Freenet as a broker for "Medical" IoT Data

by

Yannick Stebler

A thesis presented for the degree of
Bachelor of Information Science

Technology and Computer science
BFH - Bern University of Applied Sciences
Switzerland
June 17, 2021

Acknowledgements

I would like to thank my thesis advisor, Benoist Emmanuel, for his help during the thinking and writing process of this bachelor thesis. I would also like to thank all the people who helped me to improve the structure and language of this report while reading through and commenting on it.

Abstract

The goal of this thesis is to design the communication of IoT data independently from the manufacturer via a new communication path. The requirements for this new communication path are the anonymity of the transmitted data, i.e. it should not be known from whom the data originates or who has received this data. Furthermore, the data that is transmitted should also be encrypted. This prevents unauthorized parties from intercepting and reading the data. To meet these requirements, a new broker (data intermediary) was established with Freenet, which replaces the manufacturer as broker. In addition, a prototype was created that uses this new communication path to exchange data. This performs an Elliptic Curve Diffie Hellmann Key Exchange to encrypt the data. This prototype has shown us that it is possible to replace the manufacturer of the IoT data as a broker and to ensure independence from the manufacturer. Thus, it is possible to enable a more sustainable use of IoT devices.

Contents

Chapter 1

Introduction

1.1 Topic

IoT devices are on the rise and it is hard to imagine our everyday life without them. They make many everyday tasks easier, collect information or connect us with other people. New applications for these small and often practical devices are being added every day.
However, the majority of these IoT devices have a very large weak point. The data exchange of the IoT devices is often handled by the manufacturer of the devices. This means that if a manufacturer goes bankrupt, the IoT devices from this manufacturer often become useless, as the data exchange between the devices can no longer take place.
My goal is to start with exactly this rough problem. In this work we want to work out a way to break the dependency of the IoT devices to the manufacturers. It should be possible for us to operate our IoT devices without the communication via the manufacturer, and thus free us from the risk of a complete failure of our IoT infrastructure in the event of bankruptcy of the manufacturer. We will thus outsource the communication of the IoT devices to a broker. This broker will replace the manufacturer as a communication node and it should be possible for us to operate this broker locally (in the same network).
In this paper I will show you a more detailed insight into our new communication line, and I will also provide you with the most important information about the selected broker.

1.2 What is a Broker

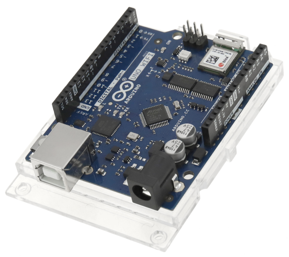
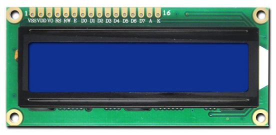
A broker is a server or service that caches data from clients and makes it available for retrieval. It therefore represents the communication interface between two end devices. In our case, the broker is the data node between the IoT device that measures the patient's vital signs and the end device that makes the measured data available to the medical staff for viewing. Data exchanged through a broker is often public and available for all to see. The challenge of storing data anonymously and making it unrecognizable so that it cannot be read by third parties is mandatory for the use of a broker when sensitive data is involved.

1.3 Freenet

Freenet is free software which lets you anonymously share files, browse and publish "freesites" (web sites accessible only through Freenet) and chat on forums, without fear of censorship. Freenet is decentralised to make it less vulnerable to attack, and if used in "darknet" mode, where users only connect to their freinds, is very difficult to detect.
Communications by Freenet nodes are encrypted and are routed through other nodes to make it extremely difficult to determine who is requested the information and what its content is.
Users contribute to the network by giving bandwidth and a portion of their hard drive (called the "data store") for storing files. Files are automatically kept or deleted depending on how popular they are, with the least popular being discared to make way for newer or more popular content. Files are encrypted, so generally the user cannot easily discover what is in his datastore, and hopefully can't be held accountable for it. Chat forums, websites, and search functionality, are all built on top of this distributed data store.[4]

1.4 Hardware

For the implementation of this work, different hardware is used. The following hardware is used:

Hardware	Overview
	The arduino UNO Wifi Rev.2 is a very basic entry to IoT. It offers many different possibilities to build and expand it's functions. The Ardunino UNO Wifi Rev.2 has 14 digital input/output pins and 6 analog inputs. With its secure ECC608 crypto chip accelerator it is possible to directly connect the Arduino UNO Wifi Rev.2 to a Wifi network of your choice.
	LDR's or Light Dependent Resistors are used in light/-dark sensor circuits. The resistance of LDR normally is very high, sometimes as high as 1 Mega ohms, if they then get illuminated the resistance drops dramatically.
	The DHT22 is low cost easy to use humidity and temperature sensor. It can measures humidity from 0-100% with and accuracy of 2-5% and the temperature from -40 to 80°C with accuracy +-0.5°C
	The LCM1602C is a 2x16 LCD display which can be used to display information up to 32 characters over 2 lines.

1.5 Goal Definitions

In this section, the goals of the bachelor thesis are defined, which are to be achieved, in addition, some optional goals are defined, which can be added as extensions if there is enough time. These are the "must" goals:

- The exchange of data should be largely anonymous. It should not be possible to know the data is being uploaded and who has read it.
- The exchange of data must be protected and data must only be readable by authorized persons / hardware.
- The data access should only be known to the authorized parties.
- Document architecture and encryption.
- Implement the documented architecture and encryption in a prototype.
- Encryption and signing of the data using elliptic curves. If possible using elliptic curves, if not possible use AES / RSA.

These are the "optional" goals:

- The security-relevant functions of Freenet should be used in the best possible way.
- Data caching on SD card at the receiver should be encrypted via AES encryption

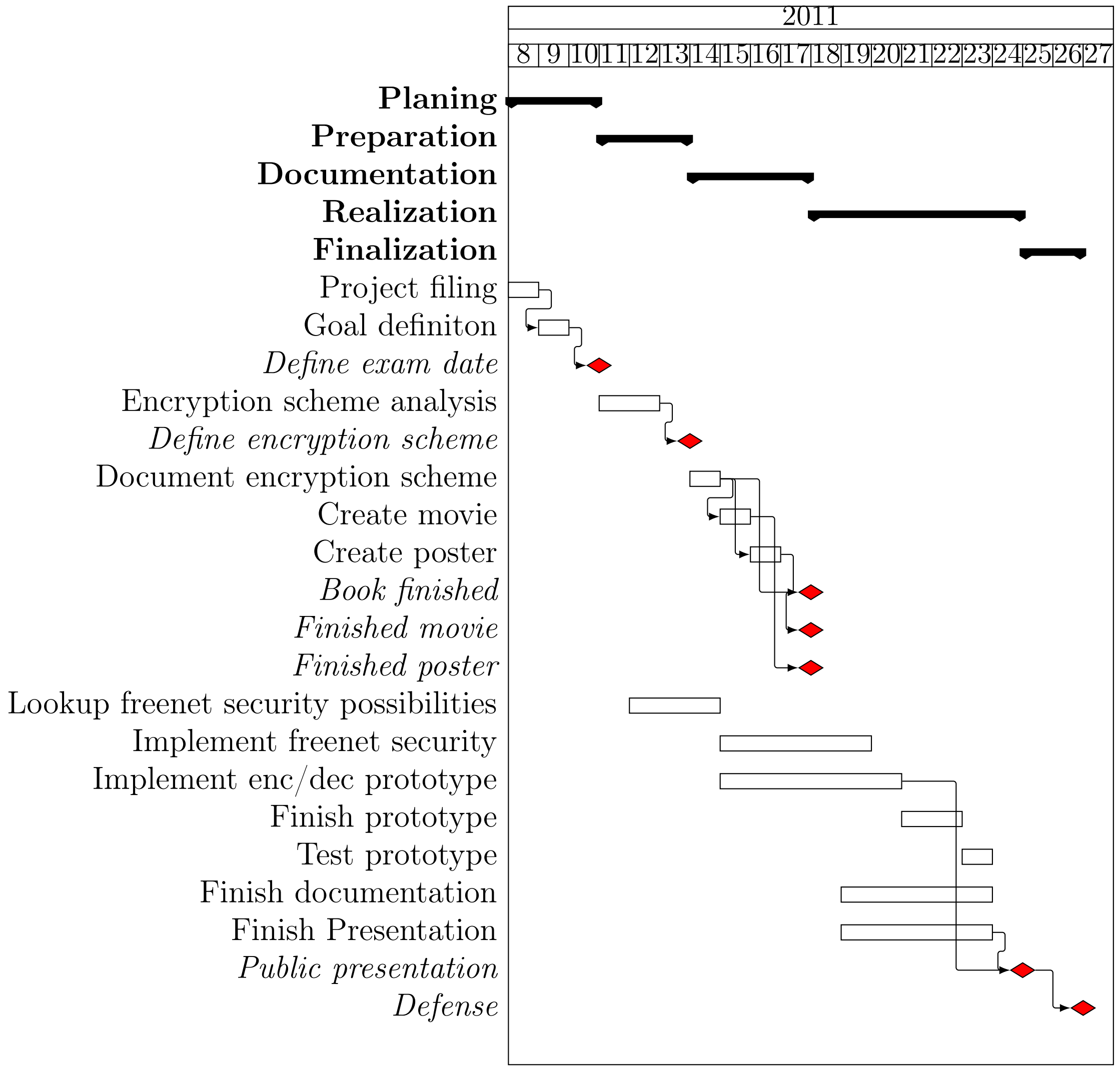
Chapter 2

Planing

2.1 Planing Methode

For the planning of my bachelor thesis, I referred to the well-known project management method, the waterfall model. The waterfall model is a classic project management model that consists of successive phases. In contrast to agile project management models, each individual phase must be completed before the next one begins, since the individual phases build on each other. Although this project planning method has some disadvantages, the advantages outweigh them. It allows you to better structure the work. Since phase after phase must be completed. It provides easy tracking, this because a waterfall model can be easily read and developed by all stakeholders without special certification or training. The main disadvantage of the waterfall model is that it is difficult to make changes during the process. However, since the requirements and goals are already defined at the start of my bachelor thesis, the likelihood of changes during the process is negligible.
On the next page you will find the waterfall model of my bachelor thesis. This was divided into 5 phases. Planning, Preparation, Documentation, Realization, Finalization. Each of these phases has at least 1 milestone, which marks the end of the respective phase and at the same time the transition to the next phase.

2.2 Timeplan


2011
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
Planing
Preparation
Documentation
Realization
Finalization
Project filing
Goal definiton
Define exam date
Encryption scheme analysis
Define encryption scheme
Document encryption scheme
Create movie
Create poster
Book finished
Finished movie
Finished poster
Lookup freenet security possibilities
Implement freenet security
Implement enc/dec prototype
Finish prototype
Test prototype
Finish documentation
Finish Presentation
Public presentation
Defense

Chapter 3

Architecture

3.1 Process architecture

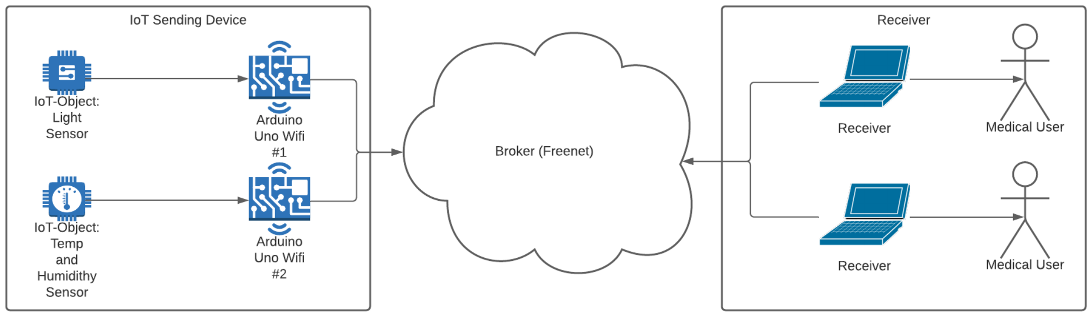


Fig. 3.1: Process overview

The figure above shows a complete process overview. Both endpoints are visible, on the left the IoT transmitter, the device that is with the patient and measures his vital signs. On the right is the endpoint device where the medical staff can evaluate the patient's values. The data exchange between the 2 endpoints takes place via a broker. In our architecture Freenet was used as broker. In order for the data to be transferred securely between the two endpoints, it must be protected from access and attack by third parties. This is done by an encryption between the two endpoints.

3.2 Arduino slot layout

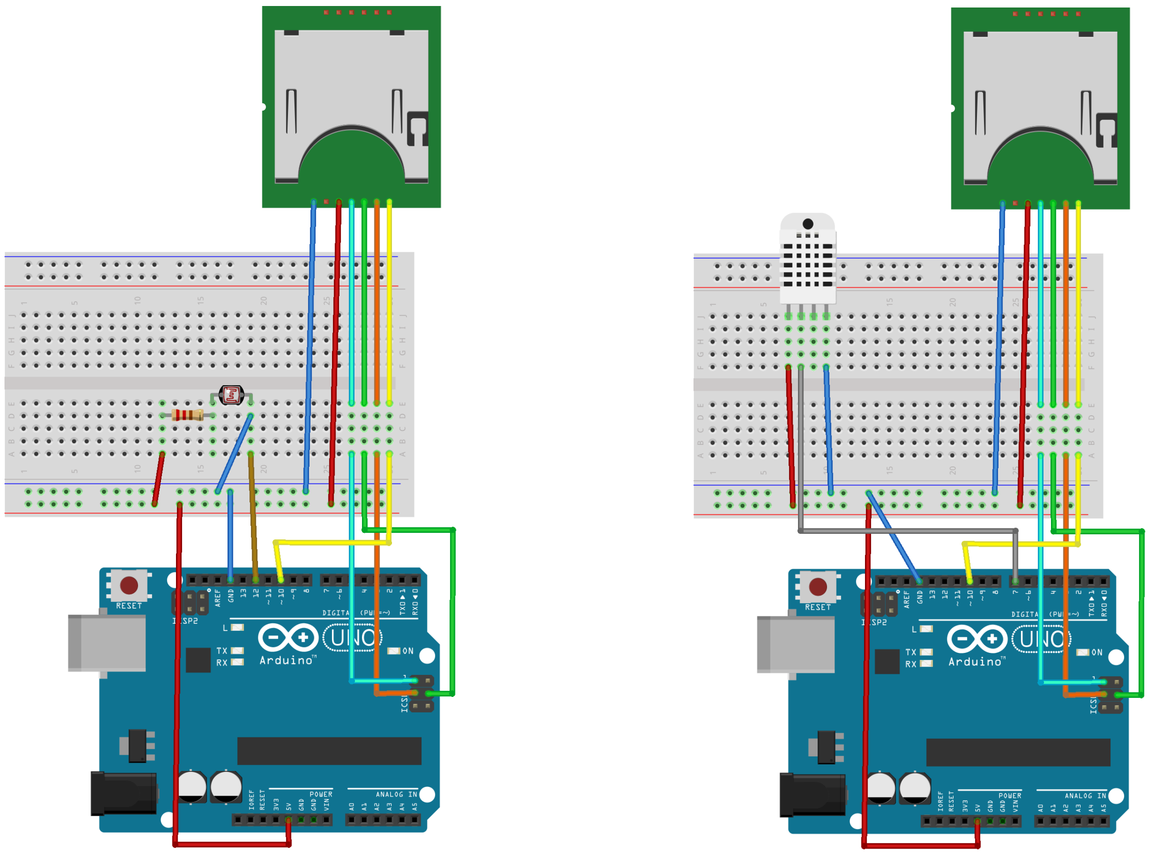


Fig. 3.2: Arduino slot layouts

The two figures show an overview of the two Arduino cabling (IoT transmitter). The left figure shows the wiring with the light sensor and the right figure the wiring with the temperature sensor. Bend Arduinos are also equipped with a micro SD card reader module, which is used to cache the communication data.

3.3 Preparations for initialization of communication

In order for communication to occur between the two devices (sender and receiver), the two devices must exchange data over an insecure channel. Since the IoT devices are quite limited in terms of inputting data, they are equipped with a QR code. This QR code is scratchable, so that when the devices are received, it can be detected whether a QR code has already been read. The QR code contains a key that is used to send data to this device via Freenet. This initial data exchange takes place via an insecure channel. As soon as a recipient has registered with the sender via the QR code. A key exchange is executed via the insecure channel. In addition to the Key Exchange, new specific paths for the data exchange between the two devices are created and transmitted encrypted via the insecure channel. These new paths are then used for the subsequent data exchange.

Chapter 4

Mathematical basics

4.1 Diffie-Hellman key exchange

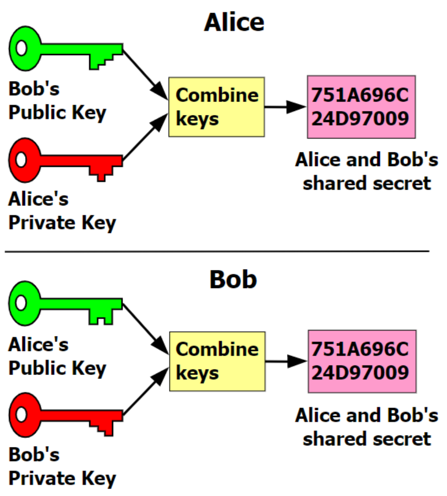


Fig. 4.1: Diffie Hellman key exchange

The Diffie Hellman Key Exchange is a key agreement protocol. It allows two communication partners to agree on a common secret key via a public non-secure channel. This agreed secret key can then be used for a symmetric cryptosystem. There are now different variants of the Diffie Hellman Key Exchange procedure, which are also used in various corners of the Internet.[12]

4.1.1 Mathematical basics

In the following chapters, the mathematical basics of the asymmetric crypto methods based on the discrete logarithm are explained. The Diffie Hellman Key Exchange is composed of various mathematical aspects. The essential aspects that are necessary for the design and security analysis of the Diffie Hellman Key Exchange are explained in more detail in the following chapters.

4.1.1.1 One-way function

A Diffie Hellman Key Exchange uses certain mathematical basics, one of these basics is the one-way function(s). A one-way function is a mathematical function that is "easy" to calculate in terms of complexity theory, but very "difficult" to reverse.

A mathematical one-way function $f : X \to Y$ must satisfy the following properties:

The calculation of $y = f(x)$ is "easy".

This means that an algorithm exists that can calculate the value in polynomial time.

The calculation of $f^-1(y) = x$is "hard".

This means that there is no "fast" algorithm for this problem that could solve it in polynomial time.[1]

4.1.1.2 Discrete exponential function and discrete logarithm

The discrete exponential function $b^x \mod m$ return the remainder when dividing b^x by m. And the inverse of the discrete exponential function is called discrete logarithm.
The big advantage of discrete exponential functions is that they can be calculated efficiently even with large exponents. So even functions with a number of more than hundred bits can be calculated within a second on the PC if implemented cleverly. For this to happen efficiently, both Euler's theorem
(For all $a, n \in \mathbb{N}$ with $ggT(a, n) = 1$ applies $a^{\varphi(n)} \equiv 1(\mod n))$
as well as the Square Multiply method are used.
But for the inverse, i.e. the discrete logarithm $f(x) = b^x$, no efficient and fast algorithm is known yet. Even with the greatest hardware effort and the best known algorithms, one only achieves calculation times that exceed the lifetime of our universe. Therefore, it can be assumed that the discrete exponential function is a one-way function according to today's knowledge. [12]

4.1.1.3 Group theory

Group theory deals with the algebraic structures of groups in mathematical space. Accordingly, a group consists of a set of things (numbers, symbols, objects, motions) and a calculus (a linkage $*$) that specifies how to deal with the set. This calculation rule is bound to certain rules, the so-called group axioms:

- The combination of two elements of the set results in an element of the same set. (Completeness of the set)
- For the combination of the elements the bracketing is not of importance, i.e., it is valid $(a * b) * c = a * (b * c)$ for all a, b, c. (associative law)
- There is an element ε in the set, which does nothing with respect to the conjunction, i.e. a $*$-neutral element:$a * \varepsilon = \varepsilon * a = a$ for all a
- For each element a there is a reverse element with respect to the link, i.e. an $*$-inverse element a^*. This element has the property of yielding the neutral element when it is linked with a: $a^* * a = a * a^* = \varepsilon$.

A group is therefore defined as follows: It is a pair $(G, *)$. Here G is a set and $*$ is the linkage to G. The following figure can be described accordingly: [12]

$$* : G \times G \to G(a, b) \mapsto a * b$$

4.1.1.4 Prime residual class group

In arithmetic, the integers relatively prime to n from the set $\{0, 1, ..., n-1\}$ of n non-negative integers form a group under multiplication of modulo n. This group is called the multiplicative group of integers modulo n.
This type of group plays a significant role in cryptography because they are finite abelian groups with respect to multiplication. An important property of the prime residue class is that for every prime residue class $a + n\mathbb{Z}$ there exists a prime residue class $b + n\mathbb{Z}$ such that holds:

$$ab \equiv 1 \pmod{n}$$

The prime residue class $b+n\mathbb{Z}$ is therefore the inverse element of $a+n\mathbb{Z}$ with respect to the multiplication in the prime residue class group $\mathbb{Z}_n^*$.[14]

4.1.1.5 Primitive root modulo n

Certain elements of prime residue class groups are called primitive roots in scale theory. The primitive root has a primitive property, through this property it is possible to represent each element of the primitive residue class group by means of one of its powers. For example, the number 3 is a primitive root modulo 7, since holds:[15]

$$\begin{aligned}
3^1 &\equiv 3 \pmod 7\\
3^2 &\equiv 2 \pmod 7\\
3^3 &\equiv 6 \pmod 7\\
3^4 &\equiv 4 \pmod 7\\
3^5 &\equiv 5 \pmod 7\\
3^6 &\equiv 1 \pmod 7
\end{aligned}$$

4.2 Elliptic Curves

Elliptic curves are special algebraic curves on which addition is geometrically defined. They play an important role in cryptography to construct secure encryption methods. This due to the various properties of elliptic curves. For example, the discrete logarithm problem is much harder to compute on an elliptic curve than on finite solids. For the representation of elliptic curves, so-called affine planes with a point at infinity are used. However, elliptic curves can also be defined in arbitrary bodies, but only elliptic curves over finite bodies are used in cryptography.

An elliptic curve is defined as follows according to Willems.
Let $\mathbb{K}$ be a body of characteristic not equal to 2 and 3.
A polynomial equation of the form E: $y^2 = x^3 + ax + b$ with $a, b \in \mathbb{K}$ with discriminant $\Delta = 4a^3 + 27b^2 \neq 0$ is called an elliptic curve.
The condition $\Delta \neq 0$ ensures that the polynomial $x^3 + ax + b$ has three pairwise different zeros.[13]

4.2.1 Elliptic Curve Diffie-Hellman (ECDH)

The elliptic curve Diffie Hellman (ECDH) differs from the general Diffie Hellman in that it is based on the discrete logarithm problem of the elliptic curve (ECDLP) as opposed to the discrete logarithm problem (DLP) of the general Diffie Hellmann. The Elliptic Key Diffie Hellmann is an anonymous key agreement protocol that allows 2 parties to share a secret over a insecure channel. Each party has a public-private key pair, of which only the public key is used for the exchange. The private key remains with the party and is needed to create the shared secret.
The ECDH works as follows. A and B agree on the elliptic curve group $\mathbb{E}$ of order n and a primitive element $\mathbb{P}$ in $\mathbb{E}$, which then also has the order n. $\mathbb{E}$, n and $\mathbb{P}$ are assumed to be known to the adversary. The ECDLP, which the ECDH is based on, is defined as the computation of the integer k given $\mathbb{P}$ and $\mathbb{Q}$ such that $\mathbb{Q} = [k]\mathbb{P}$. The ECDH let A and B compute a shared secret key S, using the property of the ECDLP as described below. A selects an integer a in the range [2, n-1], computes $\mathbb{Q} = [a]\mathbb{P}$ and sends $\mathbb{Q}$ to B. B on the other hand selects an integer b in the range [2, n-1], computes $\mathbb{R} = [b]\mathbb{P}$ and sends $\mathbb{R}$ to A. A and B receives $\mathbb{R}$ and $\mathbb{Q}$ respectively, and computes the shared secret key S; $S = [a]\mathbb{R} = [b]\mathbb{Q} = [a][b]\mathbb{P} = [ab \mod n)]\mathbb{P}$. Both A and B get the same value for S, and the shared key is established.[9]

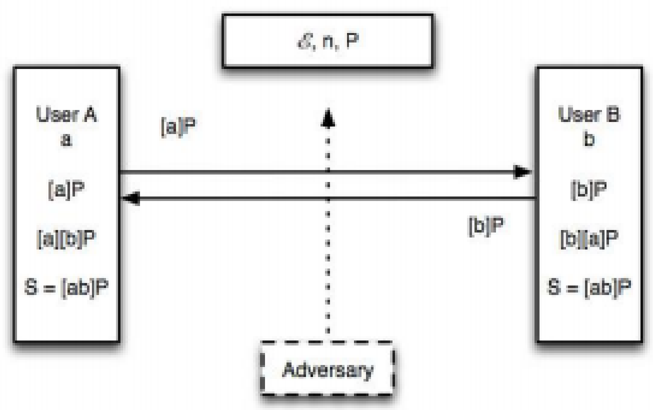


Fig. 4.2: Elliptic curve Diffie Hellman

Chapter 5

Processes and protocols

5.1 Initialization of communication

In order to establish a connection between the transmitter and receiver, these two devices must register with each other. Since the IoT transmitter is a device with very limited functions, it cannot register the receiver by input. Each transmitter is therefore equipped with a QR code. This QR code contains the address that can be used to communicate with the transmitter via Freenet. The receiver registers the transmitter via its QR code. The address of the recipient is then transmitted to the sender via Freenet. The sender registers the recipient's address. As soon as both sides have registered each other's communication path, an EC is sent to the receiver. an ECDH (Elliptic Curve Diffie Hellman) is performed by both sides using the SecP256 elliptic curve. For this, an ECC (elliptic curve cryptography) key pair is created by both sides. The public key of the key pair is then forwarded to the other side via Freenet. Both sides calculate a shared secret after receiving the public key from the other side. Up to this point, the entire communication takes place on a non-secure channel. As soon as both parties have calculated the shared secret, both sides create new connection paths on Freenet. These paths are then encrypted with the shared secret and transferred to the other side. Further communication then takes place via the secure channel created in this way.

5.1.1 SecP256r1 Elliptic curve

SECP256r1 is an elliptic curve defined in SEC 2. It is an elliptic curve in the field z_p, where p is a 256-bit prime. The "r" stands for random. The elliptic curve SECP256r1 has another sibling curve, the SECP256k1. The "k" stands for Koblitz. A Koblitz curve has some special properties that allow it to implement the group operations more efficiently. However, one assumes that a small security trade-off takes place, since often a greater "randomness" of the selected parameters leads to a higher security. For this reason, the elliptic curve SECP256r1 was used for the implementation of the prototype.

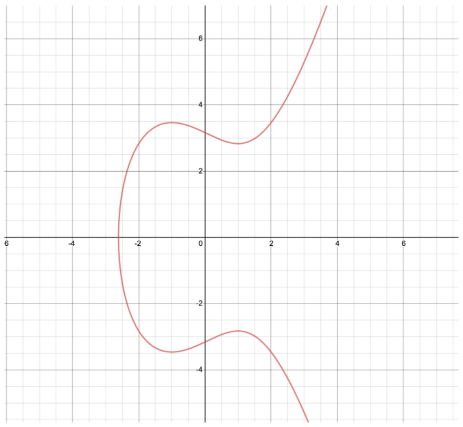
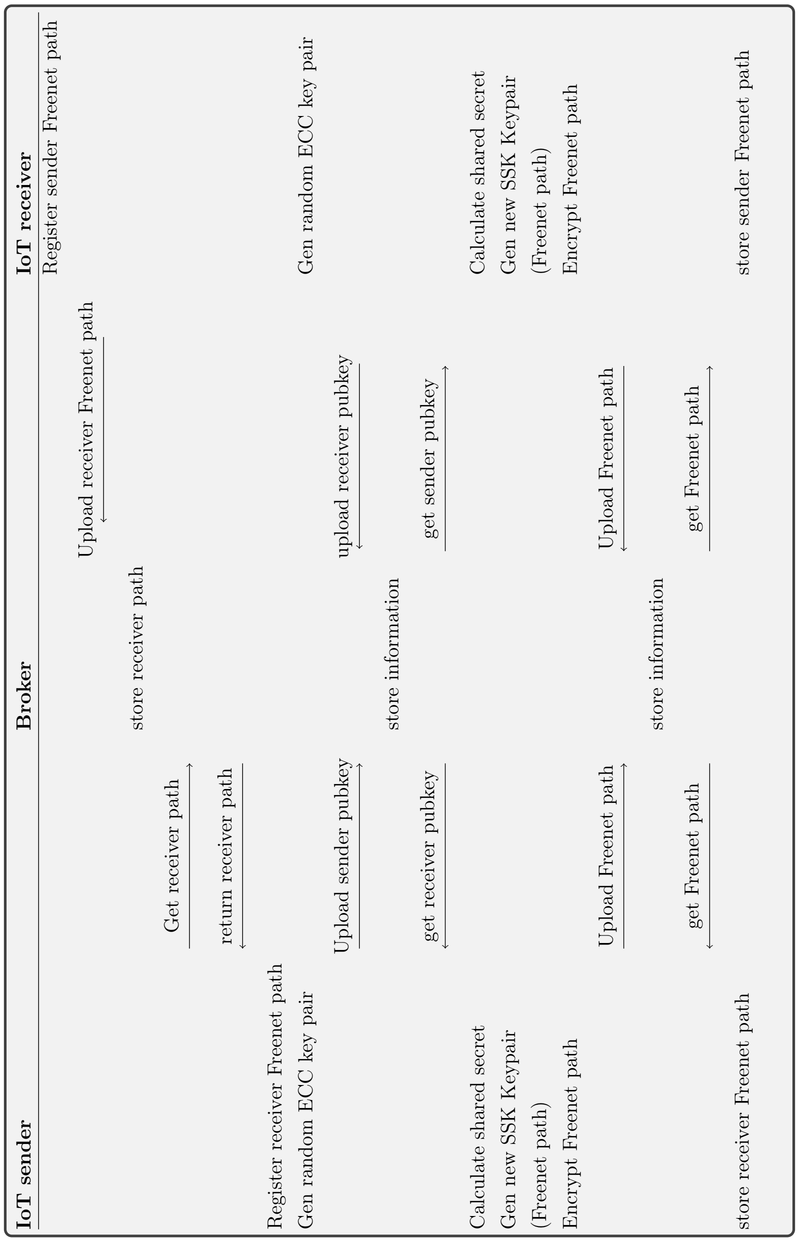


Fig. 5.1: SecP256k1 Elliptic curve

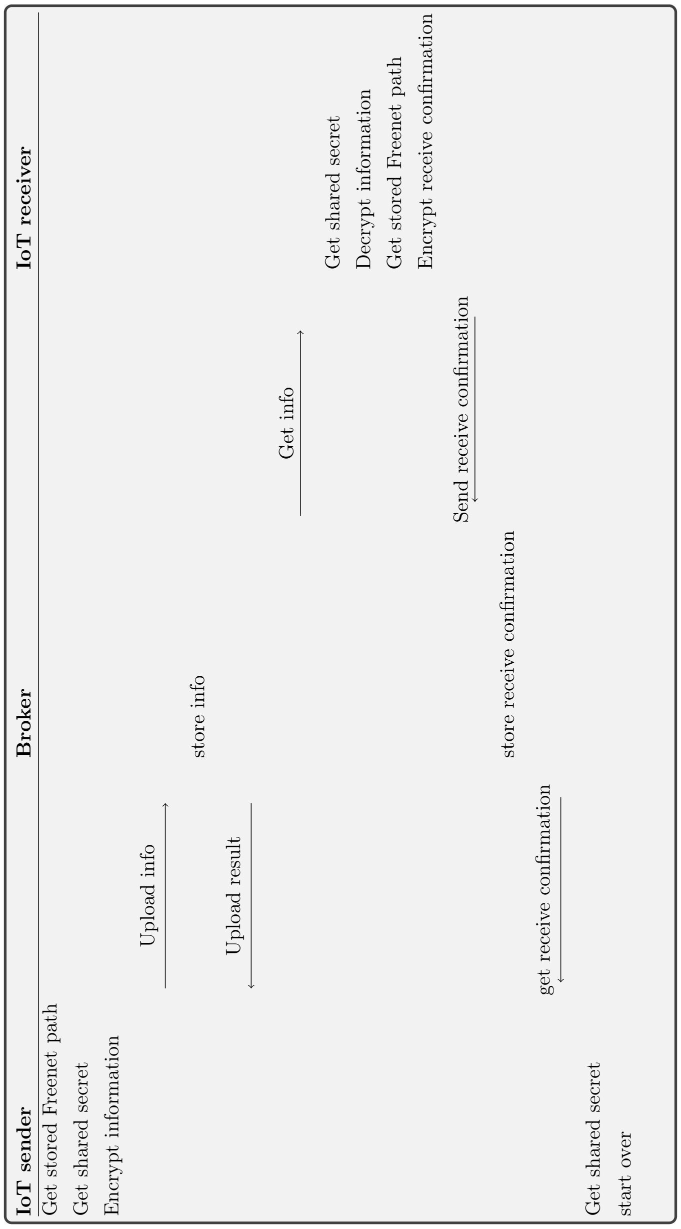
5.2 Initialization scheme



5.3 Send and receive data

Once the connection between a transmitter and a receiver is established, the two devices proceed to the next step. The IoT transmitter reads the required information from the SD card. It thus obtains the Freenet path of the receiver and the key for encrypting the data. It then reads the data from the sensor. Formats it and encrypts it. The data is then uploaded in encrypted form from the IoT transmitter to the receiver's Freenet path. The receiver regularly checks whether new data has already been uploaded from the transmitter. If it detects that new data has been uploaded, it downloads it and verifies it. For this purpose, it decrypts the received data. If the data has been downloaded correctly, the receiver sends the sender a confirmation that it has received the data via its Freenet path.
An IoT transmitter only sends the data if it has received confirmation of the previous upload of data, thus ensuring that the receiver is still active and therefore still receiving data.

5.4 Communication scheme



Chapter 6

Freenet

6.1 FCPv2 protocol

Freenet uses a simple, text-based protocol called FCPv2 (Freenet Client Protocol) to allow third-party applications to interact with Freenet.
Functionalities supported by the FCPv2 Protocol are:[6]

- Inserting of data into Freenet
- Retrieval of data from Freenet
- Querying the status of Freenet
- Managing the other Freenet nodes that are connected to your own node.

6.1.1 ClientHello

The first message sent from the third-party client to the node on a given connection is always the ClientHello message. No other messages are allowed to be sent to the node before the ClientHello is sent. The node will always respond with a NodeHello message.

```
ClientHello
Name=My Freenet Client
ExpectedVersion=2.0
EndMessage
```

Field	Possible values	Mandatory?	Description
Name	Arbitrary string, on one line	yes	A name to uniquely identify the client to the node.
ExpectedVersion	2.0	yes	The version of FCP to expect. In this case it will always be 2.0

6.1.2 NodeHello

The NodeHello message is the reply a node sends after receiving a ClientHello message from a client.

```
NodeHello
CompressionCodecs=4 - GZIP(0), BZIP2(1), LZMA(2), LZMA_NEW(3)
Revision=build01475
Testnet=false
Version=Fred,0.7,1.0,1475
Build=1475
ConnectionIdentifier=6f467be43d838f8e02877e7f176a73bd
Node=Fred
ExtBuild=29
FCPVersion=2.0
NodeLanguage=ENGLISH
ExtRevision=v29
EndMessage
```

Field	Example or [range]	Description
FCPVersion	2.0	The version of FCP being utilized for this connection.
Node	Fred	
Version	Fred, 0.7, 1.0, 1475	A comma-delimited list consisting of • node name • node version • protocol version • build number
Build	1475	
Revision	build01475	
ExtBuild	29	The build of freenet-ext.jar being utilized by the node.
ExtRevision	v29	The revision of freenet-ext.jar being utilized by the node.
Testnet	true, false	
Compression Codecs	4 - GZIP(0), BZIP2(1), LZMA(2), LZMA$_{N}EW(3)$	A list of the compression codecs supported by the node.
Connection Identifier	6f467be43d838f8 e02877e7f176a 73bd	A unique connection identifier, reported in hexadecimal format.
NodeLanguage	ENGLISH	The language currently utilized by the node.

6.1.3 ClientPut

The ClientPut message is used to specify an insert into Freenet of a single file. This file can be inserted by referring to a file on the disk, including the data directly in the message or redirecting to another key.

```
ClientPut
URI=KSK@
Metadata.ContentType=text/plain
Identifier=My Test Insert
DataLength=4
Data
<Data>
```

Field	Possible values	Mandatory?	Description
URI	CHK@ KSK@ SSK@ USK@	yes	The type of key to insert.
Metadata. ContentType	Any MIME type	no	The MIME type of the data being inserted.
Identifier	Arbitrary text string	yes	This is just for client to be able to identify files that have been inserted.
DataLength	Integer from 0 to ?	yes	The length of the data file being included in this FCP message.
Data			Indicates the end of the message. The data follows this line.

6.1.4 ClientGet

The ClientGet message is used to specify a get of a single file from Freenet.

```
ClientGet
URI=KSK@
Identifier=My Test Identifier
EndMessage
```

Field	Possible values	Mandatory?	Description
URI	CHK@ KSK@ SSK@ USK@	yes	The type of key to insert.
Identifier	Arbitrary text string	yes	This is just for client to be able to identify files that have been inserted.

6.1.5 DataFound

DataFound indicates a successful fetch of the key, but does not actually include the data.

```
DataFound
Identifier=Request Number One
Metadata.ContentType=text/plain;charset=utf-8
DataLength=37261
EndMessage
```

Field	Example or [range]	Description
Identifier	Arbitrary text string	This is just for client to be able to identify files that have been inserted.
Metadata. ContentType	Any MIME type	The MIME type of the data being inserted.
DataLength	Integer from 0 to ?	The length of the data file being included in this FCP message.

6.1.6 AllData

AllData is a message from the node returning data from the node.

```
AllData
Identifier=Request Number One
DataLength=37261 // length of data
StartupTime=1189683889
CompletionTime=1189683889
Metadata.ContentType=text/plain;charset=utf-8
Data
 <37261 bytes of data>
```

Field	Example or [range]	Description
Identifier	Arbitrary text string	This is just for client to be able to identify files that have been inserted.
DataLength	Integer from 0 to ?	The length of the data file being included in this FCP message.
StartupTime	Time	The startup time of the GetRequest.
CompletionTime	Time	The completion time of the GetRequest.
Metadata. ContentType	Any MIME type	The MIME type of the data being inserted.

6.1.7 GenerateSSK

The GenerateSSK message is used to ask the node to generate us an SSK keypair. Response will come back in a SSKKeypair message.

```
GenerateSSK
Identifier=My Identifier Blah Blah
EndMessage
```

Field	Possible values	Mandatory?	Description
Identifier	Arbitrary text string	yes	This is just for client to be able to identify files that have been inserted.

6.1.8 SSKKeypair

Is a message sent from the Freenet node to a client program in response to the client issuing a GenerateSSK command.

```
SSKKeypair
InsertURI=freenet:SSK@AKTTKG6YwjrHzWo67laRcoPqibyiTdyYufjVg54fBlWr,
           AwUSJG5ZS-FDZTqnt6skTzhxQe08T-fbKXj8aEHZsXM/
RequestURI=freenet:SSK@BnHXXv3Fa43w~~iz1tNUd~cj4OpUuDjVouOWZ5XlpX0,
           AwUSJG5ZS-FDZTqnt6skTzhxQe08T-fbKXj8aEHZsXM,AQABAAE/
Identifier=My Identifier from GenerateSSK
EndMessage
```

Field	Example or [range]	Description
InsertURI	inserting URI	Consists the URI needed to insert to freenet.
RequestURI	request URI	Consists the URI needed to request from freenet.
Identifier	Arbitrary text string	This is for the Client to Identify to witch GenerateSSK Request this Keypair is.

Chapter 7

Attacks an Prevention

The following chapter describes various possible attack scenarios and the possible solutions to prevent them. The following attacks are considered:

- Spoffing of the data: The information stored on the broker can be manipulated by anyone who can connect to the Freenet Node and also has knowledge of the Freenet URIs. To prevent this from happening, Freenet URIs must be exchanged in encrypted form for each new connection.
- Interruption of the data flow by flooding Freenet: Freenet runs as a node on a device, the exact behavior of Freenet and the effects due to the high load need to be further investigated. (Packet loss and availability)
- Physical attack: There are several possible physical points of attack, one of which is the IoT transmitter. In our prototype, it is equipped with an SD card. Stored on it are all the information about the connections to the receivers. This data is stored on the SD card in encrypted form. However, the key for this encryption is stored on the EEPROM of the IoT transmitter. Furthermore, we have a Freenet node that is also physically located on site. If this is accessed, or if the permissions are not set correctly, a lot of information can be tapped here as well.

- Man in the Middle: The man in the middle attack is very heavy here due to the use of Freenet. Since here is the only point of attack for a Man in the Middle. There is, however, the possibility that an attacker could pretend to be the recipient of the sender's QR code and thus establish his own communication with the sender. To ensure that the QR code has not yet been read out for registration before it is used at the customer's premises, it is delivered as a scratch-off zone on a piece of paper. If the QR code has already been scratched off, it can be assumed that it has already been read out.

- Shell injection: When creating the receiver software, the first attempt was made to use the same ECC library as for the transmitter. Since this is a C library, it was provided as a .dll. However, since the use of this DLL leads to a high risk of a shell incjection, it was decided to use a specific C# library that performs the calculations of the elliptic curve and thus has a much lower risk of a shell injection.

Chapter 8

Problems

During the work on my bachelor thesis I stumbled more and more over single major as well as minor problems. These are both in the creation of the prototype as well as in planning and the effort of the work in addition to the daily work. However, more problems have arisen during the creation of the prototype. Thus I had to determine with the wiring after several unsuccessful attempts, which the slots of the Arduino UNO Wifi Rev3 are not identical with the normal Arduino Uno.
So that the cabling of the so-called ICSP ports that enable the control of the Micro SD Card Reader module via the instructions described in the Arduino Uno on the digital ports of 7-13 are present, so they are not connected to the digital ports on the Arduino Uno Wifi Rev3. In order to find out where to find these ports, I spent several hours studying the schematic of the chipset. Once I understood the schematic, I was able to connect the SD card module without any problems.
One problem that was identified during the development of the prototype is the roundtrip time. Since it is very important in medicine to always get the most current data in order to always know directly about the condition of the patient. Because of the using of a broker it came to the fact that we had an increased roundtrip time. We have now decided on a maximum roundtrip time of 10s for the prototype. Since this would probably be too much for a productive implementation, we would have to take care of this problem again. Another problem was the ECDH (Elliptic Curve Diffie-Hellmann) key exchange. Since I only found a C library for the Arduino, I had to use another library for the receiver, which is a C application. The use of these 2 different libraries leads however at first also to problems, since the Shared Secret computed at the end did not agree. Furthermore, the library for using the FCP protocol for "C" was outdated and no longer current, so I decided to combine the required messages in a separate small "library".

Chapter 9

Conclusion

The goal of this work was to analyze and develop a prototype that makes it possible to detach the communication of IoT data from the manufacturer of the devices. So that they no longer have to struggle with the problem of dependence on the manufacturer. Based on this problem, a new communication path for the IoT data had to be developed. The goal was to exchange the data as anonymously and securely as possible. In addition, unauthorized parties should be prevented from gaining access to the data or its information. In order to implement these goals, communication via Freenet was targeted. Freenet itself already stands for an anonymized decentralized network with which data can be exchanged. However, since anyone who connects to my Freenet node can see this data, the data had to be additionally protected. For this an Elliptic Curve Diffie-Hellmann Key Exchange was executed. With this it was possible to transfer the data additionally in encrypted form. Even if with the work defined in this thesis and its prototypes it was shown that a communication of IoT data can be executed secured and anonymous over Freenet. Nevertheless, some further work is available. The communication over Freenet is currently not very performant, here would be one of the biggest approaches for improvement. Additionally, the current prototype only supports communication between 2 parties at a time. Therefore, it is also possible to start here to enable multiparty usage. These are only two of a few approaches that could lead to an improvement of the prototypes. During the implementation of this work, I was able to implement much of what I learned in theory in a practice-oriented prototype. In addition, my theoretical knowledge helped me a lot in creating the architecture of the new communication path. All in all, I was able to put a lot of what I had learned into practice in this prototype.

Bibliography

[1] Matthew J. B. Robshaw. “One-Way Function”. In: *Encyclopedia of Cryptography and Security*. Ed. by Henk C. A. van Tilborg and Sushil Jajodia. Boston, MA: Springer US, 2011, pp. 887–888. ISBN: 978-1-4419-5906-5. DOI: `10.1007/978-1-4419-5906-5_467`. URL: `https://doi.org/10.1007/978-1-4419-5906-5_467`.

[2] Yannick Stebler. “Freenet as a broker for IoT Data”. In: *FCPv2 protocol* (2020), pp. 13–19.

[3] Prof Bill Buchanan. *NIST P256*. URL: `https://billatnapier.medium.com/barebones-p256-1700ff5a4`. [accessed 12.06.2021].

[4] Ian Clarke. *What is Freenet?* URL: `https://freenetproject.org/pages/about.html`. [accessed: 10.05.2021].

[5] John D. Cook. *A tale of two elliptic curves*. URL: `https://www.johndcook.com/blog/2018/08/21/a-tale-of-two-elliptic-curves/`. [accessed 12.06.2021].

[6] *FCP v2 Protocol*. URL: `https://github.com/freenet/wiki/wiki/FCPv2`. [accessed 16.05.2021].

[7] David Göthberg. *DH Public-key cryptography*. URL: `https://commons.wikimedia.org/w/index.php?curid=1028579`. [accessed: 06.06.2021].

[8] Hochschule Flensburg H. W. Lang. *Elliptische Kurven*. URL: `https://www.inf.hs-flensburg.de/lang/krypto/grund/elliptische-kurven.htm`. [accessed 11.06.2021].

[9] Rakel Haakegaard and Joanna Lang. “The Elliptic Curve Diffie-Hellman (ECDH)”. In: *Description of ECDH* (), p. 1.

[10] Ken MacKay. *micro-ecc*. URL: `https://github.com/kmackay/micro-ecc`. [accessed: 16.05.2021].

[11] Svetlin Nakov. *ECDH Key Exchange*. URL: `https://cryptobook.nakov.com/asymmetric-key-ciphers/ecdh-key-exchange`. [accessed: 16.05.2021].

[12] Wikipedia. *Diffie–Hellman key exchange*. URL: `https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange`. [accessed 06.06.2021].

[13] Wikipedia. *Elliptic curve*. URL: `https://en.wikipedia.org/wiki/Elliptic_curve`. [accessed 10.06.2021].

[14] Wikipedia. *Multiplicative group of integers modulo n*. URL: `https://en.wikipedia.org/wiki/Multiplicative_group_of_integers_modulo_n`. [accessed 06.06.2021].

[15] Wikipedia. *Primitive root modulo n*. URL: `https://en.wikipedia.org/wiki/Primitive_root_modulo_n`. [accessed 06.06.2021].

[16] Don Zagier. “Elliptische kurven: Fortschritte und Anwendungen”. In: *Elliptische Kurve* (), 1–3.

List of Figures