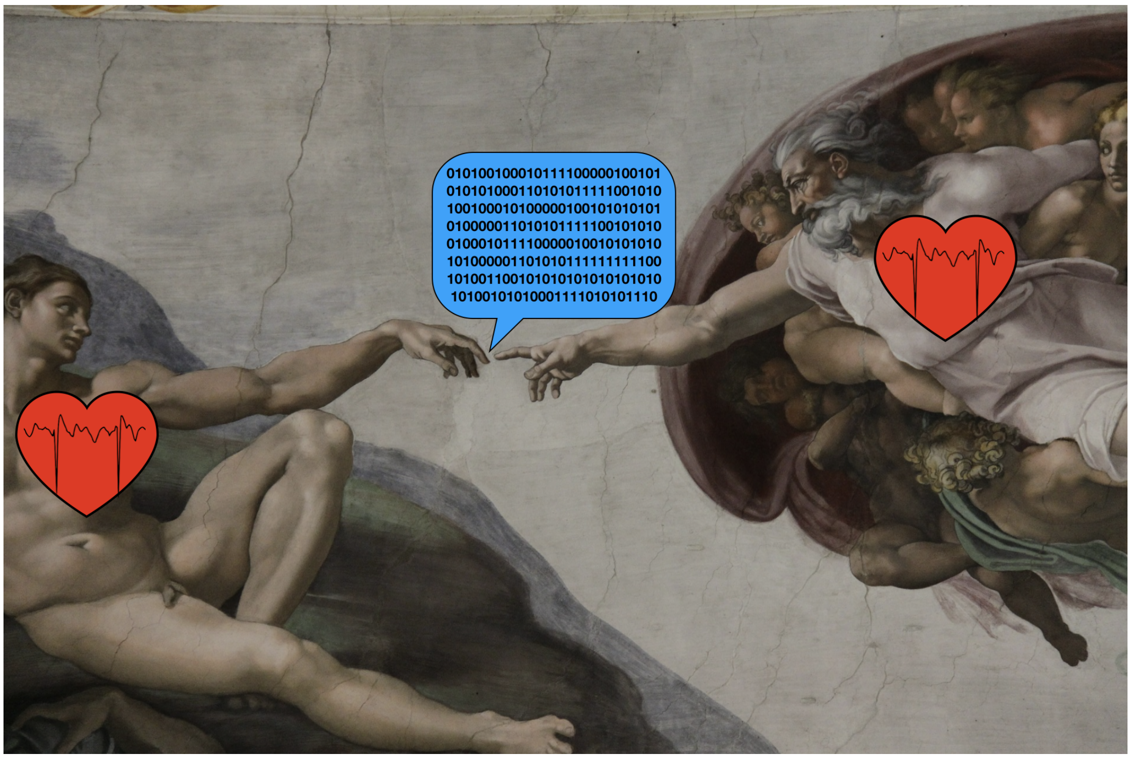
CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG



You are touching me!

A Monitoring Mechanism for Secure Key Generation

Master's thesis in Computer Science - Algorithms, Languages and Logic

JIM BENGTSSON
ALBIN GARPETUN

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

Master's thesis 2018

You are touching me!

A Monitoring Mechanism for Secure Key Generation

JIM BENGTSSON
ALBIN GARPETUN

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2018

You are touching me!
A Monitoring Mechanism for Secure Key Generation
JIM BENGTSSON
ALBIN GARPETUN

Supervisor: Pablo Picazo-Sanchez, Department of Computer Science and Engineering
Supervisor: Gerardo Schneider, Department of Computer Science and Engineering
Examiner: Wolfgang Ahrendt, Department of Computer Science and Engineering

Master's Thesis 2018
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The Creation of Adam, modified to use our proposed monitoring system

Typeset in LaTeX
Gothenburg, Sweden 2017

You are touching me!
A Monitoring Mechanism for Secure Key Generation
JIM BENGTSSON
ALBIN GARPETUN
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

The complexity of the technology we interact with in our day to day life increases, but the effort we want to spend managing these interactions does not. Secure communication is one of these interactions. This thesis proposes a design of a security protocol that generates symmetric cryptographic keys which can be used as a basis for secure connection between two people. The creation of the keys is based upon the two biometrics, ECG and EEG, and that the two people are physically touching. A proof of concept which depends on the ECG of two people has been implemented. Given the correct EEG hardware, we argue that this proof of concept could easily be extendable to a full system.

Through statistical analysis on ECG measurements and experimentation, supporting algorithms have been implemented. The biggest contribution has been the creation of a touch detection algorithm, which can detect the moment when a person is touched, based on their ECG signal.

Keywords: Biometric, Electrocardiogram, Electroencephalography, Inter-Pulse Interval, Cryptography, Symmetric key, Pan-Tompkins Algorithm, Continuous Wavelet Transform, Microcontroller

Acknowledgements

We would like to thank our two supervisors, Pablo Picazo-Sanchez, for the very valuable feedback loop, as well as providing us with crucial pieces of code and Hardware, and Gerardo Schneider for keeping Pablo in check and providing the critical eye.

In addition, we want to thank our examiner, Wolfgang Ahrendt, for his very thorough critique, which helped scoping this project.

Finally, we would like to extend an extra big thanks to our friends and families for the support.

Albin Garpetun and Jim Bengtsson, Gothenburg, 2017

Contents

Acronyms

API Application Programming Interface.

CPU Central Processing Unit.
CT Computed Tomography.
CWT Continious Wavelet Transform.

ECG Electrocardiogram.
EEG Electroencephalography.

GPU Graphics Processing Unit.
GSR Galvanic Skin Response.

IDE Integrated Development Environment.
IMD Implantable Medical Device.
IPI Inter-Pulse Interval.

LA Left Arm.
LL Left Leg.

MRI Magnetic Resonance Imaging.

PPG Photoplethysmogram.

RA Right Arm.

USB Universal Serial Bus.

WBAN Wireless Body Area Network.

1 Introduction

The development of smart devices and the internet of things have experienced tremendous growth in the past years. With this, the needs of people have become more sophisticated. We want to spend more time interacting with technology, but less time with securely managing these interactions.

Authentication is one of these time consuming problems. Authentication is the concept of verifying that an agent is who it claims to be. Another one of these problems is confidentiality, which ensures that information is obfuscated for everyone except the intended recipients. When encrypting a message and sending it, confidentiality is applied, but not necessarily authentication. The information is only readable by the recipient, but there is no guarantee that neither the recipient nor the sender is whom it claims to be [55]. An example of authentication is logging in to a system using a password.

We want to avoid having to manage passwords and carry around keys, but we still want to be able to securely communicate with each other. But in order to encrypt information, so that it can be securely transmitted, encryption keys are needed. Encryption keys are pieces of information used to encode and/or decode information. Symmetric keys are a version of cryptographic keys where the same key that is used for encryption is also used for decryption.

In order to avoid having to remember things, cryptography has started to utilize Biometrics, which is the idea that one can identify a person by their unique physical features [29]. The iris part of the eye is an example of a Biometric being used in cryptography [27], and has started to find commercial use already.

This project aims to design and develop a way to generate a symmetric key utilizing an existing Biometric, Electrocardiograms (ECG). The ECG signal is the electrical activity of the heart over a period of time. The ECG of a person can be read by using electrodes attached to the skin of that person, and is unique for every person in the world. It also acts as a source of entropy (random number generation source), which makes it well suited for cryptography key generation.

The Electroencephalography (EEG) is a monitoring method to record the electrical activity of the brain. If Alice touches Bob, it has been shown in [37] that it is possible to detect the ECG of Alice from the EEG of Bob. In [44] the authors state that the ECG of Alice also can be registered elsewhere on the body of Bob.

1.1 Goals and challenges

The main goal of this project is to create a security protocol that generates cryptographic keys from the heartbeats of two people when they are touching. This would mean that two people are able to create a symmetric key by literally just shaking hands.

We aim at implementing the above in the form of a proof-of-concept tool that handles the sensor data gathering, as well as the key generation. The implementation will be done using wearable devices that read ECG signals from the user. These wearable devices are to be prototyped on an Arduino, which is a microcontroller. This Ardunio will also have accompanying ECG sensors.

In order to create this protocol a number of questions have to be answered.

- Should the key be created from the heartbeat of Alice, Bob or both?
- At which point in time should the creation of this key start?
- What security properties can be enforced?

These questions in turn has a number of sub-problems that needs to be explored. To the best of our knowledge, neither of these questions have been answered.

- Is it possible to detect, from the ECG of Alice, when Bob starts touching Alice?
- Is it possible to detect, from the ECG of Alice, when Bob stops touching Alice?
- Is it possible to detect Bob's ECG from the ECG of Alice?

The expected impact of addressing the above questions and implementing our solution is to advance the understanding of biometric-based solutions for key generation and evaluating its practical feasibility. A lot of research has been done in this area [46, 48, 58], and having a concrete implementation would be extremely valuable.

The biggest challenge in this project stems from the number of unknown variables. All the above questions will determine what the final implementation will look like.

Another challenge is that Biometric authentication methods have a history of being broken [10]. It has also been shown that by analyzing camera footage of a person it is possible to extract the ECG R-peaks [14]. The symmetric key could then be reproduced by a third party which would break the authentication scheme. That is why we have included an evaluation part in this project: to check under which conditions our proposed protocol can enforce some secure properties. This evaluation will also consider known Man-In-The-Middle attacks like replay attacks and active eavesdropping.

1.1.1 Detailed description

After a careful literature review to assess the concrete feasibility of the approach, our aim is to answer the above questions by:

1. Perform ECG readings when two people are touching and analyze the results.
2. Depending on the results from the analysis, develop algorithms for the generation of the symmetric keys based on the ECG.
3. Implement a proof-of-concept of the approach.
4. Evaluate the practical feasibility of our implementation.
    - Is it feasible in terms of computational time?
    - Is the solution portable to small devices?
    - Is it secured against attacks?

We elaborate the above in more detail in what follows.

It is necessary to start with performing ECG tests and analyzing these results to see if it is possible to detect when another person is touching and/or releasing the person with the ECG sensors. From these tests we also need to determine if it is possible to detect the ECG of another person, and if so, how the signal looks like. For example, the resulting signal could be a compound signal consisting of both Alice's and Bob's ECG, or mainly the ECG from Alice with small interference from the ECG of Bob.

The algorithms we plan to develop will differ quite a lot depending on the results from the previous analysis. If it is possible to detect another person's ECG we will develop an algorithm which will generate a cryptographic key from the compound ECG signal when two people touch each other. If it is possible to do this directly from the compound signal this will be done. Otherwise the signal needs to be split up and used to create a cryptographic key. This could prove to be one of the main challenges of this project, due to the difficulty of splitting up a compound signal while preserving the original cryptographic properties of both signals.

If it is not possible to detect another person's ECG, but it is possible to detect when Bob starts and/or stops touching Alice, we will create the symmetric key using the ECGs from both people sent through an already trusted third party.

It will also be necessary to develop an algorithm which detects if Bob touches Alice. The reason for this is because it is possible to detect the ECG of Alice from the EEG of Bob. This makes it at least theoretically possible to recreate the same key for Bob, resulting in breaking the security if the key is used to encrypt data. This algorithm will be used to stop the key generation if a touch is detected.

The developed tool should be usable by anyone, given that they have access to the correct hardware. One could then use this to create any type of higher level protocol based on symmetric keys. Assuming the tool is proven to work for ECG sensors, we claim that it should be easy to port it to use Photoplethysmogram (PPG) sensors, which can be found in everyday smart phones [33]. Having access to a tool like this in your smart phone could be a great security asset. It would also be much more practical for every day users if all the hardware that is required to use this tool is a smart phone.

If it is possible to detect a compound ECG signal and it is necessary to split this signal to be able to create a symmetric key, this will be one of the biggest design challenges. Verifying the correctness of the signal splitting algorithm implies checking for several anomalies. First it is necessary to be able to determine whenever the ECG signal only consists of one person, the person carrying the sensor, or whether it is already a compound signal. This will be used in order to detect when there is a change from a single ECG signal to a compound signal (meaning there are at least two people touching each other). We also need to be able to verify that the compound signal is not the result of more than two people. If this is the case, a cryptographic key should not be generated because this could mean that someone outside of the intended use, an attacker, is trying to get access to the symmetric key. The single signal will also be used in the process of splitting the compounded signal into two signals.

1.2 Limitations

This project does not intend to create a fully-fledged cryptographic protocol. Thus, there will be no attempts to optimize in order to be able to port this into a wearable device such as a smart wrist band. This is merely a proof of concept. As sensor technology gets better, this will naturally improve, but that is outside the scope of this project.

There also exists hardware limitations. We have no access to EEG sensors, so if they are needed, that specific data has to be spoofed.

1.3 Related work

Using ECG in cryptography has been done before. In [46] the authors use it to authenticate programmers and external medical devices to Implantable Medical Devices (IMDs). IMDs are medical devices, such as pacemakers and defibrillators, which are embedded into the human body to support the body in some way. These devices usually have built in radio communication in order to be able to update the IMD and to read its data. Many of these IMDs lack a robust authentication protocol, making them vulnerable to over-the-air attacks, and therefore also the patient to physical harm. The system, named H2H, proposed and implemented in [46] enforces a touch-to-access policy to determine that the programmer, the person or device trying to communicate with the IMD, is actually allowed to do this. Touch-to-access policy means that when a programmer needs to communicate with the IMD it needs to have physical contact with the person equipped with the IMD. The logic is that if someone already has physical access to a patient it means it already has the ability to harm or cure. When the programmer needs to access the IMD it initiates an authentication process. The authentication process works as follows. The IMD takes a reading α of the patient's ECG. At the same time the programmer takes its own reading β of the patient's ECG. If β is "nearly equal" to α, the programmer is granted access to the IMD. Comparing this paper to what we want to explore and achieve in our thesis, one difference is that they used ECG to achieve authentication. We are trying to achieve confidentiality by creating symmetric keys from the ECG which will be used to encrypt and decrypt data. They also did not consider that ECG data could to be transferred through the act of touching.

Another paper which utilizes ECG of people to generate cryptographic keys is [59]. In this paper the authors consider how to preserve the integrity and privacy of a person's medical data over a Wireless Body Area Network (WBAN). A WBAN can easily be deployed on a person's body and can be used to monitor the person's health in real-time. The information can then be distributed to other users at anytime through handheld devices and internet. By using the ECG signal of a person when generating cryptographic keys, the authors propose a new key agreement scheme called ECG-IJS. This scheme can be implemented in such a way that no previous key distribution is needed, making it possible to use in a "plug and play" manner. The authors also show that the proposed scheme is quite energy efficient when compared to similar schemes. The scheme is thus fundamentally different from ours, since it does not create persistent keys.

There exists a multitude of schemes for distribution of symmetric keys. One of the most popular is utilizing the Diffie-Hellman, which is an asymmetric key protocol. It is used for symmetric key distribution by using the private key part of the asymmetric cryptography key pair for encrypting some symmetric key. This encrypted symmetric key can then safely be sent to the other party over a communications protocol of your choice. The receiver will then decrypt it using the public key part of the senders asymmetric cryptograph key pair. The receiver now has an unencrypted version of the same symmetric key as the sender. These schemes for key distributions all rely on some prior communications channel, which of course has to operate through some medium, such as the internet, or bluetooth. This is the main difference to our paper, which only considers the physical touch to be the medium in which information is sent [32].

2

Background

This chapter outlines the concepts and theory that make this thesis possible. It introduces cryptography, why it is needed in the first place and also exemplifies cryptographic key usage and generation. The Biometrics section gives background to what biometrics are and how these are used for information security. Finally, the theory comes together in talking about the signal processing. Here is the theory that lies the closest to the key generation system proposed in this thesis.

2.1 Cryptography

Cryptography is the practice and study about secure communication when there are so called adversaries, or third parties, present. Cryptography is an ancient art that is theorized to have found use even around 4000 years ago by the Egyptians [38]. Today there is more sensitive information distributed across the internet than ever, and thus making cryptography highly relevant. It is essential for things such as encryption, which is the act of processing data in such a way that confidentiality is achieved. Confidentiality is a term used in information security theory, which, if achieved, puts a constraint on the data. This constraint consists of disallowing non authorized consumers to access the data [55].

2.1.1 Keys

In cryptography, there are two main categories of encryption schemes: private key cryptography and public key cryptography. Most modern encryption functions rely on the concept of cryptographic keys. The goal of these schemes is to provide confidentiality, through encryption [11]. Both private key and public key cryptography rely on keys, which is the piece of data used for encryption and/or decryption. In private key cryptography this is often referred to as a symmetric key, and in public key cryptography as an asymmetric key [23].

An asymmetric cryptography scheme utilizes a different key for encryption and decryption. One encrypts with the public key, and decrypts using a private key.

Symmetric cryptography, on the other hand, relies on both keys being the same, or at least derivable from one another [12].

With symmetric cryptography, the problem often lies in distributing the private key. If one were to send the key over the internet it would be trivial to perform a Man-In-The-Middle attack, where an adversary reads the data sent between two parties, thus seeing the symmetric key in its entirety [20]. A popular method of distribution is utilizing the Diffie-Hellman scheme, which is an asymmetric key generation technique utilized to collaboratively generate a shared symmetric key [32]. Another way is to have a way to pre-distribute the keys, or information needed to create the keys, before any potentially insecure communication has happened. The distribution of the information needed to create the keys is what is considered in this report.

2.1.2 Randomness & Entropy

Symmetric keys can achieve unconditional security, meaning the cryptosystem cannot be broken even with infinitely computational resources and time, as in the discussion presented in [13]. Asymmetric keys on the other hand, can not. Asymmetric cryptography will always be secure under the condition that the underlying problem is believed (but it is still an unproven assumption) to be computationally unfeasible to solve. So when designing an asymmetric cryptography scheme, only some specific problems are applicable [13].

When designing symmetric cryptography, on the other hand, a good source of information is needed, one where a potential adversary can not figure out how this source works. One way of achieving this is what you are doing when you think of a password to use for a service. If done poorly you thought of a password that an adversary could guess given parameters known about you, but if done well an adversary would have no information of that key. This process does not work as well for computationally generated keys, however. Here, randomness is commonly used instead. If you base your key generation upon, let's say the system clock, an adversary could figure out the time span in which the key could have been generated, and then simply try all combinations in that time span. If the seed for a key is truly random, you would have to brute force every value, which is good [19].

The problem of creating truly random seeds for our key generation is a well researched problem, with a lot of different solutions. A company called Cloudflare famously utilizes a wall of lava lamps as their source of random bits [26]. They photograph the lava lamps, and the chaotic nature of the movement in the lava lamps, in combination with the noise associated with photography, makes it a perfect candidate for randomness. The software behind this is called Lavarand, and was developed by Silicon graphics, and is used in other fields than cryptography as well [30].

Given that we have a method to generate a random number, how do we know that this method is indeed random? There are a number of test suites, such as ENT [1] and NIST STS [2], to test this. ENT in particular utilizes entropy, optimum compression, chi square, arithmetic mean, Monte Carlo and serial correlation. NIST STS contains as many as 15 different tests, of which the scores are aggregated to a final score [52]. One metric that is common across both of these test suites is *entropy.*

There are many different types of entropy, but they are all a measurement of randomness at its core. In this thesis we will consider Shannon entropy, which is a specific version of Rényi entropy, and is commonly used in cryptography. Shannon entropy is, mathematically speaking, the average of some data produced given a stochastic source [13].

$$H(x) = -\sum_{x \in \mathcal{X}} P_X(x) log(P_X(x)) \tag{2.1}$$

Equation (2.1) defines Shannon entropy H(x). X is the random variable. $\mathcal{X}$ is the family of possible data choices. $P_X(x)$ denotes the probability of the random variable X having the value x.

If maximum entropy is achieved, the entropy would be equal to the number of possible forms the data can take [13]. In the case of binary data, if your source is truly random, the entropy would be equal to one bit per bit. If we plug in $\mathcal{X} = \{0, 1\}$ and $P_X(x) = \frac{1}{2}$ for every $x \in \mathcal{X}$, we see that we get 1. Another way of looking at it is that in a truly random source of bits, we have no prior knowledge to help with guessing the next bit, so $P_X(x) = \frac{1}{2}$ will always hold in this case.

When evaluating the randomness of data, entropy is a good entry point, but not the only one. Since [46] bases most of its data upon this, it is the main measurement of randomness considered in this thesis.

2.2 Biometrics

Biometrics recognition, or just biometrics, is a technical term used to describe body measurements and calculations. It describes metrics related to human characteristics. Biometrics are usually categorized into two different groups: physiological and behavioral [28]. Physiological biometrics are related to the physical aspects of the body. Examples of physiological biometrics are fingerprints, palm veins, face recognition and iris recognition. These are physical characteristics that we have related to our body. These biometrics are something that we do not have control over in the sense that we can not change them without doing some physical changes. For

[1] `http://www.fourmilab.ch/random/`
[2] `https://github.com/arcetri/sts`

example, in order to change a fingerprint it is necessary to do some physical change to the finger. Behavioral biometrics on the other hand are related to the pattern of a behavior of a person. Examples are typing rhythm, gait (movement of the body), voice and ECG. In contrast to the physiological biometrics, we can actually change the behavioral biometrics quite easily. For example typing rhythm is one thing a person can adjust whenever the person wants to.

Biometrics identification refers to the identification of an individual based on their distinctive physiological and/or behavioral biometrics, meaning it can be used as authentication. Biometric authentication has seen a lot of attention lately. One very general example are Wireless Body Area Networks (WBANs), which are lighweight body monitoring devices that communicate wirelessly [31]. Due to the wireless nature of the WBANs they have been the subject of security-related research [35]. Another very common example of a biometric authentication is using a fingerprint [18]. Many new smartphones today have a fingerprint scanner, giving a user the alternative to use the fingerprint as an authentication method when unlocking the phone. An example of perhaps a bit more unusual biometric authentication method is using the heart sound, which is what the authors in [22] did. They investigated the probability of using the heart sound as a reliable biometric for human authentication. They concluded that the most significant feature was that the sound of the heart beat is not as easily simulated or copied, when compared to more traditional biometrics as face, fingerprint or voice.

One of the positive sides when using a Biometric authentication method is that it is built into the body, where most other methods need some external information. Using a password for authentication requires the user to remember a password, which might be troublesome for some people. When opening a physical lock you need to have the key, meaning you need to carry around a physical object, which also could be cumbersome. When using the fingerprint as an authentication method it is only necessary to scan your finger against a sensor. This sensor then verifies that the scanned fingerprint matches the correct fingerprint, called the template [24]. Therefore making it more practical because it is not necessary to remember a password or to carry a physical key. The downside is that it is not replaceable. When using a password it is possible to simply change the password, and if using a lock it is also possible to change the lock. Fingerprints on the other hand is not considered a secure authentication method in the sense that once they have been copied, they can be used indefinitely [7]. It is also not so easy to change the fingerprint if it has been forged [24].

Even though most of the use cases for biometrics within cryptography are in authentication, it has also been used in a way to ensure confidentiality. In [51] the authors are using a process called *Biometric* Encryption, which is a process of secure key management. This means they are not directly using a biometric, a fingerprint in their case, to ensure confidentiality by encrypting and decrypting the data by the use of fingerprints. They are instead using fingerprints as a replacement to typical passcode key-protection protocols, which is the process of using a passcode to retreive a cryptographic key which is then used to encrypt the data. In Biometric enryption a fingerprint is therefore used to retreive a cryptographic key, which ensures confidentiality by encrypting the data.

2.3 Electrocardiography

Electrocardiography is the process of reading the electrical signals from the heart over a period of time. This can be done by placing electrodes on the skin of a person which record the electrical signals from the heart. The metric used to measure these signals is voltage. A graph that shows the voltage versus time when reading these signals is called an electrocardiogram (ECG). Figure 2.1 shows an example of an ECG from a healthy person.

An ECG is usually divided into 5 different intervals which show different waves of the ECG [46]. These are all shown in Figure 2.1. It starts with the P-wave which gives a slight peak in voltage. This is followed by the QRS-complex which consist of two sharp valleys, Q and S, that occur right before and after the R-peak. The R-peak of an ECG is the point in time where the electrical impulse is the strongest, corresponding to the "beat" in a heartbeat. Finally the T-wave finishes one period of an ECG.

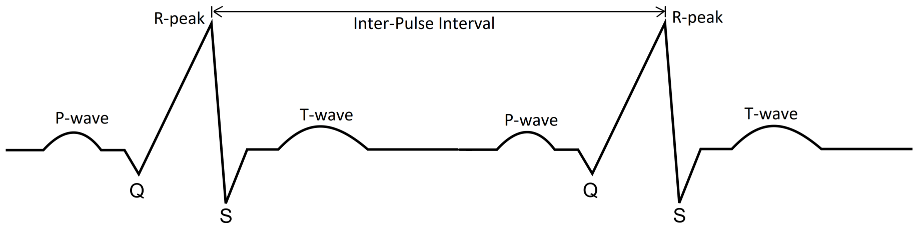


Figure 2.1: Illustration of an ECG. Original image from [17].

2.3.1 Inter-Pulse Interval

The Inter-Pulse Interval (IPI) is the time elapsed between two consecutive R-peaks, e.g the electrical impulses of the heart. In Figure 2.1 two consecutive R-peaks and the IPI between these are shown. The heart rhythm is controlled by the nervous system, where there are many non-linearly interacting processes, which gives the IPI a very chaotic structure [40]. An ECG waveform is affected by both long-term influences such as circadian rhythm and short-term influences such as temperature and respiratory changes. This result in ECG waves have both long-term patterns and short-term chaotic structure [46].

Because of the chaotic nature of IPIs it has been shown that there is a strongly random element between different IPI values [9], suggesting that IPIs can be used as a natural source of randomness [15]. In [46] it is shown that it is possible to extract four high-grade random bits per IPI, meaning bits that have maximum entropy and are fully uncorrelated, thus making it suitable for cryptographic key generation. The authors to this paper use this as a safety mechanism when an external agent has to communicate with an implanted medical device, such as a pacemaker. The person with the implanted pacemaker has to touch an external controller which is connected to the agent. This allows the agent to wirelessly communicate to the pacemaker, using a cryptographic key which is generated from a set of IPIs. Since both the pacemaker and the device has access to the IPI, they can both encrypt and decrypt using this symmetric key. Finally it is claimed that this entropy acts as a basis for key generation and it is not reproducible.

2.3.2 Measurements

To read the electrical signals from the heart it is common to use electrodes attached to the skin of a person, which detect the electrical signals. At least 3 points of contact with the subjects body has been standard, where the different electrodes consist of positive pole, negative pole and physical ground [50]. One of the most common setup is however to use the so called 12-lead ECG. This means that it gathers information about the electrical signals from 12 different perspectives, usually using 10 electrodes where 4 are attached to the arms and 6 on the chest. The data gathered from these electrodes are then used to create one cohesive ECG value. Another common setup is to only use 3 electrodes, which is not as precise compared to using 10 electrodes. When using the ECG in critical medical purposes it might be necessary to have that precision, but in other situations such as personal use, it can be good enough with the 3 electrodes setup.

As technology has become better it has emerged smaller and more elegant ways of measuring ECG. In [50] the authors came up with a solution that only requires two measuring points from the fingers, resulting in a more practical experience for the user. It has also been implemented in a mobile device in such a way that the user only needs to touch 2 electrodes for the device to detect the ECG [6].

Even though the most common way of measuring the heart signals is by using electrodes attached to the subject's skin, there are alternatives. Photoplethysmography (PPG) sensors are using a light-based technology which measures the rate of blood flow, which is controlled by the pumping action of the heart. It is even possible to get a good approximation without needing physical contact to the subject [57, 45]. The techniques used in these papers are roughly based on the amount of light that is reflected from a person's skin when blood is flowing through capillaries near the surface of the skin. The variations of reflected light is not visible to the human eye, but by using video footage of the person in combination with different video processing and computer vision techniques they are greatly increased [57]. In [14] they reached the conclusion that up to 70% of the information that was obtained when using contact-based methods also was obtained when using contactless methods of gathering the heart signals.

2.3.3 ECG as a biometric

ECG signals have been widely used in literature as a biometric authentication method. In [49] the authors showed that by only using a single lead setup on the chest, they were able to acquire 100% identification accuracy when authenticating people. It has also been implemented in a more practical environment compared to having different ECG electrodes attached to the body. The authors to [6] proposed an algorithm which was implemented in a mobile phone. The user then only had to touch two ECG electrodes of the mobile device for a duration of 4 seconds in order to acquire the signal. This method had 81.82% true acceptance rate and only 1.41% false acceptance rate.

2.3.4 Irregularities

Since the ECG signals are closely correlated to the heartbeats of a human, there might be irregularities of the ECG. These might be for different reasons. For example heart diseases, stress and drug use can impact the heartbeats, and therefore also the ECG signals. It has however been shown that an ECG identification system can be used even when the users have common cardiac irregularities, such as premature ventricular contraction [5]. In this paper the authors were able to achieve a recognition rate of 96.2% when the users had the aforementioned cardiac irregularities. To the best of our knowledge it has not been tested how good an ECG identification system would work if the users were under significant stress or using drugs.

2.4 Signal processing

One can not safely generate a cryptographic key directly from a raw ECG signal. In order to be able to utilize the ECG signal to create the key, some processing steps are needed. To create the key, a set of IPIs are needed. This can be extracted from the raw ECG signal.

2.4.1 IPI Extraction

If one can extract the R-peaks with their corresponding timestamp from an ECG signal, calculating the IPIs should be trivial. It is simply a matter of taking the difference between consecutive time intervals.

R-peaks are extracted from the R wave of a QRS complex. Since every ECG will have one QRS complex for each heartbeat, and each QRS complex will have one R-peak, we can extract as many R-peaks as we have recorded heartbeats. To extract the R-peak from a QRS complex you simply look at the time of the peak of the R wave.

In order to isolate the QRS complex from an ECG, there are a number of existing algorithms. Wavelet is an algorithm based on the wavelet transformation of an ECG signal. This method works well in settings with a low amount of noise and when the needed post-processing computation is not too great [36]. Another alternative is WQRS, which utilizes a low-pass filter and then further treated to become a curve length signal by a transformation with a nonlinear scaling factor is applied [60]. Other algorithms include GQRS, to which the algorithm is yet to be released [2], and Pan-Tompkins algorithm [43], which is considered mainly in this report.

This algorithm is popular for many reasons. One reason is that it is old but it still has not been disproved, making it unlikely that it ever will be, and thus has stood the test of time. It is also freely available online, and boasts that it can detect QRS complexes in a noisy signal, in real time. This makes it a good candidate for day-to-day measurement scenarios, where noise is a big factor due to movements of the body. Further noise is expected to come from wearable sensors not being as delicate as medical-grade equipment. Pan-Tompkins algorithm was in fact originally implemented on a micro-processor utilizing only integer arithmetic [43], so even the original implementation had performance in mind.

Pan-Tompkins algorithm works by utilizing a bandpass filter, high pass filter as well as a low pass filter. It also adapts to changes in the collected ECG signal. A graphical example of how the output might look when running Pan-Tompkins algorithm on an ECG signal can be seen in Figure 2.2.

2.4.2 Key Generation

Creation of cryptographic keys are often based on a stream of bits. The number of bits determine how strong the key is. So in order to generate a key from an ECG signal it is needed to convert the extracted IPIs to bits. The authors in [46] proposes a dynamic quantization method for this which will be described shortly in the following section.

The given ECG data is scaled from 0 to 1 by subtracting by the mean IPI value and dividing by the maximum in the scaled data. The data is then multiplied by 256, and then rounded to the nearest integer, except if it is 256, then it gets rounded to 255. This gives us data spanning 0 to 255, ever so slightly tipped in the favor of the number 255. This is then converted to binary code, the gray code encoding scheme is used in order to increase error margins of physiological parameters. Since [46] specifies that only the 4 least significant bits are highly entropic, these are extracted from every resulting byte.

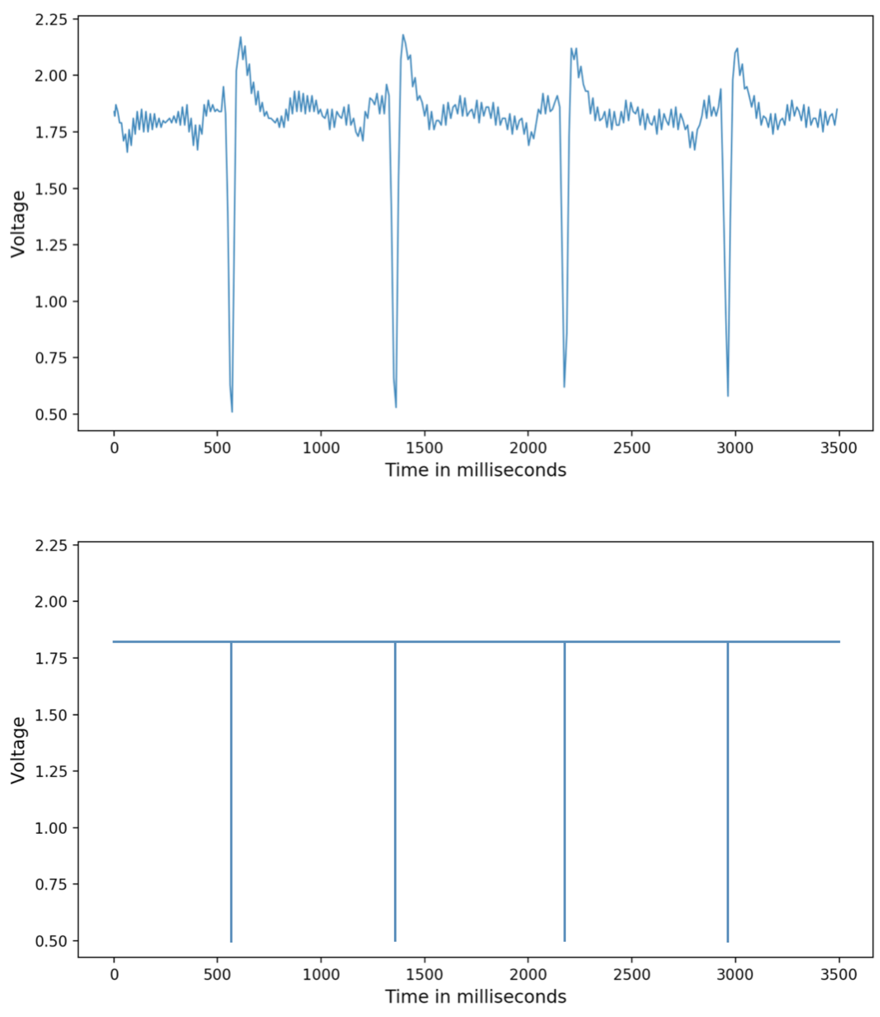


Figure 2.2: Graphical representation of the output given from running the Pan-Tompkins algorithm. The top graph is of a raw ECG signal, whereas the bottom graph is of the corresponding Pan-Tompkins output.

2.4.3 Continuous wavelet transform

A Continuous Wavelet Transform (CWT) is in mathematics used to divide a continuous time signal into so called wavelets. These wavelets are wave-like oscillations with an amplitude that begins at zero. The CWT compares the original signal to shifted and compressed or stretched versions of a wavelet. By doing this it obtains a function of two variables, the so called CWT coefficients. The CWT coefficients can be interpreted in such a way that it becomes much easier to find anomalies, or singularities, in the original signal. A signal with an abrupt transition results in CWT coefficients with large absolute value. CWT is used, among other things, to both analyze ECG signals and so called EEG signals.

2.5 Electroencephalography

The Electroencephalography (EEG) is a monitoring method to record the electrical activity of the brain. EEG measures fluctuations of the voltage which are caused by the neurons in the brain [41].

2.5.1 Measurements

EEG is usually measured in a similar way to ECG, by placing electrodes on the user's skin. In contrast to ECG however, the electrodes are usually placed on the user's scalp. There is a standard placement of all the electrodes when performing an EEG exam, called the International 10-20 system [54]. This has been formalized so that different laboratories and different exams can be compared to each other in a scientific way. The system is based on the location of an electrode and the underlying area of the brain, which can be seen in Figure 2.3. The actual placement of the electrodes are placed in such a way that the actual distances between adjacent electrodes are either 10% or 20%, hence the "10" and "20" in the name, of the total right-left or front-back distance of the skull. A minimum of 21 electrodes are used in this system.

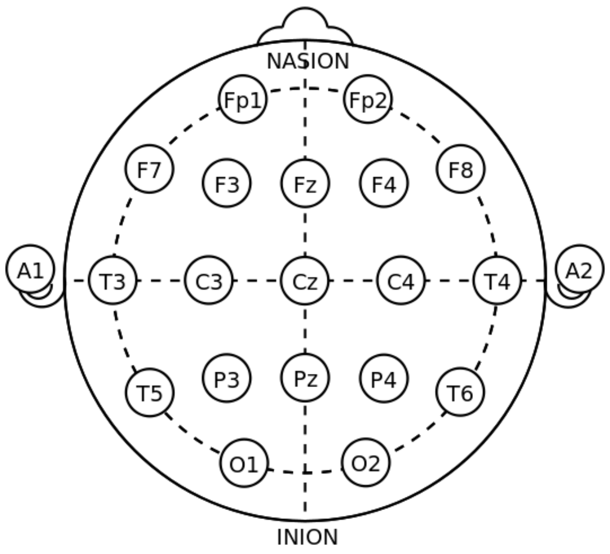


Figure 2.3: An example placement of electrodes according to the International 10-20 system. Image from [16].

The International 10-20 system is the most common way of measuring EEG, but there are other ones. Current EEG systems can have as few as 4 electrodes or up to 256 electrodes. It has however been shown that there is a distinct relationship between the number of electrodes used during an EEG exam and the quality of the signals that are detected [34].

2.5.2 Applications of EEG

EEG is most commonly used to diagnose epilepsy, which causes the EEG readings to have abnormalities [53]. Other diagnoses it is used to determine are coma, sleep disorders and brain death. It also used to be a common tool when diagnosing stroke, tumors and focal brain disorders, but recent advances in techniques such as Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) has become superior.

When being used in clinical contexts, EEG readings are usually performed in such a way that it reads the brain's spontaneous electrical activity over a period of time. Using EEG in diagnostic applications usually perform the readings either on the spectral content or on event-related potentials content of EEG. Spectral content is being used to analyze neural oscillations, also known as "brain waves", which can be observed in the frequency domain. Event-related potentials look at eventual fluctuations time that are locked to an event, for example the push of a button.

2.5.3 Detecting ECG from EEG

If person A touches person B, it has been shown that it is possible to detect the ECG of person A from the EEG of person B [44, 37]. The execution of the experiments in both of these papers are quite similar. The participants in the studies were divided into sets of two people in each set. Both EEG and ECG electrodes were attached to each participant, where [37] states they were using the International 10-20 system to measure the EEG. The two subjects in each set were first monitored when not touching, to use as a baseline, followed by a 5-minute hand holding period. When comparing the results from person A's ECG with person B's EEG, after using signal averaging to reduce the noise, it is possible to see that the EEG signal from person B is very similar to the ECG signal from person A. Thus showing it is possible to detect the ECG from another person's EEG if they are touching.

3

Design

In this chapter the design of the project is described. The work process is described in conjunction with higher level implementation design possibilities. Lastly it presents a thorough explanation of how the presented data was gathered.

3.1 Work Process

On a high level, this project will spend most of its time in the "Tests and new observations" part of the scientific method [21]. The project will consist of three phases: literature study, implementation and evaluation.

The goal of the literature study is to gather information about similar projects. In particular, how to be able to generate cryptographic keys from ECGs and, depending on the analysis from the ECG tests, the signal treatment needed to split compound signals.

The literature study will run in parallel with setting up the lab environment. The reasoning behind this being that hardware has to be delivered, and different parts may arrive at different times. Setting up, and getting more familiar with the lab environment means there will be more knowledge in how the final system can work. This knowledge will be used to create a model of the software architecture of the final system. This model should include the required languages as well as the software that is to be installed on the Arduino. If any separate processes need to communicate, this needs to be made explicit. Potential unknowns, if any, should be noted.

The second phase will be an implementation phase. The Arduino will be programmed to gather sensor data from the ECG signals. This part relies on two different hardware components. The sensors for the ECG signals, as well as the Arduino to run on. At this stage it will be necessary to perform some initial ECG readings and analyze these results to see if it is possible to extract another person's ECG from the first person's ECG when these two people are touching. Depending on the results from the analysis, different algorithms will be developed. If the analysis shows it is possible to detect another person's ECG, an algorithm that can generate a set of IPIs from a compound signal will be needed. If this is not possible the algorithm will assume ECG data extracted from the EEG instead. In both cases, however, a touch detection algorithm will be needed. This algorithm needs to be able to detect the moment of time when a person is touched by another person. A secure symmetric key generation method will also be implemented and demonstrated. The verification should be done offline, and the protocol should reject invalid signal data before trying to construct a key. The software part of the project will be split up into four modules:

- ECG monitor, for reading ECG values from the subject
- Touch detector, for verifying the ECG data and detecting when a subject gets touched
- IPI extractor, for extracting the IPIs from an ECG
- Key generator, for generating keys from the IPIs

During the key generation software development, formatted placeholder data will be utilized. This is to ensure that the development of the key creation software can proceed while the ECG reading software is under development.

The third phase will be an evaluation phase. The goal is to test the robustness of the authentication method, as well as the performance. Robustness testing includes the attacker model definition and proving how our proposed schema deals with this model. Performance testing includes a statistical analysis of the time it takes to generate a symmetric key pair on both devices, from when the sensor measurements start. This measurement could be split up into sub-measurements to see what part of the protocol is the bottle-neck. Another measurement could be the fail-rate, how many tries of the protocol it takes to successfully generate the same keys on both clients.

This project will also include the enforcement of some secure properties. An example of this would be to detect when more than two people are involved in this protocol and react against this problem in real time.

3.2 ECG readings

The ECG data used in this report comes from two different sources. The first source being a big centralized repository of existing ECG readings called PhysioBank and the second source being self-recorded ECG readings.

Physiobank is an online bank of databases containing a lot of readings of physiologic signals [4]. These databases include ECG signals, which is what is concerned in this report. More specifically, the datasets used are from the MGH/MF Waveform

Database (mghdb). This database contains a variety of readings performed at the Massachusetts General Hospital [56, 25]. Healthy-looking datasets that was recorded using 3-lead equipment are mainly concerned in this report, since the equipment utilized for the self-recorded ECG readings utilizes 3-lead equipment.

These PhysioBank ECG readings were used to create a baseline for key generation, i.e. to make sure that the self-recorded readings performed similarly to the PhysioBank ones.

The self-recorded readings were gathered using an Arduino connected to the measurement hardware MySignals, which is created by a company called Libelium. The hardware includes an Arduino shield and a 3-lead ECG sensor system [1]. This information was transferred live to Matlab software developed in this project, where the data was plotted and analysed.

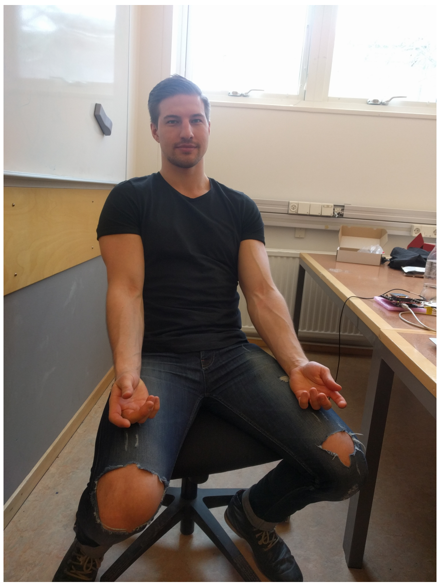
When gathering the data, many parameters had to be considered, such as surrounding environmental impact as well as the target subject's current state. In [8] a couple of actions that the subject should perform in order to mimic every day living activities are proposed. These include sitting, standing, lying and walking, which are the actions concerned in this report. The reasoning behind choosing these is that remaining activities (jogging, running, and stair climbing) proved to be hard to perform consistently with the hardware at hand. The sitting was in an office chair, the standing was done on flat ground and the lying was on top of a hardwood table. When performing the walking, we were forced to walk in the same place, so actually not moving around. This was also due to being attached to hardware.

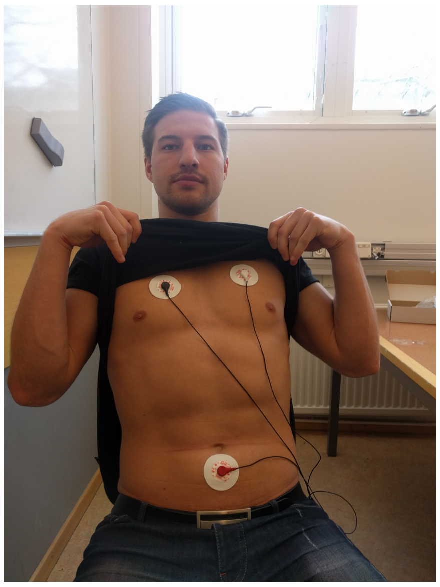
In Figure 3.1 we can see how the subject performed certain tasks. (a) shows how the subject's arms were in relation to his body during all measurements. (b) shows normal 3-lead ECG placement. The final lead placement moved the leads from the pectoral muscles to the underside of the arms. (c) shows how the subject was touched during the measurements. (d) shows how the cord was suspended during experiments to avoid noise caused by unnecessary movement of the cord. Effective in most test cases, but some movement-induced noise was not possible to avoid in the walking case.

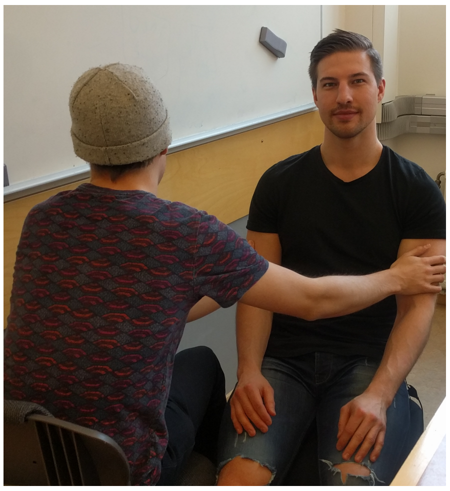
The measurements were mainly 30 seconds long, where the first 10 seconds were the subject performing the task with no interference. During the middle 10 seconds the subject was touched with two hands for the entirety of that time span, while still performing the task. The left hand was placed on the subjects corresponding lateral head (tricep), and the right hand was placed in the same manner on the right side. After 10 seconds the touching was interrupted and the subject just kept doing the task without interference, as in the first 10 seconds.

In order to keep the testing environment as consistent as possible the same room was always used. Electronics, with the exception of the hardware needed to perform the measurements, were kept at a distance about 3 meters from the measurement

[1]`http://www.libelium.com/downloads/documentation/mysignals_technical_guide.pdf`


(a) Arm pose


(b) Lead positioning


(c) Touch orientation

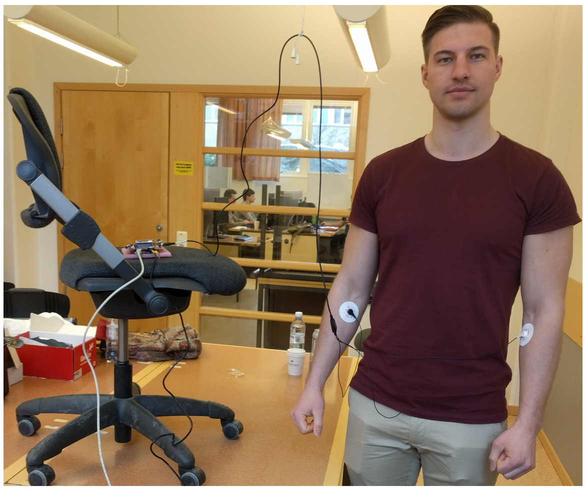

(d) Cord suspension

Figure 3.1: Measurement specifications

spot. The room also never contained any other people than the subject and the person to touch the subject. Two minute breaks were issued between each 30 second measurement. The measurement hardware was positioned so that it was made sure it was stationary.

The positioning of the 3-lead ECG sensor system was not the usual one. After experimenting with positioning we found that the best results were found when the Right Arm (RA) and Left Arm (LA) leads were positioned differently than normal. Normal placement would be to have the RA lead on the upper side of the right pectoral muscle, the LA lead on the upper side of the left pectoral muscle, and the LL lead slightly below the navel [1]. In this project, we instead positioned the RA and LA leads on the corresponding arm, on the inside of the arm, slightly below the bend of the arm. The RR lead was still below the navel. The cord used in the 3-lead setup was mostly stationary, but in the example of walking it proved hard to do.

In order to easier analyze the ECG data it was fitted around the zero-index of the y-axis (voltage) and then transformed into a continuous wavelet. Both the original ECG data and these the continious wavelet versions were then plotted. The latter being the main focus in the analysis. A continuous wavelet analysis helps to reveal patterns in a noisy image by highlighting how the frequency content varies over time.

3.3 System design

This Section concerns the design of the implementation of the monitoring mechanism for secure key generation. Here we present two possible designs with some variations. What is theoretically possible, and what was possible to us due to budget restrictions. Both of these variations assumes that a compound ECG signal from which you could extract the ECG from both involved parties from, does not exist. In figure 3.2 a high level view of the system is presented. This sketch shows one subjects side of the system.

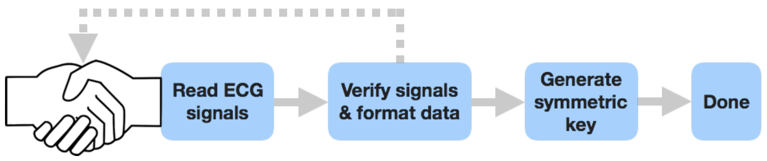


Figure 3.2: A high level view of the software communication of the proposed system. The system is mirrored on the other side of the handshake as well. The dotted line represents a possible extension in the form of retrying when the read sensor data is deemed invalid.

Both designs needs a touch detection component. This is due to the fact that while information about ECG signals is effectively distributed, which parts of that ECG signal to use is not. A touch detection component solves this issue. If touch is detected, we start generating a key. This touch detection also comes in handy for threat detection. If the key generation is already initiated and a touch is registered, it could be an adversary, to which we will respond by terminating the key generation.

Active eavesdropping is the most realistic attack vector for both concepts. There are two categories of eavesdropping that can be done, either having an adversary being part of the system (touching the two communicating parties), or remotely reading the ECG data.

3.3.1 Theoretical concept

We introduce the main design of this project as a theoretical concept rather than a concept we can implement, due to hardware limitations. We simply did not have access to EEG sensors. The implemented design on it self does not fulfill any plausible use cases, but it rather acts as a proof of concept for this more theoretical design.

The main idea behind this design is the fact that you can distribute information, namely the ECG, from a person to another without using any communication protocols or data transfer. It builds upon the fact that if Alice touches Bob, Bob can extract Alice's ECG from his own EEG, as mentioned in Section 2.5.3.

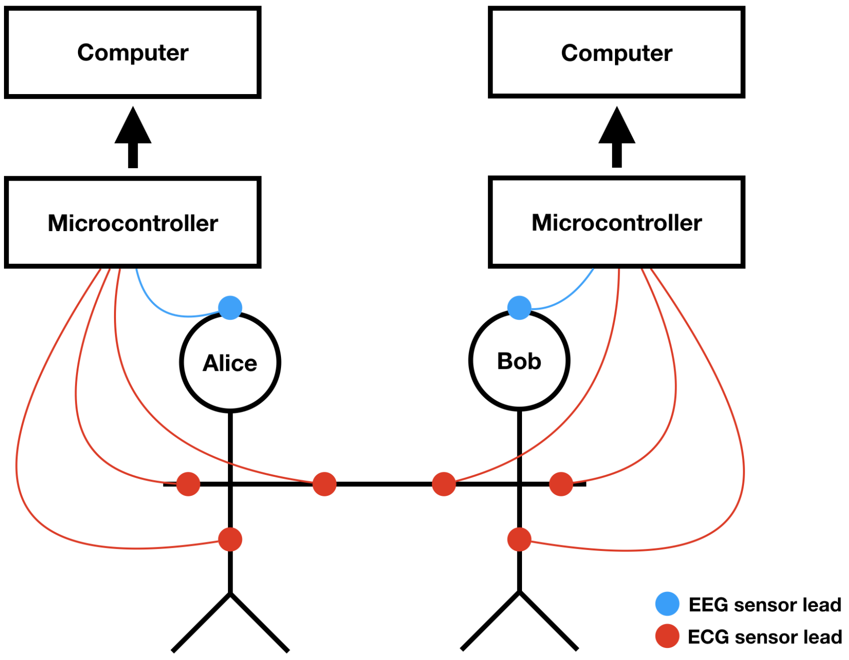


Figure 3.3: Overview of the theoretically possible system.

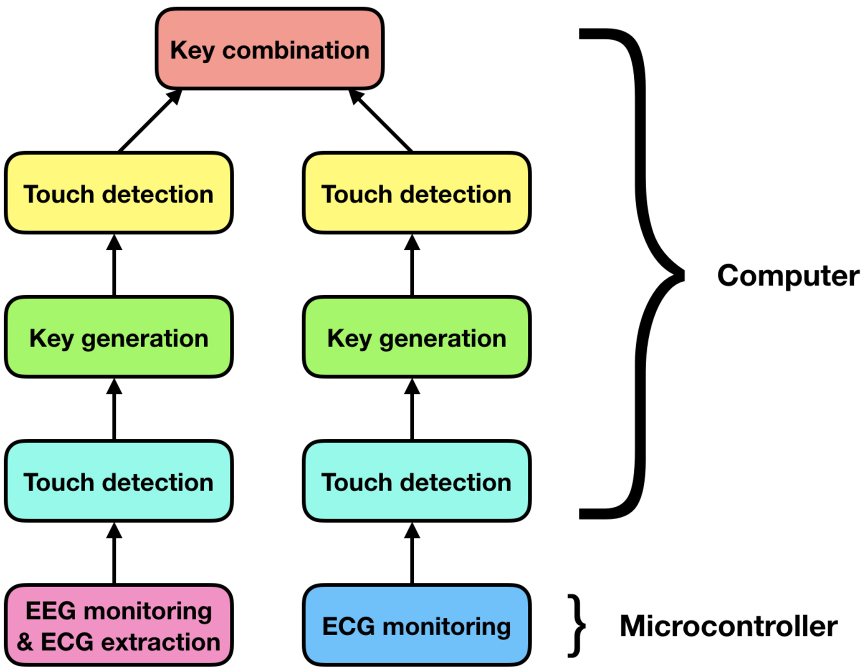


Figure 3.4: Overview of the procedures in the microcontroller and the computer of the theoretical system.

An overview of the theoretical system design can be seen in Figure 3.3, and a more detailed image of the different procedures in the microcontroller and the computer can be seen in Figure 3.4. The system is symmetrical for Alice and Bob and the system will be described from Alice's side. Alice is wearing both EEG and ECG sensors, which are connected to a microcontroller. This microcontroller will continuously read the EEG and ECG values and send them to a connected computer. The computer will receive the two signals and extract the ECG signal of Bob, once they start touching, from the EEG signal of Alice. This extracted ECG signal will be denoted ECG_b and the original ECG signal from Alice will be denoted ECG_a. When Alice and Bob start touching, the touch detector will detect this and the procedure of extracting IPIs from ECG_a and ECG_b will begin.

This will continue until the total number of extracted IPIs are high enough to create a key of desired length. This number of extracted IPIs will be the sum of IPIs from both ECG_a and ECG_b. A key is now created, and the final step is to analyze the ECG_a and ECG_b one last time by running it through the touch detection algorithm. If another touch was detected in ECG_a or ECG_b, it could mean a third person was touching Alice or Bob in order to access information about Alice's or Bob's ECG, during the gathering of IPIs. This would severely reduce the security of the generated key, therefore the entire process is aborted. If no touch was detected in ECG_a nor ECG_b, the IPIs gathered from these are deemed safe, and the key is kept. The process can now successfully finish.

Let's introduce a possible attacker to this system, Eve. We consider the system broken when Eve has gathered enough information to create or steal the same symmetric key generated by Alice and Bob. Of course, there could be different degrees of broken in this model. Eve could gain the knowledge of x number of the bits which the key is based upon. In this case, if the total number of bits used to create the symmetric key is y, Eve would have effectively reduced the security of the symmetric key to one of $y - x$ bits instead of the original x. In the case of the symmetric key being based on 256 bits, and Eve gaining knowledge about roughly half, perhaps through eavesdropping information about Alice's, but not Bob's ECG, we are left with a 128 bit key's worth of security, even if we are using a 256 bit key.

In order to gain information about these bits a couple of different attack vectors are possible. These are all presuming that Eve has total information about how this monitoring system and the key generation works. This assumption is made since security through obscurity is generally not a good idea to depend on.

If Eve gains information of the raw ECG of one or two of the people, she would have the information needed to generate the IPIs, and in turn the bits themselves. If Eve has access to both Alice's and Bob's ECG, she would have enough information to generate the same key. If she would only have one of the ECGs, she would have enough information to generate roughly half of the bits, given that both Alice and Bob has similar pulse. The person with the higher pulse will naturally contribute with more bits to the final key, due to the fact that 4 bits are generated from every IPI, which occurs once for every heart beat. If Eve gains direct knowledge about the IPIs, she could of course just use this directly and ignore the ECGs they are based upon.

We will not consider attacks on the microcontroller or computer, since infecting these is a more general security issue, which really has nothing to do with the security of this specific protocol, but rather the hardware of the computer and microcontroller used, as well as the software they are running. This means that the main attacks we are going to consider are ways that Eve can extract information about Alice's or Bob's ECG.

So how can Eve gain information about the ECGs? If Eve is wearing an EEG sensor system and is also physically touching either Alice or Bob, that person's ECG should appear in Eve's EEG, allowing her to extract the ECG of the corresponding person. She could also use different sensor technologies to gain similar information to the one given from ECGs. PPG sensors, which can be found in everyday smart phones [33], are computer vision based systems that can be used on a distance. The authors in [14] claims that 70% of the ECG data collected through touch-based systems can also be collected through contactless systems such as PPG.

There are variations of this theoretical concept of a design that can be made with less sensors. A bare bones version would be to fit Alice with ECG sensors and Bob with EEG sensors. This would change the system to only use a single ECG, Alice's. This design would be weaker to the aforementioned attacks. Since the key will only be generated based on one ECG, Eve would only need to gain access to half the amount of information to crack the key. You could vary this even further by adding an ECG sensor to Bob as well. Now Bob will have access to both ECGs, but Alice will not. In this case Alice and Bob would first establish a single-ECG key, and then use that to exchange the double-ECG key, leaving Alice and Bob with a key that is harder to crack. This would take more time, however, since two keys has to be generated, and would still be prone to an attack on the single-ECG key. The absolute best version would be to fit both Alice and Bob with both ECG and EEG sensors, which is what is mainly considered earlier in this Section.

3.3.2 Practical proof of concept

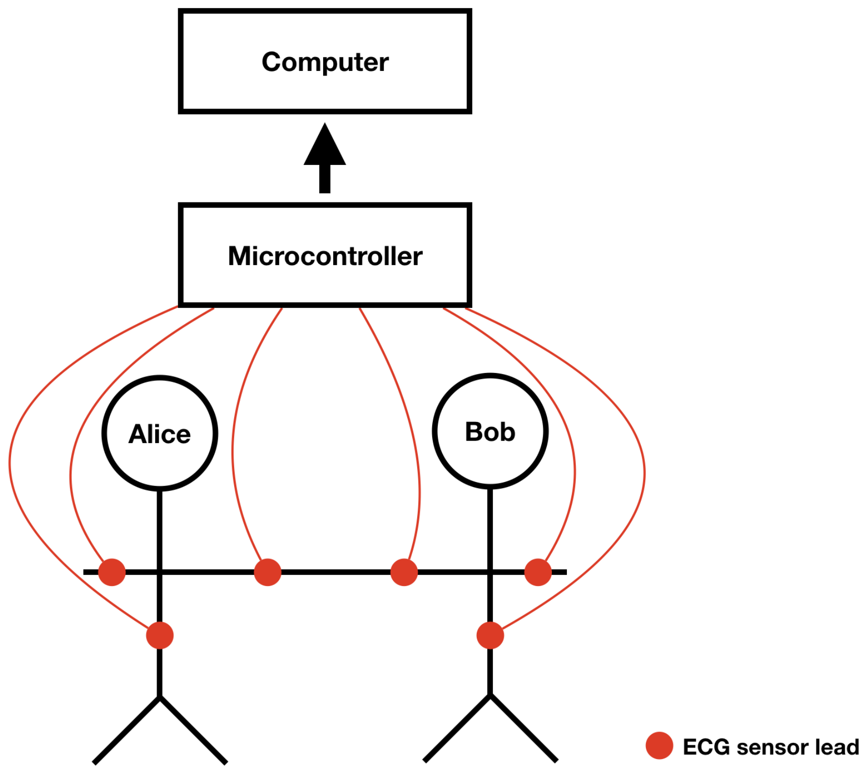


Figure 3.5: Overview of our system.

An overview of our system design can be seen in Figure 3.5, and a more detailed image of the different procedures in the microcontroller and the computer can be seen in Figure 3.6. Alice and Bob both wear ECG sensors which are connected to the same trusted third party microcontroller. One could of course fit both Alice and Bob with individual microcontrollers, which then securely communicates to a computer, which then becomes the trusted third party.

The microcontroller will continuously read the ECG values from both Alice's and Bob's ECG sensors and send these values to a connected computer. When our touch detector detects an initial touch between Alice and Bob it will start extracting IPIs from the ECG signals of both Alice and Bob. This extraction will continue until

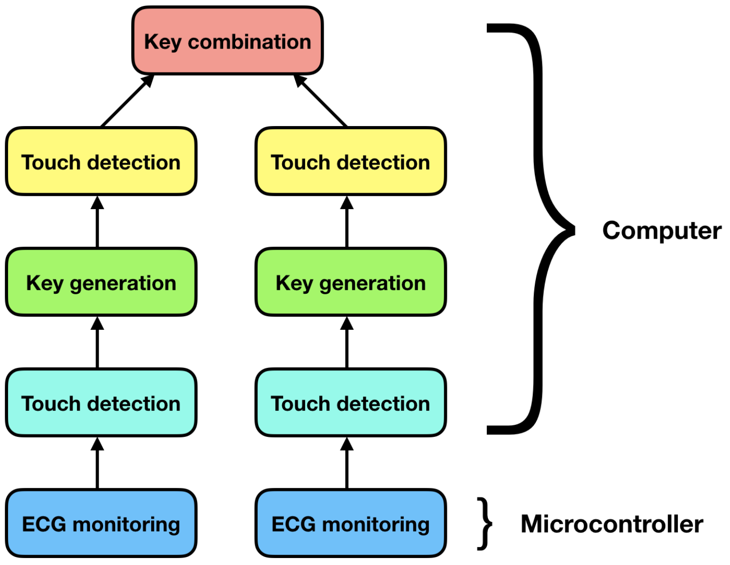


Figure 3.6: Overview of the procedures in the microcontroller and the computer of the practical proof of concept.

there are enough extracted IPIs in total to create a symmetric key, similar to what was done in the theoretical concept.

The system also checks the underlying ECG signals from which the IPIs were extracted if a touch was detected. This is due to the fact that the detected touch could be from another person who is trying to gather information about their ECG in order to break the key. When the number of total IPIs, i.e. the sum of IPIs from both Alice and Bob, is high enough, a symmetric key is created. If a touch is found in the ECG data from which these IPIs are extracted, the key is thrown away, since an adversary is suspected to have entered the system. If no touch was detected, the generated key is kept. At this stage the process is successfully finished.

When designing this system we tried to simulate the theoretical concept as much as possible with the available hardware. Comparing this design to the design presented in the previous Section, there are two main differences:

- Our system does not have any EEG sensors.
- Both Alice's and Bob's ECG signals are read from the same hardware device.

Therefore our system simulates one of the two people of the theoretical system. Assume this system simulates Alice's part. Instead of extracting Bob's ECG from the EEG of Alice when they are touching, she gets it directly from Bob's ECG sensors. In this case there is of course no reason for an ECG extractor from the EEG, which is not included in our system. Except for the extraction of ECG from the EEG, this system works as the theoretical system would on one person.

Consider that an adversary, Eve, would want to break this system in a similar fashion to what is discussed in the previous Section. Here, the microcontroller acts as a trusted third to which Alice and Bob already has secure connections. This makes the microcontroller the single point of failure, and if that was to be compromised, the system would too. Other than that Eve would have to have gain the same amount of information as in the design presented in the previous Section.

4

Implementation

This chapter concerns the practical proof of concept implemented, and the inner workings of it. It also includes systems developed solely for initial data gathering and analysis.

4.1 Initial testing

The initial testings were done to see if it was possible to detect a compound ECG signal when two people were touching. In addition we wanted to see if it was possible to detect when the touch starts and when it stops from the ECG signal, and if so with how good accuracy.

4.1.1 MySignals Arduino

For initial testing and analysis a bare-bones ECG monitoring system was created with the hardware at hand at that moment in time. For this system, an Arduino Uno Rev3 microcontroller was used. This was in turn connected to a computer via a USB connection. Attached to this microcontroller was a MySignals eHealth Medical Development shield for Arduino. This shield has 17 available sensors which can be used to monitor more than 20 biometric parameters. For our project we only had the ECG Electrocardiogram Sensor PRO, which is used to monitor the ECG of a person. The ECG sensor is connected to the shield via a 3.5 mm cord and has three electrodes, called leads. Each lead is attached to a soft patch that has pre-attached gel to it and sticks to the skin of the user.

The software that we developed to read the ECG of a user is running on the Arduino. Arduino is programmed using C++ and we used the Arduino IDE. The program that is running on the Arduino includes three different files. Wire.h and SPI.h that are used to communicate with different devices, and MySignals.h. The latter is MySignals' software library that we included so we were able to use their API in order to actually read the ECG values from the sensors. This is done by calling a single function called getECG(VOLTAGE) from the MySignals library, where VOLTAGE is the specific voltage used. The program that is running on the Arduino is two-parted. It contains a setup function that initializes the serial port and then starts the other part, an infinite loop. In this loop we continuously read the values from the ECG sensors and print this value, together with the number of milliseconds from the start, on the serial port.

4.1.2 Matlab analysis

By using Matlab, the computer which is connected to the end of the serial port is then able to read each raw ECG value together with the correct time stamp. To help determine if we were able to detect a compound ECG signal these values were plotted in Matlab, with time as the x-axis and voltage as the y-axis. Since we knew at what time we were touching, we knew between which values there should be a difference in voltage in the signal, if we detected a compound signal.

In order to determine if it is possible to detect a touch and release from an ECG signal, the ECG values were used as input to the CWT function. When applying a CWT function to an ECG signal in Matlab the CWT coefficients are returned, which is a matrix of double values. The number of columns in this matrix corresponds to the number of values in the input signal, whereas the column in the output matrix corresponds to the same value in the input signal. The bigger the values are in a column in the returned matrix, the bigger the anomaly at the corresponding value in the original signal. As seen in Figure 2.1, the typical pattern of an ECG-signal is mostly a recurring pattern, even though there usually is some deviation. This means that when performing a CWT function on an ECG-signal where no touch occurs, all the values in the resulting matrix have relative low absolute values. On the other hand if there is an anomaly, for example if there is a spike in an ECG-signal from a touch, the values in these columns where the anomaly occurred will be bigger. When plotting this matrix in Matlab, it shows a 2-D image with the time on the x-axis and the CWT values on the y-axis.

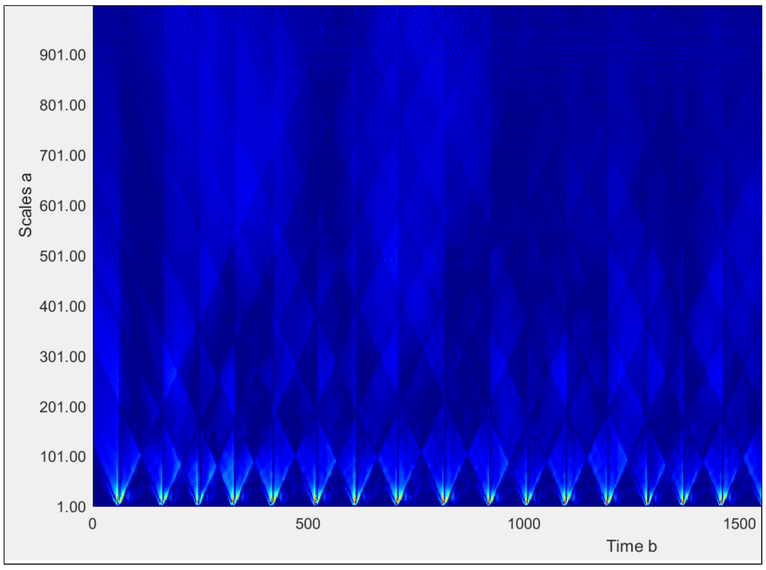


Figure 4.1: Image of CWT when anomaly is not detected.

The image is color coded, where deviations from the base color show anomalies from the original signal. If it did not detect any anomalies the entire image is pretty even with the same colors, which can be seen in Figure 4.1. If it does detect an anomaly, this will be shown by a change in color around the column where this anomaly happened, see Figure 4.2. With an increased severity of the anomaly, the colors will contain even more variation. In other words, the bigger the anomaly is, the bigger the change in color will be.

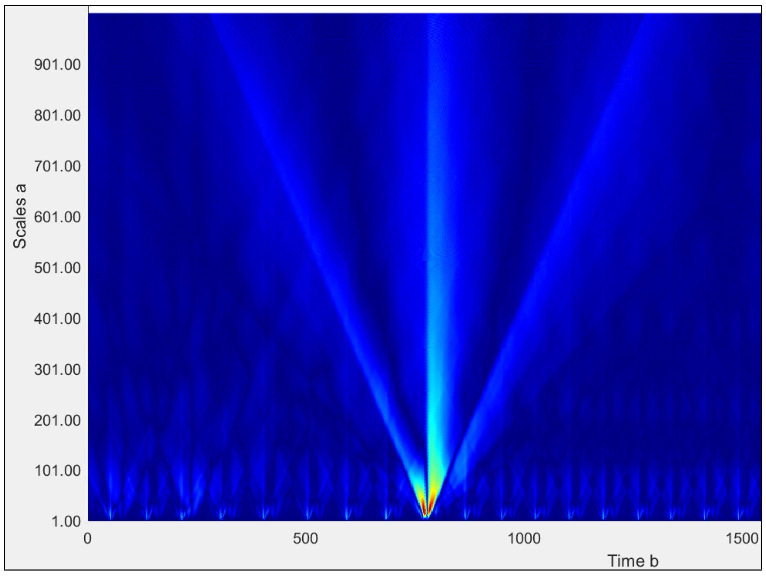


Figure 4.2: Image of CWT when an anomaly is detected.

We then looked at all of these CWT images and decided if we were able to see an anomaly at the point in time which correspond to both the start of the touch and the end of the touch. We categorized the answers to *yes*, *maybe* and *no*, and documented these answers in a table.

4.2 Practical proof of concept

This Section contains a more in depth description of the monitoring system implemented. More specifically, the hardware used and the implemented software's architecture as well as the utilized algorithms will be covered.

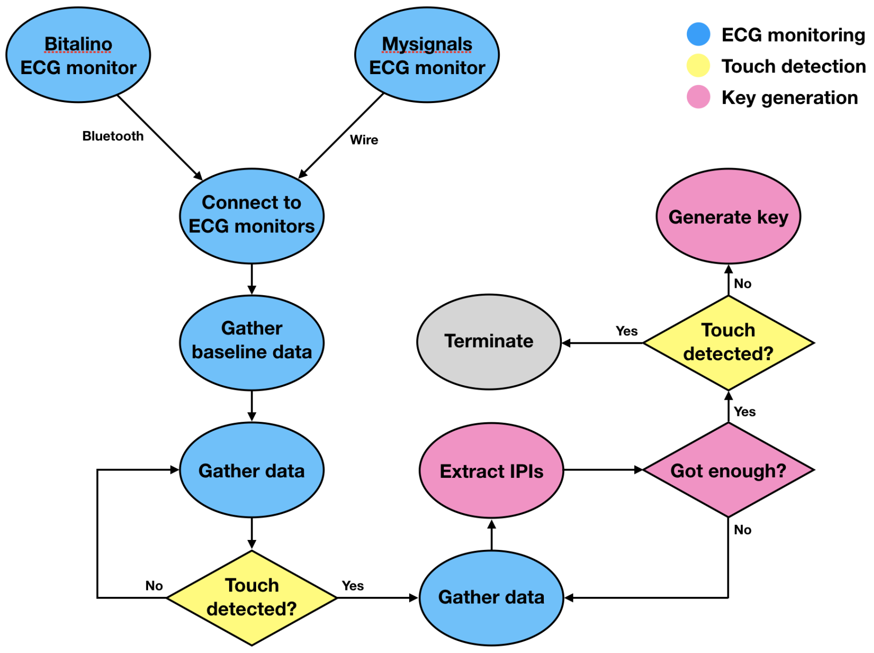


Figure 4.3: Diagram describing the information flow in the implemented software. The legends correspond to Subsections in this Section

In Figure 4.3 you can see the entire information flow of the system. The general idea is to gather ECG data until touch is detected. When touch is detected it is assumed that the two involved subjects are indeed touching in real life. Then we can start gathering data for key generation. When we have enough data for a key, which depends on the level of security needed, we check the ECG data that is the basis for this key, and make sure no additional touch has occurred during this time. Lastly, we generate the key.

The two monitoring parts at the top are essential because this is the source of the ECG data needed to generate the keys in the end. The initial gathering of baseline data exist because we need some data as input to the touch detection algorithm. The IPIs are the piece of information gathered from the ECG that the key is generated from, so we loop until we have enough IPIs to generate a key. Finally the last touch detection check is to make sure no possible adversary was introduced to the system during the time this protocol ran.

4.2.1 ECG monitoring

The hardware that was used to gather the ECG signals were a Bitalino (r)evolution plugged microcontroller and an Arduino Uno Rev3 microcontroller. The Arduino part was implemented mostly in the same way as for the initial testing, see Section 4.1.1. The only difference was in the code running on the Arduino, from which we removed the timestamps, and added a small delay of 20 ms at the end of the main loop. This was done because we noticed, from the initial testings, that we got a few extra ECG values during the first 100 ms compared to the rest of the reading. By setting a small delay we forced it to have a fixed sampling rate. By doing this we also no longer needed the timestamps, because the Arduino was now reading the ECG values at a constant sampling rate.

For the other ECG readings we were using a Bitalino (r)evolution plugged micro-controller, with attached Bitalino ECG sensors. The sensors were similar to the ones used for the Arduino. The connection between the Bitalino and the computer was wireless, more specifically Bluetooth Low Energy (BLE) was used. In contrast to the Arduino where we uploaded our C++ code to the Arduino itself, this was not the case with the Bitalino. In Bitalino the entire code is executed from Matlab. In Matlab we specified the Bitalino's MAC-address and which frequency we wanted the Bitalino to use when reading the ECG data. Finally we executed a command from Matlab when we wanted the ECG readings to begin on the Bitalino. This was all done by using a Matlab API for the Bitalino.

The first part when gathering ECG data is that we gathered what we call baseline data. This means that we gathered ECG data for a fixed amount of time (we used 10 seconds) where the user is not touching another person. Since both the Arduino and the Bitalino are reading ECG values at a constant frequency, we can use this to determine how many values we need to read for it to correspond to our 10 seconds. Reading the values is then simply done by reading the necessary number of values, from the serial port in the case of the Arduino, and via Bluetooth in the case of the Bitalino. This gathered baseline data will then be used in the touch detection algorithm, both to figure out when to start extracting IPIs from the gathered ECG values, as well as to know if there has occurred another touch during key generation, which could indicate that someone else is trying to break our key.

The second part of gathering ECG data is to detect when a touch occurs. From the point in time when the touch occurs, the program should start extracting IPIs from the ECG data. This is implemented in such a way that it reads the ECG data from the serial port and the Bluetooth connection with a certain polling rate (5 seconds in our case). When it has read a chunk of data corresponding to the polling rate, it checks if there has occurred a touch. If there has, we move on to gather the ECG data that will actually be used to extract IPIs from. If not, it goes back to gather another chunk of ECG data and then concatenates this to the already gathered ECG data after the baseline data. The concatenation part is done to improve the touch detection algorithm, see Section 4.2.3. This concatenated ECG data is then checked if a touch has occurred, and this loop goes on until it detects a touch.

The third and final part of gathering ECG data is done to gather the ECG data that is actually used to extract IPIs, which in turn is used to create a key. This is done in a similar way as the second part of gathering ECG data. It reads the ECG data with the same polling rate as above, and then checks if the gathered ECG data is enough to create a key. If it is not it reads a new chunk of ECG data and concatenates this to the last part until it has gathered enough data to create a key.

4.2.2 Key generation

The key generation part of the software is concerned with turning a continuous, raw ECG signal into a stream of bits. This stream of bits is what we will refer to as the generated key. This key generation is what is referenced in Figure 4.3. The key generation is two fold. First, the R-peaks are extracted from a raw ECG signal, and then the IPIs are calculated from the R-peaks, and finally some bits are extracted from the IPIs.

The raw ECG signal is used as input to the Pan-Tompkins algorithm, of which we used the Matlab implementation created in [47]. This outputs the points in time where the R-peaks occur. From this the IPIs can be calculated by simply computing the difference between R-peaks. The n:th IPI c can be calculated as $c = a - b$, where b is the n:th R-peak and a is the following R-peak. As a result, if we get X number of R-peaks from Pan-Tompkins algorithm, we can extract $X - 1$ number of IPIs from that. The data is then post processed with quantization code, implemented to fulfill the steps in Section 2.4.2. The resulting bits we get from this is our key.

The key generation code was tested on both of the data sets mentioned in Section 3.2. This includes both Physionet data and data gathered live from our ECG monitoring hardware. This helped us to be confident in the fact that our ECG monitoring system did gather high-grain enough information. This testing included a manual analysis of the values, as well as running the outputted bit streams through ENT, which is, as mentioned in Section 2.1.2 a pseudorandom number sequence test program. After ENT was run against both data sets a comparison was made on the output.

4.2.3 Touch detection

The implemented touch detection algorithm was based on the use of a CWT function in Matlab, see Section 4.1.2. This function was used on the gathered ECG-signals.

We start the entire process with a 10-second long baseline ECG-reading where no touch occurs. The CWT function is then applied to this signal. By extracting the biggest value in the resulting CWT matrix we have gathered a baseline maximum CWT value. This baseline max CWT value is then used as a threshold to what is considered a "normal" max CWT value. Once the baseline reading is done, the program starts to look for a touch in the ECG-signal. The polling rate decides how often ECG data is gathered, and every time the program polls, this ECG data is then given to the CWT function to get the local maximum CWT value of this chunk. A buffer value is then added to this local maximum CWT value, and if this sum is greater than the baseline maximum CWT value, a touch event will be registered. The reason we add a buffer value to the local max CWT value is to reduce the number of false positives, which are occurrences where the algorithm registers that a touch has occurred but in reality no touch occurred. So it basically acts as a buffer to decrease the sensitivity of our algorithm.

$$baselineMaxCWTvalue < localMaxCWTvalue + bufferValue$$

If this does not show that a touch occurred, the next time ECG data is gathered this data is concatenated to the already gathered data, and this is then checked in the same way with the equation above. The reason that we concatenate the data is that the CWT function gives better values the more data there is.

4.3 Testing our implementation

We chose to look at three different aspects when testing our system:

- Performance
- Randomness of the generated key
- Accuracy of the touch detection

The purpose of this implementation is not to be especially well performing, but it can still be interesting to do some tests on the implemented system in order to get a rough idea of what kind of performance one can expect from such a system. Regarding performance, we wanted to see how long time different parts of our program took to execute.

We chose to look at three different parts of our implementation regarding the performance. We tested how long time it takes to extract IPIs, which means from the moment we have gathered the necessary ECG data, until we have extracted all IPIs from said data. We also tested how long time it takes to detect a touch, meaning the time it takes from the moment the needed ECG data is gathered, until the algorithm decides if there was a touch or not. Finally we tested how long time it takes to generate a key, which is from the moment there are enough IPIs extracted, until the key is generated.

Testing the randomness simply means running the generated bits through a randomness checker. We do this partially to confirm the high level of entropy that the bits gathered from IPIs indeed holds, but also to verify that our system does not lower the randomness of the bits somehow, thus making them more predictable.

By accuracy of the touch detection we mean how many times the algorithm accurately registers a touch. This includes both detecting a touch when there actually has been a touch (true positive), and not detecting touch when there has been no touch (true negative). Of course, false negatives and false positives are also concerned.

4.3.1 Implementation of tests

The tests were done in such a way that Jim was wearing the ECG sensors from the MySignals and Albin was wearing the ECG sensors from the Bitalino. Our implementation is, as described in Section 3.3.2, a practical proof of concept of the theoretical concept, from one person's view. In our implementation we gather the baseline ECG data and perform the detection algorithm from the ECG data that is gathered from the ECG sensors of MySignals. This means that in our tests we simulated Jim's part of the theoretical concept.

A test run was then executed as a normal run of our program. It started with a 10 second long gathering of baseline ECG data from Jim's ECG sensors. Albin then touched Jim after either 2, 7 or 12 seconds after the gathering of baseline ECG data was done. We chose these times because of how our detection algorithm works, where it checks the ECG data for a touch with a polling rate of 5 seconds. This way we tested the touch detection algorithm both for if there had been a touch and if there had not been a touch, and in different polling intervals of the ECG data. If the program detected a touch, which we simply printed in the command window in Matlab, it went on to gathering new ECG data from both Jim's and Albin's ECG sensors and extracting IPIs from these until there were enough to create a key with the total size of 128 bits. We chose 128 here due to it being a small number, but really any number could have been chosen. If the program did not detect a touch we simply "forced" it to detect a touch by touching the sensors, which gives a huge spike in the ECG data. This test was performed 30 times in total, where 10 of these were done when the touch happened after 2 seconds, 5 seconds and 12 seconds.

To measure the time the different parts of our program took we simply implemented time stamps in our program at the correct locations in the code. These were then logged in a text file after each test. When all tests were completed we calculated the mean value, standard deviation and variance for each of these values.

To evaluate randomness the outputted bits were used. We ran for 128 bit keys every run, and did 30 runs. So in total the number of bits on which the randomness tests were based on was 3840. These bits were used as input to the ENT test suite, which calculates a couple of randomness measurement values, of which we mainly considered the entropy.

To determine the accuracy of the touch detection algorithm we simply filled in a table if it detected the touch correctly or not after each test. The touch detection algorithm is used during two phases in the program. First to detect the initial touch, and then to check if there has occurred another touch during the gathering of the final data to extract IPIs from. During the first phase we had the touch occur after either 2, 7 or 12 seconds. When it occurred after 2 seconds the touch detection algorithm could only give a true positive (detecting a touch that occurred) or a false negative (failing to detect a touch that occurred). In the other cases there could also be a false positive (detecting a touch when no touch occurred). This could happen if the algorithm detected a touch during the first polling intervals of ECG data, before a touch actually occurred. In the second phase the touch detection algorithm is used to detect if a touch occurred during the time when ECG data is gathered for the creation of the key. In this case there can only either be a false positive or a true negative, since no touch actually occurred during this part.

Finally we also logged the maximum baseline CWT value and the maximum CWT value during a touch from each of these tests. From these values, same as for the performance values, we calculated the mean value, standard deviation and variance.

5

Results

In this chapter results are presented, both in the form of signal readings, as well as analyses on the final system. The signal readings are presented as a basis for design choices, as well as further discussion. The analyses of the system include performance and randomness tests.

5.1 ECG readings

This Section contains the ECG measurements conducted that were used as a basis for decisions made in the project.

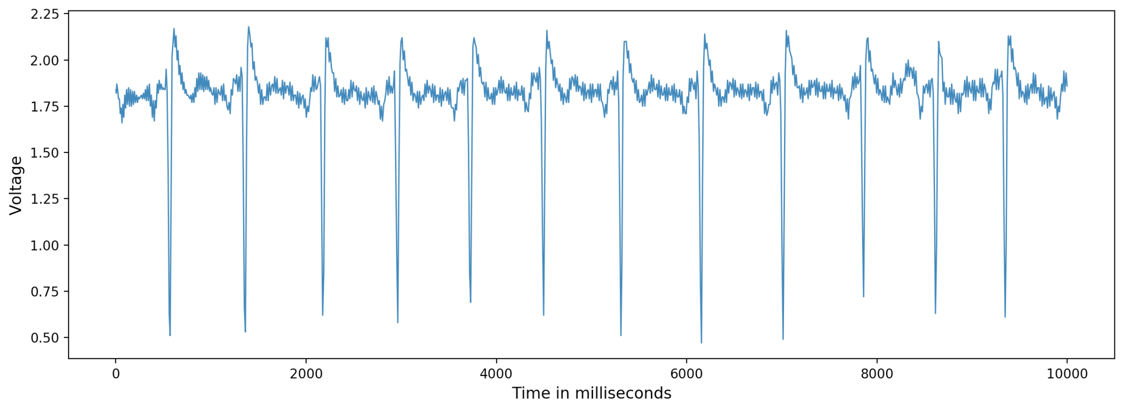
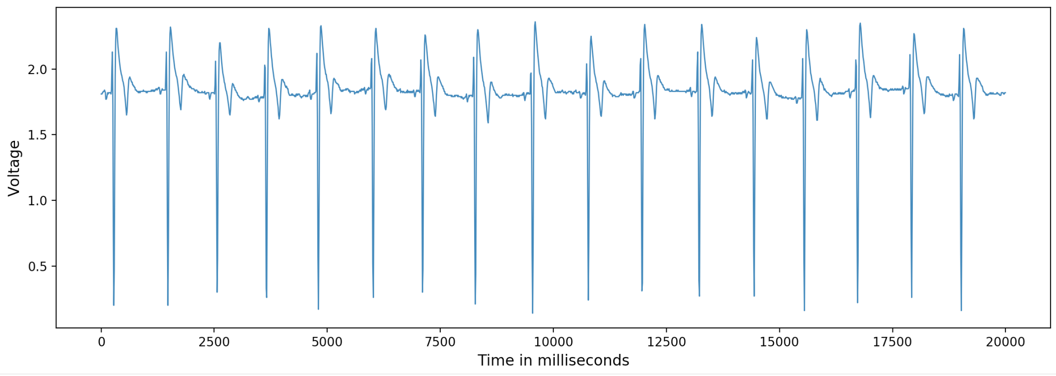


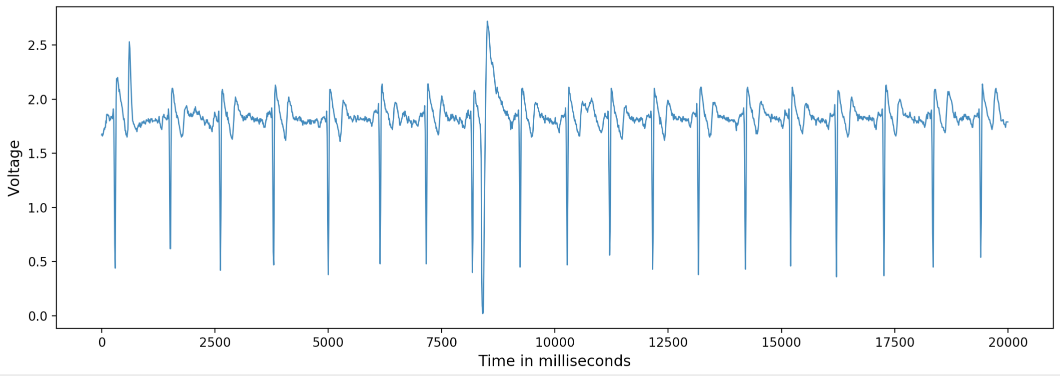
Figure 5.1: A standard 10 second ECG measurement where the subject was sitting

In Figure 5.1 we see a normal ECG measurement, where the subject is not being touched. You can see the ECG being quite regular, with the exception of some noise introduced by the sensor equipment. This noise (the small "vibrations") was not always present in our measurements, but this example highlights it.

In Figure 5.2 we see a comparison between ECG measurements using the standard 3-lead positioning versus the slightly modified positioning that we used in this report.



(a) Normal 3-lead positioning



(b) Final 3-lead positioning

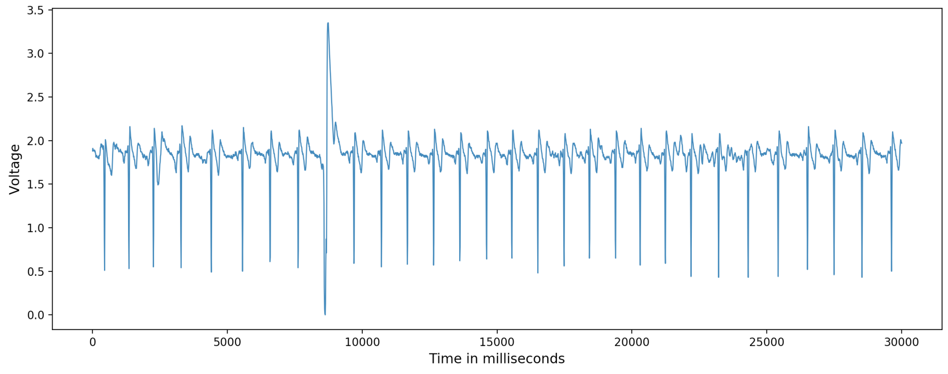
Figure 5.2: A comparison of ECG readings given different lead positioning

(a) utilizes the standard positioning, where the RA lead is positioned on the upper right pectoral muscle, the LA lead is positioned on the upper left pectoral muscle and the RR lead is positioned slightly below the navel. (b) Has the RR lead in the same position, but the RA lead is moved to the inside of the right arm just below the bend of the arm, and the LA is moved to the inside of the left arm just below the bend of the arm.

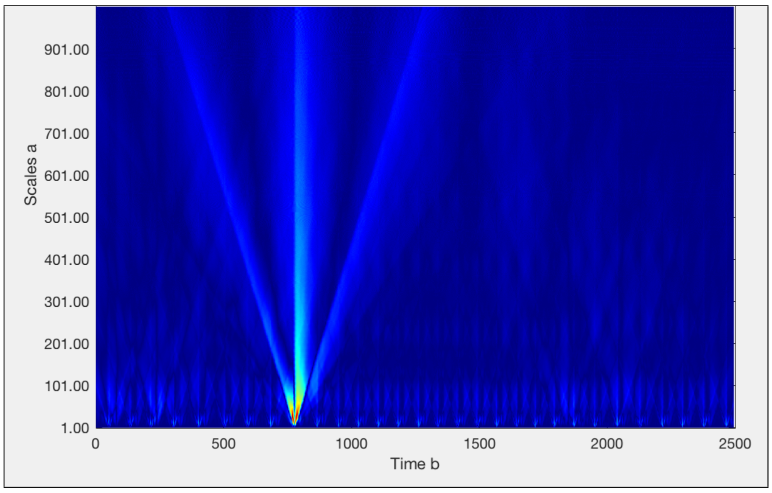
We conducted 10 experiments which were 20 seconds long each, and touched the subject once in the 7000 to 13000 millisecond time span. In neither of the tests using the regular 3-lead positioning could we find any sign of a touch occurring in the resulting output. In the modified positioning, however, we could find the touch event in a majority of the cases. This is what prompted the choice of changing from the standard 3-lead positioning.

The 10/10/10-experiments concerned in Figures [5.3, 5.4, 5.5, 5.6] were all conducted in the same way, with the only variation being the action that the subject was performing. A summary of the analysis of the data seen in these Figures can be found in Table 5.1.

The experiments are called 10/10/10 due to the first 10 seconds consists of the



(a) ECG measurement



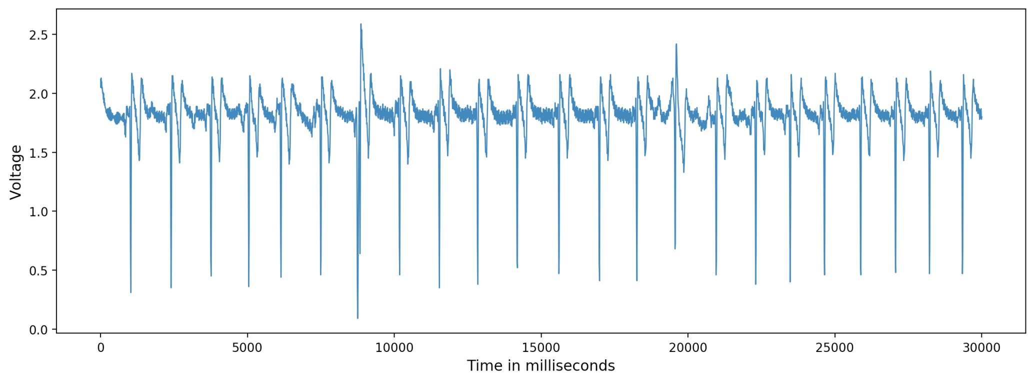
(b) Corresponding continious wavelet

Figure 5.3: The 10/10/10-experiment performed while the subject was sitting down

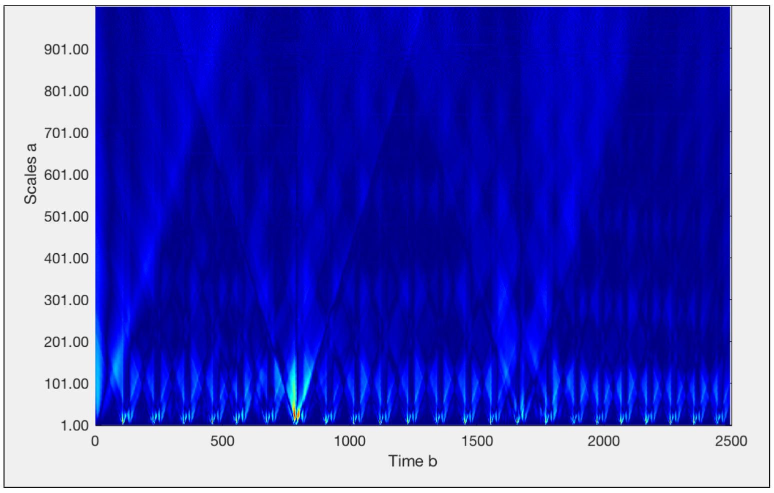
subject only performing the action, then around the 10-second mark the subject gets touched and held for 10 seconds while still performing the action, finally during the last 10 seconds the subject is no longer touched, but still performing the action. In order to generate the continuous wavelet graphs a Matlab function called CWT was used [3].

Figure 5.3 is a good sample ECG that shows a clear "Yes" for the "On touch"-detection, and a "No" for the "On let go"-detection. Figure 5.4 on the other hand is a typical scenario that shows a clear "Yes" for the "On touch"-detection, and a "Maybe" for the "On let go"-detection. Figure 5.5 shows "Maybe" for the "On touch"-detection, and a "No" for the "On let go"-detection. Finally, 5.6 shows "No" for the "On touch"-detection, and a "No" for the "On let go"-detection. Something worth noting with this last example is that it was also discovered that simply walking without introducing any touching created similar patterns.

In the ECG readings presented we could not detect a pattern that holds for the entire time of the touching period. Furthermore, this means that we could not find



(a) ECG measurement



(b) Corresponding continious wavelet

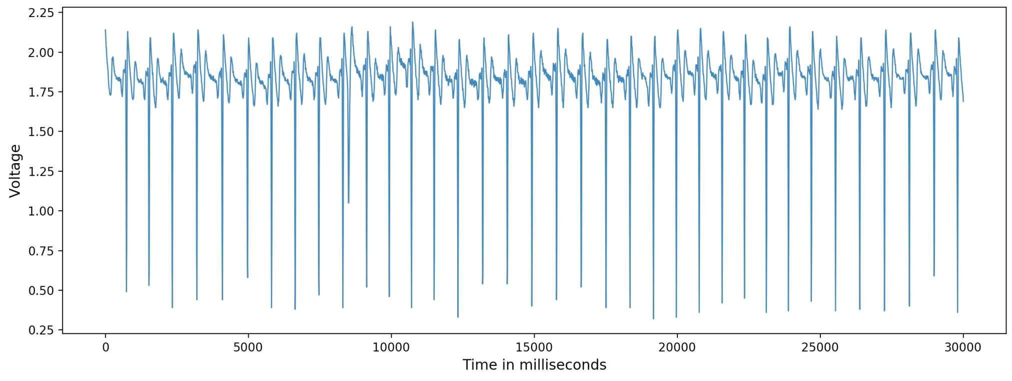
Figure 5.4: The 10/10/10-experiment performed while the subject was lying down

a compound ECG signal while the subjects were touching. In total we performed 120 of the 10/10/10-experiments, and during none of these could we detect anything resembling a compound ECG signal.

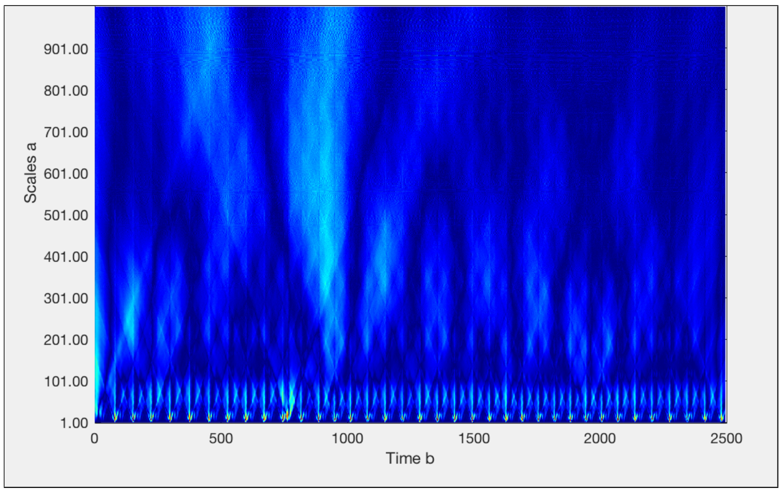
To conclude, using our hardware, we could not find a compound signal, but we could detect moment of touch with a certain percentage threshold, during certain conditions, as seen in Table 5.1.

5.2 Touch detection

In figure 5.7 the results from the touch detection algorithm are shown. How the tests were performed is described in Section 4.3.1. The different bars indicate the outcome from the touch detection algorithm. A true positive indicates it detected a touch when a touch actually occurred and a false positive means that it detected a touch, but no touch occurred. A false negative indicates it did not detect a touch when a



(a) ECG measurement

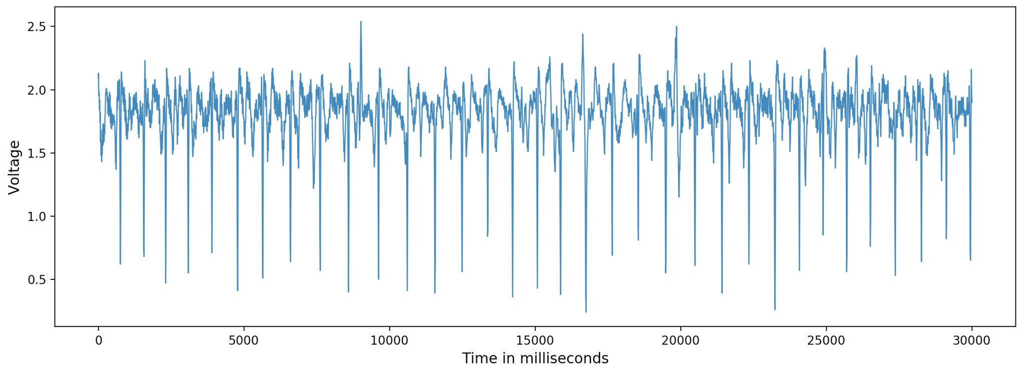


(b) Corresponding continious wavelet

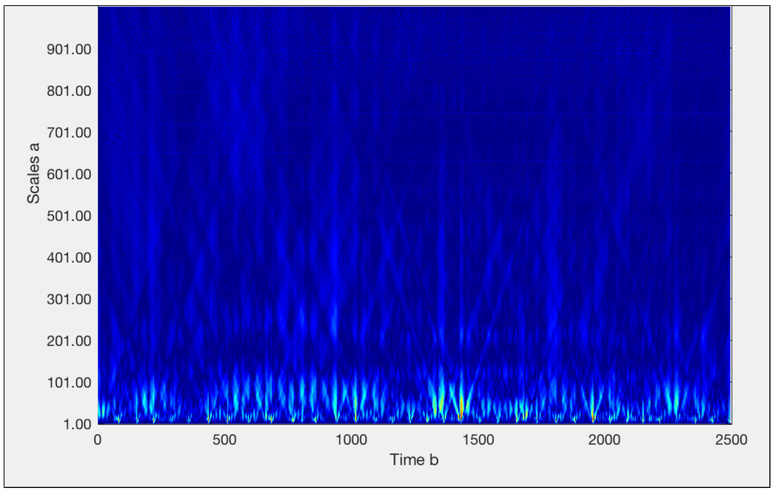
Figure 5.5: The 10/10/10-experiment performed while the subject was standing up

touch actually occurred and a true negative means it did not detect a touch when no touch occurred. So the true positives and the true negatives can be interpreted as good results, as in that the detection algorithm returned the correct answer for what happened in real life.

In Table 5.2 the CWT values that were returned from the CWT function during the tests are shown. Max baseline CWT is the maximum value found in the returned matrix after performing the CWT function on the baseline ECG data for one test. Max touch CWT is the maximum value found in the returned matrix after performing the CWT function on the ECG data where a touch event occurred for one test. The mean value, the standard deviation and the variance have then been calculated from all of the 30 tests that were performed.



(a) ECG measurement



(b) Corresponding continious wavelet

Figure 5.6: The 10/10/10-experiment performed while the subject was walking

5.3 Randomness of keys

In Figure 5.8 the results of the randomness test is presented, which is based on the ENT test suite. The data it was ran against is described in Section 4.3.1. The main value we are looking at is the entropy, which is is quite a bit away from the optimum, which is 1. However, when running against Physionet data we saw similar results. Running output data from other known randomizers, with data sets with the same size, gave similar results as well.

5.4 Performance

This Section presents performance measurements in regards to time. Here we present the overhead introduced by the software, but also take into account that the system is inherently timing-dependent. The data these performance numbers are based

	Detected?	*On touch #*	*On let go #*	*On touch %*	*On let go %*
Sitting	Yes	25	5	83.3	16.7
	Maybe	2	5	6.7	16.7
	No	3	20	10	66.7
	Total	30	30	100	100
Standing	Yes	14	7	46.7	23.3
	Maybe	7	10	23.3	33.3
	No	9	13	30	43.3
	Total	30	30	100	100
Walking	Yes	5	1	16.7	3.3
	Maybe	5	3	16.7	10
	No	20	26	66.7	86.7
	Total	30	30	100	100
Lying down	Yes	13	2	43.3	6.7
	Maybe	7	6	23.3	20
	No	10	22	33.3	73.3
	Total	30	30	100	100

Table 5.1: Table summarizing all the ECG measurement results concerning whether we could detect if a subject got touched, and when the subject got let go of

	Min	**Max**	**Mean**	**Std. dev**	**Variance**
Baseline CWT	0.62115	7.1183	1.2527753	1.4696455	2.1598578
Touch CWT	2.5122	11.7151	6.0120833	2.1746146	4.7289485

Table 5.2: Minimum value, maximum value, mean value, standard deviation and variance of the baseline CWT and the touch CWT of the 30 tests performed.

upon is described in Section 4.3.1, and the numbers referenced in this Section are presented in Table 5.3.

IPI extraction takes an ECG signal as input and outputs the IPIs found in it. The touch detection takes an ECG signal and outputs a boolean saying whether touch was detected. The key generation takes IPIs as input and outputs a stream of bits. Touch detection takes between half a second and an entire second. IPI extraction and key generation is quite quick though. These building blocks are main components in the proposed system.

Given these values one might want to calculate an estimate of the average run time of this system. This is not possible however, since it depends on the pulse of the subjects who use it and the speed of the hardware utilized. Thus, the run time will

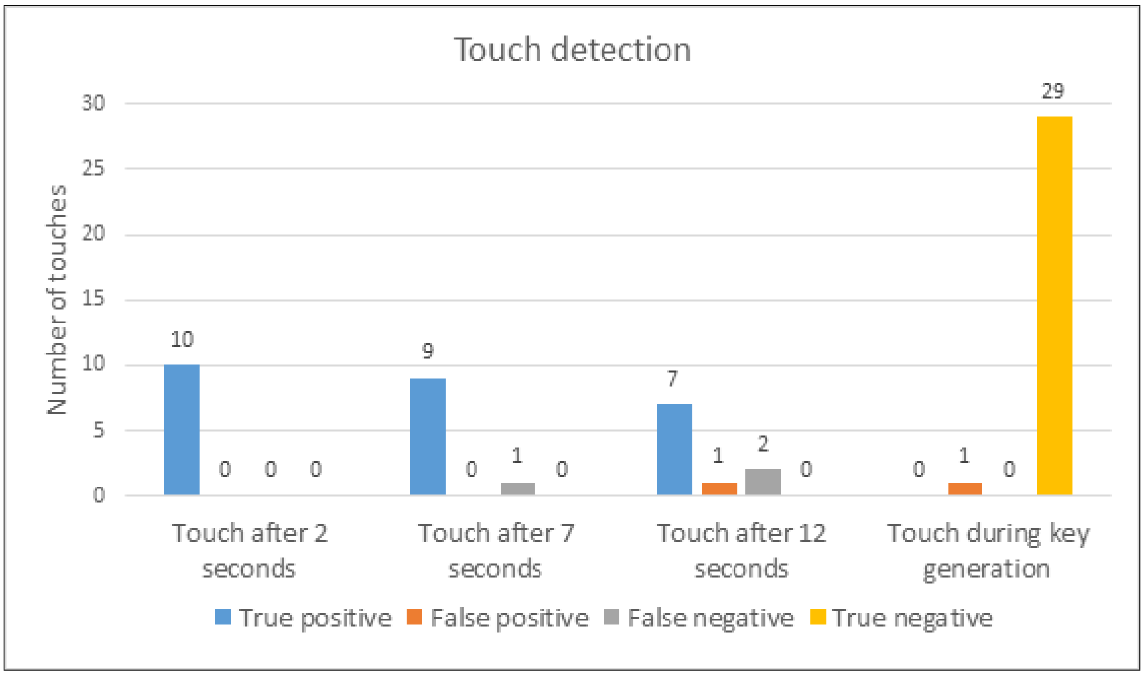


Figure 5.7: Results from the touch detection algorithm of the 30 tests performed.

	Mean	**Std. dev**	**Variance**
IPI extraction	0.0057333	0.0005121	0.0000003
Touch detection	0.6913000	0.0473428	0.0022413
Key generation	0.0139667	0.0066004	0.0000436

Table 5.3: Performance stats for the project. The data is in seconds.

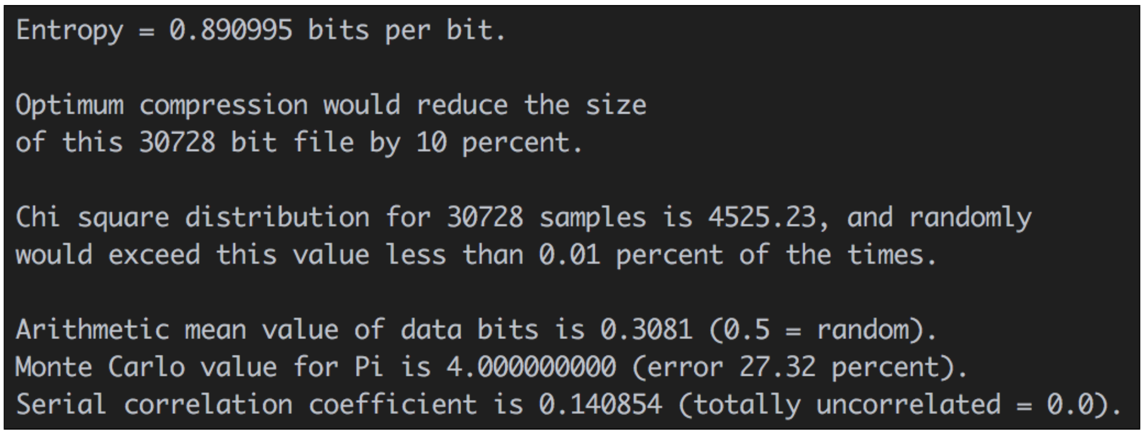
always vary a lot. What we can do is to create a formula using our building blocks, for roughly estimating the run time.

$runtime = hardwareConnect + baseline + timeUntilTouch + touchDetection + (keySize/(((bpm_a + bpm_b)/60) * 4)) + (1/(min(bpm_a, bpm_b)/60)) + IPIextraction + keyGeneration + touchDetection + hardwareDisconnect$

The timings used in this formula are all in seconds. The values we have gathered are IPIextraction, keyGeneration and touchDetection. bpm_a refers to the number of heartbeats per second on person a. The formula also assumes that the IPI extraction algorithm, which relies on the Pan-Tompkins algorithm, never misses an R-peak.

The formula is pretty straight-forward in the sense that it is the addition between a lot of parts, with the exception of the middle part, presented below

$$(keySize/(((bpm_a + bpm_b)/60) * 4)) + (1/(min(bpm_a, bpm_b)/60))$$



```
Entropy = 0.890995 bits per bit.

Optimum compression would reduce the size
of this 30728 bit file by 10 percent.

Chi square distribution for 30728 samples is 4525.23, and randomly
would exceed this value less than 0.01 percent of the times.

Arithmetic mean value of data bits is 0.3081 (0.5 = random).
Monte Carlo value for Pi is 4.000000000 (error 27.32 percent).
Serial correlation coefficient is 0.140854 (totally uncorrelated = 0.0).
```

Figure 5.8: The raw output from running the ENT test suite with the -b flag, which indicate that the input is a bit stream instead of being grouped on bytes

We can make this middle part clearer by abstracting away some details, transforming the above to something more readable.

$$(keySize/IPIsPerSecond) + oneExtraHeartbeat$$

The reason we need an extra heartbeat is the fact that IPIsPerSecond relies on the fact that the first heartbeat of that person has already occured. We calculate oneExtraHeartbeat using the minimum bpm, since that is the person that will take the longest to have an extra heartbeat. Finally, the simplified formula becomes the following formula.

$runtime = hardwareConnect + baseline + timeUntilTouch + touchDetection + (keySize/IPIsPerSecond) + oneExtraHeartbeat + IPIextraction + keyGeneration + touchDetection + hardwareDisconnect$

6 Discussion

In this chapter we reflect over what has been created. We look at how well this proof of concept actually works, both performance-wise but also from a security angle. We also present the possible implications it has for the scientific community in the form of future works.

6.1 Measurements

The measurements done with the MySignals hardware gave pretty good quality readings, given that it was commercial, non-medical grade hardware. That said, the environment in which the readings were recorded was not necessarily as ideal.

All the measurements were taken in the same environment, which we tried to alter as little as possible between readings. We were very aware that we had no prior experience with taking ECG measurements from beforehand, so the idea was that error sources from the environment will occur, but at least we can control so that all these error sources are constant across all readings.

There are some known issues with the room we used. The room was lit up with fluorescent lighting, which is known to be an electromagnetic noise source. Other noise sources include the transformers for the power supplies for our laptops. These were plugged in the entire time, and you could clearly see the difference in noise in the signal from when these were removed from the equation.

How we as test subjects conducted ourselves have also influenced the readings. The fact that two imperfect humans have to perform tasks for the test cases introduces the human error factor. Even though uniformity in how the actions were performed was sought for, small differences in the touching pattern will exist. However, we believe that these smaller error margins become irrelevant with the number of measurements. Bigger error margins such as accidentally touching the cord while performing the actions was a bigger problem. We tried to be very careful in our placements to avoid this ever happening, and we threw away all the readings were we noticed it happening. However, some recordings had suspicious activity that we believe is to be attributed to this, but since we did not notice it happening could not throw away the readings, since that would imply manually manipulating the data set.

Lastly, we suspect that clothing played a big role in how big of an impact touching had on the ECG. We got suspicious when the touch impact was especially high after removing a fleece sweater before touching. We tried rubbing hair and hands in the fleece sweater before performing readings, and sure enough, the spikes were consistently of greater impact. Thus, we tried avoiding this as much as possible, but we did not throw away prior readings where certain pieces of clothing were used.

To us, this looks like the difference in static electricity of the skin of the subjects is a key component to how well the touch detection algorithm will work. This also goes hand in hand with the fact that we chose to standardize the wait time between readings. The waiting was done because we saw significantly smaller impacts of touch the shorter time was spent waiting between readings. One could interpret this as the static electricity of the skin of the two subjects had more time to differentiate, thus giving better results.

We do not know enough about the physics behind this to conclude this, further experiments should be done with this to investigate how much impact static electricity has. However, no matter the reason, the pattern is still there. This means that the context of where our touch detection algorithm is used is very important. In a very noisy environment the failure rate would probably go up by a significant margin, so as to use this in a day to day scenario might not be plausible.

6.2 Performance

The performance is quite good, given that it was not the focus of the project to create something well performing. This means that there are a lot of low hanging fruit that could be optimized, and even better results could easily be yielded.

Looking at the individual performance numbers in Table 5.3 one can see that the touch detection took the most time. This is due to it relying on CWT analysis, which creates a huge matrix from which the graph is plotted. These big matrix operations naturally takes a lot of time. Performing these kind of operations on the GPU instead could be an idea, but the overhead from this setup would probably cause the cons to outweight the pros.

The overall performance of the system was not considered, since it depends a lot on hardware, and since the code was not especially streamlined with performance in mind. This becomes especially apparent looking at the fact that we ran the IPI extraction code for the entire ECG signal once every polling interval. In our case this was 5 seconds, so the time the IPI extraction takes compared to the polling rate caused us to have plenty of time to first perform the IPI extraction but also gather the data for the next polling interval. Perhaps another solution instead of polling could be implemented here, or one could run the IPI extraction not on the entire ECG signal, only the data gathered since the last polling. The touch detection suffers from the same problem, and as that takes a significant amount of time, this could grow out of hand the bigger the ECG input data gets.

The ECG data could get bigger in two ways. If the pulse of the two subjects is very low, or if the desired key size is set to be larger. If the pulse of the subjects is low, the amount of ECG data to generate the number of IPIs needed will grow, since less R-peaks occur in the same amount of time as for someone with a higher pulse. If the desired key size increases, the amount of IPIs needed increase, and thus naturally the size of the underlying ECG.

We only tested our system with a desired key size of 128 bits, and only with our own regular resting pulses. Thus, we do not know how the performance of these parts of our system scales. Investigating this could be crucial in the future, when industry standards for key sizes increase.

The biggest overhead introduced by our solution is the fact that it has to gather baseline data to have input for the initial touch detection. We set this baseline to 10 seconds, so any run of this system, regardless of pulse and key size, has to spend 10 seconds gathering data before it begins. This would, however, not be an issue in a deployed, real world, scenario. In a real world scenario the monitoring would happen all the time, not just when a key would need to be generated. Thus we could use pre-gathered ECG data from the subjects to create a good baseline, before the key generation is even considered. However, since this baseline could be context sensitive, we believe it is good practice that one should only use the latest 10 seconds of that data, since that has proved to be a threshold that worked well for us.

Another performance metric to keep in mind when imagining this system in real world scenario is the ECG-from-EEG extraction. Since we did not have EEG sensor hardware, and instead implemented a proof of concept for this, we could not see the performance impact of it. For us, we only gather the ECG data directly from the second subject. This extraction could perhaps take so much time that it would not be feasible to have such a low polling rate. Since this ECG-from-EEG extraction would be done in addition to extracting the IPIs from the signal, there is a lot of computation that has to be done in every polling interval.

The fact that we have been utilizing a polling scheme results in some potential performance issues, since we perform the IPI extraction and key generation every time we poll. One could perhaps instead use the baseline measurements to extract the average pulse from both subjects, and from that calculate the expected time that the system has to collect baseline ECG data. Then the system only needs to gather data until it is deemed that it probably has enough, and run the algorithms once afterwards. One ECG-from-EEG extraction, one IPI extraction and one key generation would be all that is needed. We did not try this solution, however, since the performance did not cause any functional issues for us. However, if utilizing real

EEG data and extracting ECG from that, this method might have to be considered.

Finally, not only IPIs need to be considered as the basis for key generation. In [39] another ECG-based metric for key generation is proposed. This looks to be almost twice as fast regarding performance. We did not consider this method since the article was very new when this project started. It also had no prior citations.

6.3 Touch detection

When testing our touch detection algorithm we achieved 26 true positives, meaning the algorithm detected a touch when a touch actually occurred, out of 30 tests in total. This results in a 86.6% probability of detection. In 4 of the 30 tests (13%) the results were false, where 3 of these were false negatives, meaning the algorithm did not detect a touch but a touch actually occurred, and 1 of these was a false positive, meaning the algorithm detected a touch where no touch actually occurred.

When looking at the false positives, which means the algorithm detected a touch where no touch actually occurred, this could occur in two parts of our tests: before we actually did the initial touch, and after the initial touch when the key was being generated. In the part during the creation of the key, there was only 1 false positive out of 30 tests in total. This results in 29 out of 30 tests were true negatives, giving our algorithm a specificity of 96.6%. Before the initial touch part we performed 20 tests in total. This is due to the fact that during the first 10 tests the touch happened in the first interval when gathering ECG, so there was no interval where no touch occurred. Out of these 20 tests there was also only 1 which resulted in a false positive. However, in this part there were no true negatives. This is because there always occurred a touch sooner or later in this part of the test. Therefore we cannot get a specificity from this part.

From these numbers we can suspect that the sensitivity of our touch detection algorithm was a bit too low. If we had lowered the threshold a bit we might have been able to increase the probability of detection, or at least decreased the risk of a false negative. This would also however increase the risk of getting a false positive, since the algorithm would be more prone to detecting touches. Setting the threshold depending on the context will probably yield the best results here.

Comparing these results to our initial test results, where we tested if we were able to detect a touch when sitting down, the sensitivity in both cases are almost identical. In the latter case we detected a touch in 83.3% of the tests, and 6.7% of the tests resulted in that we saw something, but the graph was a bit unclear. It is not so peculiar that the sensitivity is almost identical, since they are both based on the same method when trying to detect a touch: by using CWT. In the initial tests we simply printed a CWT graph and tried to decide by looking at it if we saw an anomaly when the touch occurred. In our touch detection algorithm we used the actual values from the CWT function and compared these to find the anomalies.

There are however some differences in these two methods. During our initial tests we were only interested to discover if we were able to detect a touch. We did not look for false positives, and therefore not true negatives as well, so we did not get a specificity from these tests. If we had counted the number of false positives, this would have led to a decrease in true positives, which would decrease the sensitivity. Another thing that might skew the results from the initial tests is the method itself. We decided if we detected a touch or not by looking at a graph and trying to see an anomaly. Since we knew what we were looking for it is reasonable to think that we might have tried to look for this pattern a bit more, and might have been prone to seeing this pattern even if this not was the case. This was not done intentionally, but it is not impossible it might have happened subconsciously. At that stage of our project we were mainly interested in if we could detect a touch or not, not the exact accuracy. Therefore we believe the method was good enough for that purpose, but it is necessary to keep this in mind when comparing it to the results we achieved from our implemented touch detection algorithm.

The implemented touch detection algorithm is comparing the greatest CWT value from some ECG data to the greatest CWT value from the baseline ECG data. Therefore the part where the baseline ECG data is gathered is a very crucial part in order for the touch detection algorithm to work as intended. If there is an anomaly in the gathered baseline ECG data, this will result in a relatively high baseline CWT value. This will then in turn impact the accuracy of the touch detection algorithm, because the resulting CWT value from when the touch occurred now needs to be that much greater in order for the algorithm to detect it as a touch.

An example of this can be seen in Figure **??**. The minimum baseline CWT value was 0.62, the maximum was 7.12 and the mean value was 1.25. As seen here, the test that was performed when the baseline CWT value was 7.12 will have a quite low chance of detecting a touch, because the touch CWT value has to be that much bigger compared to when the baseline CWT value is 0.62. Looking at the touch CWT values, we see that the minimum value was 2.51, the maximum was 11.72 and the mean was 6.01. So even if the touch CWT value was the average value, the algorithm would still not detect a touch if the baseline CWT value was as it was in the maximum case. This really stresses how important the gathering of baseline ECG data is for the touch detection algorithm to work properly. If there is too much noise or a big anomaly during this time, it will be unlikely that the touch detection algorithm, and therefore the entire process, will work.

One quite easy way to improve this would be to increase the time of the gathering of baseline ECG data. If this was done it would reduce the risk of getting a relatively high baseline CWT value, and thus increasing the effectiveness of the touch detection algorithm. In our case we prioritized to be able to perform more tests, rather than to increase the baseline time, which we set to 10 seconds. As previously mentioned, we are convinced we could have achieved better results if the baseline time was increased. The main focus was not to get the number as high as possible though, but rather to see if the concept was possible or not. Tweaking how much baseline data should be used could be done if taking this concept further.

If this was to be implemented in a real world application the time gathering baseline data would no longer be a problem. Assuming the user would use some kind of wearable, like a smart watch, which is able to always read the ECG of the user, there would be an extremely big baseline ECG data to use. This would provide an extremely accurate baseline CWT value, and thus improve the accuracy of the touch detection algorithm. However, this is a trade-off in performance, since the algorithms that run on this data most likely will take significantly more time to run. This is discussed further in Section 6.2.

6.4 Security

We will consider the system more secure the less bits from the generated key the an attacker can obtain. A potential adversary can get information of these bits in two ways. Either the adversary finds a pattern in how the bits are generated, or the attacker can remotely gather information about the bits. We will not consider other attacks, such as where the adversary gets access to the environment running the system, i.e. infecting the Arduino.

6.4.1 Randomness

If the bits generated are truly random, then it would be impossible for an adversary to find any pattern. In Section 5.3 high entropy was shown, using the ENT test suite. However, the input for this was from 30 tests of 128 bits each, which means that the underlying data for that output was 3840 bits, which is very low for it to have any statistical impact. In order for this test to give meaningful numbers the number of tests would have to be very high. Given how long a test takes to perform, this was not feasible in the scope of this project.

In order to gather a meaningful number of bits, to truly test the system, a specific testing platform could be developed. This system would continuously monitor a persons ECG and be on person throughout every day tasks, so that it can gather a big amount of data without being intrusive.

Regardless, since [46] states that these bits are of high entropy, we worked under the assumption that no matter the ENT results we would get, we would consider the bits to have high entropy.

Something to keep in mind here is that entropy is not the only factor to look at. In [42] this issue is brought up, and they argue that heartbeats perhaps does not make good random number generators. The authors of [14] also takes issue with using IPI-based protocols, but for a different reason. The issue here being that computer vision is so good right now, that you can gain up to as much as 70% of the information from a distance. That means that our 128 bit key only has an effective safety of roughly 38 bits. They also bring up the quantization step as a necessity, which fortunately is something we considered.

6.4.2 Attacker model

In order to get information about the ECG of a person we considered two models. The first model involves the adversary touching the subject, and can thus extract that persons ECG from its own EEG, similar to what is presented in Section 3.3.1. The second model involves the adversary somehow getting remote access to the ECG. We did not consider the case where an adversary manages to read the ECG of a person through physical sensors due to how intrusive and easy to detect this would be.

We tried to address the issue with the adversary touching the subject by utilizing the touch detection algorithm we developed. It serves a double purpose in the way that it is used both for initializing the protocol, and for checking for intruders. The security here depends on the reliability of the touch detection algorithm, and has to be further tested in other environments to be fully understood. However, given the context in which our tests were performed, this intrusion detection would work extremely well.

The problem with this is that touch does not necessarily imply intruder. Perhaps the intruder was touching you prior to the start of the protocol, in this case a touch event will never be registered even though the adversary is touching the subject. The best possible solution to this would be if one could would have an algorithm that could determine how many people are touching the subject at a given time. Then it would simply abort as soon as more than two are detected.

Worth noting here is that our touch detection algorithm performs better the more underlying data it is based upon. So in a real world scenario were a lot of baseline data could be gathered, the touch detection could improve even more.

For an adversary to get remote access to the underlying ECG data is really hard. This is due to the fact that the normal attack vectors, communications protocols such as the internet or bluetooth, are not an option here. Computer vision methods could be used, but also other sensors.

As mentioned in Section 2.3.2, not only ECG sensors, but also PPG sensors can be used to extract IPIs. The main difference here lies in the fact that PPG sensors can be remotely used, without physical consent from the person measured. This makes it a good attack vector. One should keep in mind that there are other issues with PPG sensors. They take a long time setting up in order to measure the ambient lighting, which it must counteract in order to get meaningful results.

6.5 Future work

This Section looks into a couple of possible future works of this project. We do not currently intend to further work on this concept but there are many possibilities of what one can do with this.

The fact that we created a system that is easy to set up means that we have paved the way for future research in this area. If a researcher wants to try and generate keys from live ECG data, that work is already done. If a researcher wants to combine ECG data to generate keys, that work is also done. More niche use cases found in our project can also be extracted.

The touch detection in particular is something new that could be used as a timing index for other touch-based protocols. An example of this is the design presented in Section 3.3.1. This is probably the biggest impact that this thesis will have.

The touch detection algorithm could also be used as a general purpose touch detector. This could for instance be useful as intrusion detection in biometric based protocols, in which an adversary could gain additional knowledge from physically touching. One such example is the pacemaker-patching mentioned in section 2.3.1.

Other future work could of course improve on the touch detection algorithm. Combining the basis for the touch detection with other input. As of now we only use CWT. We used this since it got recommended to us to facilitate the initial ECG wave analysis. By plotting it, it was easier to see anomalies. It just so happened that the data from which the plot was based on also happened to be good for our touch detection algorithm. We did not experiment much further with other options here, since we were content with the results from using CWT, so other input should definitely be considered.

One piece of hardware we did consider, was a Galvanic Skin Response (GSR) sensor, which measures electrodermal activity. We noticed a slight change in the measured GSR data from when subjects were holding. That means not only the moment of touch. This could be further investigated and perhaps used to achieve a touch detector which could determine for how long people are holding, not only the moment of touch.

One issue that we did not solve, as explained as in Section 3.3.1, is a timing issue introduced by the distributed nature of this concept. In our theoretical design there are two people who will be wearing identical ECG and EEG equipment, whereas in our implementation we are simulating one side of this design. In the theoretical design, both of these people are needed to start the creation of the keys at the exact same time, called T_0. If they start the creation of their keys at different times, this means they are not using the same ECG data to create the keys from. Since the entire creation of keys are based upon using the IPIs from the ECG data, this would likely cause the two users to create different keys, which would of course break the entire purpose of this process.

To solve this issue there needs to be some synchronization part of the protocol/design which ensures that both people get the same T_0. However, in our implementation we are only simulating one side of the theoretical design, which results in this becoming a non-issue for us.

If one would want to create a product out of this, a lot of steps would have to be taken, the first being solving the timing issue mentioned above. Other steps would include fixing the performance, porting it to something that can run on most devices. Matlab proved to be a great proof of concept platform for us, but is not the best choice for IoT devices in general. Using the PPG sensors found in many modern smart phones might be a better idea then using ECG, to reduce the amount of hardware needed. However, the users would still need to carry EEG sensors on them, which would still not be feasible in day to day life, with todays EEG sensor technology.

Finally, a more general version of this system could be considered. The system created works for two people, but in theory it should be able to extend to multiple people. We do not know what happens with the EEG of a person when multiple people touch, so that would have to be investigated as well.

7

Conclusion

In this thesis we wanted to answer the research question if it is possible to create a security protocol which generates cryptographic keys from the heartbeats of two people when they are touching. The answer to this is yes, the design of such a security protocol has been created, as well as a practical implementation which simulates one side of this protocol. This implementation was evaluated in terms of security and performance. From these results we conclude that it is indeed feasible to create such a security protocol.

There still exists a few problems to be solved before before this is viable in a practical setting. The main problem to solve is a timing issue introduced in the touch detection, which decides when the two people should start generating their keys. This should be further investigated in future work. Another problem is variation in the used measurement devices, which could result in generating different keys. Optimization for the entire system to allow it to run on smaller devices and improvements to the touch detection algorithm would also need to be considered for this to be deployable on smart devices. In the future, we might all be able to communicate securely with just a handshake.

Bibliography

[1] 3-lead ECG positioning. `https://lifeinthefastlane.com/ecg-library/basics/lead-positioning/`. [Online; accessed 22-Mar-2018].

[2] GQRS documentation. `https://www.physionet.org/physiotools/wag/gqrs-1.htm`. [Online; accessed 14-Mar-2018].

[3] Matlab's CWT function. `https://se.mathworks.com/help/wavelet/ref/cwt.html/`. [Online; accessed 26-Mar-2018].

[4] Physiobank Databases. `https://physionet.org/physiobank/database/`. [Online; accessed 12-Dec-2017].

[5] Foteini Agrafioti and Dimitrios Hatzinakos. Ecg biometric analysis in cardiac irregularity conditions. *Signal, Image and Video Processing*, 3(4):329, 2009.

[6] Juan Sebastian Arteaga-Falconi, Hussein Al Osman, and Abdulmotaleb El Saddik. Ecg authentication for mobile devices. *IEEE Transactions on Instrumentation and Measurement*, 65(3):591–600, 2016.

[7] Jorge Blasco, Thomas M Chen, Juan Tapiador, and Pedro Peris-Lopez. A survey of wearable biometric recognition systems. *ACM Computing Surveys (CSUR)*, 49(3):43, 2016.

[8] Justin Boyle, Niranjan Bidargaddi, Antti Sarela, and Mohan Karunanithi. Automatic detection of respiration rate from ambulatory single-lead ecg. *IEEE transactions on information technology in biomedicine*, 13(6):890–896, 2009.

[9] KA Brownley, BE Hurwitz, and N Schneiderman. Cardiovascular psychophysiology. in 1. t. cacioppo & lg tassinary & gg berntson (eds.), handbook ofpsychophysi0 logy (pp. 224-264), 2000.

[10] Kelsey L Bruso and Glen E Newton. Secure biometric authentication from an insecure device, June 7 2012. US Patent App. 12/960,511.

[11] William J Buchanan. Private-key encryption. In *Mastering Networks*, pages 300–310. Springer, 1999.

[12] Johannes Buchmann. *Introduction to cryptography*. Springer Science & Business Media, 2013.

[13] Christian Cachin. *Entropy measures and unconditional security in cryptography.* PhD thesis, ETH Zurich, 1997.

[14] Alejandro Calleja, Pedro Peris-Lopez, and Juan E Tapiador. Electrical heart signals can be monitored from the moon: Security implications for ipi-based protocols. In *IFIP International Conference on Information Security Theory and Practice*, pages 36–51. Springer, 2015.

[15] Sriram Cherukuri, Krishna K Venkatasubramanian, and Sandeep KS Gupta. Biosec: A biometric based approach for securing communication in wireless networks of biosensors implanted in the human body. In *Parallel Processing Workshops, 2003. Proceedings. 2003 International Conference on*, pages 432–439. IEEE, 2003.

[16] Wikimedia Commons. File:21 electrodes of international 10-20 system for eeg.svg — wikimedia commons, the free media repository, 2015. [Online; accessed 15-May-2018].

[17] Wikimedia Commons. File:ekg diagram 1.svg — wikimedia commons, the free media repository, 2018. [Online; accessed 4-June-2018].

[18] Shimal Das and Jhunu Debbarma. Designing a biometric strategy (fingerprint) measure for enhancing atm security in indian e-banking system. *International Journal of Information and Communication Technology Research*, 1(5), 2011.

[19] Hans Delfs and Helmut Knebl. Symmetric-key encryption. In *Introduction to Cryptography*, pages 11–31. Springer, 2007.

[20] Yvo Desmedt. Man-in-the-middle attack. In *Encyclopedia of cryptography and security*, pages 759–759. Springer, 2011.

[21] Gordana Dodig-Crnkovic. Scientific methods in computer science. In *Proceedings of the Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden, Skövde, Suecia*, pages 126–130, 2002.

[22] Nashwa El-Bendary, Hameed Al-Qaheri, Hossam M Zawbaa, Mohamed Hamed, Aboul Ella Hassanien, Qiangfu Zhao, and Ajith Abraham. Hsas: Heart sound authentication system. In *Nature and Biologically Inspired Computing (NaBIC), 2010 Second World Congress on*, pages 351–356. IEEE, 2010.

[23] Diaa Salama Abd Elminaam, Hatem Mohamed Abdual-Kader, and Mohiy Mohamed Hadhoud. Evaluating the performance of symmetric encryption algorithms. *IJ Network Security*, 10(3):216–222, 2010.

[24] Papri Ghosh and Ritam Dutta. A new approach towards biometric authentication system in palm vein domain. *International Journal of Advance Innovations, IJAITI*, 1(2):1–10, 2012.

[25] AL Goldberger, LAN Amaral, and L Glass. 1. m. hausdorff, pc ivanov, rg mark, je mietus, gb moody, c.-k. peng, he stanley, physiobank, physiotoolkit,

and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101:23, 2000.

[26] Khaled Hamze. Cloudflare is protecting the internet using groovy lava lamps. *TechChrunch*.

[27] Feng Hao, Ross Anderson, and John Daugman. Combining crypto with biometrics effectively. *IEEE transactions on computers*, 55(9):1081–1088, 2006.

[28] Anil Jain, Lin Hong, and Sharath Pankanti. Biometric identification. *Communications of the ACM*, 43(2):90–98, 2000.

[29] Anil K Jain, Sharath Pankanti, Salil Prabhakar, Lin Hong, and Arun Ross. Biometrics: a grand challenge. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 935–942. IEEE, 2004.

[30] Anders J Johansson and Henrik Floberg. Random number generation by chaotic double scroll oscillator on chip. In *Circuits and Systems, 1999. ISCAS'99. Proceedings of the 1999 IEEE International Symposium on*, volume 5, pages 407–409. IEEE, 1999.

[31] Emil Jovanov, Aleksandar Milenkovic, Chris Otto, and Piet C De Groen. A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation. *Journal of NeuroEngineering and rehabilitation*, 2(1):6, 2005.

[32] Tero Kivinen. More modular exponential (modp) diffie-hellman groups for internet key exchange (ike). 2003.

[33] Yuriy Kurylyak, Francesco Lamonaca, and Domenico Grimaldi. Smartphone based photoplethysmogram measurement. *Digital image and signal processing for measurement systems*, pages 135–164, 2012.

[34] Troy M Lau, Joseph T Gwin, and Daniel P Ferris. How many electrodes are really needed for eeg-based mobile brain imaging? *Journal of Behavioral and Brain Science*, 2(03):387, 2012.

[35] Ming Li, Wenjing Lou, and Kui Ren. Data security and privacy in wireless body area networks. *IEEE Wireless communications*, 17(1), 2010.

[36] Juan Pablo Martínez, Rute Almeida, Salvador Olmos, Ana Paula Rocha, and Pablo Laguna. A wavelet-based ecg delineator: evaluation on standard databases. *IEEE transactions on biomedical engineering*, 51(4):570–581, 2004.

[37] Rollin McCraty, Mike Atkinson, Dana Tomasino, and William A Tiller. The electricity of touch: Detection and measurement of cardiac energy exchange between people. *Brain and values: Is a biological science of values possible*, 1998:359–379, 1998.

[38] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[39] Sanaz Rahimi Moosavi, Ethiopia Nigussie, Marco Levorato, Seppo Virtanen, and Jouni Isoaho. Low-latency approach for secure ecg feature based cryptographic key generation. *IEEE Access*, 2017.

[40] Joachim H Nagel, Kedu Han, Barry E Hurwitz, and Neil Schneiderman. Assessment and diagnostic applications of heart rate variability. 1993.

[41] Ernst Niedermeyer and FH Lopes da Silva. *Electroencephalography: basic principles, clinical applications, and related fields*. Lippincott Williams & Wilkins, 2005.

[42] Lara Ortiz-Martin, Pablo Picazo-Sanchez, Pedro Peris-Lopez, and Juan Tapiador. Heartbeats do not make good pseudo-random number generators: An analysis of the randomness of inter-pulse intervals. *Entropy*, 20(2):94, 2018.

[43] Jiapu Pan and Willis J Tompkins. A real-time qrs detection algorithm. *IEEE transactions on biomedical engineering*, (3):230–236, 1985.

[44] T Pishbin, SMP Firoozabadi, N Jafarnia Dabanloo, F Mohammadi, and S Koozehgari. Effect of physical contact (hand-holding) on heart rate variability. *autonomic nervous system*, 3:20, 2012.

[45] Ming-Zher Poh, Daniel J McDuff, and Rosalind W Picard. Non-contact, automated cardiac pulse measurements using video imaging and blind source separation. *Optics express*, 18(10):10762–10774, 2010.

[46] Masoud Rostami, Ari Juels, and Farinaz Koushanfar. Heart-to-heart (h2h): authentication for implanted medical devices. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1099–1112. ACM, 2013.

[47] Hooman Sedghamiz. Matlab implementation of pan tompkins ecg qrs detector., 03 2014.

[48] Jinyang Shi and Kwok-Yan Lam. Vitacode: electrocardiogram representation for biometric cryptography in body area networks. In *Ubiquitous and Future Networks, 2009. ICUFN 2009. First International Conference on*, pages 112–115. IEEE, 2009.

[49] Hugo Silva, Hugo Gamboa, and Ana Fred. Applicability of lead v2 ecg measurements in biometrics. *Proceedings of Med-e-Tel*, 2007.

[50] Hugo Silva, André Lourenço, Renato Lourenço, Paulo Leite, David Coutinho, and Ana Fred. Study and evaluation of a single differential sensor design based on electro-textile electrodes for ecg biometrics applications. In *Sensors, 2011 IEEE*, pages 1764–1767. IEEE, 2011.

[51] Colin Soutar, Danny Roberge, Alex Stoianov, Rene Gilroy, and BVK Vijaya Kumar. Biometric encryption. *ICSA guide to Cryptography*, pages 649–675, 1999.

[52] Marek Sỳs and Zdeněk Říha. Faster randomness testing with the nist statistical test suite. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 272–284. Springer, 2014.

[53] William O Tatum IV. *Handbook of EEG interpretation.* Demos Medical Publishing, 2014.

[54] Vernon L Towle, José Bolaños, Diane Suarez, Kim Tan, Robert Grzeszczuk, David N Levin, Raif Cakmur, Samuel A Frank, and Jean-Paul Spire. The spatial location of eeg electrodes: locating the best-fitting sphere relative to cortical anatomy. *Electroencephalography and clinical neurophysiology*, 86(1):1–6, 1993.

[55] Rossouw Von Solms and Johan Van Niekerk. From information security to cyber security. *computers & security*, 38:97–102, 2013.

[56] J Welch, P Ford, R Teplick, and R Rubsamen. The massachusetts general hospital-marquette foundation hemodynamic and electrocardiographic database–comprehensive collection of critical care waveforms. *Clinical Monitoring*, 7(1):96–97, 1991.

[57] Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John Guttag, Frédo Durand, and William Freeman. Eulerian video magnification for revealing subtle changes in the world. 2012.

[58] Guang-He Zhang, Carmen CY Poon, and Yuan-Ting Zhang. Analysis of using interpulse intervals to generate 128-bit biometric random binary sequences for securing wireless body sensor networks. *IEEE Transactions on Information Technology in Biomedicine*, 16(1):176–182, 2012.

[59] Zhaoyang Zhang, Honggang Wang, Athanasios V Vasilakos, and Hua Fang. Ecg-cryptography and authentication in body area networks. *IEEE Transactions on Information Technology in Biomedicine*, 16(6):1070–1078, 2012.

[60] W Zong, GB Moody, and D Jiang. A robust open-source algorithm to detect onset and duration of qrs complexes. In *Computers in Cardiology, 2003*, pages 737–740. IEEE, 2003.