in order to save even more power, the module has a special stand-by mode which has not been used in this project. More information about this mode can be found in [43].

are ideal and section 3.3 provides a more extensive analysis of the system's practical behaviour. In order to provide a design as flexible as possible, the desired mode of operation can be selected in the configuration files and all the rest is taken care of automatically.

Software Design

One of the most time-consuming parts of this thesis was about writing the drivers for the nRF24 module in order to make it compatible with both the ARM1176JZF-S CPU of the Raspberry Pi B+ and the ATMEGA328P CPU of the Arduino UNO. A great help in this task was the draft open source library found in [46], which has been completed and optimised in order to produce two drivers, one working on Linux (compilable by g++ through a custom Makefile) and one working on Arduino (compilable by the Arduino IDE [67]). At this point, in order to have a design as generic as possible and to encourage further usages, a single nRF24 driver wrapper has been built. The scope of this additional abstraction layer is to automatically select (through pre-processor macros) which driver has to be compiled, given the underlying hardware. Therefore, any developer can now create a nRF24 wrapper's object and benefit from the functions shown in Fig.(3.9) without having to worry about the implementation and the underlying hardware.


RF24 WRAPPER

constructor
```
+ RF24_wrapper( byte RX_addr[6] )
```
methods
```
+ void set_TX( byte RX_addr[6] )
+ bool send( void* load, uint8_t length )
+ bool receive( void* load, uint8_t length )
+ bool get_sync( byte packet[SYNC_LEN] )
+ bool send_sync( byte packet )
+ bool send_byte( byte load )
+ void change_power_level( byte power_level)
+ bool ping()
+ void pong()
+ void print_details()
```

Figure 3.9: RF24 driver wrapper outline

First of all, the driver is initialised with a fixed 6-byte RX address; i.e., the address used for reception. Afterwards, the user is free to call the method `set_TX` as many times as needed to set the destination address. The wrapper possesses a simple `send` function to transmit the desired data. To facilitate even further the wrapper's utilisation, this method accepts a pointer to any kind of data: the developer is then free to send a pointer to his own custom data structure. The only limitation is about the load length, which is limited to 32 bytes by the hardware (the maximum size of the internal nRF24l01+ FIFO is 32 bytes). In addition, this method includes an automatic Cycle Redundancy Check (CRC) and an *ack* functionality that allows the user to detect if his packet has correctly been received. The function `receive` works similarly to the previous one. In order to make these two methods as flexible as possible, the timeout values for reception (`RX_TIMEOUT`) and transmission (`TX_TIMEOUT`) are arbitrarily configurable through a pre-processor macro, whose default values are 1 s and 200 ms, respectively.

The method `get_sync` allows to automatically receive a sync packet of length `SYNC_LEN`. In the default configuration, `SYNC_LEN` has been set to seven bytes: one byte determining the kind of request and six bytes identifying the message's transmitter. Similarly, `send_sync` sends a sync request to a target device. Without surprise, the method `change_power_level` is used to change the transceiver mode (0x00, 0x01, 0x02 or 0x03) according to Tab.(3.1) above.

The last three methods, `ping`, `pong` and `print_details` have debugging purposes. Indeed, they are used to ping a target device and to display debugging information on the standard output.

3.2.2 Prototype's Implementation - Server's Hardware and Software

This subsection focusses on the server's implementation. As before, the hardware needed to build the server is discussed in the first paragraph and the software implementation in the second.

Hardware Design

Fig.(3.10) here below presents the full hardware design. The server is powered at 5V and implements the SPI interface with the nRF24 module from PINs 16, 20, 22, 24, 25 and 26.

The nRF24 module is powered by the Raspberry Pi through a lm1117-3.3 voltage regulator. In fact, as explained in section 3.2.1, the nRF24 has to be powered at 3.3V. However, the Raspberry Pi's power PIN 3.3V does not provide a current stable enough to reliably run the nRF24l01+ chip. For that reason, the voltage regulator is in charge of converting the 5V coming from the Raspberry Pi's PIN 1 to a suitable 3.3V output to feed the nRF24 module. For further stabilisation, three capacitors are designed to smooth the power signal as much as possible.

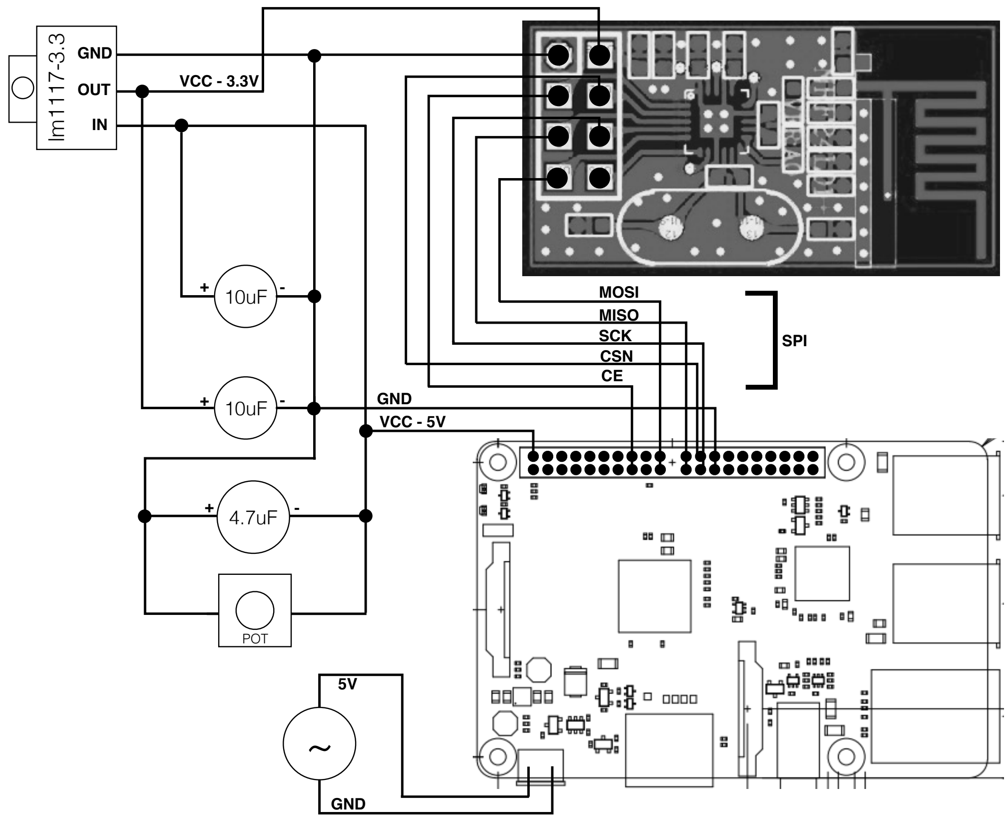
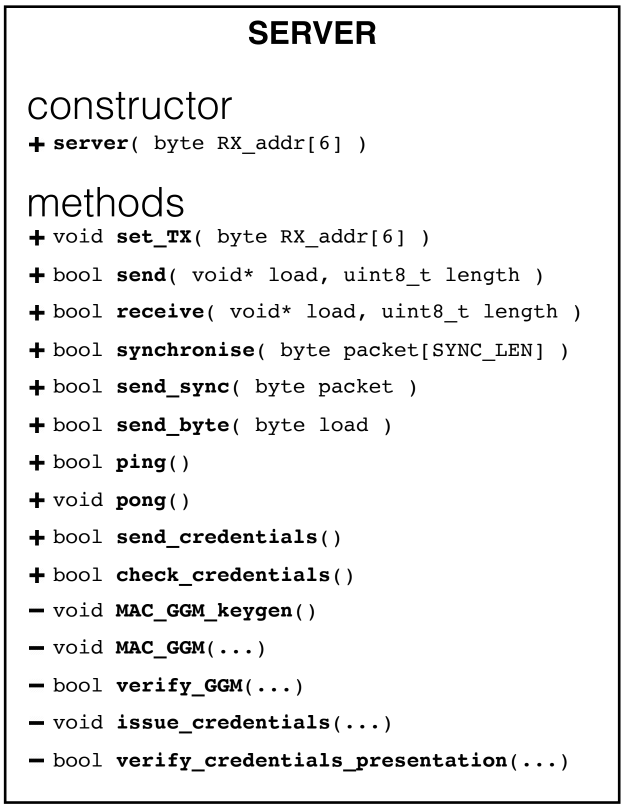


Figure 3.10: Server hardware reference design [58, 52]

It is of utmost importance to reduce the transmitting power of the server when issuing new credentials in order to give them only to a close targeted wearable, and not to all the devices around. Ideally, this interaction should have a range of a few centimetres. To that purpose, a potentiometer is used to regulate the server's transmitting power by limiting the current flowing through the regulator.

Software Design

Fig.(3.11) presents the methods available when a server object is created. With regard to the RF24 wrapper, the server is initialised with its 6-byte RX address and possesses the methods `set_TX`, `send`, `receive`, `send_sync`, `send_byte`, `ping` and `pong`. The method `synchronize` is a an extension of the RF24 wrapper's methods `get_sync` that automatically updates the TX address according to the address received in the last sync packet. The methods `send_credentials` and `check_credentials` rely on all other private functions to issue and verify the user's credentials. These methods implement the algorithms described in section 1.3.2.



SERVER

constructor

```
+ server( byte RX_addr[6] )
```

methods

```
+ void set_TX( byte RX_addr[6] )
+ bool send( void* load, uint8_t length )
+ bool receive( void* load, uint8_t length )
+ bool synchronise( byte packet[SYNC_LEN] )
+ bool send_sync( byte packet )
+ bool send_byte( byte load )
+ bool ping()
+ void pong()
+ bool send_credentials()
+ bool check_credentials()
- void MAC_GGM_keygen()
- void MAC_GGM(...)
- bool verify_GGM(...)
- void issue_credentials(...)
- bool verify_credentials_presentation(...)
```

Figure 3.11: Server software outline

The FSM shown in Fig.(3.12) completely illustrates the server's software behaviour. Once the server is turned on, it starts by initialising all its modules and peripherals. Then, it issues the targeted credentials and enters in a main loop. This loop runs forever and its purpose is to detect when the server needs to react. In fact, the main loop is waiting for the reception of a sync packet addressed to the server's RX address. Then, if the sync indicates a wearable is willing to show its credentials, the server verifies them and if the verification succeeds, it contacts the external entity as explained in section 3.1.4. Similarly, when the server is pinged, it answers with a *pong* packet.

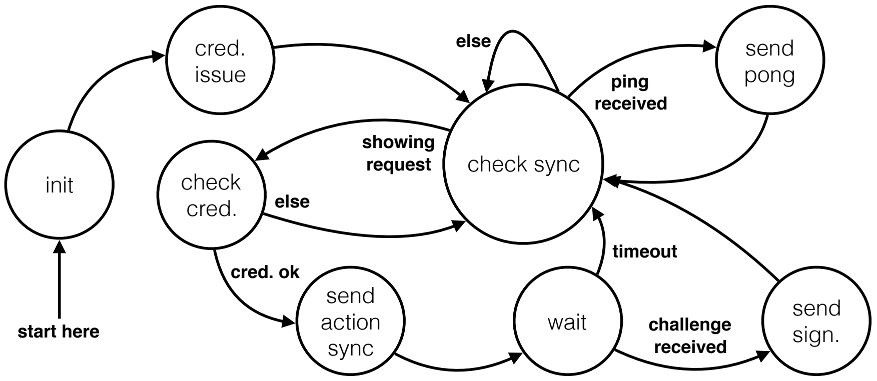


Figure 3.12: Server software FSM

3.2.3 Prototype's Implementation - Wearable's Hardware and Software

This section describes in details the complete wearable's hardware and software implementation.

Hardware Design

Similarly to the previous section, Fig.(3.13) below illustrates the wearable's reference hardware design. The SPI is interfaced with the Arduino through the PINs 7, 8, 11, 12 and 13. Exactly as for the server, the nRF24 is powered through a lm1117-3.3 voltage regulator. Note that the real-world wearable should store the issued credentials in a non-volatile memory (like an EEPROM or a flash memory) in order to not lose these data after each device's reboot. For the more enthusiastic readers, appendix I illustrates the final hardware design of the industrial wearable.

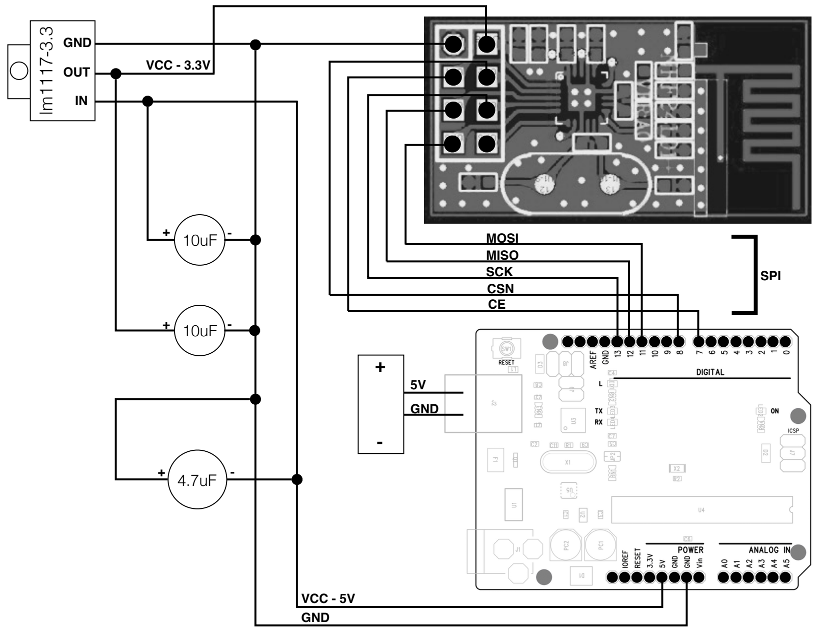


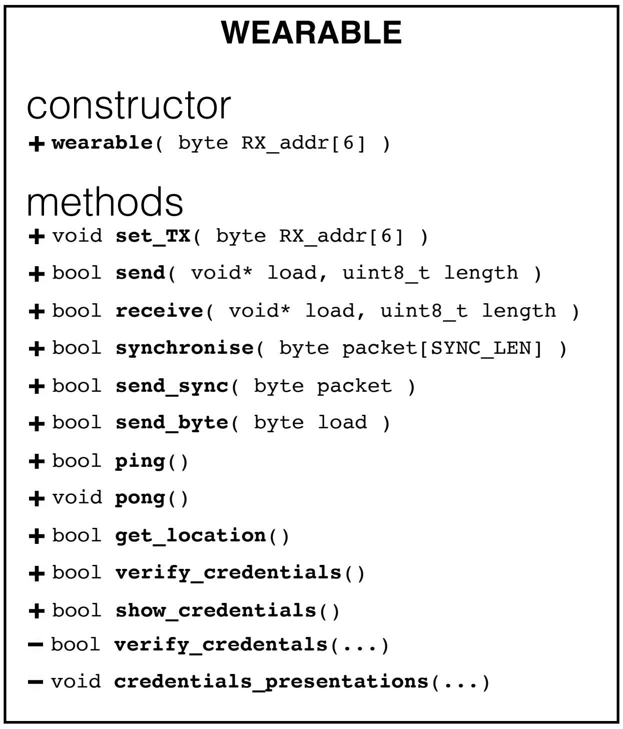
Figure 3.13: Wearable hardware reference design [56, 52]

Software Design

Fig.(3.14) presents the methods available when a wearable object is created. Most of them have already been discussed. Without surprise, the methods `get_location`, `get_credentials` and `show_credentials` rely on the other private functions to respectively verify the beacon's location, verify the validity of new credentials and show the credentials to the server. The underlying methods are the exact implementation of the protocols described in section 1.3.2. The wearable's method `synchronize` is implemented exactly as the server's one.

The FSM of Fig.(3.15) illustrates the wearable's behaviour. When the wearable is turned on, it starts by testing the connection with the main server by pinging it. Once a sync packet indicates the credentials' issuance process, the wearable leaves the main loop, gets the credentials and verifies them: if the verification succeeds, it tests immediately the new credentials. Similarly, when a new location is received, the wearable challenges the beacon as described in section 3.1.3, and if the location is legitimate, it starts the credentials' showing phase.

The wearable's software had to be massively optimised in order to fit the very limited Arduino's resources. For instance, one of the main optimisations was to play with the program and dynamic memory (see section 3.3.2).


WEARABLE

constructor

```
+ wearable( byte RX_addr[6] )
```

methods

```
+ void set_TX( byte RX_addr[6] )
+ bool send( void* load, uint8_t length )
+ bool receive( void* load, uint8_t length )
+ bool synchronise( byte packet[SYNC_LEN] )
+ bool send_sync( byte packet )
+ bool send_byte( byte load )
+ bool ping()
+ void pong()
+ bool get_location()
+ bool verify_credentials()
+ bool show_credentials()
− bool verify_credentals(...)
− void credentials_presentations(...)
```

Figure 3.14: Wearable software outline

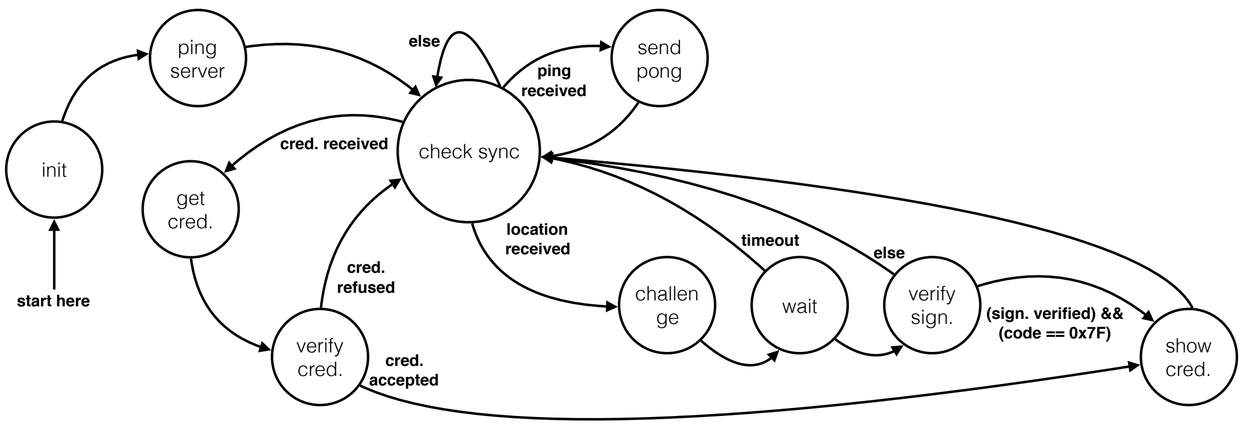


Figure 3.15: Wearable software FSM

3.2.4 Prototype's Implementation - Beacon's Hardware and Software

The last part of this section describes the beacons composing the localisation system. The beacon's hardware reference design is exactly the same as the wearable's design presented in Fig.(3.13).

Software Design

The beacon's software FSM is simpler than the previous one. Indeed, after the initialisation phase, the beacon enters in a main loop where it sends a location sync packet and then waits for a wearable to answer. If an answer is received, the beacon computes a DSA signature from the hash of the challenge and its location code, as explained in section 3.1.3. Afterwards, the FSM gets ready to play as interface between the wearable and the server for a possible credentials' showing process. If the wearable decides to presents its credentials, the beacon transfers them to the server and then waits for an ack to pass back to the wearable.

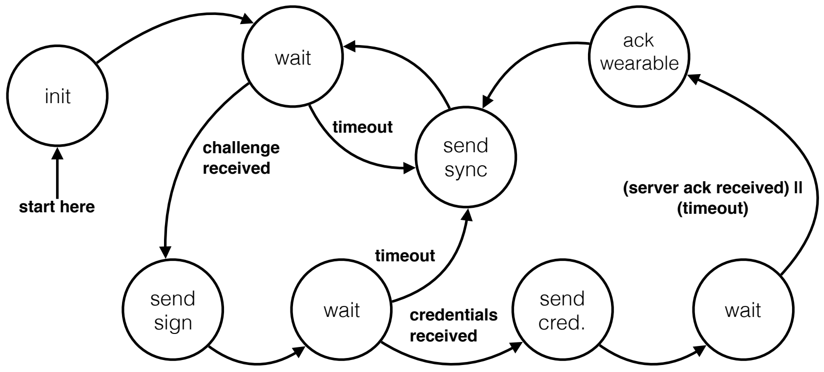


Figure 3.16: Beacon software FSM

3.3 Prototype's Resources and Performances

This subsection is devoted to the system's resources and performances. More specifically, the system will be analysed in term of financial cost, speed, temperature dependency, operational range and its sensitivity to radio interferences. Screenshots of the final prototype can be found in appendix E.

3.3.1 Prototype's Resources and Performances - Financial Cost

One of the main objective during the realisation of these prototypes was to keep them as cheap as possible. Indeed, concerning the software development, only free software have been used during the whole projects. As explained in appendix A, the simulator has been realised with Apple Xcode 7 [66]. The Arduino boards have been programmed with the free Arduino IDE [67] and debugged with Saleae Logic [70] (free software compatible with many logic analysers).

From the hardware point of view, the prices of each component composing the system are detailed in appendix F, and Tab.(3.2) here under summarises the total cost of the system[3].

Server	**Wearable**	**Localisation Beacon**
£33.25	£4.89	£4.89

Table 3.2: Summary of the prototype's financial cost

On one hand, the implementation of the server requires two 10uF capacitors, a 4.7uF capacitor, a nRF24L01+ based module, a lm1117-3.3, a potentiometer, 9 jumpers and the Raspberry Pi[4]. Therefore, it is responsible for more than 75% of the total cost, which is quite a lot but is needed in only one exemplar. On the other hand, each wearable and beacon cost less than £5 and are built from two 10uF capacitors, a 4.7uF capacitor, a nRF24L01+ based module, a lm1117-3.3, 9 jumpers and an Arduino board.

3.3.2 Prototype's Resources and Performances - Memory Usage

This paragraph gives a special focus on the wearable and on the beacons, since it is the entity requiring the most delicate optimization due to all the problematics debated in section 1.1. The embedded system's memory has two main parts: the program memory and the dynamic memory. The first one is what limits the size of the code, while the dynamic memory is what actually consists in the device resources during execution. Tab.(3.3) below reports the exact memory usage of the

[3]Some irrelevant costs like ethernet cables, plug adapters, USB cables and breadboards have been neglected.

[4]For better understanding, these references should be read in parallel with section 3.2, which provides a full detailed diagram of each entity.

wearable device and of the localisation beacons (about 5 KB of program memory are due to the boot loader). The memory usage of the server is comparable to the wearable, since the implemented code is analog.

	Program Memory	**Dynamic Memory**
Wearable	20.080 KB	284 B
Localisation Beam	14.422 KB	196 B

Table 3.3: Wearable's memory usage

As explained in section **??** the code as been optimized to move all fixed variables on the program memory (which is much wider than the dynamic memory). However, this choice has to be carefully balanced since program memory is not volatile[5], and is therefore much slower to read and consumes more power.

3.3.3 Prototype's Resources and Performances - Power Consumption

Contrarily to the memory usage, the power consumption is much harder to determine (at least, without the appropriate tools). Nevertheless, a theoretical approximation can be made based on the circuits presented in section 3.2. Therefore, the following results represent the ideal behaviour and consist in a lower bond approximation.

The first load consuming power is the nRF24 module. More specifically, as explained before, the nRF24l01+ chip has four modes of power operation that allow the user can programatically trade the amount of power drawn by the chip for higher range or reliability. In fact, the nRF24l01+ can be programmed to sink a current I_{sink} of 7, 7.5, 9 or 11.3 mA [43]. Therefore, knowing that the

[5]For modern devices, the program memory is generally made of flash or EEPROM memory, but old systems may still use EPROM.

core chip is powered at $V_{nRF24} = 3$V, Eq.(3.5) provides the total power P_{nRF24} consumed by the chip [43].

$$P_{nRF24} = I_{sink} * V_{nRF24} \tag{3.5}$$

As explained in section 3.2, the nRF24 module is connected to the 5V V_{cc} Arduino PIN through a voltage regulator, which consumes a non-negligible amount of power. Indeed, the power P_{volt_reg} dissipated by the lm1117 voltage regulator[6] can be calculated as given by Eq.(3.6).

$$P_{volt\_reg} = (V_{in} - V_{out}) * (I_{sink} + I_{Quiescent}) \tag{3.6}$$

where $I_{Quiescent}$ equals 5 mA [45], V_{in} is the 5V source of the Arduino board, V_{out} equals 3.3V (since the selected voltage regulator model is the lm1117-3.3 [45]), and I_{sink} is the current sunk by the nRF24 module[7]. Tab.(3.4) here below summarises the power consumption for the four possible nRF24's modes of operation.

Modes	I_{sink}	P_{nRF24}	P_{volt_reg}	**Total**
0x03	11.3 mA	33.9 mW	27.71 mW	61.61 mW
0x02	9 mA	27.0 mW	23.80 mW	50.80 mW
0x01	7.5 mA	22.5 mW	21.25 mW	43.75 mW
0x00	7 mA	21.0 mW	20.04 mW	41.04 mW

Table 3.4: Transceiver's power consumption (nRF24l01+ and lm1117-3.3)

[6]For easy understanding the following equation should be read in parallel with Fig.(3.13) of section 3.2.3.

[7]For the sake of completeness, it has to be mentioned that the nRF24 module has an on-chip additional voltage regulator dropping the input voltage from 3.3V to 3.0V, but its power dissipation is negligible and has been ignored.

Therefore, the total power saved by selecting the nRF24 mode 0x00 instead of the 0x03 is

$$\begin{aligned}\Delta_{0x03-0x00} &= P_{tot\_0x03} - P_{tot\_0x00} \\ &= 61.61[mW] - 41.04[mW] \\ &= 20.57[mW]\end{aligned} \tag{3.7}$$

where P_{tot_0x03} and P_{tot_0x00} are found in the last column of Tab.(3.4) at the first and last row, respectively. The result provided by Eq.(3.7) is not negligible knowing that a standard Coin Lithium Battery provides about 90 mAh. The above calculations have to be preciously kept since they will be unchanged during a future migration to an industrial system (see section 3.4).

Concerning the prototyped system, an original Arduino UNO board can source at most a current I_{max} of 200mA from its 5V V_{cc} PIN [62, 61], delivering then a maximum power P_{ard_PIN} of

$$\begin{aligned}P_{ard\_PIN} &= I_{max} * V_{cc} \\ &= 200[mA] \times 5[V] \\ &= 1[W]\end{aligned} \tag{3.8}$$

Therefore, Eq.(3.8) proofs that the Arduino board is theoretically strong enough to power the nRF24 module connected through the voltage regulator in any mode of operation. Note that the Arduino boards also posses a 3.3V V_{cc} PIN which has not been used in the wearable's design for the two following reasons:

- The 3.3V V_{cc} PIN can source at maximum 100 mA;
- The 5V V_{cc} PIN provides a far more stable DC current.

Now, the total power consumption[8] of the wearable device and the localisation beacons can be computed by adding to the results of Tab.(3.4) the power consumed by the Arduino CPU itself. Indeed, an Arduino UNO board that comes, as in the current case, with an ATMEGA328P CPU, a resonator oscillating at 16 MHz and powered (through USB) at 5V, draws a current I_{ard_CPU} of 16.32 mA [40]. Therefore, the total power P_{ard_CPU} dissipated by the board is

$$P_{ard\_CPU} = I_{ard\_CPU} * V_{USB} = 81.6[mW] \tag{3.9}$$

[8]The losses due to the jumper cables, socket solders and other board's imperfections are ignored.

Unfortunately, the power consumption of the Raspberry Pi is much harder to estimate due to the complexity of its design. However, a rapid over-approximation of the power P_{RPi} consumed by the board when the HDMI is disabled, all LEDs are disabled, an ethernet cable is plugged in but unused, leads to an upper bond of

$$P_{RPi} \leq 1[W] \tag{3.10}$$

which is far bigger than the power consumed by the nRF24 module [39, 65]. Moreover, the maximum current that can be drawn from the board is 700mA [65], which is again enough to power the nRF24 module connected through the voltage regulator in any mode of operation.

Finally, from the above equations and the results displayed in Tab.(3.4), the final power consumption of each entity of the prototyped system is summarised by Tab.(3.5) below .

nRF24 Operation Modes	**Server**	**Wearable & Beacon**
0x03	$\leq$ 1.062 W	$\approx$ 143.21 mW
0x02	$\leq$ 1.051 W	$\approx$ 132.40 mW
0x01	$\leq$ 1.044 W	$\approx$ 125.35 mW
0x00	$\leq$ 1.041 W	$\approx$ 122.64 mW

Table 3.5: Summary of the system's power consumption

3.3.4 Prototype's Resources and Performances - System's latency

This paragraph is devoted to analyse the latency of the system. In that purpose the server's and wearable's speed are summarised by the box plots of Figs.(**??**) and (**??**), respectively. These figures have been realised from data taken in the following working conditions:

- The tests have been realised at room temperature (25 °C);

- The transmitter and the receiver where situated exactly three meters apart;
- Each box of the box plot is generated from 50 independent samples.

Fig.(3.17) illustrates the timing of the methods `send_credentials`, `get_credentials` and `get_location` (see sections 3.2.2 and 3.2.3), and shows that sending the credentials is slower than verifying them. Moreover, the wearable can get a location's update about three time faster than how it gets new credentials. Fig.(3.18) presents the timing for the credentials' presentation, which is by far the slowest process.

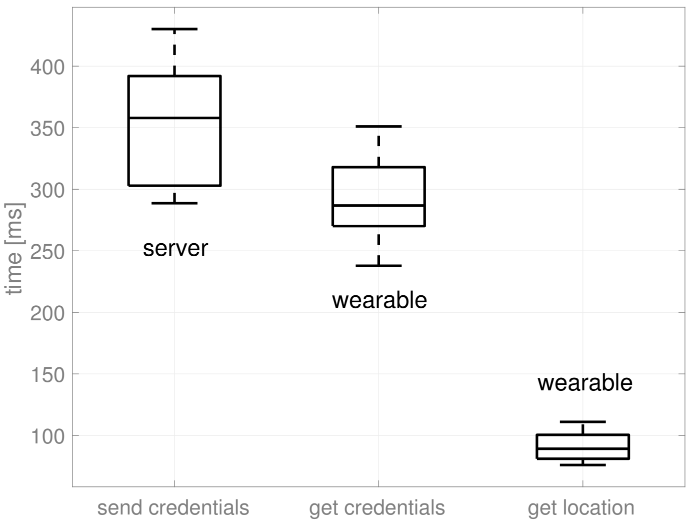


Figure 3.17: Credentials' issuance latency

The server's methods `issue_credentials` and `verify_credential_presentation` have not been pictured because their latency is negligible. However, contrarily to the server's case, the two wear-

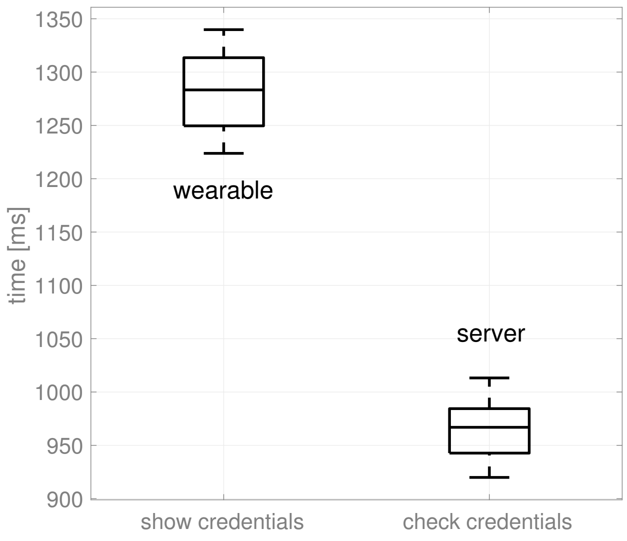


Figure 3.18: Credentials' showing latency

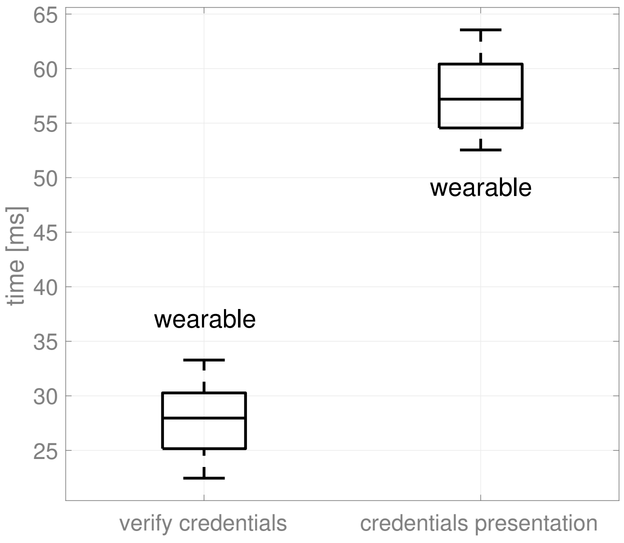


Figure 3.19: Wearable's inner methods

able's methods `verify_credentials` and `credential_presentation` (which are direct implementations of [13]) don't have a negligible latency due to the limited CPU resources of the wearable, but are still much faster than the methods displayed in the other plots.

In addition to the purely experimental data illustrated above, there is a theoretical maximal latency that can be computed for the current implementation of the credentials' showing and issuance, as well as for the reception of a new location. Indeed, each of these methods has a maximum timeout as described in section 3.2.1 that can be exploited for Denial of Service (DoS) attacks (see appendix G.4 for further details). Tab.(3.6) below summarises this theoretical maximum latency for each of these methods. This table has been computed using the default RX and TX timeout values (1s and 200 ms, respectively).

Methods	**Median**	**Theoretical Max**
`send_credentials`	357.94 ms	1.2 s
`get_credentials`	286.75 ms	2.0 s
`get_location`	89.19 ms	1.0 s
`show_credentials`	1283.3 ms	1.2 s
`check_credentials`	966.9 ms	2.0 s
`verify_credentials`	27.96 ms	-
`credentials_presentation`	60.89 ms	-

Table 3.6: Median and maximal theoretical latency

The power mode of the nRF24 module influences directly the data rate of the transceiver, allowing a transmission from 250 kbps to 2 Mbps (see Tab.(3.1) of section 3.2.1). However, the experiments showed that it does not influence the system's latency significantly.

3.3.5 Prototype's Resources and Performances - Operational Range

This paragraph aims to analyse the prototype's range of operation. In that purpose, the error rate (computed as the percentage of the number of failures divided by the number of successes) has been observed for each nRF24's mode of operation and reported in Fig.(3.20). The dots on the graphs represent the experimental data and the dotted lines are a cubic splines interpolation. The splines show a quasi exponential relation between the error rate and the operational range.

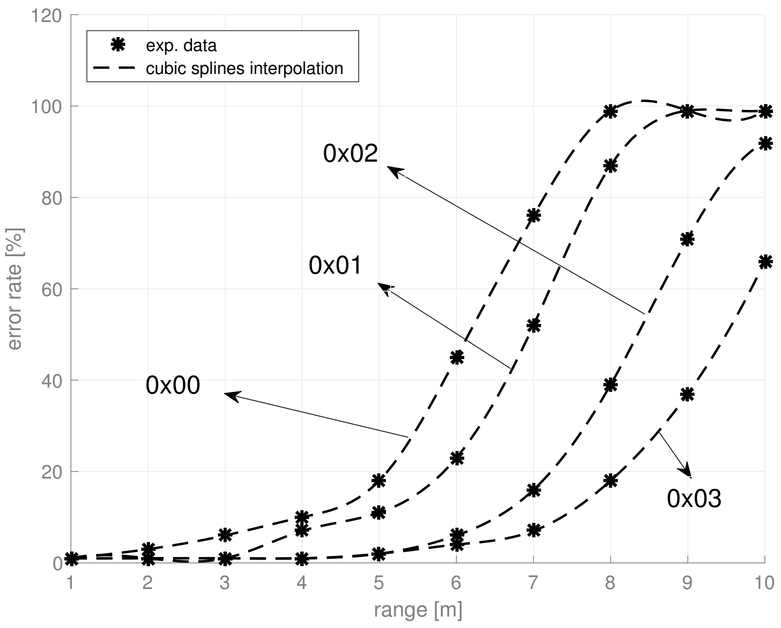


Figure 3.20: Influence of the distance on the error rate

More specifically, the tests have been run under the following conditions:

- The tests have been realised at room temperature (25 °C);
- A success is accounted only when the complete operation chain is observed on the wearable: get credentials from the server → get location from the beacons → show credentials to the server.

The above experiment show a high error rate when the distance reaches barely 10 meters. Indeed, it is likely to have a single failure during one of the transmissions. However, this range is perfectly sufficient since the architecture requires the wearable to perform only short-ranges communications.

A last additional ping test has been run to purely test the range of the nRF24l01+ chip[9]. In this last case, the nRF24 modules were able to ping each other up to a distance of 46 meters while

[9]To avoid any possible radio interferences, this test has been realised inside the Eurostar, when passing through the Eurotunnel (at about 40m under the sea bed) [63, 64].

keeping an error rate below 5%. This last test shows that the weak link in the communication chain is probably the unofficial Arduino board. A possible explanation is that the board struggles with communicating large data through SPI to the nRF24 module fast enough to avoid errors.

3.3.6 Prototype's Resources and Performances - Temperature Dependency

The next analysis concerns the system's temperature dependency. For that experiment, the wearable has been placed into an oven to experimentally test its temperature dependence. The following test settings have been applied:

- The wearable was situated inside the oven;
- The transmitter and the receiver were situated exactly one meter apart;
- A success is accounted only when the complete operation chain is observed on the wearable: get credentials from the server → get location from the beacons → show credentials to the server.

Fig.(3.21) shows the results of the tests. As before, the dots on the graphs represents the experimental data and the dotted lines are a cubic splines interpolation. The graph shows that the system is completely stable until about 38°C and then the error rate suddenly increases.

However, in theory, the wearable should work correctly until 85°C. Indeed, the nRF24 module, the voltage regulator, the

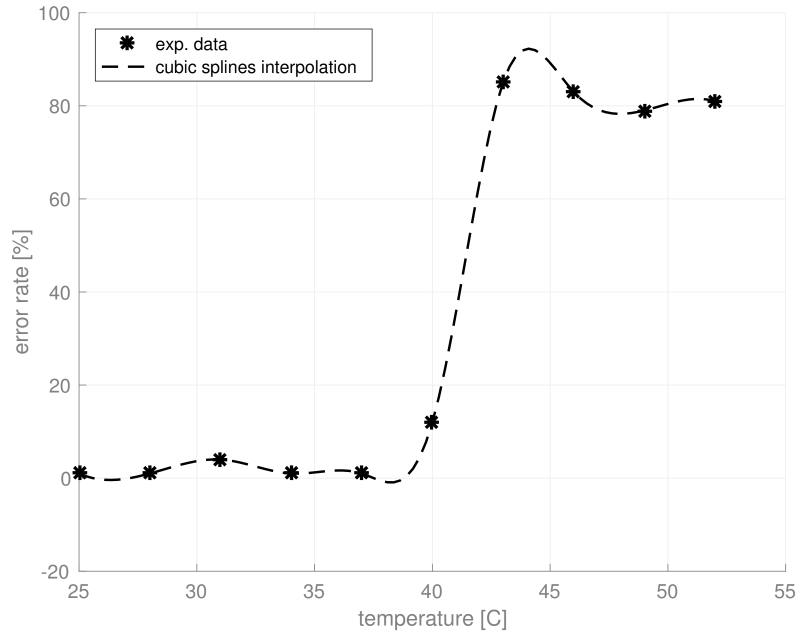


Figure 3.21: Influence of the temperature on the error rate

Arduino USB chip and the ATMEGA328P are designed to operate until 85°C, 125°C, 85°C and 85°C, respectively [43, 45, 41, 40].

This huge performance's difference could be explained by the fact that despite all the chips have been designed to withstand such temperatures, the other (cheap) components composing the board are not. Indeed, the following well-known facts are always applicable:

- The resonator's frequency may derive and not produce the exact value (16 MHz);
- IO protocols like I2C and SPI may perform badly;
- Resistors may not produce the exact value (may change up to 8%);
- The jumpers' resistivity increases with temperature;
- Capacitors may not produce the exact value (may change by a lot more than 8%).

The first and the last point are the most problematic. Indeed, the resonator regulates the CPU operations and the role of the capacitors in the Arduino board are to smooth the signals arriving to the CPU. If these operations are poorly performed, the CPU's behaviour may become unpredictable.

With the above considerations in mind, an additional test that produced a very noticeable result has been run on the board at a temperature around 40 - 45°C: despite the location beacons were switched off, the wearable started showing its credentials, by its own, at a rate of about 0.5% ! This observation consists in an important vulnerability and is further discussed in section G.3. The server's temperature dependency has not been analysed since servers generally remain all the time locked in temperature-controlled rooms.

3.3.7 Prototype's Resources and Performances - Sensitivity to Interferences

This last paragraph aims to analyse the system's sensitivity to radio interferences in order to test the system's resistance to trivial DoS (jamming).

Fig.(3.22) shows the results of these tests. As usual, the dots on the graphs represents the experimental data and the dotted lines are a cubic splines interpolation. The graph shows a critical point at around 0.7 m. Indeed, the system seems sensitive to jamming only when the jammer is situated at less than 0.7 m from the median formed by the transceivers. In fact, once this this critical point is reached, the system appears to be immune to interferences.

Moreover, even when the jammer is placed right in between the two transceivers (at position 0), the error rate does not exceed 45%.

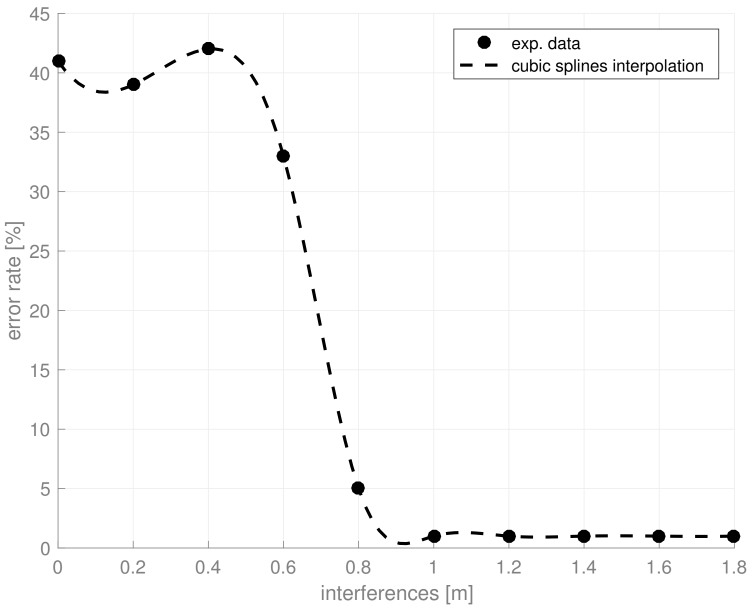


Figure 3.22: Influence of radio interferences on the error rate

This experiment has been run with the setup shown in Fig.(3.23) and each transceiver were set to the power mode 0x01. As usual, the test has been run at room temperature and under the same success' conditions as in the previous paragraphs.

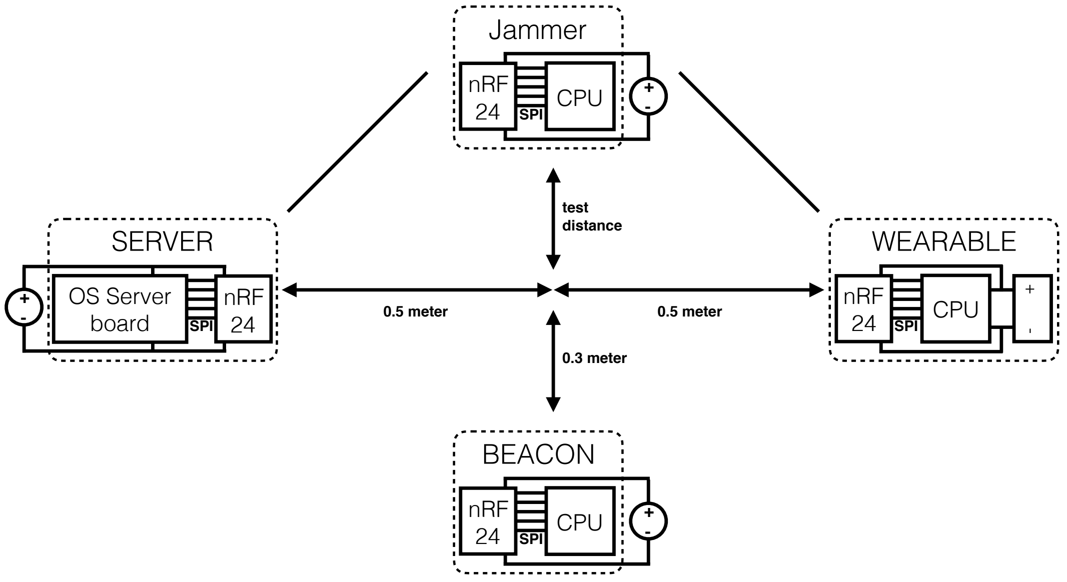


Figure 3.23: Sensitivity to radio interferences experiment setup

3.4 From the Prototype to an Industrial System

This section explains how to transform the studied prototype into a fully real-world system. Indeed, the extremely limited resources of the unofficial Arduino boards make impossible to meet any industrial requirements.

On one hand, the server needs to migrate to a more powerful platform and bridge the gap between the prototyped implementation and the web interface depicted in appendix C (fairly straightforward). On the other hand, the wearables and the beacons should be modified in order to include all the functionalities from the original architecture shown in Fig.(2.1) of section 2.1 that have been skipped in the prototype. To that purpose, all Arduinos should be replaced by a Microchip PIC32MZ (Fig.(3.24)), the latest Microchip PIC32 release that has been natively designed for security. Indeed, in addition to various hardware implementations of AES and DES (CBC, ECB, CTR, CFB, OFB, as well as AES-GCM), the chip also provides a hardware implementation of SHA-2, a pseudorandom number generator, a true random number generator and a 512-byte OTP array not readable from other memory spaces (ideal for storing cryptographic keys) [42]. Moreover, Microchip has recently released a next version of its free IDE, MPLAB Harmony, allowing an automatic inclusion of the WolfSSL library [69, 51]. In conclusion, the modifications to migrate the wearables from the Arduino to the PIC32MZ can be summarised as follow:

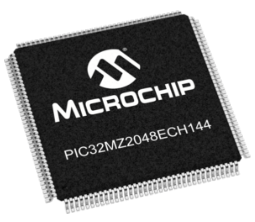


Figure 3.24: Example of Microchip PIC32MZ [57]

1. Replace the software implemented SHA function by the PIC32MZ hardware implementation and the current prototype ECC library by the original WolfSSL;
2. Encrypt all wearable-beacon and server-beacon data transfers as discussed in section 2.1 using the hardware implementations of AES, AES-GCM and WolfSLL.
3. Link the prototype's `get_random` function to the MCU hardware random number generator;

A full Printed Circuit Board (PCB) of the final industrial wearable can be found in appendix I.

Conclusion and Further Directions

This paper presented a flexible privacy enhancing system from its architectural to its prototyping level, based on the anonymous credentials protocols developed by [13]. Three kind of entities are involved in this system: a main server, wearable devices and localisation beacons. In this multi-purpose architecture, the server firstly issues some anonymous credentials to the wearables. Then, when the wearable reaches a particular location, it automatically presents its credentials to command the server the execution of a particular action. Both the architecture of the wearable and of the server remain generic and scalable in order to encourage further enhancements and easy integration into real-world applications.

In this work

This work started by presenting the complete system's architecture and the security analysis. Afterwards, the prototype's behaviour is explained through analysis of the communication between each entity. Then, the hardware and software implementation have been discussed as well as all the components needed to build the system. The prototype has been widely tested to carefully determine the resources it requires and its performances under different circumstances. Finally, this work ends by showing how to migrate from the current prototype to a fully industrial system.

Further Works

The final goal of this architecture is its implementation on industrial hardware. Moreover, a further improvement would consists in modifying the current Non-Interactive Zero Knowledge (NIZK) proof to add a proof of non-negativity over the credentials' attributes. Indeed, a proof of non-negativity would allow the wearable to keep in memory a single attribute representing the maximum user's authorisation, and use it to perform diverse actions. Even further, the introduction of this proof would make possible the implementation of a full Bell-LaPadula model [4]. Finally, an additional improvement would consist in the integration of a credentials' revocation mechanism and in applying the architectural improvements suggested in appendix G in order to improve the system's security.

Appendices

A Appendix. Design Steps and Strategy

This appendix is devoted to the explanations of the design steps and methodology applied during the realisation of this work. The design strategy can be separated in the heigh main steps detailed below.

Step 1 - Background Researches

The first step realised during this term was about summarising all the main papers and researches in the target area in order to build a literature review on the field of wearable security and anonymous credentials for embedded devices. This literature review gave a pretty complete idea of what are the existing tools, what has already been done, and what is still missing. Therefore, this first step indicated the main directions to follow and is the foundation of the whole project.

Step 2 - Architecture and Security Analysis

The next step consisted in designing the system's architecture and in setting up the desired security properties. The summary of the work done during this step can be found in section 2.2.

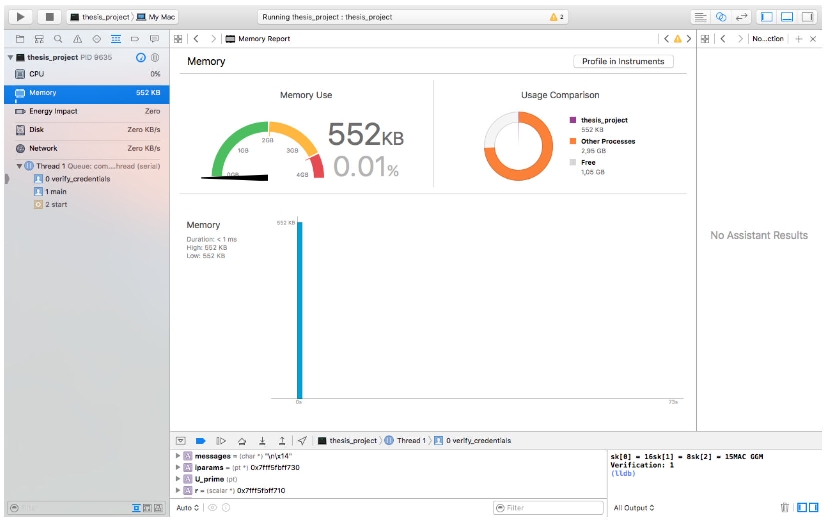
Step 3 - Anonymous Credential Protocol Choice

Many anonymous credential protocols are available and well documented. However, since the current system is an example of situation where the issuer of the credentials is also the verifier, the aMAC protocol [13] is a great choice due to its simplicity and the elegance of its design. This step also required to write down all the Zero-Knowledge proofs involved in the protocol and translating them into Elliptic Curve Cryptography (EEC) to allow efficient implementations (see sections 1.3.1 and 1.3.2).

Step 4 - Simulator's Construction

Now that all the theoretical parts have been done and the problematic is correctly understood, the first building step was the construction of a software simulator on a standard desktop computer.

Indeed, the debugging tools available on computers, the code auto-completion tools and syntax analysers greatly facilitated the software development, since they are not available when programming for embedded processors. The simulation has been developed on Apple Xcode 7 [66]. As shown by the screenshot on Fig.(25), the debugging tool automatically displays the resource usages and permits to execute the program step by step in order to gain a complete and deep control over the software under development. However, spending to much time developing a first simulator happened to be a mistake. Indeed, most of the code realised during this step could not be embedded into the wearables because of their very limited amount of memory. A better approach would have been to first select the hardware, and then simulate only small portions of the code at a time in order to test them directly on the device and verify that they meet all the hardware constraints.



```
messages = (char *) "\n\x14"
iparams = (pt *) 0x7fff5fbff730
U_prime (pt)
r = (scalar *) 0x7fff5fbff710
```

```
sk[0] = 16sk[1] = 8sk[2] = 15MAC GGM
Verification: 1
(lldb)
```

Figure 25: Illustration of user-friendly debugging tools

Step 5 - Hardware Choice

The fifth step consisted in selecting the hardware to build the prototypes. Due to their very advantageous costs, efficiency and flexibility, the main server has been prototyped on a Raspberry Pi B+ and the other entities have been implemented on unofficial Arduino UNO boards [65, 61].

However, many different possibilities had to be investigated for the component's interactions and for the localisation system. More specifically, the communication methods for the server-wearable link and the kind of positioning system had to be carefully chosen since the software implementation strictly depends on it. Many choices like Wi-Fi, Bluetooth, RF 433MHz or RF 2.4GHz technologies were available and have been considered in term of cost, power consumption and operational range. A summary of all the possible options for the communications technologies can be found in Tab.(7) below[10].

Technologies	**Advantages**	**Drawbacks**
Wi-Fi (P2P)	Standard technology;	Hard to set up with cheap components; Low range;
Wi-Fi (with router)	Standard technology; High range achievable with low power;	Need to trust the router; Hard to set up with cheap components; Range hard to control;
Bluetooth	Standard technology; Low power; Cheap (£3.85);	Low range;
XBee	Easy to set up; Low power;	Expensive (£25.58);

[10]This analysis has been done with some of the cheapest and most common components on the market. The considered Wi-Fi, Bluetooth, XBee, RF 433MHz and RF 2.4GHz modules are respectively referenced by [74], [75], [76], [77] and [78].

RF 433MHz	Very cheap module (∼ £1); Easy to ensure security (low level protocols);	Big and need and external antenna (about 17 cm); High power consumption; Low range;
RF 2.GHz	Very cheap module (∼ £1); Easy to ensure security (low level protocols); High range; Low power;	Hard to set up; Need circuit drivers; Need very stable current;

Table 7: Possible server-wearable communication technologies

A careful investigation led to the choice of RF 2.4GHz waves for the localisation system and all the communications[11]. On a more organisational side, the very high shipping delay needed to receive the components in London was under serious consideration. In fact, some piece of hardware achieving the best trade-off between performances and price required more than 45 working days to arrive and had to be regretfully discarded.

Step 6 - Migration to Embedded Systems

Once all the electronic components have been chosen, it is time to assemble de system. This step was one of the most complicated tasks of the project for all the reasons explained in section 1.1. Moreover, specific and targeted drivers had to be built in order to make all the peripherals communicate with other specifics embedded CPUs.

[11]Actually, this choice is mainly due to the thigh current restriction imposed by the unofficial Arduino, that made impossible to plug more than one module on a single board. A more evolute version with hardware of better quality is encouraged and investigated in section 3.4.

An additional difficulty came from the lack of debugging tools available for embedded systems: there are no nice and user-friendly debuggers like the ones discussed in step 4. In fact, when dealing with embedded systems, often, the only tool available is a logic analyser (see Fig.(26)).

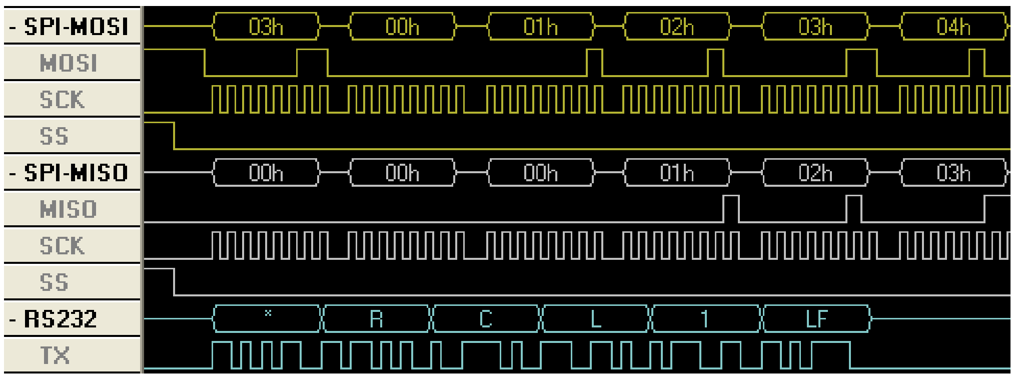


Figure 26: Illustration of logic analyser [68]

Step 7 - Testing & Validation

The final phase in the realisation of all embedded systems consists in testing and validating the design. Indeed, before any industrialisation, the complete behaviour of the device should be verified, even under extreme conditions. Section 3.3 is entirely dedicated to the presentation of the experimental results performed by the circuit.

Step 8 - Guidelines for an Industrial System

As a complement, this additional step consisted in describing how to transform the developed prototype into a real-world system meeting industrial standards.

B Appendix. Credentials' Issuance and Presentation Proof

This appendix comes as complement of section 1.3.2 by presenting the complete Non-Interactive Zero Knowledge (NIZK) proofs used in the credentials' issuance and presentations. This appendix refers to the same notations defined in section 1.3.2 and is totally inspired from pages 27 and 28 of [13].

B.1 Credentials' Issuance and Presentation Proof - Issuance Proof

This section aims to develop the proof π_1 introduced section 1.3.2. For easy reference, this proof is recalled below.

$$\begin{aligned}\pi_1 \quad := \quad & PK\left\{(\boldsymbol{x}, \widetilde{x_0}) : U' = x_0 U + \sum_{i=1}^{n} x_i(m_i U) \quad \wedge \quad C_{x_0} = x_0 G + \widetilde{x_0} H \quad \wedge \quad X_i = x_i H\right\} \\ & \forall i \in \{1, \ldots, n\}\end{aligned}$$

The proof's development is divided in five parts: the witnesses' computations, the challenge's computation, the responses' computation, the information to send, and finally, the proof's verification.

- **Witnesses Computation :**
  1. pick at random $(\widetilde{w_0}, w_1, \ldots, w_n) \in \mathbb{F}_p^{n+1}$
  2. compute $W_{U'} = w_0 + \sum_{i=1}^{n}(w_i m_i U)$
  3. compute $W_{C_{x_0}} = w_0 G + \widetilde{w_0} H$
  4. compute $W_{X_i} = w_i H \quad \forall i \in \{1, \ldots, n\}$
- **Challenge's Computation :**
  1. compute $\mathcal{H}\left(C_{x_0} || W_{U'} || W_{C_{x_0}} || W_{X_i}\right) \quad \forall i \in \{1, \ldots, n\}$, where $\mathcal{H}$ is a cryptographically secure hash function
- **Responses Computation :**
  1. compute $r_{x_i} = w_i - c x_i \quad \forall i \in \{1, \ldots, n\}$
  2. compite $r_{\widetilde{x_0}} = \widetilde{w_0} - c\widetilde{x_0}$

- **Information to Send :**

  1. send $(c, r_{x_i}, r_{\widetilde{x_0}}, C_{x_0}, X_i, U, U') \quad \forall i \in \{1, \ldots, n\}$, where $(U, U') \leftarrow \mathsf{MAC_{GGM}}$

- **Verification :**

  1. recompute $W_{U'}$ as follow:

$$\begin{aligned} W_{U'} &= r_{x_0}U + cU' + \sum_{i=1}^{n}(r_{x_i}m_iU) \\ &= \left(w_0 + \sum_{i=1}^{n}(w_im_iU)\right) - c\left(x_0 + \sum_{i=1}^{n}(x_im_iU)\right) + cU' \\ &= w_0U + \sum_{i=1}^{n}(w_im_iU) \end{aligned}$$

  2. recompute $W_{C_{x_0}}$ as follow:

$$\begin{aligned} W_{C_{x_0}} &= r_{x_0}G + r_{\widetilde{x_0}}H + cC_{x_0} \\ &= (w_0G + \widetilde{w_0}H) - c\,(x_0G + \widetilde{x_0}H) + cC_{x_0} \\ &= w_0G + \widetilde{w_0}H \end{aligned}$$

  3. recompute $W_{X_i} \quad \forall i \in \{1, \ldots, n\}$ as follow:

$$\begin{aligned} W_{X_i} &= r_{x_i}H + cX_i \\ &= w_iH - c(x_iH) + cX_i \\ &= w_iH \end{aligned}$$

  4. accept the proof if $c == \mathcal{H}\left(C_{x_0}||W_{U'}||W_{C_{x_0}}||W_{X_i}\right) \quad \forall i \in \{1, \ldots, n\}$

B.2 Credentials' Issuance and Presentation Proof - Presentation Proof

Similarly to the previous section, the equations below develop the proof π_2 introduced section 1.3.2. For easy reference, the proof is recalled below.

$$\begin{aligned} \pi_2 \; := \; & PK\left\{(\boldsymbol{m}, \boldsymbol{z}, -r) : \phi(\boldsymbol{m} = 1) \quad \wedge \quad C_{m_i} = m_iU + z_iH \quad \wedge \quad V = \left(\sum_{i=1}^{n} z_iX_i\right) - rG\right\} \\ & \forall i \in \{1, \ldots, n\} \end{aligned}$$

As before, the proof's development is divided in five parts: the witnesses' computations, the challenge's computation, the responses' computation, the information to send and finally, the proof's verification.

- **Witnesses Computation :**

  1. pick at random $(a, r, z_1, \dots, z_n, w_r, w_{z_1}, \dots, w_{z_n}, w_{m_1}, \dots, w_{m_n}) \in \mathbb{F}_p^{3n+3}$
  2. compute $U_a = aU$
  3. compute $U'_a = aU'$
  4. compute $C_{m_i} = m_i U_a + z_i H \quad \forall i \in \{1, \dots, n\}$
  5. compute $C_{U'} = rG + U'_a$
  6. compute $W_{C_{m_i}} = w_{m_i} U_a + w_{z_i} H \quad \forall i \in \{1, \dots, n\}$
  7. compute $W_V = w_r G + \sum_{i=1}^{n} (w_{z_i} X_i)$

- **Challenge's Computation :**

  1. compute $\mathcal{H}\left(C_{m_i} || C_{U'} || W_{C_{m_i}} || W_V\right) \quad \forall i \in \{1, \dots, n\}$, where $\mathcal{H}$ is a cryptographically secure hash function

- **Responses Computation :**

  1. compute $r_s = w_r + cr$ (note the '+' sign)
  2. compute $s_{z_i} = w_{z_i} - cz_i \quad \forall i \in \{1, \dots, n\}$
  3. compute $s_{m_i} = w_{m_i} - cm_i \quad \forall i \in \{1, \dots, n\}$

- **Information to Send :**

  1. send $(c, s_r, s_{z_i}, s_{m_i}, C_{m_i}, C_{U'}, U_a) \quad \forall i \in \{1, \dots, n\}$

- **Verification :**

1. recompute $W_{C_{m_i}} \quad \forall i \in \{1, \ldots, n\}$ as follow:

$$\begin{aligned} W_{C_{m_i}} &= s_{m_i} U_a + s_{z_i} H + c C_{m_i} \\ &= (w_{m_i} U_a + w_{z_i} H) - c\,(m_i U_a + z_i H) + c C_{m_i} \\ &= w_{m_i} U_a + w_{z_i} H \end{aligned}$$

2. compute $V = x_0 U_a - C_{u'} + \sum_{i=1}^{n} (x_i C_{m_i})$

3. recompute W_V as follow:

$$\begin{aligned} W_V &= cV + s_r G + \sum_{i=1}^{n} (s_{z_i} X_i) \\ &= cV + \left(w_r G + \sum_{i=1}^{n} (w_{z_i} X_i) \right) - c \left(-rG + \sum_{i=1}^{n} (z_i X_i) \right) \end{aligned}$$

substituting $V = x_0 U_a - (rG + U'_a) + \sum_{i=1}^{n} (x_i [m_i U_a + z_i H])$, we obtain:

$$\begin{aligned} &= c \left(x_0 U_a - (rG + U'_a) + \sum_{i=1}^{n} (x_i [m_i U_a + z_i H]) \right) + \left(w_r G + \sum_{i=1}^{n} (w_{z_i} X_i) \right) \\ &\quad -c \left(-rG + \sum_{i=1}^{n} (z_i X_i) \right) \end{aligned}$$

replacing $U'_a = x_0 U_a + \sum_{i=1}^{n} (x_i m_i U_a)$, the previous expression becomes:

$$\begin{aligned} &= c \left\{ x_0 U_a - \left(x_0 U_a + [rG + \sum_{i=1}^{n} (x_i m_i U_a)] \right) + \sum_{i=1}^{n} (x_i [m_i U_a + z_i H]) \right\} \\ &\quad + \left(w_r G + \sum_{i=1}^{n} (w_{z_i} X_i) \right) - c \left(-rG + \sum_{i=1}^{n} (z_i X_i) \right) \end{aligned}$$

finally, recalling $X_i = x_i H$, we have:

$$= w_r G + \sum_{i=1}^{n} (w_{z_i} X_i)$$

4. accept the proof if $c == \mathcal{H} \left(C_{m_i} || C_{U'} || W_{C_{m_i}} || W_V \right) \quad \forall i \in \{1, \ldots, n\}$

Note that the credentials's verification's proof in the original paper [13] presents a little mistake at page 28. Indeed, the expression 3.(b) of appendix E.2 should be

$$c' = H(param||\{C_{m_i}\}_{i=1}^{n}||C_{u'}||\{\underbrace{C_{m_i}^{c}}_{exponent}g^{s_{m_i}}h^{s_{z_i}}\}_{i=1}^{n}||\underbrace{V^{c}}_{exponent}X^{s_{z_1}}\dots X^{s_{z_n}}g^{s_r}) \tag{11}$$

instead of

$$c' = H(param||\{C_{m_i}\}_{i=1}^{n}||C_{u'}||\{C_{m_i}g^{s_{m_i}}h^{s_{z_i}}\}_{i=1}^{n}||VX^{s_{z_1}}\dots X^{s_{z_n}}g^{s_r}) \tag{12}$$

C Appendix. Web Interface

This appendix presents the web interface and the databases used to manage the system. Two kind of users can log in in this website: admins and wearables' users. The admins have the possibility to grant credentials to the users and to manage the devices' assignments, while standard users are only able to observe the credentials they own.

First of all, in oder to include any web interface to the current system, the server's FSM illustrated in Fig.(3.12) of section 3.2.2 should be modified to become as shown in Fig.(27) here below. Indeed, the server's main loop must listen to potential commands coming from the website. In the current case, the server has to be ready to issue some credentials when demanded by a logged in admin.

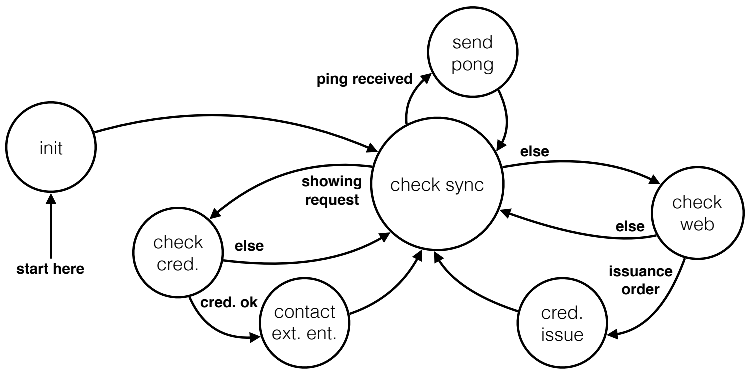


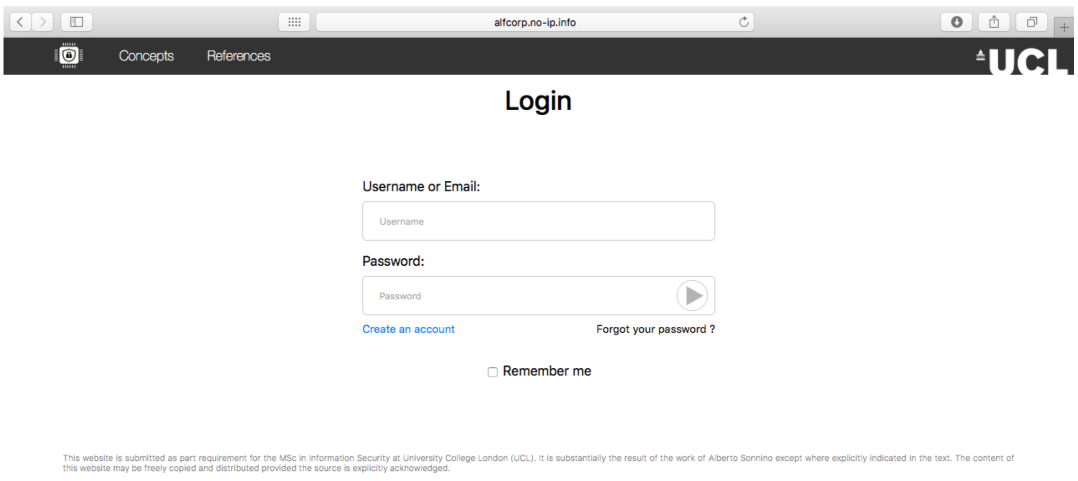
Figure 27: Industrial server's software FSM

The next sections presents respectively the realised secure login system, the web credentials' management, and some additional features available through the developed web interface.

C.1 Web Interface - Secure Login System

The PHP secure login system used in this website comes from the *panique* open source library referenced in [50]. All the expected login features have been implemented: login form, registration through email verification, password reset mechanism, edit personal user data, etc.

The login system relies on a SQL database containing the users' ID, usernames, hashed and salted passwords, registration date, number of failed login attempts (used for protection agains brute force attacks), and the role of the user (classic user or administrator).


alfcorp.no-ip.info

Concepts References

Login

Username or Email:

Username

Password:

Password

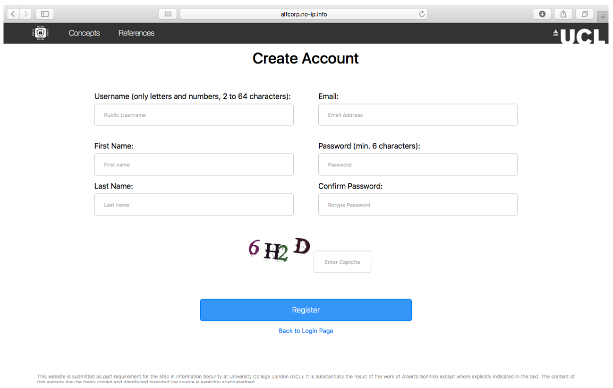
Create an account

Forgot your password ?

☐ Remember me

This website is submitted as part requirement for the MSc in Information Security at University College London (UCL). It is substantially the result of the work of Alberto Sonnino except where explicitly indicated in the text. The content of this website may be freely copied and distributed provided the source is explicitly acknowledged.

(a) Login interface


alfcorp.no-ip.info

Concepts References

Create Account

Username (only letters and numbers, 2 to 64 characters):	Email:
Public Username	Email Address
First Name:	Password (min. 6 characters):
First name	Password
Last Name:	Confirm Password:
Last name	Retype Password

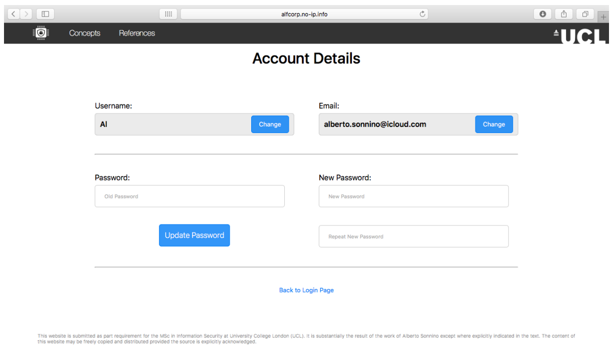
6 H2 D Enter Captcha

Register

Back to Login Page

This website is submitted as part requirement for the MSc in Information Security at University College London (UCL). It is substantially the result of the work of Alberto Sonnino except where explicitly indicated in the text. The content of this website may be freely copied and distributed provided the source is explicitly acknowledged.

(b) Account's registartion


alfcorp.no-ip.info

Concepts References

Account Details

Username:	Email:
Al (Change)	alberto.sonnino@icloud.com (Change)
Password:	New Password:
Old Password	New Password
Update Password	Repeat New Password

Back to Login Page

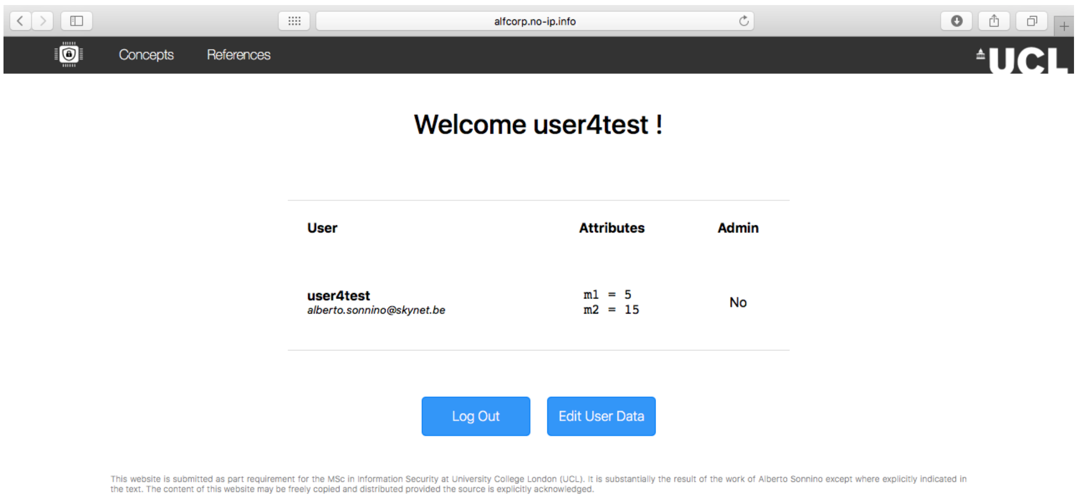
This website is submitted as part requirement for the MSc in Information Security at University College London (UCL). It is substantially the result of the work of Alberto Sonnino except where explicitly indicated in the text. The content of this website may be freely copied and distributed provided the source is explicitly acknowledged.

(c) Edit account

Figure 28: Login System

C.2 Web Interface - Credentials' Management

The credentials' management interface is accessible only after proper login and is different depending on the type of account. Fig.(29) picture a typical user and an admin account. As explained above, the user can only see its credentials while the admin can list all the registered users. To that purpose a second SQL database holds records of the credentials issued to each user. The two buttons under the *Edit* column on Fig.(29b) are designed to allow the admin to edit or revoke the user's credentials. However, this features are not yet implemented in the prototype.

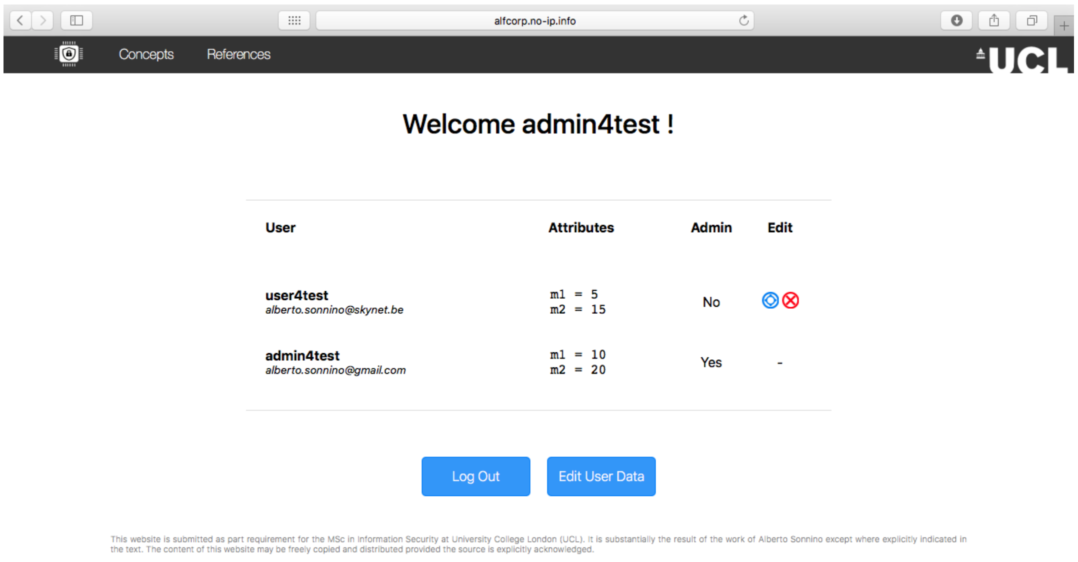


Welcome user4test !

User	Attributes	Admin
user4test *alberto.sonnino@skynet.be*	`m1 = 5` `m2 = 15`	No

This website is submitted as part requirement for the MSc in Information Security at University College London (UCL). It is substantially the result of the work of Alberto Sonnino except where explicitly indicated in the text. The content of this website may be freely copied and distributed provided the source is explicitly acknowledged.

(a) Example of user account



Welcome admin4test !

User	Attributes	Admin	Edit
user4test *alberto.sonnino@skynet.be*	`m1 = 5` `m2 = 15`	No	◎ ⊗
admin4test *alberto.sonnino@gmail.com*	`m1 = 10` `m2 = 20`	Yes	-

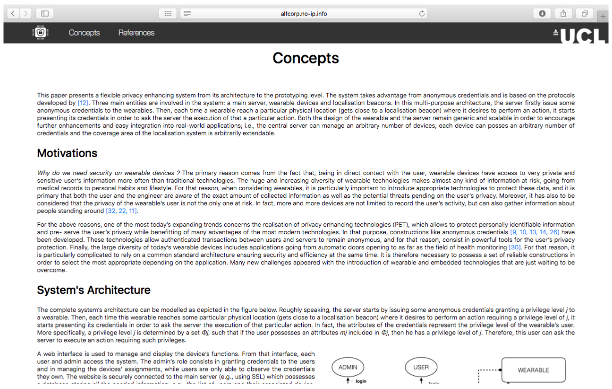
This website is submitted as part requirement for the MSc in Information Security at University College London (UCL). It is substantially the result of the work of Alberto Sonnino except where explicitly indicated in the text. The content of this website may be freely copied and distributed provided the source is explicitly acknowledged.

(b) Example of admin account

Figure 29: Web credential's management

C.3 Web Interface - Additional Features

As additional features, two extra tabs have been added to the website. The first provides an brief and easy-to-read overview of the system, and the second presents the list of references used in this work.



Concepts

This paper presents a flexible privacy enhancing system from its architecture to the prototyping level. The system takes advantage from anonymous credentials and is based on the protocols developed by [12]. Three main entities are involved in the system: a main server, wearable devices and localisation beacons. In this multi-purpose architecture, the server firstly issue some anonymous credentials to the wearables. Then, each time a wearable reach a particular physical location (gets close to a localisation beacon) where it desires to perform an action, it starts presenting its credentials in order to ask the server the execution of that a particular action. Both the design of the wearable and the server remain generic and scalable in order to encourage further enhancements and easy integration into real-world applications; i.e., the central server can manage an arbitrary number of devices, each device can posses an arbitrary number of credentials and the coverage area of the localisation system is arbitrarily extendable.

Motivations

Why do we need security on wearable devices ? The primary reason comes from the fact that, being in direct contact with the user, wearable devices have access to very private and sensitive user's information more often than traditional technologies. The huge and increasing diversity of wearable technologies makes almost any kind of information at risk, going from medical records to personal habits and lifestyle. For that reason, when considering wearables, it is particularly important to introduce appropriate technologies to protect these data, and it is primary that both the user and the engineer are aware of the exact amount of collected information as well as the potential threats pending on the user's privacy. Moreover, it has also to be considered that the privacy of the wearable's user is not the only one at risk. In fact, more and more devices are not limited to record the user's activity, but can also gather information about people standing around [32, 22, 11].

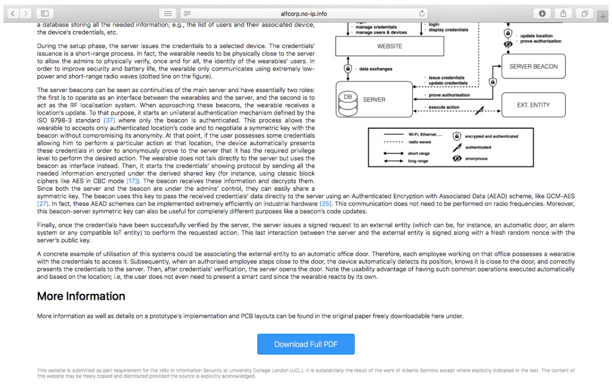
For the above reasons, one of the most today's expanding trends concerns the realisation of privacy enhancing technologies (PET), which allows to protect personally identifiable information and pre- serve the user's privacy while benefitting of many advantages of the most modern technologies. In that purpose, constructions like anonymous credentials [9, 10, 13, 14, 26] have been developed. These technologies allow authenticated transactions between users and servers to remain anonymous, and for that reason, consist in powerful tools for the user's privacy protection. Finally, the large diversity of today's wearable devices includes applications going from automatic doors opening to as far as the field of health monitoring [30]. For that reason, it is particularly complicated to rely on a common standard architecture ensuring security and efficiency at the same time. It is therefore necessary to possess a set of reliable constructions in order to select the most appropriate depending on the application. Many new challenges appeared with the introduction of wearable and embedded technologies that are just waiting to be overcome.

System's Architecture

The complete system's architecture can be modelled as depicted in the figure below. Roughly speaking, the server starts by issuing some anonymous credentials granting a privilege level *j* to a wearable. Then, each time this wearable reaches some particular physical location (gets close to a localisation beacon) where it desires to perform an action requiring a privilege level of *j*, it starts presenting its credentials in order to ask the server the execution of that particular action. In fact, the attributes of the credentials represent the privilege level of the wearable's user. More specifically, a privilege level *j* is determined by a set *Φj*, such that if the user possesses an attributes *mj* included in *Φj*, then he has a privilege level of *j*. Therefore, this user can ask the server to execute an action requiring such privileges.

A web interface is used to manage and display the device's functions. From that interface, each user and admin access the system. The admin's role consists in granting credentials to the users and in managing the devices' assignments, while users are only able to observe the credentials they own. The website is securely connected to the main server (e.g., using SSL) which possesses

(a) Brief system's overview - Part 1



a database storing all the needed information; e.g., the list of users and their associated device, the device's credentials, etc.

During the setup phase, the server issues the credentials to a selected device. The credentials' issuance is a short-range process. In fact, the wearable needs to be physically close to the server to allow the admins to physically verify, once and for all, the identity of the wearables' users. In order to improve security and battery life, the wearable only communicates using extremely low-power and short-range radio waves (dotted line on the figure).

The server beacons can be seen as continuities of the main server and have essentially two roles: the first is to operate as an interface between the wearables and the server, and the second is to act as the RF localisation system. When approaching these beacons, the wearable receives a location's update. To that purpose, it starts an unilateral authentication mechanism defined by the ISO 9798-3 standard [37] where only the beacon is authenticated. This process allows the wearable to accepts only authenticated location's code and to negotiate a symmetric key with the beacon without compromising its anonymity. At that point, if the user possesses some credentials allowing him to perform a particular action at that location, the device automatically presents these credentials in order to anonymously prove to the server that it has the required privilege level to perform the desired action. The wearable does not talk directly to the server but uses the beacon as interface instead. Then, it starts the credentials' showing protocol by sending all the needed information encrypted under the derived shared key (for instance, using classic block ciphers like AES in CBC mode [17]). The beacon receives these information and decrypts them. Since both the server and the beacon are under the admins' control, they can easily share a symmetric key. The beacon uses this key to pass the received credentials' data directly to the server using an Authenticated Encryption with Associated Data (AEAD) scheme, like GCM-AES [27]. In fact, these AEAD schemes can be implemented extremely efficiently on industrial hardware [25]. This communication does not need to be performed on radio frequencies. Moreover, this beacon-server symmetric key can also be useful for completely different purposes like a beacon's code updates.

Finally, once the credentials have been successfully verified by the server, the server issues a signed request to an external entity (which can be, for instance, an automatic door, an alarm system or any compatible IoT entity) to perform the requested action. This last interaction between the server and the external entity is signed along with a fresh random nonce with the server's public key.

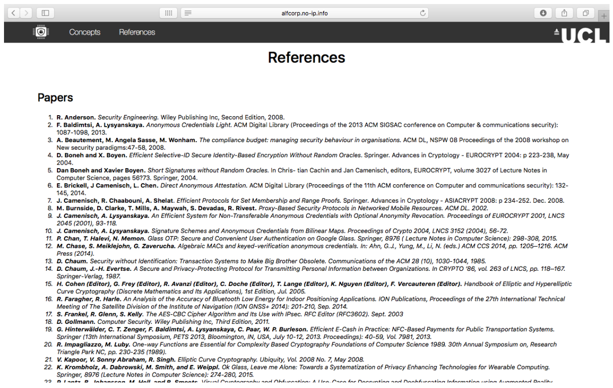
A concrete example of utilisation of this systems could be associating the external entity to an automatic office door. Therefore, each employee working on that office possesses a wearable with the credentials to access it. Subsequently, when an authorised employee steps close to the door, the device automatically detects its position, knows it is close to the door, and correctly presents the credentials to the server. Then, after credentials' verification, the server opens the door. Note the usability advantage of having such common operations executed automatically and based on the location; i.e, the user does not even need to present a smart card since the wearable reacts by its own.

More Information

More information as well as details on a prototype's implementation and PCB layouts can be found in the original paper freely downloadable here under.

This website is submitted as part requirement for the MSc in Information Security at University College London (UCL). It is substantially the result of the work of Alberto Sonnino except where explicitly indicated in the text. The content of this website may be freely copied and distributed provided the source is explicitly acknowledged.

(b) Brief system's overview - Part 2



References

Papers

1. **R. Anderson.** *Security Engineering.* Wiley Publishing Inc, Second Edition, 2008.
2. **F. Baldimtsi, A. Lysyanskaya.** *Anonymous Credentials Light.* ACM Digital Library (Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security): 1087-1098, 2013.
3. **A. Beautement, M. Angela Sasse, M. Wonham.** *The compliance budget: managing security behaviour in organisations.* ACM DL, NSPW 08 Proceedings of the 2008 workshop on New security paradigms:47-58, 2008.
4. **D. Boneh and X. Boyen.** *Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles.* Springer. Advances in Cryptology - EUROCRYPT 2004: p 223-238, May 2004.
5. **Dan Boneh and Xavier Boyen.** *Short Signatures without Random Oracles.* In Chris- tian Cachin and Jan Camenisch, editors, EUROCRYPT, volume 3027 of Lecture Notes in Computer Science, pages 56?73. Springer, 2004.
6. **E. Brickell, J Camenisch, L. Chen.** *Direct Anonymous Attestation.* ACM Digital Library (Proceedings of the 11th ACM conference on Computer and communications security): 132-145, 2014.
7. **J. Camenisch, R. Chaabouni, A. Shelat.** *Efficient Protocols for Set Membership and Range Proofs.* Springer. Advances in Cryptology - ASIACRYPT 2008: p 234-252. Dec. 2008.
8. **M. Burnside, D. Clarke, T. Mills, A. Maywah, S. Devadas, R. Rivest.** *Proxy-Based Security Protocols in Networked Mobile Resources. ACM DL. 2002.*
9. ***J. Camenisch, A. Lysyanskaya.*** *An Efficient System for Non-Transferable Anonymous Credentials with Optional Anonymity Revocation. Proceedings of EUROCRYPT 2001, LNCS 2045 (2001), 93-118.*
10. ***J. Camenisch, A. Lysyanskaya.*** *Signature Schemes and Anonymous Credentials from Bilinear Maps. Proceedings of Crypto 2004, LNCS 3152 (2004), 56-72.*
11. ***P. Chan, T. Halevi, N. Memon.*** *Glass OTP: Secure and Convenient User Authentication on Google Glass. Springer, 8976 (Lecture Notes in Computer Science): 298-308, 2015.*
12. ***M. Chase, S. Meiklejohn, G. Zaverucha.*** *Algebraic MACs and keyed-verification anonymous credentials. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 2014, pp. 1205–1216. ACM Press (2014).*
13. ***D. Chaum.*** *Security without Identification: Transaction Systems to Make Big Brother Obsolete. Communications of the ACM 28 (10), 1030-1044, 1985.*
14. ***D. Chaum, J.-H. Evertse.*** *A Secure and Privacy-Protecting Protocol for Transmitting Personal Information between Organizations. In CRYPTO '86, vol. 263 of LNCS, pp. 118–167. Springer-Verlag, 1987.*
15. ***H. Cohen (Editor), G. Frey (Editor), R. Avanzi (Editor), C. Doche (Editor), T. Lange (Editor), K. Nguyen (Editor), F. Vercauteren (Editor).*** *Handbook of Elliptic and Hyperelliptic Curve Cryptography (Discrete Mathematics and Its Applications), 1st Edition, Jul. 2005.*
16. ***R. Faragher, R. Harle.*** *An Analysis of the Accuracy of Bluetooth Low Energy for Indoor Positioning Applications. ION Publications, Proceedings of the 27th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2014): 201-210, Sep. 2014.*
17. ***S. Frankel, R. Glenn, S. Kelly.*** *The AES-CBC Cipher Algorithm and Its Use with IPsec. RFC Editor (RFC3602). Sept. 2003*
18. ***D. Gollmann.*** *Computer Security. Wiley Publishing Inc, Third Edition, 2011.*
19. ***G. Hinterwälder, C. T. Zenger, F. Baldimtsi, A. Lysyanskaya, C. Paar, W. P. Burleson.*** *Efficient E-Cash in Practice: NFC-Based Payments for Public Transportation Systems. Springer (13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10-12, 2013. Proceedings): 40-59, Vol. 7981, 2013.*
20. ***R. Impagliazzo, M. Luby.*** *One-way Functions are Essential for Complexity Based Cryptography Foundations of Computer Science 1989. 30th Annual Symposium on, Research Triangle Park NC, pp. 230-235 (1989).*
21. ***V. Kapoor, V. Sonny Abraham, R. Singh.*** *Elliptic Curve Cryptography. Ubiquity, Vol. 2008 No. 7, May 2008.*
22. ***K. Krombholz, A. Dabrowski, M. Smith, and E. Weippl.*** *Ok Glass, Leave me Alone: Towards a Systematization of Privacy Enhancing Technologies for Wearable Computing. Springer, 8976 (Lecture Notes in Computer Science): 274-280, 2015.*
23. ***P. Lantz, B. Johansson, M. Hell, and B. Smeets.*** *Visual Cryptography and Obfuscation: A Use-Case for Decrypting and Deobfuscating Information using Augmented Reality*

(c) List of references

Figure 30: Website additional features

D Appendix. Example of External Entity's Implementation

This appendix presents the detailed hardware and software implementation of an example of external entity compatible with the developed system.

D.1 Example of External Entity's Implementation - Hardware Design

The hardware needed to build this entity is close to the circuitry composing the wearable and the beacon presented in Figs.(3.13) of section 3.2.3.

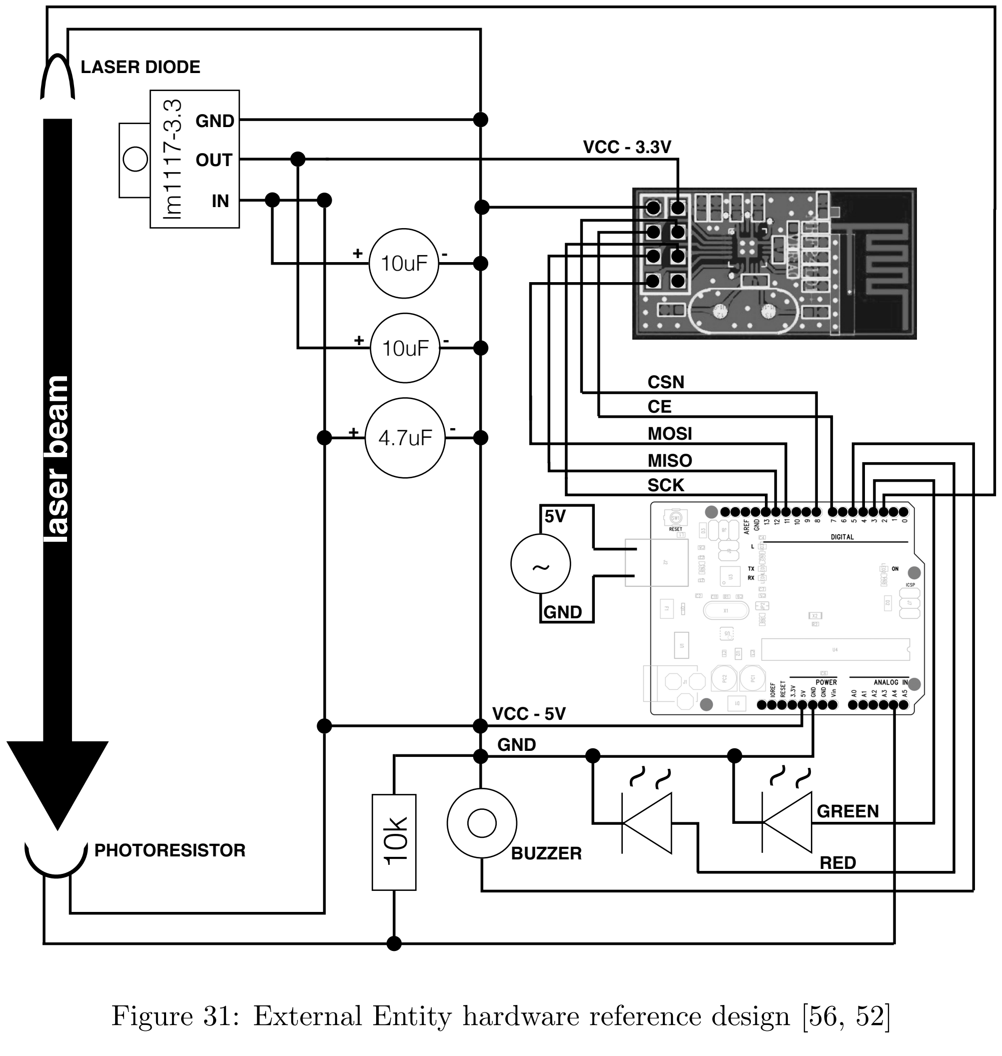


Figure 31: External Entity hardware reference design [56, 52]

The Arduino powers the nRF24 module though the usual lm1117-3.3 voltage regulator and activates a Laser Diode (LD) simulating a closed door. The LD is oriented directly toward a photoresistor that is connected to the Arduino's analog PIN A4: as long as the LD is activated and the photoresistor registers a high light's concentration, the door is closed, the system is armed, and no intruders are detected. If an intruder steps through the doorand interrupts the laser beam, the buzzer (simulating an alarm) is activated. The red and green LEDs represent the door's status.

This example of external entity has been realised with the exact same base components as the server, the wearable or the beacon. The additional hardware as the LD, the photoresistor, the buzzer, the resistor and the LEDs come from [83], [84], [85], [86], and [87], respectively.

D.2 Example of External Entity's Implementation - Software Design

The software behaviour of this external entity operates as shown by the FSM of Fig.(32). Once the entity is powered up, it arms the alarm, closes the door, and enters in a main loop waiting for a server's request and verifying the laser's feedback. Then, as explained in the last paragraph

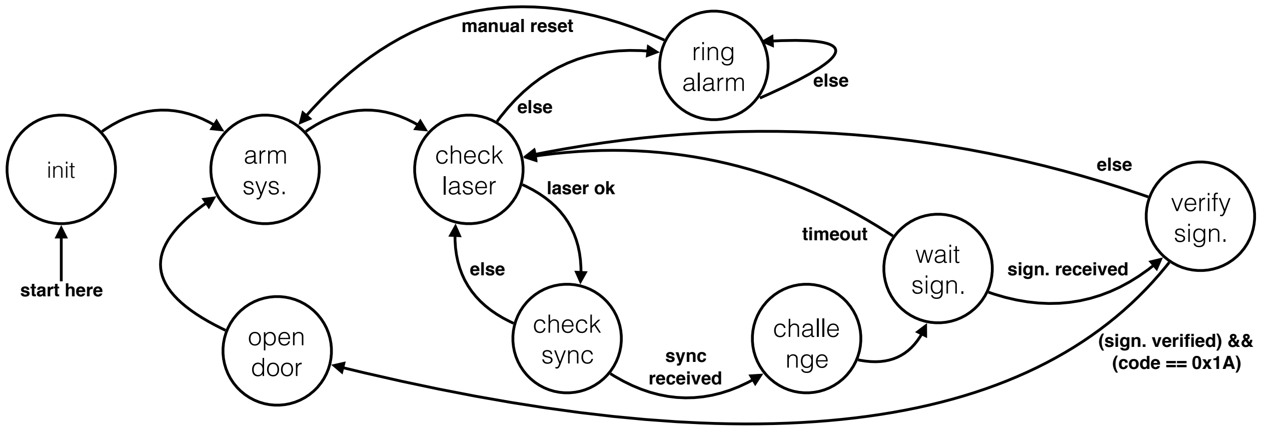


Figure 32: External Entity software FSM

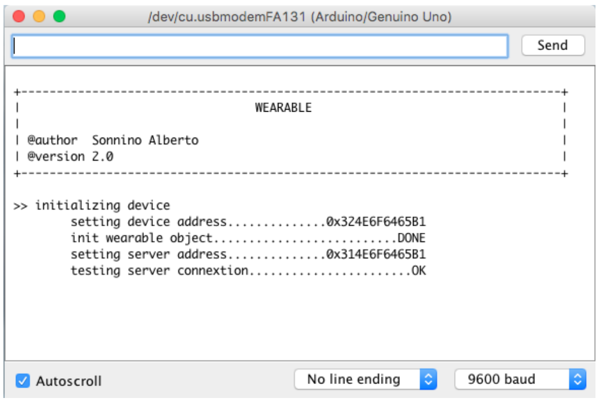
of section section 3.1.5, when the server contacts the external entity, it sends a 7-byte *action-sync* packet composed of the byte 0x17 followed by its RX address. Then, the external entity challenges the server and waits for a signed request asking to open its specific door (i.e., the door associated with the action code 0x1A). If the signature's verification succeeds, the laser beam is interrupted for a time equal to `OPEN_DELAY`, then the system is armed again.

E Appendix. Final Prototype's Screenshots

This appendix shows some screenshots of the prototype's console. The first pictures present the boot message of each entity, then the complete actions' chain is illustrated from the reception of new credentials to the credentials' showing phase.

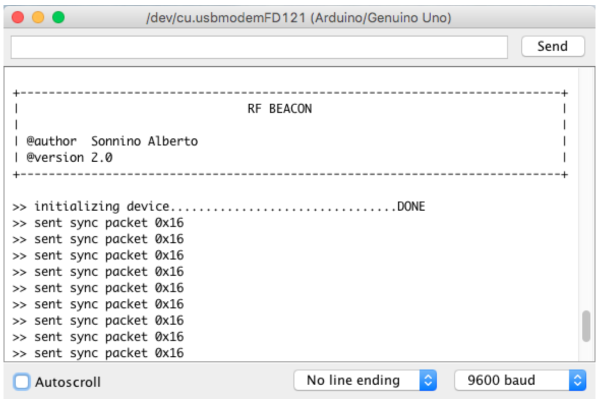
E.1 Final Prototype's Screenshots - Boot Messages

Fig.(33) shows the boot messages appearing when starting the system. The beacon, enters in a main loop issuing *location-sync* packets until a wearable answers it. For testing purposes, the server has been configured to issue the credentials when turned on (see Fig.(33c)).



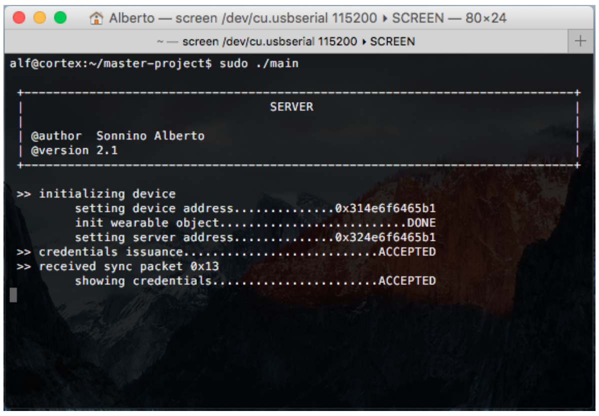
```
+---------------------------------------------------------------------+
|                              WEARABLE                               |
|                                                                     |
| @author  Sonnino Alberto                                            |
| @version 2.0                                                        |
+---------------------------------------------------------------------+

>> initializing device
        setting device address.............0x324E6F6465B1
        init wearable object.........................DONE
        setting server address.............0x314E6F6465B1
        testing server connextion......................OK
```

(a) Wearable's boot message



```
+---------------------------------------------------------------------+
|                              RF BEACON                              |
|                                                                     |
| @author  Sonnino Alberto                                            |
| @version 2.0                                                        |
+---------------------------------------------------------------------+

>> initializing device...............................DONE
>> sent sync packet 0x16
>> sent sync packet 0x16
>> sent sync packet 0x16
>> sent sync packet 0x16
>> sent sync packet 0x16
>> sent sync packet 0x16
>> sent sync packet 0x16
>> sent sync packet 0x16
>> sent sync packet 0x16
```

(b) Beacon's boot message


Alberto — screen /dev/cu.usbserial 115200 ▸ SCREEN — 80×24
~ — screen /dev/cu.usbserial 115200 ▸ SCREEN

```
alf@cortex:~/master-project$ sudo ./main

 +---------------------------------------------------------------------+
 |                              SERVER                                 |
 |                                                                     |
 | @author  Sonnino Alberto                                            |
 | @version 2.1                                                        |
 +---------------------------------------------------------------------+

 >> initializing device
        setting device address.............0x314e6f6465b1
        init wearable object.........................DONE
        setting server address.............0x324e6f6465b1
 >> credentials issuance.........................ACCEPTED
 >> received sync packet 0x13
        showing credentials......................ACCEPTED
```

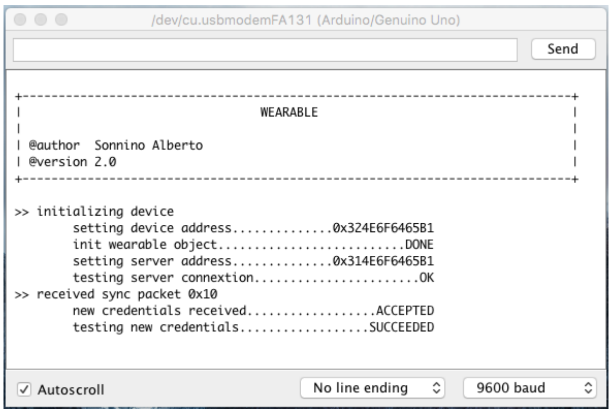
(c) Server's boot message - Screenshot

Figure 33: Boot messages

Both the server and the wearable display the default registered TX and RX addresses, and as indicated in section 3.2.3, the wearable tests its connection with the server.

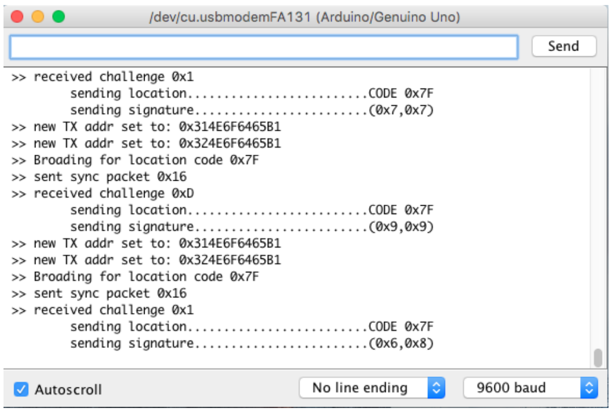
E.2 Final Prototype's Screenshots - Communication Chain

Fig.(34) below depicts the complete prototype's communication chain. It stats with the wearable receiving new credentials (Fig.(34a)) and challenging the beacon (Fig.(34b)). Then, the wearable shows its credentials to the server (Figs.(34c) and (34d)).



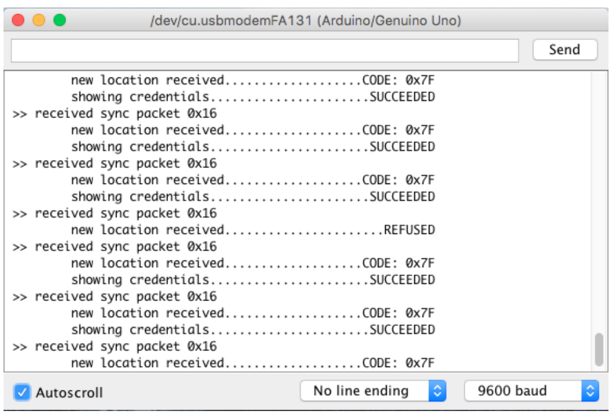
```
+----------------------------------------------------------------------+
|                              WEARABLE                                |
|                                                                      |
| @author  Sonnino Alberto                                             |
| @version 2.0                                                         |
+----------------------------------------------------------------------+

>> initializing device
       setting device address.............0x324E6F6465B1
       init wearable object.........................DONE
       setting server address.............0x314E6F6465B1
       testing server connextion......................OK
>> received sync packet 0x10
       new credentials received.................ACCEPTED
       testing new credentials.................SUCCEEDED
```

(a) Wearable getting new location



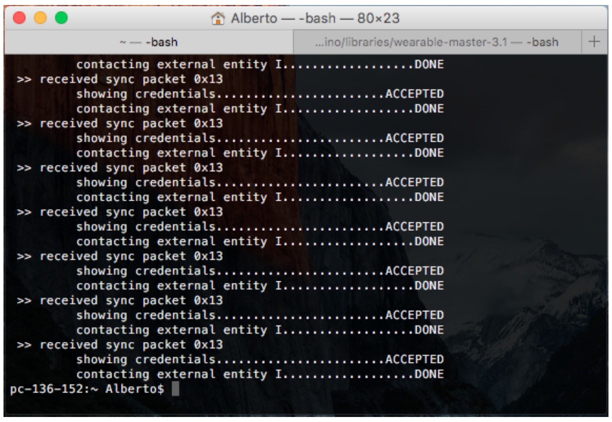
```
>> received challenge 0x1
       sending location.......................CODE 0x7F
       sending signature......................(0x7,0x7)
>> new TX addr set to: 0x314E6F6465B1
>> new TX addr set to: 0x324E6F6465B1
>> Broading for location code 0x7F
>> sent sync packet 0x16
>> received challenge 0xD
       sending location.......................CODE 0x7F
       sending signature......................(0x9,0x9)
>> new TX addr set to: 0x314E6F6465B1
>> new TX addr set to: 0x324E6F6465B1
>> Broading for location code 0x7F
>> sent sync packet 0x16
>> received challenge 0x1
       sending location.......................CODE 0x7F
       sending signature......................(0x6,0x8)
```

(b) Beacon sending signed location



```
       new location received..................CODE: 0x7F
       showing credentials.....................SUCCEEDED
>> received sync packet 0x16
       new location received..................CODE: 0x7F
       showing credentials.....................SUCCEEDED
>> received sync packet 0x16
       new location received..................CODE: 0x7F
       showing credentials.....................SUCCEEDED
>> received sync packet 0x16
       new location received.....................REFUSED
>> received sync packet 0x16
       new location received..................CODE: 0x7F
       showing credentials.....................SUCCEEDED
>> received sync packet 0x16
       new location received..................CODE: 0x7F
       showing credentials.....................SUCCEEDED
>> received sync packet 0x16
       new location received..................CODE: 0x7F
```

(c) Wearable showing credentials



```
        contacting external entity I..................DONE
 >> received sync packet 0x13
        showing credentials.......................ACCEPTED
        contacting external entity I..................DONE
 >> received sync packet 0x13
        showing credentials.......................ACCEPTED
        contacting external entity I..................DONE
 >> received sync packet 0x13
        showing credentials.......................ACCEPTED
        contacting external entity I..................DONE
 >> received sync packet 0x13
        showing credentials.......................ACCEPTED
        contacting external entity I..................DONE
 >> received sync packet 0x13
        showing credentials.......................ACCEPTED
        contacting external entity I..................DONE
 >> received sync packet 0x13
        showing credentials.......................ACCEPTED
        contacting external entity I..................DONE
 >> received sync packet 0x13
        showing credentials.......................ACCEPTED
        contacting external entity I..................DONE
pc-136-152:~ Alberto$
```

(d) Server verifying credentials

Figure 34: Communication chain - Screenshots

F Appendix. Details of the Prototype's Hardware Total Cost

This appendix shows the cost's details of all the components needed to assemble the prototyped system[12]. Most of them come from Amazon UK and are referenced in [72, 73, 78, 79, 80, 82, 81].

Component	**Price**	**Quantity**
(Unofficial) Arduino Board	£3.39	2
Raspberry PI B+	£31.60	1
nRF24L01+	$\sim$ £1	3
lm1117-3.3	$\sim$ £0.20	3
Capacitors (10uF and 4.7uF)	$\sim$ £0.01	9
Potentiometer	$\sim$ £0.15	1
Male-to-Female Jumpers	$\sim$ £0.03	27
Total Cost:	**£43.01**	

Table 8: Prototype's hardware total cost detail [72, 73, 78, 79, 80, 82, 81]

[12]Some irrelevant costs like ethernet cables, plug adapters, USB cables and breadboards have been neglected.

G Appendix. Potential Attacks and Mitigations

This appendix is dedicated to the prototype's limitations and vulnerabilities. More specifically, four possible attacks are presented as well as some possible mitigations. The scope of this analysis is to know how to protect as much as possible a future industrial system.

G.1 Potential Attacks and Mitigations - Low Noise Amplifier Based Attack

The first and simplest attack consists in amplifying the power emitted by the localisation beacons in order to reach a far away wearable. To realise this attack, two Low Noise Amplifiers (LNA) are needed to amplify all 2.4GHz radio waves, without amplifying the associated radio noises. The scope of this manipulation is to make the wearable believe that it is close to a localisation beacon, and therefore, making it start the credentials' showing phase even if the user is far away from the access point. Moreover, such attacks are very cheap since LNAs as the one shown in Fig.(35) can be bough for less than £0.50 [44, 59]. A sketch of this attack is shown in Fig.(36) below. However, this trick is easily detectable. Indeed, amplifying

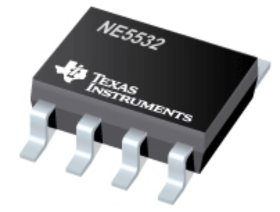


Figure 35: Example of Texas Instruments LNA chip [59]

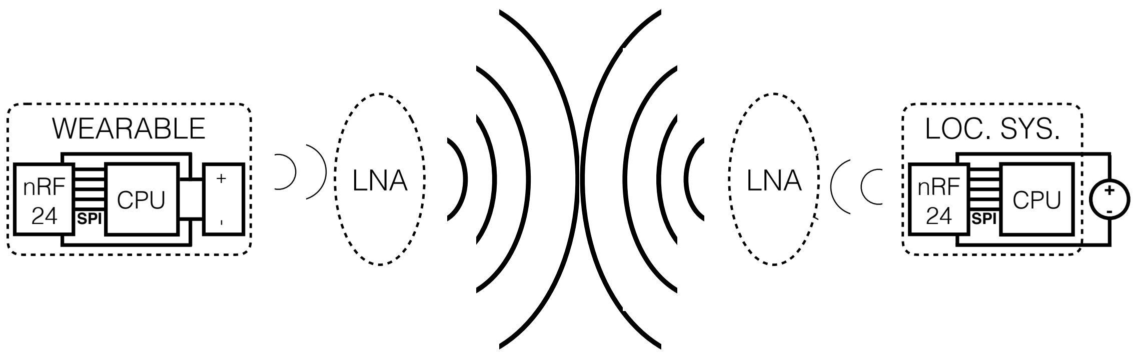


Figure 36: Sketch of LNA based attack

radio waves generates lots of signals that are unlikely to remain undetected for long. To mitigate this threat, one possible solution could be to exploit the other localisation beacons to detect it. Indeed, since most of the time they only emit low-power sync packets and wait for a response, their code could be easily adjusted to detect any abnormal radio activity on the 2.4GHz radio band and warn the main server if any. In other words, the beacon's FSM depicted in Fig.(3.16) of section 3.2.4 would become as illustrated by Fig.(37) below.

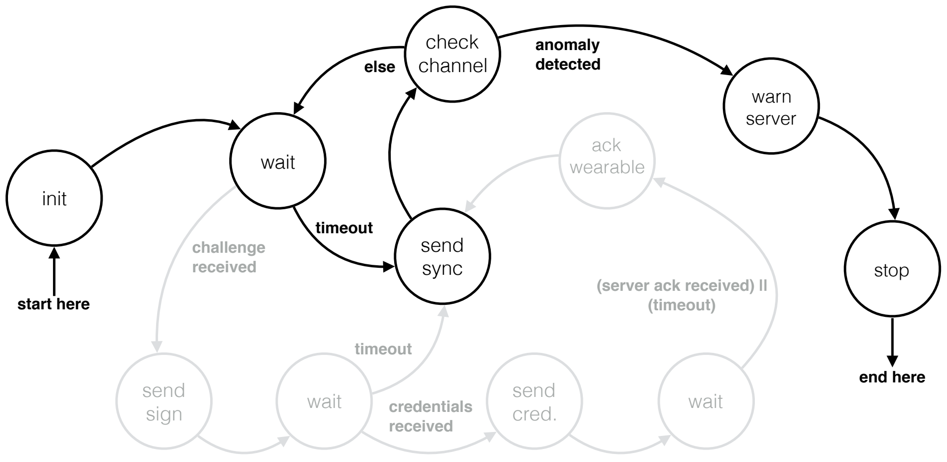


Figure 37: Modified localisation beacon's FSM

G.2 Potential Attacks and Mitigations - Man-In-The-Middle Attack

A more advanced version of the above attack consists in implementing a Man-In-The-Middle (MITM) between a localisation beacon and the wearable. This has the advantage of not being detectable as the LNA based attack but comes at the price of a higher sophistication and cost. As shown in Fig.(38), this attack demands two actors: one close to the victim and one close to the targeted localisation beacon. The first step is to send a sync-location packet to the victim's wearable. The device will then believe being in proximity of a beacon and issue a challenge. At this point, the first MITM operator has to quickly (before the RX timeout occurs) transmit the challenge to the second operator, who will then present it

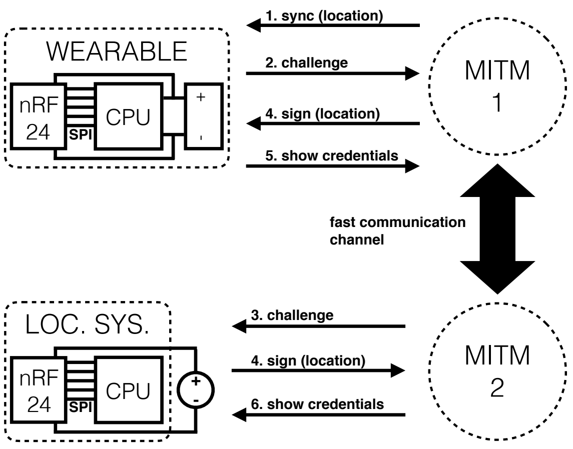


Figure 38: MITM attack sketch

to the beacon. Then, the beacon will output a legitimate signature on that challenge, and the second MITM operator must quickly retransmit it to the first operator. Finally, the signature is provided to the wearable, who will start showing the associated credentials.

To defeat this attack, the most straightforward mitigation is to reduce the timeout values in order to make impossible for the attacker to retransmit the information from one MITM operator to the other in time (but this requires reliable hardware). The timeouts' optimal values are discussed at the end of this appendix.

G.3 Potential Attacks and Mitigations - Temperature Fault Attack

This attack results from the observation made in section 3.3. In fact, experiments showed that when the wearable is heated above 40°C, the program flow may become unpredictable. Therefore, the FSM shown in Fig.(3.15) of section 3.2.3 is no longer applicable. More specifically, the situation of interest is the one pictured in Fig.(39). Indeed, as specified in section 3.3, this situation happens with a probability of about 0.5% which is not huge, but big enough to be exploited. When this situation occurs, the wearable starts showing its credentials without the need of a localisation beacon in proximity.

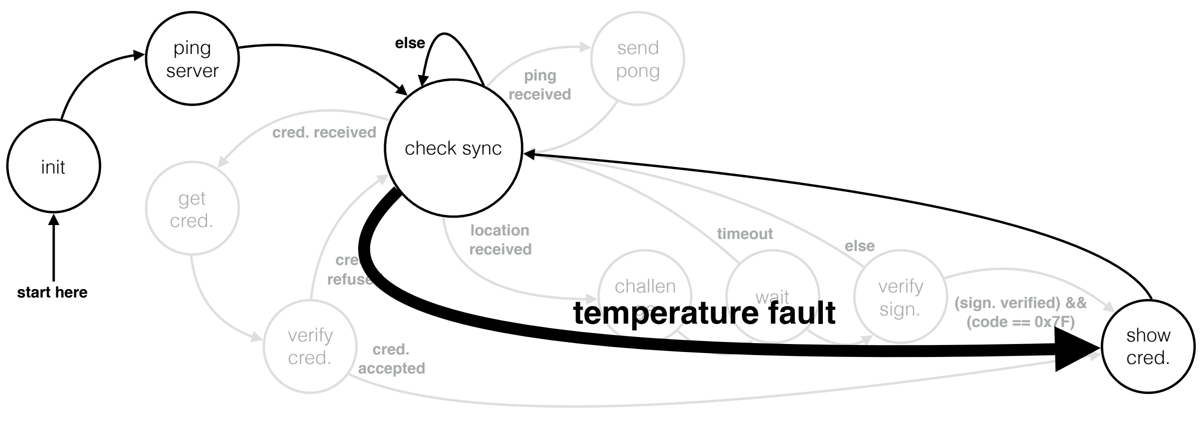


Figure 39: Possible wearable's FSM under high temperatures

However, it is clear that in practice it is not trivial to heat a person's wearable device to such temperatures without raising any suspicion, but this can naturally happen in some very hot countries. The only easy mitigation to that problem is to build the wearable with reliable hardware, and more particularly, with resistant resonators and capacitors.

G.4 Potential Attacks and Mitigations - Denial of Service Attack

The last attack discussed in this paper is the Denial of Service (DoS) attack, which is the more important. Despite the system has been proved resistant against jamming, an attacker could exploit the TX and RX timeouts (see Tab.(3.6) of appendix 3.3.4) to keep busy the server or the beacon. Indeed, one of the easiest way to exploit this weakness is to attack the server's method `check_credentials`, which is structured as depicted by the pseudocode of Fig.(40). To fully take advantage of this weakness, the attacker could send an showing-sync packet to the server, wait for *slightly* less than 1 s, and then send k random byte (considering some k-byte credentials) to make the server believe it received the first load and make him access the second while loop. In that case, the server will be busy for about 2 s. Therefore, the attacker would only need to send (7+k) bytes (sync + credentials) each 2 s to keep the server busy. The same attack could be applied to the wearable's `get_credentials` method, but this is hardly impossible to exploit since the credentials' issuance is done in private.

```
while (not RX_TIMEOUT)
    if (first load received)
        while (not RX_TIMEOUT)
            if (second load received)
            {
                [perform some actions]
                return
            }
return
```

Figure 40: Pseudocode exploitable for DoS

The beacon is the weakest link concerning DoS attacks. Indeed, when acting as an interface between the wearable and the server, 3 RX and 1 TX timeouts are involved. Exploiting these timeouts similarly to above, the adversary could keep the beacon busy for 3.2 s at the cost of k+1 bytes (credentials + ack).

The first easy enhancement to mitigate DoS attacks is to wire all the connection between the server and the beacons, but this could be expensive and greatly reduces the flexibility of the system. An alternative solution consists in optimising the timeout values to have the best trade off between low theoretical latency and good success rates in oder to force the adversary to invest much more resources in his attack. These optimal timeout values are discussed in the next section.

Note that the advantage of using raw radio waves instead of Wi-Fi is that it forces the adversary to come close to the victim to perform the DoS attack, contrarily to the general case scenario where the attacker can safely stay home and operate through the internet. Being aware of the high sensibility of embedded systems against DoS attacks [32], this is one of the main reasons why the RF 2.4GHz waves have been emphasised during this project.

G.5 Potential Attacks and Mitigations - Optimal Timeouts Values

An optimal RX timeout value can be compute by observing the box plots in Figs.(3.17) and (3.18) of section 3.3.4. Indeed, a possible solution consists in choosing the value `RX_TIMEOUT` as follows:

$$\mathsf{RX\_TIMEOUT} = \frac{Q_3}{n_{RX}} \tag{13}$$

where Q_3 is the statistical third quartile and n_{RX} is the number of RX timeouts involved in the method. In fact, choosing the RX timeout as shown in Eq.(13) guarantees a success rate of about 75%. For instance, the box plot of the method `check_credentials` indicates to choose a RX timeout of

$$RX_{check\_cred} = \frac{984}{2} = 492ms \tag{14}$$

Therefore, after this optimisation, the attacker will need to send (7+k) bytes every 984 ms (on average, (14+2k) bytes very 2 s) to keep the server busy, which requires an investment of twice more resources than before.

Unfortunately, the TX value has to be determined empirically. Its purpose it to mark a transmitter's delay between each sent packet in order to be sure the receiver got it. This value will therefore strictly depend on the quality of the radio channel and of the receiver's hardware.

H Appendix. Proposition of Revocation Mechanisms

This appendix discusses some possibilities of revocation mechanisms for the considered architecture along with their main advantages and drawbacks. In general, revocation mechanisms are used to take back a right previously granted. In the current case, it is about taking back some credentials from a given wearable in order to prevent its user from acting at a previously authorised privilege level.

Note that when considering revocation in this architecture, the wearable's user automatically becomes a potential attacker since he might try to keep its credentials despite revocation. The Threat Model depicted in Tab.(2.2) of section 2.2 has therefore to be updated.

- **Possibility 1 :** Physically take back and reset the device.

This solution consists in reaching out the user and convince him to give back or reset his device. Depending on the situation, the above could be an acceptable solution. However, it is not always easy to convince a potentially dishonest user to gently give back his privileges.

- **Possibility 2 :** Reset all the devices and start over.

In extreme situations, an alternative to the above mechanism is to reset all devices and start over with the issuance process. Despite this solution is extremely effective, it is often the most expensive. In the current case, this would consist in deleting the aMAC key, generating a new one, and asking each authorised user to come over to start the credentials' issuance process from the beginning.

- **Possibility 3 :** Embed an expiration date.

The third solution consists in programming the devices with an embedded expiration date such that once reached, the device stops working. Note that this mechanism works only under the hypothesis that the devices are tamper-proof, as it is assumed in this architecture (see section 2.2); i.e., the user cannot modify the device's expiration date. The drawback of this mechanism is its poor flexibility. Indeed, the expiration date has to be inserted when programming the device and cannot be easily changed afterwards.

- **Possibility 4 :** Modify the wearable's software FSM to accept revocation commands.

This last solution is more specific to the current prototype's implementation. Roughly speaking, a public UID number has to be associated with each wearable (obviously, to protect its anonymity, the wearable never shows this ID during the credentials' showing phase). Then, the wearable's Finite State Machine (FSM) depicted in Fig.(3.15) of section 3.2.3 should be modified in order to include a state where the device checks whether the server commanded a revocation of privileges. This modified FSM is illustrated in Fig.(41) below.

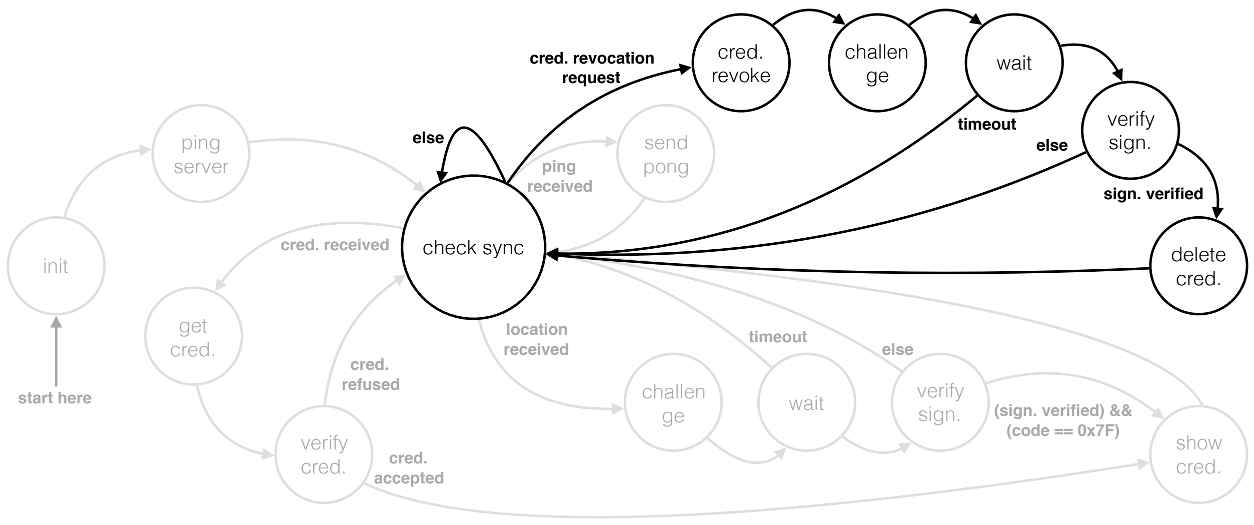


Figure 41: Wearable's FSM enhancement for revocation mechanism

More specifically[13], when the main server wishes to revoke a wearable's credential, it sends a *revocation sync* packet containing the target UID to each wearable. At that point, the concerned device reacts and challenges the server with a fresh random nonce. Then, the server answers with a signature made of a hash of the concatenation of this random nonce, the revocation command, and the device's UID. Finally, the wearable verifies the signature, and if it is valid it deletes its credentials. As in the previous case, this mechanism works only under the assumption of tamper-proof devices. Note that this server-wearable revocation exchange is very similar to the discussion between the server and the external entity presented in section 3.1.5.

[13]For easier understanding what follows should be read in parallel with section 3.2.3.

The main advantages of this revocation method is its flexibility. Moreover, the above mechanism could be easily and efficiently implemented on the wearables. However its main drawback is the fact that the wearable would then need to perform a long-range communication, while the original design avoided it for efficiency and security reasons. Indeed, such long-range transactions are power-consuming and more vulnerable to some of the attacks described in appendix G.2.

I Appendix. Industrial Wearable - Printed Circuit Board

This appendix presents a suggestion of Printed Circuit Board (PCB) for an industrial application of the presented wearable device.

As explained in section 3.4, the industrial wearable could be implemented around a Microchip PIC32MZ microcontroller. In particular, the PIC32MZ1024ECH144 has been chosen because, in addition of coming in a LQFP package (which is relatively easy to solder), it is a good trade off between price, size and performances. However, the presented PCB remains compatible with any LQFP package of PIC32MZ0512EC(E/F/K)144, PIC32MZ1024EC(G/H/M/E/F/K)144 and PIC32MZ2048EC(G/H/M)144 [42]. Fig.(42) below depicts the complete schematic of the design.

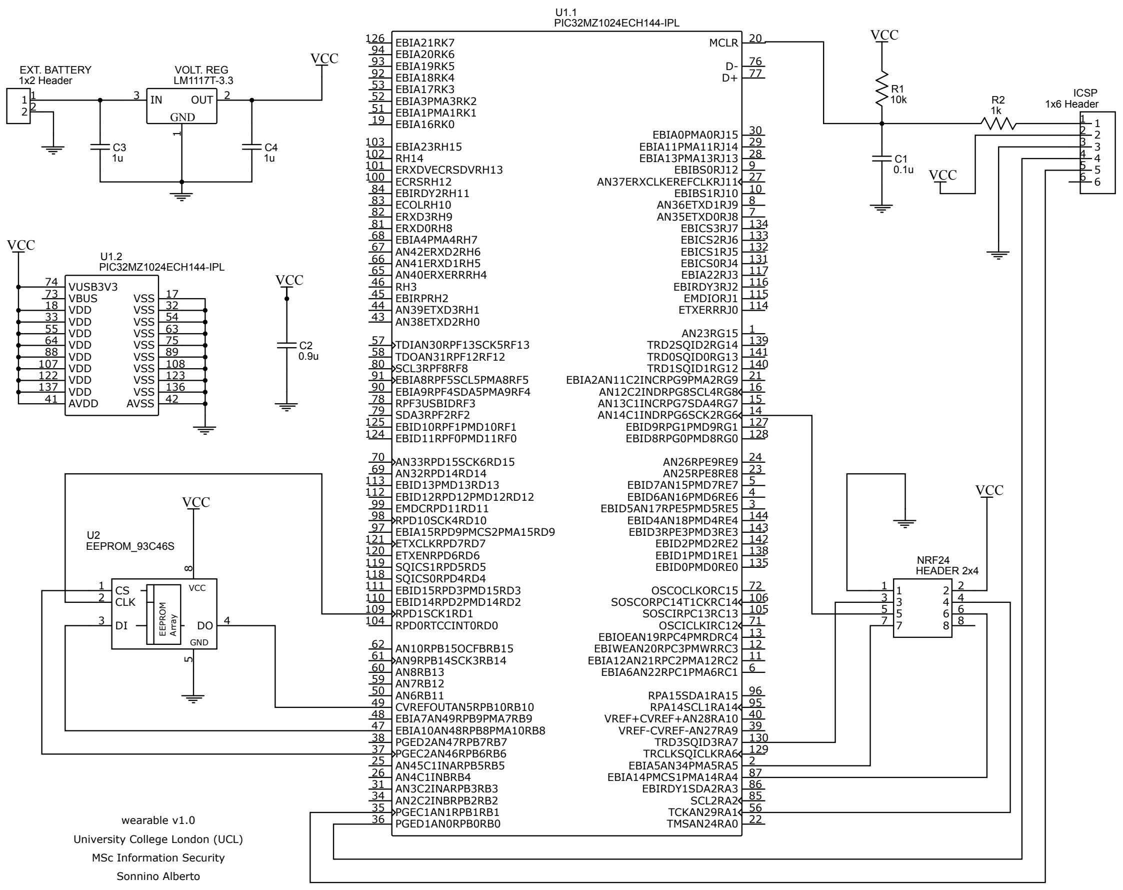


Figure 42: Industrial wearable's PCB schematics

The microcontroller is powered through the same 3.3V voltage regulator used in the prototypes and the SPI interface with the nRF24 module is achieved through the PINs `RA1`, `RA4`, `RA5`, `RA7` and `SCK2`. As discussed in section 3.2.3, the final design is equipped with an EEPROM memory (non-volatile) in order to permanently store the user's credentials. This memory is interfaced with another SPI from PINs `SCK1`, `RB10`, `RB8` and `RB6`, as indicated in Fig.(42). PINs `PGEC1`, `PGED2` and `MCLR` are used to program the PIC through the classic Microchip ICSP. The PCB pictured in Fig.(43) has been realised with the free online software *easyEDA* and can be purchased through that same website for less than £3[14] [71]. Figs.(44a) and (44b) present respectively a preview of the PCB's front and back side, as it would come after manufacturing.

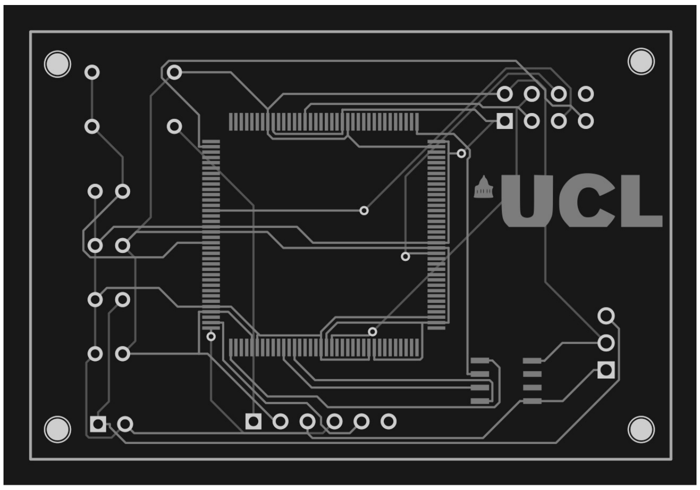
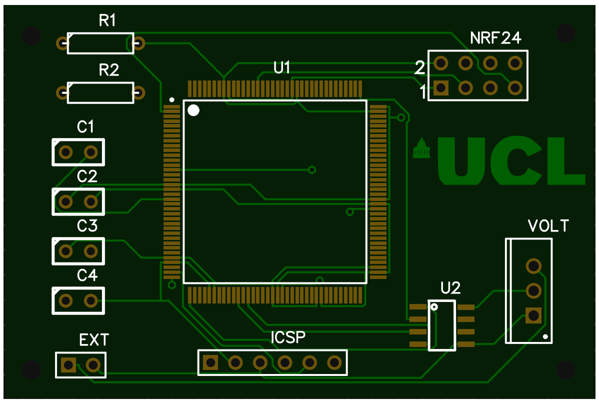
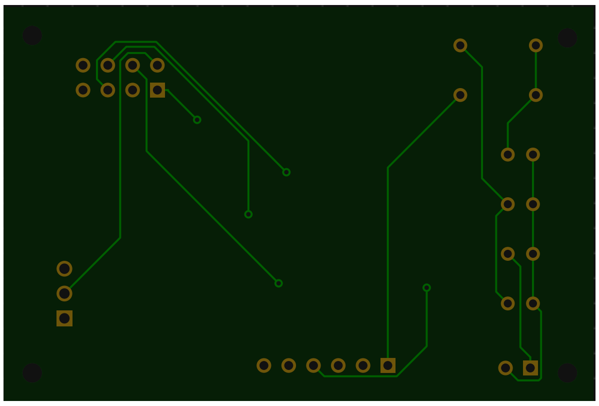


Figure 43: Industrial wearable's Printed circuit board



(a) PCB preview - Front side



(b) PCB preview - Back side

Figure 44: Industrial wearable's PCB preview

[14]Provided that at least five copies of the PCB are purchased. Shipping fees not included.

Bibliography

[1] R. Anderson. Security Engineering. Wiley Publishing Inc, Second Edition, 2008.

[2] F. Baldimtsi, A. Lysyanskaya. Anonymous Credentials Light. ACM Digital Library (Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security): 1087-1098, 2013.

[3] A. Beautement, M. Angela Sasse, M. Wonham. The compliance budget: managing security behaviour in organisations. ACM DL, NSPW 08 Proceedings of the 2008 workshop on New security paradigms:47-58, 2008.

[4] Bell, D. The Bell-Lapadula Model. Journal of computer security 4.2 (1996): 3. APA

[5] D. Boneh and X. Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. Springer. Advances in Cryptology - EUROCRYPT 2004: p 223-238, May 2004.

[6] D. Boneh and X. Boyen. Short Signatures without Random Oracles. In Christian Cachin and Jan Camenisch, editors, EUROCRYPT, volume 3027 of Lecture Notes in Computer Science, pages 56–73. Springer, 2004.

[7] E. Brickell, J Camenisch, L. Chen. Direct Anonymous Attestation. ACM Digital Library (Proceedings of the 11th ACM conference on Computer and communications security): 132-145, 2014.

[8] M. Burnside, D. Clarke, T. Mills, A. Maywah, S. Devadas, R. Rivest. Proxy-Based Security Protocols in Networked Mobile Resources. ACM DL, 2002.

[9] J. Camenisch, R. Chaabouni, A. Shelat. Efficient Protocols for Set Membership and Range Proofs. Springer. Advances in Cryptology - ASIACRYPT 2008: p 234-252, Dec. 2008.

[10] J. Camenisch, A. Lysyanskaya. An Efficient System for Non-Transferable Anonymous Credentials with Optional Anonymity Revocation. Proceedings of EUROCRYPT 2001, LNCS 2045 (2001), 93-118.

[11] J. Camenisch, A. Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. Proceedings of Crypto 2004, LNCS 3152 (2004), 56-72.

[12] P. Chan, T. Halevi, N. Memon. Glass OTP: Secure and Convenient User Authentication on Google Glass. Springer, 8976 (Lecture Notes in Computer Science): 298-308, 2015.

[13] M. Chase, S. Meiklejohn, G. Zaverucha. Algebraic MACs and keyed-verification anonymous credentials. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 2014, pp. 1205–1216. ACM Press (2014).

[14] D. Chaum. Security without Identification: Transaction Systems to Make Big Brother Obsolete. Communications of the ACM 28 (10), 1030-1044, 1985.

[15] D. Chaum, J.-H. Evertse. A Secure and Privacy-Protecting Protocol for Transmitting Personal Information between Organizations. In CRYPTO '86, vol. 263 of LNCS, pp. 118–167. Springer-Verlag, 1987.

[16] H. Cohen (Editor), G. Frey (Editor), R. Avanzi (Editor), C. Doche (Editor), T. Lange (Editor), K. Nguyen (Editor), F. Vercauteren (Editor). Handbook of Elliptic and Hyperelliptic Curve Cryptography (Discrete Mathematics and Its Applications). 1st Edition, Jul. 2005.

[17] R. Faragher, R. Harle. An Analysis of the Accuracy of Bluetooth Low Energy for Indoor Positioning Applications. ION Publications, Proceedings of the 27th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2014): 201-210, Sep. 2014.

[18] S. Frankel, R. Glenn, S. Kelly. The AES-CBC Cipher Algorithm and Its Use with IPsec. RFC Editor (RFC3602), Sept. 2003

[19] D. Gollmann. Computer Security. Wiley Publishing Inc, Third Edition, 2011.

[20] G. Hinterwälder, C. T. Zenger, F. Baldimtsi, A. Lysyanskaya, C. Paar, W. P. Burleson. Efficient E-Cash in Practice: NFC-Based Payments for Public Transportation Systems. Springer (13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10-12, 2013. Proceedings): 40-59, Vol. 7981, 2013.

[21] R. Impagliazzo, M. Luby, One-way Functions are Essential for Complexity Based Cryptography, Foundations of Computer Science 1989. 30th Annual Symposium on, Research Triangle Park NC, pp. 230-235 (1989).

[22] V. Kapoor, V. Sonny Abraham, R. Singh. Elliptic Curve Cryptography. Ubiquity, Vol. 2008 No. 7, May 2008.

[23] K. Krombholz, A. Dabrowski, M. Smith, and E. Weippl. Ok Glass, Leave me Alone: Towards a Systematization of Privacy Enhancing Technologies for Wearable Computing. Springer, 8976 (Lecture Notes in Computer Science): 274-280, 2015.

[24] P. Lantz, B. Johansson, M. Hell, and B. Smeets. Visual Cryptography and Obfuscation: A Use-Case for Decrypting and Deobfuscating Information using Augmented Reality. Springer, 8976 (Lecture Notes in Computer Science): 261-273, 2015.

[25] K Lauter. The Advantages of Elliptic Curve Cryptography for Wireless Security. IEEE Wireless communications, Vol. 11 Issue 1 pp. 62-67, Feb. 2004.

[26] S. Lemsitzer, J. Wolkerstorfer, N. Felber, M. Braendli. Multi-gigabit GCM-AES Architecture Optimized for FPGAs. Springer. Cryptographic Hardware and Embedded Systems - CHES 2007. Volume 4727 (Lecture Notes in Computer Science): 227-238, Sept. 2007.

[27] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym Systems. In Selected Areas in Cryptography, vol. 1758 of LNCS, 1999.

[28] D. McGrew, Galois Counter Mode. Springer. Encyclopedia of Cryptography and Security: 506-508, 2011.

[29] A. J. Menezes, T. Okamoto, S. A. Vanstone, ”Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field”, IEEE Transactions on Information Theory, Vol. 39 Issue 5 pp. 1639-1646, Sep. 1993.

[30] V. S. Miller, Use of Elliptic Curves in Cryptography, Lecture Notes in Computer Science, 218, pp. 417-426 (2000).

[31] T. Mills, M. Burnside, J. Ankcorn and S. Devadas. A proxy-based architecture for secure networked wearable devices. Core, 2001.

[32] H. Modares, R. Salleh, A. Moravejosharieh. Overview of Security Issues in Wireless Sensor Networks. IGI Global, PeerReviewed, 2010.

[33] V. Mott, K. Caine. Users' Privacy Concerns About Wearables: impact of form factor, sensors and type of data collected. Springer, 8976 (Lecture Notes in Computer Science): 231-244, 2015.

[34] D. Niculescu, B. Nath. Ad hoc Positioning System (APS) using AOA. IEEE. INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies (Volume:3): 1734-1743. Mar. 2003.

[35] N. P. Smart. The Discrete Logarithm Problem on Elliptic Curves of Trace One. Journal of Cryptology, Vol. 12 Issue 3 pp. 193-196, Jun. 1999.

[36] M. Sterckx, K. Arenberg , B. Gierlichs, B. Preneel, I. Verbauwhede. Efficient Implementation of Anonymous Credentials on Java Card Smart Cards. IEEE (2009 First IEEE International Workshop on Information Forensics and Security (WIFS)): 106-110, Dec. 2009.

[37] D. K. Yadav, B. Ionascu, S. V. K. Ongole, A. Roy, N. Memon. Design and Analysis of Shoulder Surfing Resistant PIN Based Authentication Mechanisms on Google Glass. Springer, 8976 (Lecture Notes in Computer Science): 281-297, 2015.

[38] ISO/IEC 9798-3: 1997, Information technology - Security techniques - Entity authentication - Part 3: Mechanisms using digital signature techniques.

[39] ARM. ARM1176JZF-S. Revision: r0p7. Technical Reference Manual.

[40] AVR. ATmega48A/PA/88A/PA/168A/PA/328/P. ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES IN-SYSTEM PROGRAMMABLE FLASH.

[41] AVR. 8-bit Microcontroller with 8/16/32K Bytes of ISP Flash and USB Controller. ATmega8U2 ATmega16U2 ATmega32U2. 7799ES–AVR–09/2012.

[42] Microchip Technology Inc. PIC32MZ Embedded Connectivity (EC) Family. 32-bit MCUs (up to 2 MB Live-Update Flash and 512 KB SRAM) with Audio and Graphics Interfaces, HS USB, Ethernet, and Advanced Analog. http://ww1.microchip.com/downloads/en/DeviceDoc/60001191F.pdf. isited on the 7th of Aug. 2016.

[43] Nordic Semiconductor. nRF24L01+ Single Chip 2.4GHz Transceiver. Preliminary Product Specification v1.0.

[44] Texas Instruments. NE5532x, SA5532x Dual Low-Noise Operational Amplifiers. NE5532, NE5532A, SA5532, SA5532A. SLOS075J – NOVEMBER 1979 – REVISED JANUARY 2015

[45] Texas Instruments. LM1117 800-mA Low-Dropout Linear Regulator. SNOS412N – FEBRUARY 2000 – REVISED JANUARY 2016

[46] Arduino. nRF24l01+. http://playground.arduino.cc/InterfacingWithHardware/Nrf24L01. Visited on the 7th of Aug. 2016.

[47] G. Danezis (gdanezis). petlib/examples/amacscreds.py. https://github.com/gdanezis/petlib /blob/master/examples/amacscreds.py. Visited on the 7th of Aug. 2016.

[48] NaCl: Networking and Cryptography library. https://nacl.cr.yp.to. Version 2016.03.15. Visited on the 7th of Aug. 2016.

[49] Oryx Embedded. TCP/IP Solutions for Embedded Systems. http://www.oryx-embedded.com/#&panel1-1. Visited on the 7th of Aug. 2016.

[50] Panique. php-login-advanced. https://github.com/panique/php-login-advanced. Visited on the 24th of Aug. 2016.

[51] WolfSSL. Embedded SSL Library. https://www.wolfssl.com/wolfSSL/Home.html. Visited on the 7th of Aug. 2016.

[52] Gizmosnack. Tutorial - nRF24L01 and AVR. Monday, April 8, 2013. http://gizmosnack.blogspot.co.uk/2013/04/tutorial-nrf24l01-and-avr.html. Visited on the 7th of Aug. 2016.

[53] IconFinder. Encryption, firewall, lock, safe, secure, security. https://www.iconfinder.com. Visited on the 23rd of Aug. 2016.

[54] IconFinder. Eye, hidden, invisible, off icon. https://www.iconfinder.com. Visited on the 23rd of Aug. 2016.

[55] IconFinder. Signature icons. https://www.iconfinder.com. Visited on the 23rd of Aug. 2016.

[56] Mentor Graphics. PCB Design, Analysis, & Package Integration. Arduino Uno. https://www.mentor.com/pcb/reference-designs/a000066. Visited on the 7th of Aug. 2016.

[57] Microchip, Inc. PIC32MZ2048ECH144. http://www.microchip.com/wwwproducts/en/PIC32MZ2048ECH144. Visited on the 7th of Aug. 2016.

[58] Raspberry Pi. Raspberry Pi B+ schematic. https://www.raspberrypi.org/forums/viewtopic.php?t=81922. Visited on the 7th of Aug. 2016.

[59] Texas Instruments. NE5532, Dual Low-Noise High-Speed Audio Operational Amplifier. http://www.ti.com/product/NE5532. Visited on the 7th of Aug. 2016.

[60] F. Baldimtsi. Brown University - Computer Science. CS Blog. PAY-AS-YOU-GO or How can we get private, secure and efficient payments in public transportation, Aug. 2013. http://blog.cs.brown.edu/2013/08/05/pay-you-go-or-how-can-we-get-private-secure-and-efficient-payments-public-transportation/. Visited on the 21st of Jul. 2016.

[61] Arduino. Arduino UNO & Genuino UNO. https://www.arduino.cc/en/Main/ArduinoBoardUno. Visited on the 7th of Aug. 2016.

[62] Arduino. Arduino PIN Current Limitation. http://playground.arduino.cc/Main/ArduinoPinCurrentLimitations Visited on the 7th of Aug. 2016.

[63] Eurostar. http://www.eurostar.com/uk-en. Visited on the 7th of Aug. 2016.

[64] Groupe Eurotunnel. The Channel Tunnel infrastructure. http://www.eurotunnelgroup.com/uk/the-channel-tunnel/infrastructure/. Visited on the 7th of Aug. 2016.

[65] Raspberry Pi Foundation. Raspberry Pi 1 Model B+. https://www.raspberrypi.org/products/model-b-plus/. Visited on the 7th of Aug. 2016.

[66] Apple. Xcode. Mac App Store Preview. https://itunes.apple.com/gb/app/xcode/id497799835?mt=12. Visited on the 7th of Aug. 2016.

[67] Arduino. Download the Arduino Software. https://www.arduino.cc/en/Main/Software. Visited on the 7th of Aug. 2016.

[68] Intronix. LogicPort Interpreters. http://www.pctestinstruments.com/logicport/interpreters.htm. Visited on the 7th of Aug. 2016.

[69] Microchip, Inc. MPLAB Harmony Integrated Software Framework. http://www.microchip.com/mplab/mplab-harmony. Visited on the 7th of Aug. 2016.

[70] Saleae. Debug hardware like the pros with the logic analyzer you'll love. https://www.saleae.com. Visited on the 7th of Aug. 2016.

[71] easyEDA. Do circuit simulation, PCB design, Electronic circuit design online for free. https://easyeda.com. Visited on the 21st of Aug. 2016.

[72] Amazon UK. XCSOURCE UNO R3 Rev3 Board Development Board ATmega328P CH340G AVR Arduino Compatible Board +Cable for Arduino DIY

TE113. https://www.amazon.co.uk/XCSOURCE®-Development-ATmega328P-Compatible-TE113/dp/B00SR4FLMI/ref=sr_1_10?ie=UTF8&qid=1470612600&sr=8-10&keywords=arduino. Visited on the 2nd of Jun. 2016.

[73] Amazon UK. Raspberry Pi B+ Desktop (700MHz Processor, 512MB RAM, 4x USB Port). https://www.amazon.co.uk/Raspberry-Pi-Desktop-700MHz-Processor/dp/B00LPESRUK/ref=sr_1_1?ie=UTF8&qid=1470612787&sr=8-1&keywords=raspberry+pi+B%2B. Visited on the 2nd of Jun. 2016.

[74] Amazon UK. SunFounder ESP8266 Serial Wifi Wireless Transceiver Module for Arduino UNO R3 Mega2560 Nano. https://www.amazon.co.uk/dp/B00U293Y5M/ref=sr_ph_1?ie=UTF8&qid=1470585960&sr=sr-1&keywords=wifi+module+arduino. Visited on the 2nd of Jun. 2016.

[75] Amazon UK. Ecloud Shop HC-05 wireless Bluetooth Arduino host for serial transceiver module. https://www.amazon.co.uk/Ecloud-wireless-Bluetooth-Arduino-transceiver/dp/B015CJJPR2/ref=sr_1_4?s=electronics&ie=UTF8&qid=1470586253&sr=1-4&keywords=bluetooth_module+arduino. Visited on the 2nd of Jun. 2016.

[76] Amazon UK. Xbee 2mW Module with Whip Antenna (Series ZB). https://www.amazon.co.uk/Xbee-2mW-Module-Whip-Antenna/dp/B00AQNXSV4/ref=sr_1_1?s=electronics&ie=UTF8&qid=1470586129&sr=1-1&keywords=ZigBee+module. Visited on the 2nd of Jun. 2016.

[77] Amazon UK. XCSOURCE 5 PCS 433Mhz RF Transmitter Module + Receiver Kit for Arduino ARM MCU WL TE122. https://www.amazon.co.uk/XCSOURCE-Transmitter-Receiver-Arduino-TE122/dp/B00V4ISS38/ref=sr_1_3?s=electronics&ie=UTF8&qid=1470586401&sr=1-3&keywords=433Mhz++arduino. Visited on the 2nd of Jun. 2016.

[78] Amazon UK. Kuman 10pcs nRF24L01+ 2.4GHz Antenna Wireless Transceiver RF Transceiver Module Arduino Compatible

K19. https://www.amazon.co.uk/Kuman-nRF24L01-Wireless-Transceiver-Compatible/dp/B01BVAAASY/ref=sr_1_1?s=electronics&ie=UTF8&qid=1470586467&sr=1-1&keywords=rf24+arduino. Visited on the 2nd of Jun. 2016.

[79] Amazon UK. 10pcs LM1117T-3.3 LM1117T LD1117 3.3V TO-220 Voltage Regulator. https://www.amazon.co.uk/dp/B016M5MNVA/ref=sr_ph_1?ie=UTF8&qid=1470613078&sr=sr-1&keywords=voltage+regulators+3.3. Visited on the 2nd of Jun. 2016.

[80] Amazon UK. Tenflyer Pack of 210 25 Value 0.1uF-220uF Electrolytic Capacitors Assortment Kit Set. https://www.amazon.co.uk/Tenflyer-0-1uF-220uF-Electrolytic-Capacitors-Assortment/dp/B00R43VP4Q/ref=sr_1_1?s=electronics&ie=UTF8&qid=1470613239&sr=1-1&keywords=capacitors. Visited on the 2nd of Jun. 2016.

[81] Amazon UK. 40P Conductor Male to Female Jumper Wire 20CM,40P Color Wires Ribbon Cable. https://www.amazon.co.uk/Conductor-Female-Jumper-Color-Ribbon/dp/B00ATMHU52/ref=sr_1_4?s=electronics&ie=UTF8&qid=1470613444&sr=1-4&keywords=Jumpers. Visited on the 2nd of Jun. 2016.

[82] Amazon UK. 10 Pcs 10K Ohm Top Adjustment Variable Resistors Potentiometer. https://www.amazon.co.uk/Pcs-Adjustment-Variable-Resistors-Potentiometer/dp/B00EDK2OF2/ref=sr_1_4?ie=UTF8&qid=1471200905&sr=8-4&keywords=potentiometer. Visited on the 2nd of Jun. 2016.

[83] Amazon UK. WINOMO 10pcs Mini Laser Dot Diode Module Heads WL Red 650nm 6mm 5V 5mW. https://www.amazon.co.uk/WINOMO-10pcs-Laser-Diode-Module/dp/B0151KP0JY/ref=sr_1_1?ie=UTF8&qid=1472137982&sr=8-1&keywords=laser+diode. Visited on the 4th of Jun. 2016.

[84] Amazon UK. 20PCS Photoresistor GL5528 LDR Photo Resistors Light-Dependent. https://www.amazon.co.uk/20PCS-Photoresistor-GL5528-Resistors-Light-Dependent/dp/B00GKEKMF8/ref=sr_1_1?ie=UTF8&qid=1472138061&sr=8-1&keywords=photoresistor. Visited on the 4th of Jun. 2016.

[85] Amazon UK. 12mm Dia DC 5V 2 Terminals Electronic Continuous Sound Buzzer 30pcs. https://www.amazon.co.uk/Terminals-Electronic-Continuous-Sound-Buzzer/dp/B01BNIHDNE/ref=sr_1_1?ie=UTF8&qid=1472138129&sr=8-1&keywords=buzzer+5V. Visited on the 4th of Jun. 2016.

[86] Amazon UK. 1120Pcs 1/4W 1% Metal Film Resistors Assorted Kit Set 56 Values (1 ohm 10M ohm). https://www.amazon.co.uk/1120Pcs-Metal-Resistors-Assorted-Values/dp/B00GIFDWYC/ref=sr_1_2?ie=UTF8&qid=1472138247&sr=8-2&keywords=resistor. Visited on the 4th of Jun. 2016.

[87] Amazon UK. 150 x 3mm Red Green Yellow 2 Pin LED Light Emitting Diodes. https://www.amazon.co.uk/Green-Yellow-Light-Emitting-Diodes/dp/B0087ZT24A/ref=sr_1_2?ie=UTF8&qid=1472138286&sr=8-2&keywords=leds. Visited on the 4th of Jun. 2016.

[88] Massachusetts Institute of Technology (MIT). The Cricket Indoor Location System. http://nms.csail.mit.edu/cricket/. Visited on the 6th of Jun. 2016.

[89] Pozyx Labs. Pozyx Accurate Positioning. https://www.pozyx.io. Visited on the 6th of Jun. 2016.