



DEGREE PROJECT IN INFORMATION TECHNOLOGY, SECOND LEVEL  
STOCKHOLM, SWEDEN 2018

# Network Authentication to the Physical World

JOAKIM SANDBERG

# Network Authentication to the Physical World

Joakim Sandberg

2018-01-14

Master's Thesis

Examiner

Gerald Q. Maguire Jr.

Academic adviser

Anders Västberg

Industrial adviser

Lars Larsson - Axis Communications – Lund,  
Sweden

## **Abstract**

Quick Response (QR) codes have been used for both non-authentication purposes and authentication & authorization of a user. The visual representation of a QR code requires a reader/decoder to convert the code to a readable resource for an application. This raises some concerns, such as: What kind of information and how much information can be stored in this representation? What kind of vulnerabilities are there when using this technology in some type of authentication?

The concrete problem addressed in this thesis is whether a mobile device displaying a QR code can be used as an authenticator for an Axis Communications A8105-E Network Video Door Station. To solve this problem the thesis investigates the use of QR codes in authentication & authorization based upon displaying a QR code on a mobile device, scanning this code via a camera, and then verifying the code using a validation server. Two important issues were what information to put into the QR code (given that the QR code is to be read by a camera) and where the decoding process should be carried out. This thesis also considers multiple types of authentication. Moreover, the system contains multiple components which themselves rely on secure communication and well-designed protocols to ensure security (against popular methods of attack) and stability.

A prototype of the proposed authentication process was evaluated using a testbed consisting of three door stations, an Android app, and a backend service for analysing QR codes and making an authentication & authorization decision. QR codes proved to be as secure as the current solutions, such as magnetic stripe or RFID access cards. Using QR codes together with the user's mobile device also offered additional functionality, such as easy management of the device registration/deregistration and compatibility with multifactor authentication. The conclusion is with the current door station products and the finalized design of the software; it is possible to have a secure and scalable system which is also cost-effective by eliminating the need of human verification as well as equipment required for access card systems.

### **Keywords**

Authentication, Authorization, Accounting, Security applications, QR-code, door station, Validation Server



## Sammanfattning

Quick-Response (QR) koder har använts både för icke autentiseringssyften men även för just autentisering av en användare. Den visuella representationen av en QR-kod behöver en läsare för att kunna omvandla koden till en läsbar resurs för en applikation. Detta leder till frågeställningarna: Vad för information och hur mycket kan man lagra i en QR-kod? Vilka sårbarheter finns det med användningen av denna teknologi inom autentisering?

Det konkreta problemet i detta examensarbete är huruvida en mobil enhet som visar en QR-kod kan användas med en Axis Communications A8105-E Network Video Door Station. För att lösa detta problem så undersöker detta arbete användningen av QR-koder inom autentisering baserat på att visa QR koden på den mobila enheten, skanna denna kod med en kamera och validera denna kod med en valideringsserver. Två viktiga frågor var vilken information som skulle lagras i QR-koden samt vart avläsningen av en QR-kod tog plats. Detta arbete undersöker också olika typer av multifaktor autentisering. Systemet består vidare av flera komponenter som förlitar sig på säker kommunikation och väl designade protokoll som ger säkerhet (mot de mest populära nätverksattackerna) och stabilitet.

Den tänkta autentiseringsprocessens prototyp evaluerades i en testmiljö bestående av tre Door Station, en Android applikation och en backend service för att analysera QR-koder samt hantera autentiserings och behörighetslogik. QR-koder bevisades vara lika säkert som befintliga lösningar som till exempel kort accesskort med magnetremsa och RFID chip. Användning av QR-koder med de mobila enheterna gör dessutom att hantering av registrering/avregistrering av enheterna kan ske på ett enkelt sätt samt även integreras med multifaktor autentisering. Slutsatsen är att med de befintliga Door Station enheterna och den slutgiltiga designen av mjukvaran ger det säkert och skalbart system som dessutom är kostnadseffektivt genom att behovet av en mänsklig verifiering samt de fysiska komponenterna av befintliga accesskortsystem, inte längre finns.

### Nyckelord

Authentication, Authorization, Accounting, Säkerhetsapplikationer, QR-kod, IoT, Door Station, Valideringsserver



## Acknowledgments

First, I would like to thank Axis Communications for giving me the opportunity to write my thesis there. Many thanks to the industrial adviser Lars Larsson at the company who has been a great support during this time.

I would also like to thank Professor Gerald Q. Maguire Jr. at KTH as the examiner and additional support for this time as well.

Stockholm, January 2018  
Joakim Sandberg





## Table of contents

<b>Abstract</b> .....	<b>i</b>
<b>Keywords</b> .....	<b>i</b>
<b>Sammanfattning</b> .....	<b>iii</b>
<b>Nyckelord</b> .....	<b>iii</b>
<b>Acknowledgments</b> .....	<b>v</b>
<b>Table of contents</b> .....	<b>vii</b>
<b>List of Figures</b> .....	<b>xi</b>
<b>List of Tables</b> .....	<b>xiii</b>
<b>List of acronyms and abbreviations</b> .....	<b>xv</b>
<b>1 Introduction</b> .....	<b>1</b>
<b>1.1 Motivation</b> .....	<b>1</b>
<b>1.2 Background</b> .....	<b>2</b>
<b>1.3 Problem definition</b> .....	<b>3</b>
<b>1.4 Purpose</b> .....	<b>4</b>
<b>1.5 Goals</b> .....	<b>4</b>
<b>1.6 Research Methodology</b> .....	<b>5</b>
<b>1.7 Delimitations</b> .....	<b>5</b>
<b>1.8 Structure of the thesis</b> .....	<b>6</b>
<b>2 Background</b> .....	<b>7</b>
<b>2.1 Network attacks</b> .....	<b>7</b>
2.1.1 Replay attack .....	7
2.1.2 Brute force attack.....	7
2.1.3 Chosen plaintext attack.....	7
2.1.4 Chosen ciphertext attack .....	7
2.1.5 Timing attack .....	8
<b>2.2 Authentication</b> .....	<b>8</b>
<b>2.3 Authorization</b> .....	<b>8</b>
<b>2.4 Accounting</b> .....	<b>9</b>
<b>2.5 Nonce</b> .....	<b>9</b>
<b>2.6 QR-codes</b> .....	<b>9</b>
<b>2.7 Cryptography</b> .....	<b>12</b>
2.7.1 Asymmetric key encryption .....	12
2.7.2 Symmetric key encryption .....	12
<b>2.8 Certificates</b> .....	<b>15</b>
<b>2.9 SSL/TLS</b> .....	<b>15</b>
<b>2.10 Related work</b> .....	<b>17</b>
2.10.1 Transportation.....	17
2.10.2 Banking.....	19

2.10.3	QR-based work .....	19
<b>2.11</b>	<b>Impact.....</b>	<b>21</b>
<b>2.12</b>	<b>Summary .....</b>	<b>22</b>
<b>3</b>	<b>Methodology .....</b>	<b>25</b>
<b>3.1</b>	<b>Research Process .....</b>	<b>25</b>
<b>3.2</b>	<b>Research Paradigm .....</b>	<b>25</b>
<b>3.3</b>	<b>Data Collection .....</b>	<b>27</b>
3.3.1	TLS .....	27
3.3.2	QR code images .....	27
<b>3.4</b>	<b>Experimental design/Conducted Measurements.....</b>	<b>28</b>
3.4.1	QR-code experiments .....	28
3.4.2	Encryption experiments .....	32
3.4.3	Test environment .....	32
3.4.4	Hardware/Software to be used .....	33
<b>3.5</b>	<b>Assessing reliability and validity of the data collected.....</b>	<b>35</b>
3.5.1	Reliability .....	35
3.5.2	Validity .....	35
<b>3.6</b>	<b>Data Analysis .....</b>	<b>36</b>
3.6.1	Data Analysis Technique .....	36
3.6.2	Software tools .....	37
<b>3.7</b>	<b>Evaluation framework .....</b>	<b>37</b>
<b>4</b>	<b>Developing a proof-of-concept validation service .....</b>	<b>39</b>
<b>4.1</b>	<b>System components .....</b>	<b>39</b>
<b>4.2</b>	<b>Optimizing the QR decoder service .....</b>	<b>40</b>
4.2.1	QR decoder in door station .....	40
4.2.2	VM as a service .....	40
4.2.3	Spreading out services .....	41
<b>4.3</b>	<b>Database .....</b>	<b>41</b>
<b>4.4</b>	<b>Implementing security .....</b>	<b>42</b>
4.4.1	Lower level security .....	42
4.4.2	Application level security.....	42
<b>4.5</b>	<b>Complete design.....</b>	<b>46</b>
4.5.1	No authentication example .....	46
4.5.2	Secure and optimized validation procedure .....	47
<b>4.6</b>	<b>Bootstrapping service.....</b>	<b>56</b>
<b>5</b>	<b>Results and Analysis .....</b>	<b>59</b>
<b>5.1</b>	<b>Metrics.....</b>	<b>59</b>
5.1.1	QR code.....	59
5.1.2	Environment.....	60
5.1.3	Encryption.....	60
<b>5.2</b>	<b>Major results .....</b>	<b>61</b>
5.2.1	QR code results .....	61
5.2.2	Environment.....	65

5.2.3	Encryption results .....	70
<b>5.3</b>	<b>Reliability Analysis.....</b>	<b>78</b>
5.3.1	Light conditions of QR code.....	78
5.3.2	Detectable area .....	79
5.3.3	Encryption effects .....	79
<b>5.4</b>	<b>Validity Analysis .....</b>	<b>80</b>
<b>5.5</b>	<b>Discussion .....</b>	<b>80</b>
5.5.1	Preparation of the optimal system layout.....	81
5.5.2	Improved solution.....	84
5.5.3	Evaluation and analysis of the optimized system.....	87
5.5.4	Comparison with MiFare access card.....	88
<b>6</b>	<b>Conclusions and Future work .....</b>	<b>93</b>
6.1	Conclusions .....	93
6.2	Limitations .....	94
6.3	Future work .....	94
6.4	Reflections .....	96
	<b>References .....</b>	<b>97</b>
	<b>Appendix A: code snippets .....</b>	<b>101</b>
	<b>Appendix B: Equations .....</b>	<b>103</b>



## List of Figures

Figure 2.1:	Captured packets to and from a server with HTTP GET .....	9
Figure 2.2:	Basic authorization method sends the credentials in plain text.....	9
Figure 2.3:	Visual representation of a QR code .....	10
Figure 2.4:	TLS handshake without and with client authentication .....	16
Figure 3.1	The three key definitions used in the research and the related considerations.....	26
Figure 3.2:	Spherical coordinate system.....	29
Figure 3.3:	Setup of experiment to measure readability of an angled device .....	30
Figure 3.4:	Network setup of test environment.....	33
Figure 4.1:	Insecure and secure channels in system layout .....	43
Figure 4.2:	Symmetric keys encryption and decryption.....	44
Figure 4.3:	Asymmetric keys encryption and decryption.....	45
Figure 4.4:	Signing and verifying procedure .....	45
Figure 4.5:	System component overview .....	47
Figure 4.6:	Registration procedure .....	48
Figure 4.7:	User authentication to the service.....	49
Figure 4.8:	Detailed flow of challenge between mobile device and validation service .....	50
Figure 4.9:	Scanning and QR decoding procedure .....	51
Figure 4.10:	Validating content of a QR code .....	52
Figure 4.11:	QR code generation in mobile device.....	53
Figure 4.12:	QR code validation in server, by the hash values.....	54
Figure 4.13:	Mobile connection management with validation parameters .....	55
Figure 4.14:	Time synchronization between mobile device and server .....	56
Figure 5.1:	Readability of QR codes with different sizes in pixels, during four different light conditions. ....	62
Figure 5.2:	Readability of QR codes with different stored content lengths, during four different light conditions.....	63
Figure 5.3:	Readability of QR code on a tilted device. The plot on the left shows when device is tilted forward and backward along the doorstation while the right plot show tilting left and right.....	64
Figure 5.4:	Rotation along the Z-axis of the door station, this yielded a successful reading of the code .....	64
Figure 5.5:	Detectable and readable area around the door station, based on coarse measured values .....	65
Figure 5.6:	RSA encryption which shows correlation of maximum input data with different key sizes and padding options.....	71
Figure 5.7:	Correlation between plaintext input length and encrypted output length for RSA encryption with 2048-bit key. ....	72
Figure 5.8:	RSA encryption time with best fit lines with regards to whole data set. Note equation times are in microseconds. ....	73
Figure 5.9:	RSA encryption time with best fit lines of subset of data. Input length up to 1 kilobytes. Note equation times are in microseconds. ....	73

Figure 5.10:	RSA decryption time with best fit lines for whole data set. Note equation times are in microseconds. ....	74
Figure 5.11:	RSA decryption time with best fit lines, for data subset up to 1 kilobytes. Note equation times are in microseconds. ....	74
Figure 5.12:	Symmetric encryption times for three different algorithms with best fit lines. Note equation times are in microseconds. ....	75
Figure 5.13:	AES encryption times for two modes with best fit lines. Note equation times are in microseconds. ....	76
Figure 5.14:	AES encryption times for different key sizes, with best fit lines. Note equation times are in microseconds. ....	76
Figure 5.15:	OTP encryption during four different attempts. One of these attempts (black points and line) shows a large deviation in time. Note equation times are in microseconds. ....	77
Figure 5.16:	Decryption times of both symmetric and asymmetric algorithms, with best fit lines. Note that equation times are in microseconds. ....	78
Figure 5.17:	Optimized system layout .....	86
Figure 5.18:	MiFare DESFire EV1 authentication protocol .....	90
Figure 6.1:	Sequence diagram of a situation where an eavesdropper could potentially gain access .....	95

## List of Tables

Table 2-1:	A QR code's storage capabilities.....	10
Table 2-2:	10-character QR code for different error correction levels.....	11
Table 2-3:	50-character QR code for different error correction levels.....	11
Table 2-4:	100-character QR code for different error correction levels.....	11
Table 2-5:	Summary of related work, techniques, and protocols used in different applications.....	23
Table 3-1:	Hardware and software used.....	33
Table 3-2:	Evaluation framework table .....	37
Table 4-1:	System components .....	39
Table 4-2:	Configuration of database tables .....	42
Table 5-1:	Image snapshots from various experiments, taken from door station viewpoint .....	66
Table 5-2:	Maximum data block lengths in bytes of what can be encrypted with the chosen padding and the used key size .....	70
Table 5-3:	Resulted encrypted content lengths in bytes, for chosen padding and key sizes.....	71
Table 5-4:	Previous designs of QR code format.....	83
Table 5-5:	Time taken from when a QR code is scanned at door station to unlock signal being received.....	87
Table 5-6:	Time taken from when a QR code is scanned at door station to the remain locked signal being received .....	88





## List of acronyms and abbreviations

AAA	Authentication, Authorization and Accounting
AES	Advanced Encryption Standard
CA	Certificate Authority
CBC	Cipher Block Chaining
CPU	Central Processing Unit
CSR	Certificate Signing Request
DDOS	Distributed Denial of Service
DES	Data Encryption Standard
3DES	Triple Data Encryption Standard
ECB	Electronic Code Book
GUID	Global Unique Identifier
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
MAC	Message Authentication Code
NONCE	Number Only Used Once
NTP	Network Time Protocol
OAEP	Optimal Asymmetric Encryption Padding
OTP	One Time Pad
PIN	Personal Identification
PKCS	Public Key Cryptography Standard
PKI	Public Key Infrastructure
RFID	Radio-frequency Identification
QR	Quick Response
SHA	Secure Hash Algorithm
SL	StorStockholms Lokaltrafik
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
USB	Universal Serial Bus
UL	Upplands Lokaltrafik
VPN	Virtual Private Network



# 1 Introduction

This chapter describes the specific problem that this thesis addresses, the context of the problem, the goals of this thesis project, and outlines the structure of the thesis.

## 1.1 Motivation

Systems where people are verified and validated as legitimate users, have been used in our society for a long time; we refer to these systems as *validation services*. These systems are adapted to their operating domain and can be both hardware based and software based. An example of a hardware based system is when a person needs to enter a building by using an access card together with a personal identification (PIN) code (here after referred to as a PIN code) entered via a physical keypad, this could also be referred as *physical authentication*. A software based verification system can be a login session to an online service which requires a username and a password.

The process of people validating their own identify is known as *authentication*. In this process they must provide sufficient information about themselves to prove to the service that they are who they say they are. This information is often described as the following:

- *Something you know* is most commonly a password or PIN code that people remembers and ought to not be physically written down since it increases the risk that others can obtain access to this information.
- *Something you have* is generally provided by devices that the user uses when they want the service. An advantage of this type of information is that user does not need to remember it and this information can be dynamically changing (as long as both the device, often called an authenticator, and the authentication service agree upon the information).
- *Something you are* is the concept of user providing information that can be uniquely found on the user themselves, such as biometrics (e.g. fingerprints).

All three types of information are not required in order to prove who you are to the validation services, but the more that is required, the higher the security is it considered to be. For instance, if an ATM was based on only the bank card (something you have) and no PIN code, then anyone that access the card can make money withdrawals as long as there was money on that card. Adding the required PIN code (something you know) makes that problem go away and thus requires the card user to know the correct PIN code.

In many cases, authentication is insufficient for a person to gain access to the resource. The access to the resource might be limited to subset of people, which means the validation system must determine *whether* this user should have access. This process is known as *authorization*, and could be used together with the authentication method. For instance, an employee at company must have both a valid access card and a valid PIN code. He or she can enter different areas in the company's building but at some locations, the authorization service may require that a higher level of validation is required to continue, meaning that although the

employee previously authenticate themselves - the authentication did not provide sufficient information to enable the system to authorize access, hence the user had to perform some additional validation step. For example, at this access control point the user might need to present their index finger to a finger print reader. Moreover, this specific user might not be authorized to access the resource, thus despite providing additional validation they would be denied access to the resource.

Both authentication and authorization are processes that require cross-references to work with, some sort of stored information that are accessed by these processes. In some systems, a database of valid entities is stored in the stationary units that are used as validation points (e.g. card readers at doors, ATMs, etc.), but this storage could also be implemented in a *backend service* somewhere else, hence the communication between the stationary units and the backend services must be protected.

*Admission control* ensure that the authentication and authorization must be successful before the user can proceed to access a resource. If either process fails for some reason, then it is up to the implementation to take the necessary actions to restrict access to the resource or to execute a failure protocol of some kind (for example, raise an alarm that results in a security guard checking the user).

## 1.2 Background

Validation services are concerned with authentication and authorization, hence they focus on security and thus need to be consistent, have a minor risk of security breaches, and keep up with the latest technology. The security aspect among other aspects such as time must therefore be prioritized when developing validation services.

For example, a banking application on a mobile device provides customers with an overview of their account and the ability to carry out transfers from their accounts to other accounts. In this case, security is a crucial factor in the user's acceptance and usage of this application. The user might not care whether a transaction takes two seconds or five seconds, as long as the transaction is secured such that nobody else sees the account information and that no one else can initiate such a transfer or repeat this transfer (therefore, the transaction must happen once and only once).

Applications based on authentication such as login, resource access, secure communications, etc. use cryptographic algorithms to ensure a private conversation between a client and a server, ensure that the client and server trust each other's identities, that the transaction cannot be repudiated\*, and that the information sent between them is complete and valid.

Environments where authentication is most likely required include entry to companies and even home residences. There are different methods to deal with the authentication process, such as having the building door locked at all times and people without a key must contact the persons they have come to visit in order to gain access to the building. There are also keyless solutions where the door is locked, but can be opened via some other means, such as RFID tag or biometrics. Another solution is having a *door station* which is basically a wall-mounted controller next to the door. These door stations can have different specifications,

---

\* Server cannot deny any transaction with client afterwards and vice versa.

but all have the same purpose: to provide admission control for entrance to the building. Additionally, the door station may incorporate an intercom so that visitors can communicate with persons inside the building.

Various approaches have been made to develop door stations, such as Control4 [1] who made a door station consisting of a camera and keypad. This approach is very flexible since it covers all the use cases of entering a building: visitors can ring the doorbell button, an authorized person can enter a code to get inside (assuming that this feature is enabled), and people inside can see as well as speak with a visitor without having to physically go to the door. While this product might be useful to companies with multiple employees each of whom can have their own code, it could also be used at people's residences. Another product, called iVision+ [2], is more suitable for residences. This product was developed by Optex and is a very simple device which is more or less plug-and-play. The product consists of two devices: a unit mounted outside and a corresponding inside unit that the external device communicates with.

The door stations that will be used in this thesis project was developed by Axis Communications. These products are wall-mounted units that are based on network communication, thus this door station can communicate with services inside the building [3]. Because it is a network-based authentication unit, it is especially suitable when designing and deploying an application where physical authentication is common.



**A8105-E**

Image reference [3]



**A8004-VE**

Image reference [4]

*(Images used with permission from Axis Communications.)*

### 1.3 Problem definition

Validation services basically compare two values, one of which is provided as input to the system and one that has previously been entered into the system. The new input is compared with the previously recorded data for deciding to grant access to some resource, for instance a website, a bank account, an area in a building, etc. The way we supply each of these values to the system varies as a function of different parameters, such as security, environment, and user interface.

When designing these systems, we must consider the system's goals very carefully. For example, in a validation process for a bank transaction; security is very important and thus authentication might require specific validation checks before the user is authenticated and the transaction is authorized. This will take time and users are often aware of this delay. In another situation, time might be more critical, hence the process should be completed in a short time. While in a use case where very high security is required, how long the validation may not matter (in practice).

A common situation today is that the requirements for a user to authenticate himself/herself to enter a building, are based on older methods, for example a security guard needs to crosscheck the user's validity of entity against a stored entity (a list of users authorized to enter the facility) and this process might take considerable time. Conversely, there might be a door station installed at the entrance door which can initiate a video and/or voice session between an arriving visitor and the recipient, allowing the visitor to authenticate themselves to the recipient who can make a decision to admit the visitor or not. In these situations, it takes a long time for a validation session to complete. Using existing devices and correctly optimizing their operation, a new and faster way of authenticating/authorizing users to enter a building or to access some other resource can be designed. In the example of access to a building, an efficient computer assisted validation process for a visitor *before* they can enter the building could minimize the time required by people hired to validate visitors, thus saving both time and money.

## 1.4 Purpose

This thesis should introduce readers to a new innovative design for a validation system that is easy to implement, scales, and ensures high security. Additionally, by implementing this functionality using existing devices and software, it will also be cost effective, thus increasing its potential in the market. To complement this project, some background in this area will be described along with some related work and facts that are required background knowledge for this project.

## 1.5 Goals

The main goal of this thesis project is to design a validation system based on Quick Response (QR) codes and mobile devices. A QR code reader is implemented as part of a door station by exploiting the camera's ability to capture the QR code and some additional software that can decode the QR code to use this as input to a validation service. This high-level goal has been divided into the following sub-goals:

1. Design a module for QR decoder that can rapidly decode the image of the QR code taken by a camera.
2. Design a validation service which can:
  - Handle encrypted and signed content and
  - Offers good scalability, thus enabling it to simultaneously handle multiple external devices.
3. Ensure high security between each external device and the validation service, hence lowering the risk for malicious usage of the images from the camera. This

- will be done using existing security protocols for communication between a client and server.
4. Differentiate between authorized and unauthorized users in a straightforward way, while allowing multiple mobile devices to be used by a single user. This will require some means to distinguish between these different devices.
  5. Provide an administrative interface to ensure easy accounting, user management, and configuration of the system's properties.

## 1.6 Research Methodology

This project will utilize a physical testbed and create designs that will serve as a pre-study for future improvements and as the base for a design of a functional product. Because this project is the first in the chain of steps towards a product, it will consist of different tests and trials to rule out ideas that are unsuitable. The most suitable methodology for this project is design science; hence we planned and conducted a set of tests, along with additional tests that were identified along the way, in order to iterate through different system designs and implementations. Each prototype was evaluated from several aspects, such as security, usefulness, speed, complexity, scalability, cost, and ethical points of views. Through this iterative process the project refined a design and implementation to realize a prototype that is suitable for the task and fulfil the project's goals.

In contrast to other methods, such as researching existing designs and looking for improvements, having only one design and analysing it very thoroughly from various aspects, or starting with a simple theoretical method; the chosen method lays the groundwork for a future product. Another alternative method would be to analyse the desired product from a marketing point of view, then conduct surveys and from the results of these surveys analyse potential alternatives. These other methods are more suitable when the contemplated system is closer to a functional product. However, in the current state of the process towards a product, the functional product phase is still far in the future.

## 1.7 Delimitations

Since the results of this project will not to be released as a product, but rather will lay the groundwork for other researchers, there are some aspects that will not be covered. A concept that is used today is *load balancers* which spreads load over all copies of the application on the network, in order to improve performance. However, using load balancing will not be covered in this thesis due to the simplicity of components and resources used.

Additionally, the hardware used here as physical authentication are the door stations from Axis Communications only, which means the system application is designed from many theoretical perspectives that is can be used with other types of hardware, but that is not tested here.

Software based system design will be restricted to not modify or use any hardware modules other than the intended products, so any solutions that requires additional hardware are out of the scope of this thesis, although they might be discussed.

Generally, the test environment is restricted to a lab environment which means real network will not be used, rather a limited local network.

## 1.8 Structure of the thesis

The structure of the thesis is as follows. Chapter 2 presents relevant background information regarding secure channels and its components. It also presents previous work regarding usage of QR codes in mobile devices as well as other physical media used for authentication purposes.

Chapter 3 presents the research methodology and method in detail. Why and how the selected methodology and method are used in this project and the underlying components.

Chapter 4 describes the tests and groundwork laid during this project. It is divided into subsections describing the development of different areas of the complete (prototype) authentication system.

The results of the tests described in Chapter 4 are stated and analysed in Chapter 5 together with their theoretical base.

Chapter 6 completes the thesis by presenting the conclusion of the project in detail and states some general points of view regarding authentication systems. Moreover, the chapter states how this project can be generalized and suggests future work.



## 2 Background

This chapter provides basic background information about technologies developed and used in internet security such as nonces, cryptography, Secure Sockets Layer (SSL)/Transport Layer Security (TLS), authentication, authorization, accounting, and certificates. Before these sections, some common network attacks will be described briefly. Additionally, this chapter briefly describes QR-codes, related work based on QR codes as well general physical-based authentication applications. Also, this chapter describes what has not (yet) been done in the area, what kinds of authentication are missing and should be developed, etc.

### 2.1 Network attacks

Here are described some of the most common network attacks.

#### 2.1.1 Replay attack

Replay attack is the kind of attack of which a resource between a client and a server is being recorded by an attacker with the purpose of reusing this resource at a later point in time. This attack will succeed if the system is not able to handle multiple copies of the same resource, meaning sending the exact same resource to the server will grant the same access as it did the first time.

#### 2.1.2 Brute force attack

Brute force attacks are commonly known as a method of trial-and-error until the system grants access. The resources (usernames, passwords, nonces etc) are basically guessed by an attacker multiple times and works where systems are able to handle multiple requests in short time. Naturally, there are intelligent systems of which works against this sort of attack by having a restricted number of attempts before the system is locked for a short while.

#### 2.1.3 Chosen plaintext attack

Chosen plaintext attack has the purpose of achieving the key used for the encryption which is used in the system. If the attacker knows the encryption algorithm used in the system, he/she can then create a set of plaintexts and encrypt these which then results in the attacker having a pair of {plaintext, ciphertext} which then can be used to trying to figure out the key. This is done by simply comparing the own created ciphertexts, with the system's outputted ciphertexts, where a match then results in the attacker knowing which plaintext gave the correct ciphertext.

#### 2.1.4 Chosen ciphertext attack

Chosen ciphertext attack has same purpose as chosen plaintext attack, achieving the key used in the encryption scheme. This sort of attacks works quite similar but with a slight difference. Instead of the attacker sending a set of plaintexts, he/she sends a set of ciphertexts for the system to decrypt. The decrypted result is then

achieved by the attacker by the same method he/she used do achieve the ciphertexts sent on the channel. Now, similar to plaintext attack, the attacker can try to determine the correct key used for the decryption, by analysing the pair {ciphertext, plaintext}.

### 2.1.5 Timing attack

Timing attack is the sort of attack of which an attacker is able to learn the encryption key without actually having to break the encryption itself. The attacker simply sends plaintexts of different lengths to the system and observes how much time it takes before the encryption operation is completed. By having different encryption times on different plaintexts, the attacker can carefully deduce what parameters is used in the encryption and in long term, be able to calculate the key itself used. RSA is an encryption algorithm which is a victim of this sort of attack [5].

## 2.2 Authentication

When designing systems which dependent on users with varied limitations on use along with different purposes and goals, there are three distinct aspects that can be considered: authentication, authorization, and accounting.

Authentication is the concept of having unique user credentials and the ability to authenticate a user in order for the user to gain access to resources. The system waits for a user's credentials to be entered and then compares these with stored credentials. In buildings where large companies are located, there is often a system to authenticate users before they can access the building. Smartcards along with PIN codes are widely used and the system makes it easy to authenticate users (or at least their cards).

Adding additional authentication methods together, such as the use of PIN codes together with the cards, is known as *2-factor authentication*. In this case both the authentication methods used must be successful for the validation service to authenticate the user.

## 2.3 Authorization

Authorization is the process of determining what resources an authenticated user is allowed to access. A system administrator might have authorization to manage every feature in a system, while another user might only have access to part of the system. Authorization with regarding to a door station depends upon parameters such as time and/or place and generally distinguishes a guest from an employee.

As an example, if HTTP is used (*without* SSL/TLS) with a site whose administrators have "secured" the site with the basic HTTP authorization method then there is little security. Figure 2.1 shows a captured HTTP request and response. Figure 2.2 shows the corresponding authorization method information which was used in the authentication process. Since in this example the credentials entered were incorrect, the server sends an unauthorized response. However, the major issue is that credentials in the captured packets are in plain text so *anyone* with access to the network could record this and subsequently use it in a replay attack.

No.	Time	Source	Destination	Protocol	Length	Info
490	44.478648	192.168.0.100	[REDACTED]	HTTP	635	GET / HTTP/1.1
491	44.499279	[REDACTED]	192.168.0.100	HTTP	795	HTTP/1.1 401 Unauthorized (text/html)

**Figure 2.1:** Captured packets to and from a server with HTTP GET

```

Authorization: Basic dGVzdDoxMjM=\r\n
Credentials: test:123
\r\n

```

**Figure 2.2:** Basic authorization method sends the credentials in plain text

## 2.4 Accounting

Accounting records who, what, and when specific resources were accessed. Additionally, accounting can be used to collect data about attempted usage of a resource. For instance, logging how many attempts a user has tried to gain access to a specific resource which this user does not have authorization to access, can be used to determine whether this user is a betrayer or might simply have forgot the password required for that resource.

During normal use, accounting can be used to form baselines for every (valid) user's activity. When an abnormal use is detected, actions could be taken to inform administrators. In this way, attacks such as brute-force attacks can be detected. If all activity is recording by the accounting process, then potentially an attack can be tracked to the source.

## 2.5 Nonce

Nonces are widely used to prevent replay attacks and are based on timestamps or simply be random numbers. They are a very simple component that has large impact on encryption and the protocols needed to prevent the replay attack [6]. The implementations of nonces and the validation of nonces can differ in systems, but if timestamps are used to create nonces, it is easy to validate a nonce, simply check that the nonce's timestamp has a just slightly earlier time than the current time. In order to do this, it is necessary to have enabled time synchronization of the communicating devices (e.g. using an NTP server to provide common source of timing information). If just an accumulative counter or a random number was used as a nonce, the assurance that the value is only used once requires the validation service to crosscheck with some database of used nonces.

## 2.6 QR-codes

A Quick Response Code is a type of 2D barcode, invented by Denso Wave Inc. [7]. These codes provide a visual representation of data using black and white squares forming a large square (as shown in Figure 2.3).



**Figure 2.3:** Visual representation of a QR code

Because a QR code provides a visual representation of a code, it has some limitations, such as a bounded maximum storage. The maximum data capacity [8] depends on the type of data, as shown in Table 2-1.

**Table 2-1:** A QR code's storage capabilities.

Type	Maximum Data (characters)
<b>Numeric</b>	7,089
<b>Alphanumeric</b>	4,296
<b>Binary, 8bits</b>	2,953

The general structure of the QR code consists of a version code, format information, timing patterns, data, and alignment as well as detection points [8]. These fields enable codes to be readable even if some errors occur in the encoding or in the reading, because a QR code is designed to allow up to 30% error correction [9]. This error correction is done with *Reed-Solomon error correction codes*, a type of cyclic-code [10]. Calculations in this fashion are made by polynomial divisions. If the remainder of a polynomial representation of the QR code's format, divided by a generator polynomial is non-zero, then an error has occurred, and then the encoded error-correction bytes can be used to recover the data. This process is explained in detail in [11]. Table 2-2 to Table 2-4 show how QR code generation will respond to the input with regards to the error correction level. These tables show codes with three different input lengths and with different amounts of error correction data.

Since the image follows the QR code format with three corner squares and additional squares in the middle, every black and white dot of the rest of the code will have a size that depends on how much content is stored. This size is compared with the code as a whole; hence there is a certain requirement for image resolution depending upon the complexity of the code. It is very important when using these codes that the reader has enough resolution to resolve a given QR code. For instance, if an image contains a complex code, such as that shown in Table 2-4, the reader needs an image with very high resolution to be able to detect every single black and white dot in the QR code.

Table 2-2: 10-character QR code for different error correction levels





<b>QR code with 10 characters</b>			
<b>Error correction 7%</b>	<b>Error correction 15%</b>	<b>Error correction 25%</b>	<b>Error correction 30%</b>
			

Table 2-3: 50-character QR code for different error correction levels









<b>QR code with 50 characters</b>			
<b>Error correction 7%</b>	<b>Error correction 15%</b>	<b>Error correction 25%</b>	<b>Error correction 30%</b>
			

Table 2-4: 100-character QR code for different error correction levels

<b>QR code with 100 characters</b>			
<b>Error correction 7%</b>	<b>Error correction 15%</b>	<b>Error correction 25%</b>	<b>Error correction 30%</b>
			

## 2.7 Cryptography

Cryptography is a very old concept of manipulating information from the original state to a completely different state by applying different methods. It is widely used within information technology, especially within the area of security protocols. There are two main types of cryptography used by internet security protocols today: asymmetric key encryption (used in public key infrastructure, PKI) and symmetric key encryption.

### 2.7.1 Asymmetric key encryption

Asymmetric key encryption has the property of utilizing one public key and one private key. These keys are generated in pairs; hence they have a relationship to each other. One key can be used to encrypt the original data while the other key is used to decrypt the encrypted data back to the original data. The private keys are kept secret, while the public keys can be open to anyone to use. Asymmetric encryption is very powerful since the sender can encrypt some data with the receiver's public key and only the receiver (which ought to be the owner and sole possessor of the corresponding private key) can decrypt this data. A public key infrastructure (PKI) provides a means for anyone to get public keys. Additionally, PKI can be used in conjunction with a certificate authority (CA) to establish who the owner of a given public key is (see Section 2.8).

Typical key sizes for asymmetric key encryption, for instance for RSA, are 1024, 2048, and 4096 bits. This high number of bits is because the calculation of keys is based on a mathematical approach with prime numbers. Basically, two prime numbers form a modulus which is used to derive the key. As the key size increases, this computation becomes more expensive which is why asymmetric keys cannot be used when frequently encrypting messages.

### 2.7.2 Symmetric key encryption

In symmetric key encryption, one key is used both to encrypt and decrypt data. Symmetric keys are widely used because of their high performance and security. The symmetric key encryption computations typically utilize key-sizes of 128-256 bits, compared to the asymmetric key sizes of roughly ten times more. Symmetric encryption does not rely on any prime number computation but instead about different complex mathematical procedures of the data which are based on the key. For example, the operations could be shifting each letter in the plaintext a certain number of times before adding a number and then shifting again some number of times.

When two parties wish to communicate using symmetric key encryption to secure their communication: How do the two parties agree on the same key? In order for these two parties to establish a secure and encrypted communication session, they must either have a pre-shared key or they would have to communicate over an insecure channel to decide upon a symmetric key (if nothing more is configured for deciding and sharing the keys). However, sharing the secret-key via an insecure channel would enable everyone else listening to the channel to be able to see this secret key. The result is that symmetric key encryption cannot be used on its own but rather is used in combination with asymmetric key encryption, as described in Section 2.9.

Types of symmetric key encryption algorithms are AES, DES, 3DES, and OTP, described in the following subsections.

#### 2.7.2.1 AES

Advanced Encryption Standard (AES) [12], is also known as Rijndael and was developed in 2001 by Vincent Rijmen and Joan Daemen. It is based on key sizes of 128, 192, or 256 bits and block sizes of 128 bits. AES works by so called substitution-permutation method of a matrix form of the data. The substitution part is defined by a series of different operations on the matrix such as *sub-bytes* where basically each byte in the matrix is replaced by another byte according to a pre-calculated table called S-box. Other operations on the matrix are by switching elements in a row in a certain way as well as columns. All these operations on the matrices are then repeated a certain amount of times (10 for 128-bit key, 12 for 192-bit key and 14 for 256-bit key) which then results in a complete different matrix, and generally an indeterministic ciphertext. This algorithm is very fast and secure but requires the input data to be a multiple of the block size which means extra care needs to be taken in the implementation.

#### 2.7.2.2 DES

DES, Data Encryption Standard [13], was invented by IBM in 1977. It uses only 56-bit keys, however in Java, it must be supplied with 64-bit key (8 bytes) since there is one parity bit per byte which is removed, resulting in a 56-bit key. It works by also having a permutation of 16 rounds of an operation, which in this case is a so-called Feistel-round.

The feistel round is also based on substitution which here is done by first having the right half of the input data shifted to the left side of the output. The left half of the input is XORed with the right half of input going through a mangler function. That function is based on an S-box of mapping one input symbol to another one, resulting in a complete random output. Since DES is considered to be vulnerable to brute force attacks, it has the security improvement of the following algorithm 3DES.

#### 2.7.2.3 3DES – a.k.a DESede

Triple DES – is precisely what the name refers to, three times normal DES which is why the key size of it is also three times of normal DES ( $3 \times 56 = 168$ ). Generally, there is not one 168-bit key but rather three different 56-bit keys used in the order *encryption-decryption-encryption*. Optionally, the encryption key can be reused, resulting in a total of  $2 \times 56 = 112$  bit key size.

#### 2.7.2.4 OTP

A One Time Pad (OTP) is a stream cipher, unlike AES/DES/DESede which are all block ciphers. This means that encryption and decryption happen in a stream rather than dividing the input into blocks and encrypt each block. In a stream cipher each bit is XORed with a key bit resulting in a complex ciphertext from a simple operation.

Advantages	Encryption and decryption is a quick operation since it is based on bitwise XOR. If the key is kept secure and implemented correctly (only used once and true randomness), OTP is a very high secure encryption technique.
Disadvantages	Key size must be (at least) length of plaintext. This is a problem if there is large data to encrypt. Another issue with OTP is if in fact the keys are used more than once, it will then increase the potential risk of keys being revealed.

$$Plaintext1 \oplus Key = Cipher1 \rightarrow Plaintext1 \oplus Cipher1 = Key$$

$$Plaintext2 \oplus Key = Cipher2 \rightarrow$$

$$Cipher1 \oplus Cipher2 = Plaintext1 \oplus Plaintext2$$

The result of Plaintext1 XORed with Plaintext2 might not be that revealing but these relationships make it possible to use brute force attacks on plaintexts to eventually reveal the key used (i.e. *known plaintext attack*). For this reason the key must never be re-used, hence the key must be as long as the input string – thus there will never be key re-use and hence there is no means to break the encryption – other than having a copy of the key.

#### 2.7.2.5 Modes

There are two modes that could be used on with the encryption, CBC and ECB. The modes that the algorithms (AES/DES/3DES) use have pros and cons themselves which can be used for determining the better for the application usage.

Cipher block chaining (*CBC*) works by having an initial vector (IV) that is XORed with the first block where the resulted cipher block is used as vector to XOR the next block and so on, hence the word “chaining”. The advantages and disadvantages are:

Advantages	Due to the use of chaining, every block is linked to another, making it harder for an attacker to manipulate an individual block.
Disadvantages	Requires the IV to be sent along the ciphertext because decryption needs the same vector. Cannot be parallelized since every block is dependent on the previous block (except initial block which needs IV).

Electronic Code Book (*ECB*) is a very simple encrypting mode where plaintext blocks simply are encrypted only by the key and no additional factor such as initial vector. The advantages and disadvantages are:

Advantages	Can be parallelized since block encryption is independent of other blocks.
Disadvantages	Same plaintext block results in same cipher block since no IV is used



### 2.7.2.6 Paddings

Using padding with encryption means appending a random content to the encrypted result in the purpose of making it more difficult to learn about the key and parameters (i.e. making it more non-deterministic). For the symmetric encryption DES and 3DES, there is either the option of PKCS5 padding [14, p. 2] or no padding. If padding is used, it will append an 8 bytes header to the data. This is preferably when using the algorithm with ECB mode, since otherwise the system would be a target for a *Padding Oracle Attack* [15].

For asymmetric encryption RSA, the padding options can be either no padding, PKCS1 [16] or OAEP (Optimal asymmetric encryption standard [17]).

Although paddings are excellent to make the encrypted content even more random, it does this by the cost of reducing the available content as input, which Table 5-2 describes with RSA as algorithm.

## 2.8 Certificates

Certificates contain public keys that are generated by clients and then signed by servers acting as so-called certificate authorities (CA). Each certificate contains a variety of information including the name of the owner, a date for its creation, a date indicating how long this certificate is valid, and an address that can be matched to the domain of the owner, along with some metadata (for example, stating which algorithm to use for encryption).

The purpose of certificates is to form trust between Internet users. Since there are many kinds of users, certificates are divided into categories which represent a certain level of trust. By combining several certificates in applications, a *trust chain* can be formed. At the top of the chain, i.e., the root, is a certificate that every other certificate down the chain relies on. The root certificate is signed by itself, which could be considered as a security loophole if manipulated by the issuing certificate authority. There is also the possibility that an *intermediate CA* can exist between the root CA and the end user. This intermediate CA issues a certificate to the user. The trust chain is formed by checking that each certificate in the chain is signed by the trusted root CA, the intermediate CA's, and all the way down until the end certificate which is the user's certificate.

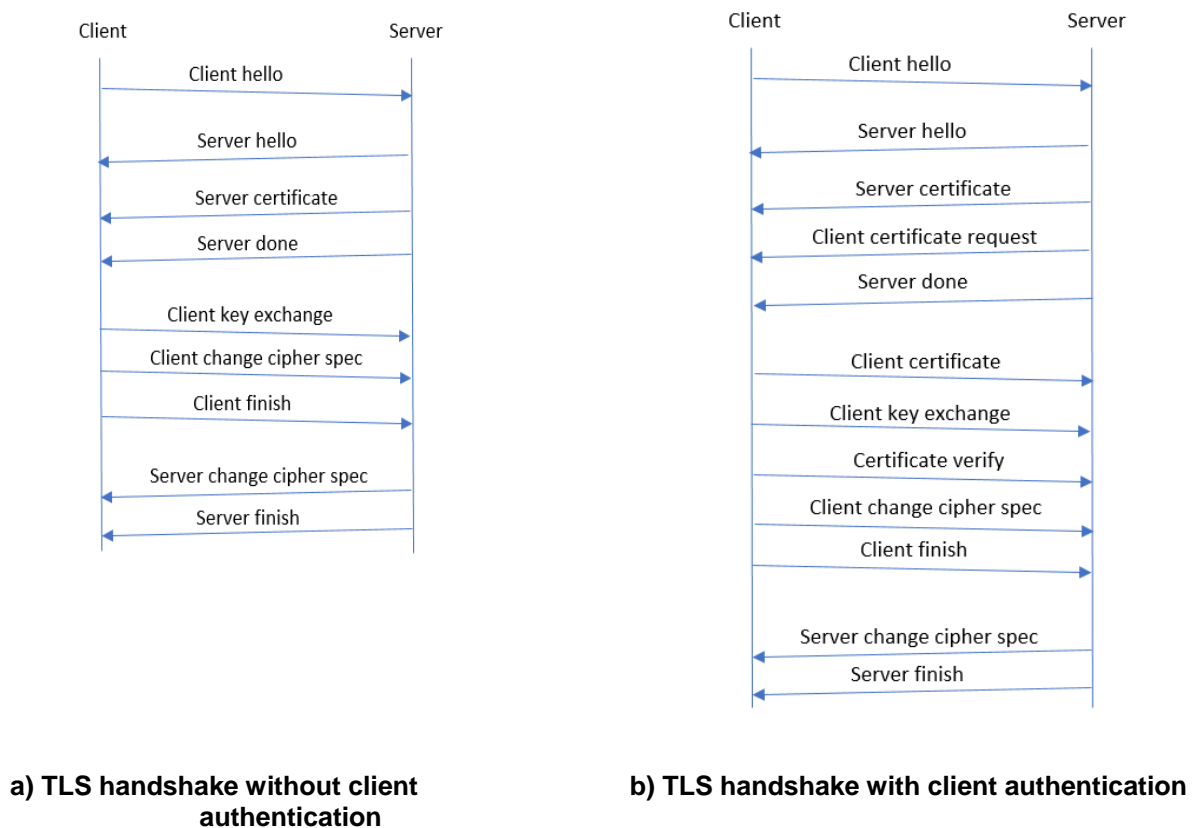
Since there are both clients and servers in the Internet, trust is not always mutual. Most clients might trust the servers, but not vice versa. This can be seen in web browsers where multiple certificates, representing trusted servers, can be installed. These certificates are used by the web browsers to establish a secure communication channel as described in Section 2.9.

## 2.9 SSL/TLS

SSL and TLS [18] are basically the same technology, but use different naming conventions due to their different development paths (after SSL released version 3, the following improvements became TLS version 1.0, 1.1, and so on). TLS establishes a secure communication channel between two parties, most commonly a client and a server. These secure channels can be used when the communication between client and server needs to be secure as well as having the ability to authenticate each other.

The process of establishing a secure channel begins with a so-called TLS handshake. This handshake involves multiple steps between client and server. There are two types of TLS handshakes. The choice of which type of handshake an application uses depends on the application's specific use. A standard TLS handshake enables a trusted communication channel from a client to a trusted server, but not the other way around. This handshake may be used when the application does not require the server to trust the client and hence the client does not need to authenticate itself with a signed certificate. The other type of handshake is used when client authentication is necessary. In this case, the TLS handshake has additional steps and results in *mutual authentication*.

The handshake first uses asymmetric encryption to securely exchange a symmetric key, this symmetric key is subsequently used as a session key. A visual representation of this handshake is shown in Figure 2.4 (without client authentication in (a) and with client authentication in (b)).



**Figure 2.4:** TLS handshake without and with client authentication

When a connection needs to be established, the client sends a “hello” message, consisting of a list of cipher suites [19] that it is compatible with, along with some metadata and a random string, as a nonce. The server responds with a “hello” consisting of the chosen cipher suite (selected from among the received list), which it supports, some metadata, and its own nonce. In addition to this *hello*, the server sends its certificate and its public key in order for the client to trust the server as well as to compute the session key. The client sends the concatenation of the server’s and client’s nonces encrypted with the server’s public key, which the server can decrypt with its private key. The two parties now have the information

necessary to generate the same session key. The client and server complete the handshake by sending a “finish” message which then enables the secure channel.

In the client-authentication mode of the TLS handshake, the server sends a certificate request in the server hello. This certificate request mainly consists of parameters that the client certificate must match (such as the signer of the certificate and what type of certificate it is) so that the server can validate it. If the client’s certificate matches these specifications, it sends along its certificate before the key exchange and adds a certificate to verify the message, which is the signature of the certificate using the client’s private key; this tells the server that this client is the owner of this certificate. If the server validates the client’s certificate, then the handshake completes as previously mentioned.

## **2.10 Related work**

Previous work on networked validation services for physical access is summarized in this section. Moreover, this section covers many different usage areas.

### **2.10.1 Transportation**

In Sweden, there are several public transportation companies that have implemented QR codes as part of their authentication systems. In [20], Tatjana Apanasevic describes many aspects of the use of tickets as realized by mobile devices.

The cities of Stockholm, Linköping, Malmö, and Lund public transportation systems are equipped with optical devices for scanning QR-code based tickets as displayed on mobile phones.

#### **2.10.1.1 Skånetrafiken**

Skånetrafiken runs the public transport in southern Sweden among Lund, Malmö, and other cities. In the city of Lund, buses are equipped with Aztec-code (very similar code to QR-code [21]) readers for authenticating mobile tickets. This technique demands both reliable and secure communication between the mobile device and the authentication service as the buses are changing location all the time. Moreover, network coverage is essential. A couple techniques could be used. The bus could download all the user data (public keys, device information etc) and store it during the day so that subsequently the Aztec-code readers can access this data; thus, avoiding the need for coverage and constant connectivity to an authentication server. The disadvantage is that the buses will not have up to date information about the users since the actual database is likely to be updated multiple times a day. A second approach, which is more likely used, is that the Aztec-code readers on the buses communicate with an internet server using 4G. In this approach, there must be continuous connectivity. Another alternative is that the Aztec-code readers could be disconnected from the backend service, but have a copy of the authentication system’s public key (which could be updated once a day or less when there is some connection between bus and server). Using this public key for validating a ticket could be done if the bus is communicating with the app in some way. Since the Aztec-code reader is simply a detector of Aztec codes, it is unlikely that it can communicate with the app and validate the ticket in this way, although by using nonces this would offer a means to prevent replay attacks.

In retail stores as well as in SQRL systems (see Section 2.10.3.2), the mobile phone is the reader which scans a (possibly) static image of a QR-code, while in Skånetrafiken, the Aztec-code is generated by the app and the scanning devices are mounted on the buses.

Two problems that Skånetrafiken has been dealing with are stability and security. Their newly designed app [22] is based on when the traveller wants to buy a ticket. Their system of tickets and travel search engine was the victim of a DDoS attack [23] [24]. This attack was successful due to the fact that they could not handle the volume of traffic sent to their servers. Another attack, more relevant to their app, was a replay attack that was actually known to them and had been tested by journalists [25]. They had recently launched their new ticket system with their app which was based on Aztec-codes. This leak in the system encouraged some journalists to test if the Aztec-code based ticket, received by a purchaser, could indeed be used later despite the fact it had already been used. In a perfect system, that Aztec-code should have been based on a nonce. However, the transport company wants both security and to persuade users that ticket can be used throughout its designated period of validity. Hence it is likely that time of day based nonces were used by Skånetrafiken in their Aztec-code generation process, perhaps with the intention of allowing for some delay and processing time between the issuance of the ticket and the server's processing of the ticket for authentication. In the replay-attack test, the journalist successfully tried the same Aztec-code 15 minutes after generation, which indicates a substantial risk for a replay attack being successful. This attack does not need the cleartext of the underlying data to be visible to the attacker, as the attacker can simply re-use the resulting QR-code.

Public key cryptography is likely to be used in this system, just as in the retail system, whether the Aztec-code is generated in the app or it sent from the server (either as an image or as a set of parameters that are used by the app to generate the actual Aztec-code\*). Subsequently, the Aztec-code reader scans the mobile ticket and compares it with a code provided by the server.

### **2.10.1.2 SL**

Like Skånetrafiken, StorStockholms Lokaltrafik (SL) has a ticket based on the mobile device's app showing a QR-code instead of an Aztec-code. Since Stockholm is close to Uppsala and there are many daily commuters between the two cities, the two transit system operators have integrated their apps, thus enabling users to seamlessly purchase tickets in either the Upplands Lokaltrafik (UL) or SL region and then using the ticket in either or both service regions. This option of using QR codes as tickets was a great benefit for SL since they had a lot of problems with people hacking their physical RFID ticket cards [26].

Also, similar to Skånetrafiken, SL has had flaws in the authentication part of their ticketing system. A thesis by Tiina Loukusa [27] describes the general RFID systems from a security point of view and mentions that two of the RFID techniques used in access cards, EM4100 & MiFare, are fairly easy to hack. SL utilized MiFare Classic for their RFID based ticketing system. As Tiina states in her thesis and was well known, MiFare Classic based cards were indeed hackable.

---

\* Most likely the server simply sends a public key and parameters for generating a private key, while the app itself generates the QR-code image.

These cards used cryptographic security based on a modified version of ISO 14443A; however, this customized version made it predictable and easy to understand. The result of this security flaw made it easy to clone this type of access cards, thus a copy of the real card could be used at transit stations. After this flaw was discovered, SL improved their security to use Mifare Plus as it utilizes longer keys for the cryptographic algorithms, thus making it harder to crack the keys [28].

### **2.10.2 Banking**

Banking systems have always been an area where there is a focus on security. Time and resources are spent by an attacker to circumvent the security of the systems. This means such a system must resist even the latest types of attacks, otherwise, there is likely to be a compromise of the system's integrity and the attacker will gain some profit from their attack.

One type of identity authentication which banks use is BankID [29]. By connecting a BankID to a user's bank account, both the bank and the user gain a sense of greater security and safer transactions.

The most common issue today related with BankID is spoofing and phishing, where hackers attempt to prove to the user that they are a legitimate authority [30]. This issue can be divided into two parts with different seriousness. Both parts require multiple attempts and are not trivial for success for the hackers. The first part, where the actual spoofing/phishing occurs, is when hackers use both conventional and unconventional methods for getting a hold of the BankID. They can watch while the users enter his credentials without the user even knowing it, also known as shoulder surfing. Other methods include trying to get the user to give them their credentials, either by telling them or getting the user to input them into a fake form online. While these methods only give the BankID data to the hackers, they need this information for the second part of actually trying to retrieve useful information about a user or get a hold of money.

### **2.10.3 QR-based work**

This subsection describes previous work that is QR-code based.

#### **2.10.3.1 ICA QR**

ICA is one of Sweden's large retail food chains. In 2015, ICA launched a payment system which uses QR codes. According to the steps described by ICA [31], the *Mobilbetalning* system for payment using a mobile device consists of a two-factor initiating protocol followed by single-factor authentication.

First, the user must log in to the ICA app where his/her payment cards can be registered. When a payment is to be made, the user selects the card via the app followed by a scan of the QR code at the payment station. The first time, the user has to input their card's PIN code which completes the two-factor authentication. Subsequently, a scan of the QR code is sufficient. An authentication system such as this provides simplicity for the user but can be high risk as well, since the device is connected to a payment card and hence connected with the user's account information.

In her licentiate thesis [32], Tatjana Apanasevic describes the company Seamless that has done a similar system for retail payments.

### **2.10.3.2 SQRL**

Secure QR Login (SQRL) is an authentication system using QR codes. This type of authentication uses a combination of a standard QR code and public cryptography technology. In Steve Gibson's description [33] of the SQRL system, a device which uses this system generates a secret and then uses the QR code, which contains the site's URL, to form a public and private key. This is done by feeding a hash function with the secret and QR code. The output of that function is a public and private key pair which can then be used to communicate with the site. Note that the server and the user's mobile device must share a unique secret.

The use of a public and private key pair together with a PKI is more convenient and secure than the use of symmetric keys. A public key is generated and known for the site which the user is authenticating to. The QR code is signed by the user's private key. Then instead of the user entering a username and password, the QR code is scanned. The site does not know the private key of the user, but by using the public key the site can confirm that the user possesses the private key associated with the username that has been provided.

While many systems (such as TLS) use symmetric keys for each session, the use of symmetric keys to authenticate each user does not scale well but is useful for securing each session after the user has authenticated and been authorized to establish a session. These symmetric keys can be freshly generated for each session and only need to be retained during the session.

### **2.10.3.3 Hotek door lock**

Hotek Hospitality group is a company that designs hotel accessories and products that will improve hotels' internal systems for guests as well as staff. One of their products is a QR-code based door station [34]. This system is used by hotel guests to enter their rooms. Hotek designed their system to be backward compatible with the current magnetic swipecard technology, which is standard in most hotels. The advantage of using a QR-code is that a hard copy can be printed for guests without a smartphone. Use of a QR-code has lower cost as there is no need for a physical key or mechanical maintenance, while still covering those situations where a swipecard is usually needed. They state that no network connectivity is needed to open the doors, hence the QR-code readers on the locks does the authentication locally. The ability to use printed copies of the QR-code also means that a mobile does not need to remain connected with the authentication service.

Internally, their system is likely to rely on asymmetric key encryption to secure the QR codes. This also suggests that the actual time of scanning the QR-code has little to do with the security protocol, but the actual time of scanning may be compared to the interval during which the QR-code is valid. This valid period is the time this user intends to stay at the hotel. The QR-code also contains the room number. The QR-code message can utilize encryption and/or signature to protect the privacy of the message contents and to ensure that they cannot be altered without the system detecting this. However, creating another valid QR-code would require that the attacker must know the private key that was used for the encryption and signing.

#### 2.10.3.4 Libetech

Jeremy Blub and some associates worked on a project called Libetech [35]. That project consists of a standard doorlock, webcam, and a beaglebone development board [36]. For the software, they have written a python script that runs on the board and there is a server-based application along with a database that stores information about the users who have been granted access to enter this door. They have implemented their system by using a ZBar [37] scanner along with their own algorithm for securing the communication and data in the database. Users will receive an email with a QR code which they can show to the scanner to gain access.

#### 2.10.3.5 Access control

Matrix Security Solutions developed a QR-code based door passage system [38] in which the QR-code is a sticker that can be put on a wall next to the designated passage. The user scans the QR-code with a mobile device and sends the code to the main server which parses the QR-code. This code contains information such as IP-address of the door controller and door operation mode (i.e. enter/exit building area). A security check is also made by the server, indicating that the QR code was received from a registered user. Then the server sends the door controller on the specified IP-address the signal to unlock the door. Subsequently they are granted access granted or denied access via the connected door lock controller. The requirement for this is that the door controller and server operates on the same local network since any NAT would probably become a restriction regarding the access between server and door controllers.

### 2.11 Impact

When implementing a system that has the intentions of authentication, authorization, and possibly accounting, it is important to understand what effects it will have. Both physical effects on the environment surrounding it, but also the effects it will have on people (specifically users, non-users, and administrators). Physical effects might (in the case of buildings) be the need to adapt the newly implemented system to the existing infrastructure or networks may need re-configuration. Other effects can be that users and/or administrators require different devices to be able to use the new technology. Moreover, these are devices that they might not already own or use. That leads to a need to buy the necessary devices and a certain time to adapt to the specific device.

Since people today usually have at least one mobile device that they regularly use, people will want and need to be able to control systems, login to accounts, or have monitor functionality (e.g. integrated home surveillance ability) via these devices. This makes it very easy for developers to just add new features that will connect these mobile devices to a service. Moreover, because people regularly use these devices, they are likely to adopt the new service very quickly. However, the introduction of these new services comes with a great responsibility. Because people have integrated almost their entire daily life with their mobile devices, there could be situations in which very sensitive information might be accessible to the new application. For this reason, developers need to understand how to handle this sensitive information correctly, should it be available. It is expected that users will quickly lose faith - if it appears that sensitive data has been leaked or even intentionally becomes accessible to others *without* the users' approval, then not

only will the user's faith be lost but the cost may potentially be enormous. The impact of recent inventions within the mobile device area has lead people to continue to use login system,s banking management, and other highly sensitive applications despite the fact that many attacks against these applications have occurred and continue to occur. Despite the fact that popular applications have strengthened their security features, news occasionally reveals yet more successful attacks.

The impact of new inventions will only have good results if these inventions have sufficient security features to gain the users' trust and that these systems are implemented to be secure against well-known attacks. Moreover, the systems must offer something of value to the users.

## **2.12 Summary**

Looking at the different related works and technologies described above, there are certainly well-designed examples that give great benefits in their area. Table 2-5 summarizes the systems and techniques presented in this chapter.

The next chapter will describe the methodology and methods used for this thesis project.



**Table 2-5: Summary of related work, techniques, and protocols used in different applications**

<b>Application/ Technique</b>	<b>Description</b>
Authentication, Authorization, and Accounting (AAA)	A protocol used for client-server models. AAA is widely used since it gives the most important functionalities required for a client-server session.
TLS	A security protocol designed to improve common HTTP and/or socket communications between client and server.
Nonces, PKI, shared key, certificates	<p>All of these are components of TLS consists of. They are themselves very reliable and scalable to use. The certificates with the corresponding trust chain are simple to use for many kinds of environments, such as testing in large networks, small networks, real-world application etc. They do not require a real CA to issue for testing purposes.</p> <p>Asymmetric and symmetric key encryption have opposite functionality from each other and they have advantages as well as disadvantages themselves. Despite this, they both benefit TLS. Asymmetric key encryption offers higher security but at the cost of more computation, while the shared symmetric key is smaller while maintain a good strength.</p> <p>The use of a nonce helps protect against replay attacks by contributing a unique value to each communication session.</p>
QR codes	QR codes are 2D visual representations of a barcode. Compared to the standard 1D barcode which has very low storage capacity, QR codes can store up to 7,089 characters.
Related Work	Previous works related to QR codes have been varied and range from areas such as hotels to tickets in mobile devices for traveling by bus and train. This work also includes hobby projects which might provide the groundwork for other applications. All of these innovations have one common goal: simple management for users which can enable modern hardware/software to realize a more secure environment. Their use of QR codes to provide this security differs from project to project. Unfortunately, it requires post-release application attacks to identify weaknesses, since every possible attack might not be covered when the application was designed.



## 3 Methodology

The purpose of this chapter is to provide an overview of the research method used in this thesis. Section 3.1 describes the research process. Section 3.2 details the research paradigm. Section 3.3 focuses on the data collection techniques used for this research. Section 3.4 describes the experimental design. Section 3.5 explains the techniques used to evaluate the reliability and validity of the data collected. Section 3.6 describes the method used for the data analysis. Finally, Section 3.7 describes the framework selected to evaluate the optimized prototype design.

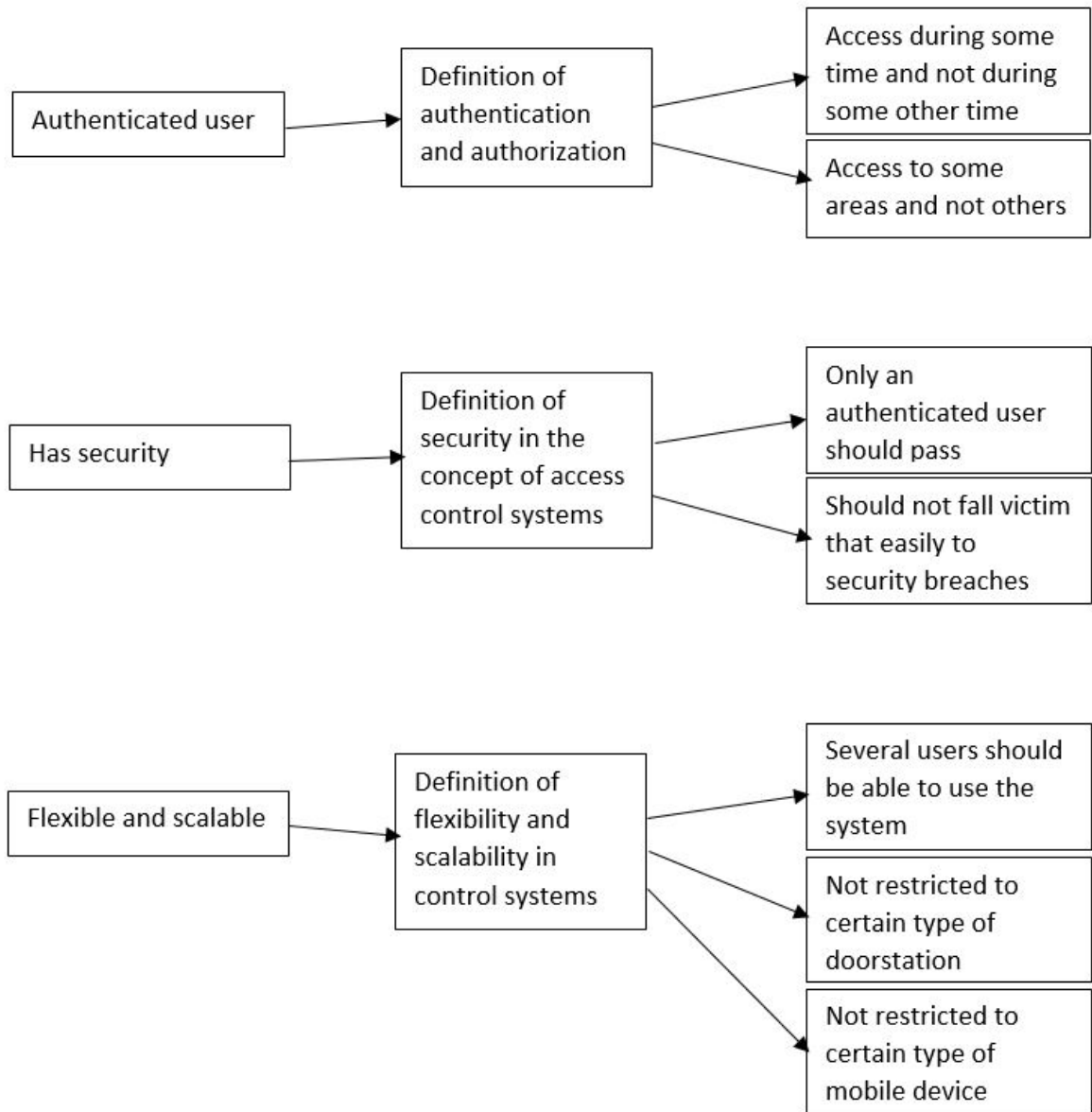
### 3.1 Research Process

The research in this thesis will be based on designing a series of prototypes and evaluating each one of them. The final goal is to build an application consisting of well-tested components. The prototypes could become fully integrated systems or just minor components of a larger system depending on the final purpose of the product. However, the details of going from the final prototype of this project to a product are outside the scope of this thesis project.

### 3.2 Research Paradigm

This thesis will focus on designing and evaluating a series of prototypes along a path from idea to the final application. This process begins with background research and this research should not stop until a definitive version of the prototype is tested. Hence, various concepts that are evaluated must each be tested. This makes a very pragmatic view of the research for this thesis project, hence this iterative design and evaluate methodology is the research paradigm that has been followed.

Some concepts that must be considered throughout the entire process include goal-related considerations, such as; the proposed systems must not be *less* secure than current authentication methods (i.e. access cards) and proof-of-concept validation can be structured into several parts to enable subsystem-wise development. An important requirement underlying this thesis project is a clear definition who is an authenticated user in the process of the validation. These concepts and definitions are briefly described in Figure 3.1. Together, these considerations form the main goal which is to achieve a proof-of-concept validation service that is sufficiently secure to be considered as the prototype of a future product.



**Figure 3.1** The three key definitions used in the research and the related considerations

### 3.3 Data Collection

This section describes the data collection of the project which basically are the TLS packets described in Section 3.3.1 followed up by the QR codes image description in Section 3.3.2.

#### 3.3.1 TLS

Since security is central to the project, all the physical network communication channels must be established using the TLS protocol to ensure that all the communication is appropriately secured. From Figure 4.1, the available channels can be seen as a triangular model of which two of the channels are physical network channels - from the mobile device to the server and from door station to the server. The data sent via these channels will be encrypted and encoded and thus require specific keys to be decrypted. This ensures that the plain text of the data sent via these channels is indecipherable to anyone without the specific keys and only the designated client and server can access this information. This is very important when communication deals with user credentials.

The mobile device and door station are connected to the server via a Wi-Fi connection and Ethernet connection respectively and both of them run applications that are capable of creating TLS connections, hence this will be the underlying protocol for each communication channel.

As mentioned in Section 2.9, TLS initiates a handshake to exchange the symmetric keys among other reasons. These symmetric keys will be sent via asymmetric encryption to ensure their secure transmission. Among the available cipher suites for the TLS connections, we selected the cipher suite *TLS\_RSA\_WITH\_AES\_128\_GCM\_SHA256*. This means that RSA keys will be used as asymmetric encryption keys and AES used for the symmetric encryption (based upon a shared secret key). Furthermore, we specified 2048 bit RSA keys because this is the minimum preferable length with today's available algorithms and it suits both the computing power of the client & server, while being sufficiently strong against the expected adversaries\*.

#### 3.3.2 QR code images

The major type of input that will be collected as data is images. As these images contain the authentication information this raises concerns about privacy, hence we must ensure that the content of the images is properly secured. As we assume that in every use case that a user holds up a device displaying a QR code in front of a camera or reader and because collecting images of the QR code is an essential step for the prototype, we must ensure that these images cannot be manipulated in any way (hence we must ensure their integrity) and at the same time we must ensure their privacy. When the image of the QR code is collected by door station, this data could be decoded at the door station or the image could be sent via the network to another place where a server decodes the image.

---

\* For the purposes of this thesis, the expected adversaries are assumed to be capable of intercepting the Wi-Fi and network traffic, injecting malicious traffic, and presenting invalid QR-codes to the door station. However, they are assumed to not have physical access to the validation server or the authentication server.

For this project, a key requirement is that the data in QR codes is the authentication that the system is concerned with, which motivates the use of encryption to protect this data.

Although securing the QR code itself and securing all the communication are good practice, there remain situations that must be considered (even though they might occur rarely or offer very low risk). For instance, if the image, that will be decoded to extract QR codes, contains more than one QR code, the system should detect this and handle such exceptions in an appropriate and secure way. Also, when an image is received and decoded, it should be discarded as soon as it is no longer needed, hence it should never be stored. There are three major reasons for this: (1) only the encoded data should be further handled, (2) avoiding storage of the QR code reduces the risk of a replay attack, and (3) this reduces the risk of collecting and retaining data that is unnecessary and might even be a violation of personal privacy and integrity.

Not retaining the images is not a problem with respect to accounting (the third part of AAA) because the accounting should only use the authentication data and not any other information that might be contained in an image. As noted above, the user's privacy and integrity need to be balanced with the AAA requirements in the proper way in order to minimize undesirable problems from occurring.

### 3.4 Experimental design/Conducted Measurements

This section describes each experiment that was conducted in order to evaluate the evolving design. Section 3.4.1 describes all of experiments related to the QR code analysis, while Section 3.4.2 describes the encryption tests.

#### 3.4.1 QR-code experiments

In this section, the testbed for analysing the QR code is described as well as all of the parameters used for the specific test.

##### 3.4.1.1 *Measurements of detectable area*

Testing QR code readability in front of a door station was necessary in order to determine what areas provide readable spots and which do not. This is used to form a three dimensional model of the detectable area. We assume that the device that displays the QR code will be aimed towards camera, which means that device's angle in relation to x-,y- and z-axis (of the door station itself) will vary, but not to the camera objective. In general, this test is sufficient to learn how close to a door station a device displaying a QR code must be for the QR code to be read correctly.

The testing parameters are:

- Door station lies on the table with camera pointing up (as illustrated later in Figure 3.3).
- Horizontal coordinates  $[x,y]$  denotes coordinates relative camera  $[0,0]$  with centimetres as unit (cm).
- **Different height settings:**  $[0 < z < h]$  where  $h$  denotes the maximum height in centimetres (cm) above table, must be below the maximum readable height  $z_{max}$ .
- **Ambient light settings:** [Low, High]

- **Mobile screen brightness:** [Low, High]
- **QR code content:** 85-character string
- **QR code size:** 700 pixels (mobile used is Nokia 3, refer to Table 3-1)
- **QR colour:** black & white

### 3.4.1.2 Measurements of rotations

Testing readability in front of a door station for determining what angles are allowed for a successful read of a QR code. This test is important because a device with a QR code might not always be held perfectly aligned to the door station. In addition, the camera objective itself could be set as viewing a widescreen view which will cause straight contours to be rounded as well as angled in the viewpoint. This experiment has the setup shown in Figure 3.3. The test server is configured to only receive data from the QR decoder and mobile device and no validation procedure is configured, since the QR code does not need to have actual validation information for these measurements. In this figure, there are four angles to be measured:  $XZ_{\alpha}$ ,  $XZ_{\beta}$ ,  $YZ_{\alpha}$ , and  $YZ_{\beta}$ . The angles of XZ-plane will correspond to left-right tilting action where YZ corresponds to forward-backward tilting action. The angles themselves in each plane can be considered to have same maximum value assuming that left/right or forward/backward tilt of the device does not matter to the decoding procedure. However, for a more detailed result, these two angles on each plane are separated to see if there is any significant difference in the decoding.

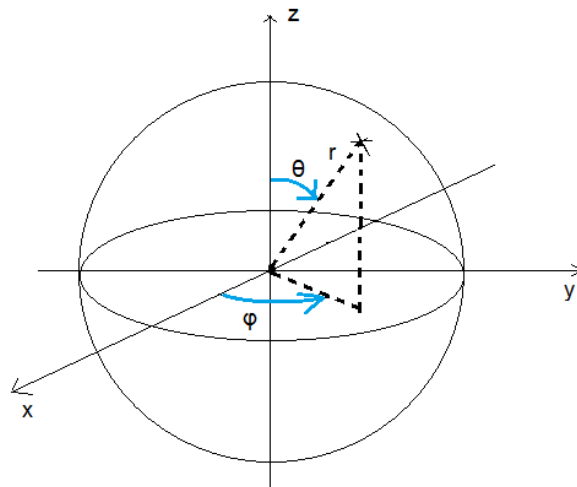
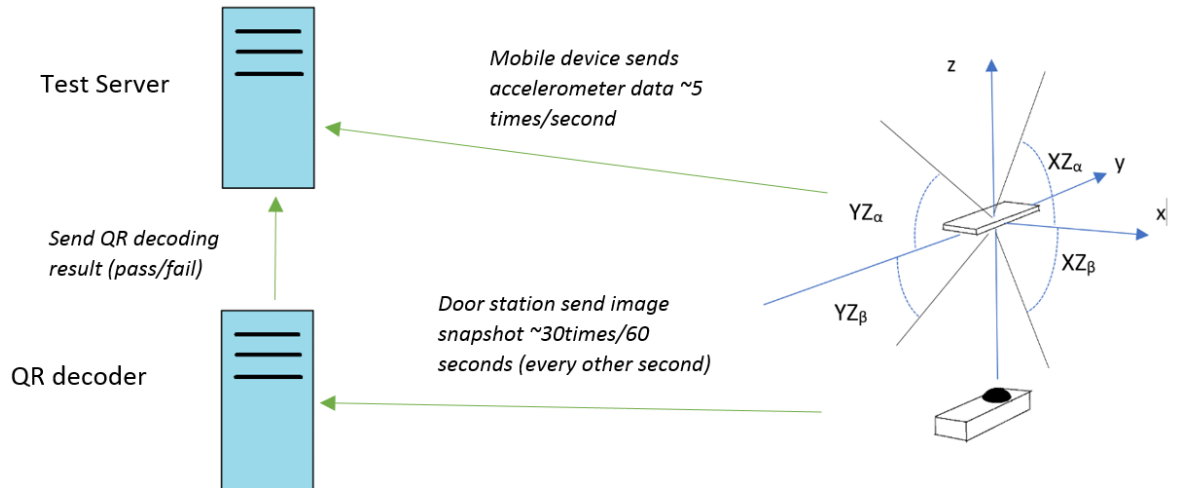


Figure 3.2: Spherical coordinate system



**Figure 3.3: Setup of experiment to measure readability of an angled device**

The testing parameters are:

- Door station lies on the table with camera pointing up.
- Horizontal coordinates  $[x,y]$  denotes coordinates relative camera  $[0,0]$  with centimetres as unit (cm).
- **Static height setting:**  $z = 20$  the maximum height in centimetres (cm) above table from the center of the phone.
- **Angle** to be measured is *theta* ( $\theta$  in Figure 3.2), but since mobile device is held facing down towards door station, theta will be measured with xy-plane as a reference. This is illustrated more clearly in Figure 3.3.
- **Ambient light settings:** [Low, High]
- **Mobile screen brightness:** [Low, High]
- **QR code content:** 85-character string
- **QR code size:** 700 pixels
- **QR colour:** black & white

#### 3.4.1.3 Measurements of colour variation

Testing of QR code readability in front of a door station was done to determine what colours of QR codes can be used and how this will affect their readability. This test is relevant since some colours could affect readability in different ambient light conditions. A QR code has two colours, normally black and white, but in this test combinations of some standard colours are used.

The testing parameters are:

- Door station lies on the table with camera pointing up.
- Horizontal coordinates are set to be directly over camera  $[0,0]$ .
- **Static height setting:**  $z = 20$  denotes the height in centimetres (cm) above table from the center of the phone.
- **Ambient light settings:** [Low, High]
- **Mobile screen brightness:** [Low, High]
- **QR code content:** 85-character string
- **QR code size:** 700 pixels
- **QR colours:**



Black:	[r = 0, g = 0, b = 0]
White:	[r = 255, g = 255, b = 255]
Red:	[r = 255, g = 0, b = 0]
Green:	[r = 0, g = 255, b = 0]
Blue:	[r = 0, g = 0, b = 255]
Yellow:	[r = 255, g = 255, b = 0]
Purple:	[r = 255, g = 0, b = 255]
Cyan:	[r = 0, g = 255, b = 255]
Dark grey:	[r = 50, g = 50, b = 50]

#### 3.4.1.4 Measurements of QR code sizes

Testing of QR code readability in front of a door station for determining which sizes of QR codes are readable. The position of mobile device was fixed because if the device were in moved it would lead to different sizes of the QR code in the image.

The testing parameters are:

- Door station lies on the table with camera pointing up.
- Horizontal coordinates are set to be directly over camera [0,0].
- **Static height setting:**  $z = 20$  denotes the height in centimetres (cm) above table from the center of the phone.
- **Ambient light settings:** [Low, High]
- **Mobile screen brightness:** [Low, High]
- **QR code content:** 85-character string
- **QR code maximum size:** 700 pixels, (mobile screen has 720 pixels width, QR codes also need padding outside, known as *quiet zone*, to be able to work)
- **QR code minimum size:** 32 pixels
- **QR code colour:** black & white

#### 3.4.1.5 Measurements of QR code content length

Testing of QR code readability in front of a door station for determining which complexities of QR codes are readable. A simple string with length of 50 characters will be tried as the least complex code for the QR code up to a 1000-character string as the most complexity. This test is important because another test will consider different encryption types, and these will generate different content lengths which whose content is to be stored inside a QR code (i.e. effecting the complexity of the QR code).

The testing parameters are:

- Door station lies on the table with camera pointing up.
- Horizontal coordinates are set to be straight over camera [0,0].
- **Static height setting:**  $z = 20$  denotes the height in centimetres (cm) above table from the center of the phone.
- **Ambient light settings:** [Low, High]
- **Mobile screen brightness:** [Low, High]
- **Dynamic QR code content:** character string lengths [50 ... 1000]
- **QR code size:** 700 pixels
- **QR code colour:** black & white

### 3.4.2 Encryption experiments

Testing of encryption techniques with focus on processing time but also concerns the resulting lengths of the specific content that is to be stored inside a QR code. Standard encryption techniques such as asymmetric encryption and symmetric encryption are tested.

These encryption experiments will be run on the Nokia 3 device, with specifications in Table 3-1.

The code for the encryption test is described in Code Snippet 1 in Appendix A.

The testing parameters are:

- **Different types:** PKI encryption, standard symmetric encryption, customized encryption
- **Encodings:** Base64, HEX
- **Key lengths:** 128bit, 256bit, 512bit, 1024bit, 2048bit, 4096bit
- **CipherTransformations:**

RSA/ECB/PKCS1Padding

RSA/ECB/OAEPWithSHA-1AndMGF1Padding

RSA/ECB/OAEPWithSHA-256AndMGF1Padding

AES/CBC/NoPadding

AES/CBC/PKCS5Padding

AES/ECB/NoPadding

AES/ECB/PKCS5Padding

DES/CBC/NoPadding

DES/CBC/PKCS5Padding

DES/ECB/NoPadding

DES/ECB/PKCS5Padding

DESede/CBC/NoPadding

DESede/CBC/PKCS5Padding

DESede/ECB/NoPadding

DESede/ECB/PKCS5Padding

### 3.4.3 Test environment

The project's network configuration is shown in Figure 3.4. This network is divided into two separate networks: one wireless and one wired. The wired network consists of a power over Ethernet (PoE) switch which is connected to one or two door stations and a laptop via a USB-Ethernet adapter. The wireless network is created by a smartphone that sets up a wireless hotspot for the laptop to connect to. Two other smartphones will connect to the same hotspot enabling them to communicate with the laptop. There is no physical (wired or wireless) connection between any smartphone and door station.

Door stations have static IP-addresses (in this case 192.168.1.90 and 192.168.1.95). Additionally, the USB-Ethernet adapter has a static IP-address. The

smartphone which provides the hotspot accesspoint will utilize subnet 192.168.43.0 and will run a DHCP server, hence the laptop and the other smartphones get dynamically assigned address from this DHCP server.

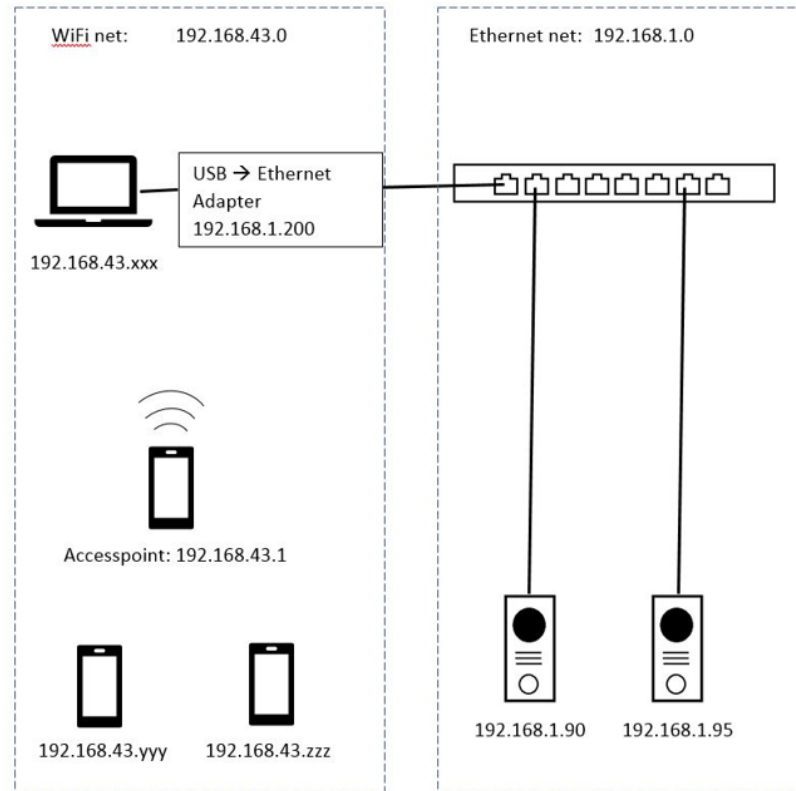



Figure 3.4: Network setup of test environment


#### 3.4.4 Hardware/Software to be used

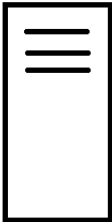
Table 3-1 shows the details of the hardware and software that were used in the testing.

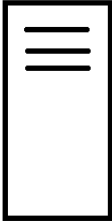
Table 3-1: Hardware and software used

Component	Description				
<b>Mobile device</b> 	<b>Model</b>	<b>Memory</b>	<b>CPU</b>	<b>OS</b>	<b>Physical specifications</b>
	<b>Samsung Galaxy J3</b>	1.5 GB	Cortex A7, 1.5 GHz Quad-core	Android 5.1.1	Resolution: 1280 x 720 pixels Screen size: 5.0 inches, 68.9 cm <sup>2</sup> (~68.2% screen-to-body ratio)

	<b>Nokia 3</b>	2.0 GB	MTK 6737, 1.3 GHz Quad-core	Android 7.0	Resolution: 1280 x 720 pixels Screen size: 5.0 inches, 68.9 cm <sup>2</sup> (~67.3% screen-to-body ratio)
	<b>HTC One (M7)</b>	2.0 GB	Qualcomm® Snapdragon™ 600, 1.7 GHz Quad-core	Android 5.0.2	Resolution: 1920 x 1080 pixels Screen size: 4.7 inches, 60.9 cm <sup>2</sup> (~65.0% screen-to-body ratio)

<b>Door station</b>	<i>Model</i>	<i>Memory</i>	<i>Image sensor</i>	<i>Lens</i>
	<i>A8105-E</i>	256 MB	1/ 2.8" progressive scan RGB CMOS	1.56mm, F2.8 Horizontal field of view: 180° Vertical field of view: 120° Fixed focus
	<i>A8004-VE</i>	256 MB	1/ 3.0" progressive scan RGB CMOS	2.8 mm, M12 Mount, F2.8 Fixed iris, Horizontal field of view: 97°

<b>Validation server</b>	<i>Model</i>	<i>Memory</i>	<i>CPU</i>	<i>OS</i>
	<i>Dell XPS 13 Laptop</i>	8 GB	Intel® Core™ i7-3687U, 2.10 GHz, Hyperthreading on	Windows 8.1 64-bit
	<i>Software</i>	<i>Description</i>		<i>Extra</i>
	<i>Java</i>	version 8 (build 1.8.0_144-b01)		
<i>Eclipse</i>	Developing environment for java applications	JAR libraries: - JSON - MySQL connector - BouncyCastle (for JDK 1.5 or higher)		
<i>Android Studio</i>	Developing environment for android applications.	JAR libraries: - BouncyCastle version 1.58 - SpongyCastle core 1.51 - SpongyCastle pkix 1.51 - XZing core 3.2		

<b>QR decoder server / Database</b>	<i>Model</i>	<i>Memory</i>	<i>CPU</i>	<i>OS</i>
		<i>Intel® NUC</i>	8 GB	Intel® Celeron J3455, 1.5 GHz
<i>Software</i>		<i>Description</i>		
	<i>Java</i>	version 8 (build 1.8.0_131-8u131-b11-2ubuntu1.16.04.3-b11)		
	<i>ZBar QR decoder</i>	version 0.10		
	<i>MySQL database</i>	version 14.14 distribution 5.7.20		

### 3.5 Assessing reliability and validity of the data collected

In this section is the reliability assessment and validity assessment of the data collected, described. Section 3.5.1 describes the reliability was assessed while Section 3.5.2 describes how validity was assessed.

#### 3.5.1 Reliability

The data that will be collected will be collected via different techniques. For instance, in the experiment to measure angle influence on readability it would be difficult to measure angles in an analogue way (e.g. with a ruler or similar), since it would introduce many deviation and it would be hard to determine relevant values. Instead, the mobile device's built-in accelerometer was used to measure the correct tilt values which were sent simultaneously as a stream in parallel with the images of the QR codes. This increases the reliability of the values and enabled rapid & high quantity measurement of this data.

Since experiments were conducted to analyse the effect of different encryption methods with regards to time and content size, one way of determining reliability of these results is to analyse the graphs for unexpected deviations. For example, if an encryption algorithm X should take T seconds to encrypt data Y, but takes 50% longer for one attempt but  $\pm 5\%$  every other time, then timing for that algorithm might be unreliable. Moreover, the testing conditions must be taken into account as well as the hardware used.

#### 3.5.2 Validity

To be able to ensure the validity of the data results, there are various approaches used depending on the type of data.

*QR code images* The images of QR codes are decoded by a QR code algorithm not developed in this thesis. The response from the decoder is a message telling us whether a QR code was found in the image or not. If a QR code is found, then the message also contains

the data encoded by the QR code. This is a straightforward way of knowing whether an image was valid (i.e. contained a QR code).

<i>TLS packets</i>	Since TLS is to be used for the communication channels every packet arriving at the server should arrive on port number 443 and have the correct the TLS protocol version. For the purposes of this thesis project the version must be TLS 1.2, hence any other TLS version is invalid.
<i>Encryption</i>	Various encryption techniques were tested and analysed, which means it is important to be able to determine the validity of this data. The first thing to check is whether the output format (i.e. size) is correct. Since time is a metric used in the analysis, having the correct unit and feasible values can be determined with the help of multiple attempts to avoid certain influence from hardware. For additional feasibility, loggings of the clock during these attempts could be done if necessary clock calibration is required for even better results, though it was not done during these tests.
<i>Mobile devices</i>	The data related to mobile devices is partly validated by algorithms. These algorithms will ascertain that the data is time synchronized as well as verify that the data comes from a verified mobile device. The received messages are considered potentially valid if the time synchronization indicates a feasible time difference. For this purpose the time difference is limited to 25 ms. The time difference is checked manually by examining the clocks of the devices. For challenges sent to mobile devices, they are only valid if the challenge is verified by the mobile device and invalid otherwise.

The PIN codes input via the mobile devices (input by the users) are considered valid if and only if the verification succeeds (i.e. the received PIN code matches the PIN code stored in the database).

Another part of the mobile device's validity concern the alignment with respect to the doorstation's camera - when the devices is placed before the camera. Since it is very important that the mobile device be placed within the limited angles, distances, and coordinates necessary to produce a correct (i.e., correctly readable) scan, otherwise the data will be considered invalid. The limits of alignment will be assessed experimentally.

### **3.6 Data Analysis**

Section 3.6.1 describes the analysis techniques, while Section 3.6.2 describes tools used for conducting the experiments in this thesis.

#### **3.6.1 Data Analysis Technique**

The research in this thesis is based on analysing the readability of QR codes in different situations as well as how different security parameters and environmental conditions affect this readability. To be able to determine solutions and

optimizations for different methods, a broad range of the parameters were used, yielding lots of data. Therefore, the most suitable data analysis technique was to use a quantitative approach for managing all the data. Then conclusions were drawn based on statistical analysis rather than a qualitative analysis.

### 3.6.2 Software tools

Wireshark [39] is a popular tool for packet analysis. Wireshark was used to capture the packets sent between a doorstation and validation server as well as between a mobile device and validation server.

The statistical program R [40], was used to calculate statistics and render graphs for the analysis.

## 3.7 Evaluation framework

The evaluation of the prototype is based upon a security analysis of the components used to build the prototype as well as the performance of these components need to operate in the expected working environment. Table 3-2 defines the current framework and questions for the research.

**Table 3-2: Evaluation framework table**

Objective	Evaluation Question	Method
<b>Define a secure communication between client and server</b>	Can the physical link be secured using existing protocol?	Gather data using existing protocols
	Do we have to design the secure link ourselves?	Design protocol and analyse security
<b>Define a readable QR code</b>	Do physical parameters affect in the test environment?	Gather data from reading angles, distances etc.
	What parameters for the QR code rendering affects readability?	Try different parameters such as pixels, content lengths etc.
<b>Make the system reliable and fast</b>	Can we try many devices of door stations and mobile units?	Multiple connections can affect reliability
	What parameters in the system design can affect the performance?	Trying different hardware and software
		Tweaking the functions of different applications used





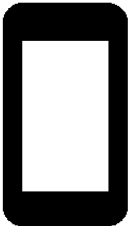



## 4 Developing a proof-of-concept validation service

In this chapter, the development process is thoroughly described. The chapter also describes each of the system components with their purposes, the QR decoding service properties with respect to its advantages and disadvantages, the system from a security perspective, and finally the complete system.

### 4.1 System components

The components used for this project are listed in Table 4-1. Each of them are described from the point of view of their purpose in the complete system.

**Table 4-1: System components**

Component	Model	Description	Purpose
	Samsung Galaxy J3		Setup a Wi-Fi hotspot
	Nokia 3		Test device for QR codes
	HTC One (M7)		Test device for QR codes
	A8105-E		Scan the QR codes
	A8004-VE		Scan the QR codes
	Dell XPS 13 Laptop		Developing environment, running validation service, host for guest VM
	Intel NUC mini PC		Running QR decoder server and MySQL database
	VM		Developing environment, running QR decoder server and MySQL database
MySQL database			
			

The doorstations used are the Axis A8105-E and Axis A8004-VE which are devices with an integrated camera, speaker, microphone, and a button. Their purpose is to provide a one-way video stream together with a two-way audio session between host and guest. Their additional purpose in this project is to act as a reader for the QR code displayed by a mobile device that is positioned in front of the camera module. For this to work, there must be an application running in the doorstation that either scans and decodes the images or sends these images to another server which performs the decoding. Nevertheless, the minimum requirement is an additional application running on the door station for this purpose.

The software application for the doorstation was developed in this project and is written in Go. This application also provides different application functions such as evaluating various metrics and different QR decoding methods.

## **4.2 Optimizing the QR decoder service**

A QR decoder is necessary to read the content of a QR code. Details of how this QR decoder was implemented varied with the design prototypes in order to test and then evaluate each of these alternatives in order to select an optimal choice. These alternatives can be grouped into the three described in the following subsections.

### **4.2.1 QR decoder in door station**

The initial design choice was to have the door station itself act as the QR decoder. This was done to reduce the work load on a potential server. Additionally, the thought that having the door station do this decoding would balance the system.

Axis Communications had already developed a built-in application for their door stations which is able to detect when a QR code is held up to the camera. This application was tested and evaluated.

Another method was to install XZing's ZBar library, which is a C-library that decodes QR codes. In order to utilize this library I developed a wrapper in Go for this library. In this way the application could be fed images which then were analysed for QR codes.

### **4.2.2 VM as a service**

Migrating the QR decoding service from the door station to another computer was done by deploying a virtual machine (VM) to run this service in. In this approach the ZBar library could be installed in a Linux environment as a native program. However, this library was not that compatible with Windows. Placing the QR decoder in a VM was motivated by the fact that such a VM could potentially perform much better than running the same code in an embedded system, specifically a door station.

### 4.2.3 Spreading out services

The final design alternative was to place the QR decoder in a completely separate computer. The reasons for this were that the laptop was until this point hosted the development environment, managed connections with door stations, acted as the validation service, and hosted the VM. This put a lot of pressure on the laptop that was affected in the performance by running multiple applications, one of which was a VM. This made the memory consumption as well as CPU utilization to increase to a high point and motivated the choice to have an additional computer, in this case a mini PC, to run only the QR decoder.

## 4.3 Database

A database is required for this system to work properly. This database stores user's mobile device information that is necessary to check against when a validation is in progress for a newly arrived user. This database contains information that makes it possible for the server to map incoming QR codes to a corresponding decryption key, along with functionality to determine who is to be authorized access via a specific door station and determine whether encryption parameters are correct. The database stores a table of checked nonces (in order to prevent replay attacks). The complete database configuration tables are listed in Table 4-2 together with their column names and types. Every table is configured with an incrementor *id* for performance reasons, as when tables grow very large this value will speed up the search.

The *Doorstations* table has only two relevant columns: *guid* and *restriction\_mode*. The latter describes what authorization type the door station has. This is designed to have two values: "standard access" or "restricted access", where the first one means a normal case situation where anyone who successfully authenticates themselves will be granted access, but a door station with restricted access will granted only a few users access. The decision to these two values was to simulate the two types of authorization modes that can currently be set for a door station. The column *guid* represents the door station's guid which will be used as a reference for the restriction mode column when searching the table.

The table *devices* stores any relevant (to this system) information about a device. When a mobile device is first introduced in the system, a *device\_guid* will be assigned to this device. Subsequently the system uses this value to fetch other information about a mobile device from the table, for example to use in the validation procedure. The last column in this table, *access\_mode* is used along with the *doorstations* table to realize the restriction access mode, explained earlier.

The last table, *nonces*, is mainly used to check if a provided nonce exists in the table or not.

**Table 4-2: Configuration of database tables**

<i><b>Table</b></i>	<i><b>Columns</b></i>	
	Name	Type
<i><b>doorstations</b></i>	guid	varchar(64)
	restriction_mode	varchar(16)
<i><b>devices</b></i>	device_public_key	varchar(512)
	device_guid	varchar(64)
	access_mode	varchar(16)
<i><b>nonces</b></i>	device_guid	varchar(64)
	nonce	varchar(64)

## 4.4 Implementing security

This section describes a number of different methods for applying security to the system components and flows. Section 4.4.1 describes the protocols for securing the underlying communication, while Section 4.4.2 describes the application's security components that have been developed.

### 4.4.1 Lower level security

As described in Section 2.9, TLS it is a great protocol for achieving security for a TCP channel. Also since there is the option of having either standard authentication or mutual authentication, a choice can be made dependent on the application in question. Alternatively, IPsec could be implemented. However, this would mean having to work during the development more at the OS-level with the network security of the components. Moreover, IPsec is more suitable for networks such as VPNs where application security is a problem [41]. In contrast, in this case, all of the components are already TLS-compatible, thus motivating the use of TLS for the secure underlying channels between components.

TLS utilizes certificates, as described in Section 2.8. These certificates are not hardcoded in this system's components, only in the door station and backend (validation server, QR decoder server and database). The application in the mobile device(s) will not contain certificates, but certificates will be a part of registration to the service, as explained in more detailed in Section 4.5.2.

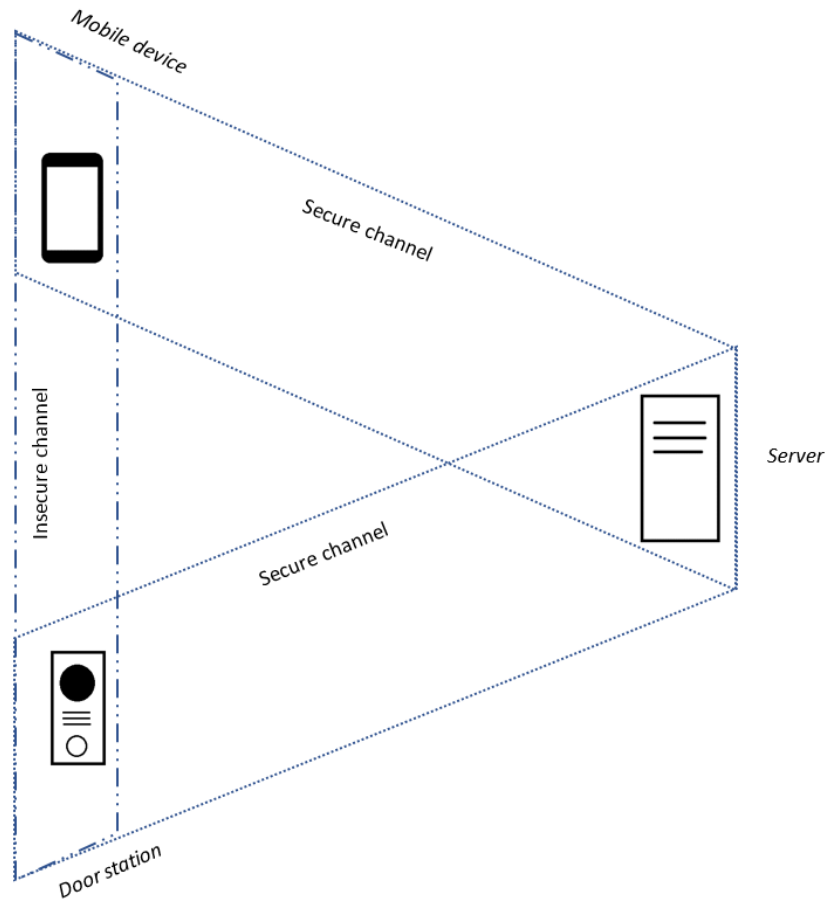
### 4.4.2 Application level security

This section describes the security that is implemented from an application aspect.

#### 4.4.2.1 Choice of algorithms

The location of the QR decoder was not the most important issue but rather how to secure the proposed system was. From a security perspective, the system can be seen as a three-channel system, where two of these channels are secure whilst the

third is not. Figure 4.1 demonstrates this scenario where the insecure channel is the channel between the handheld device (mobile) and the door station. This is simply because that channel is based wireless transmission channel between the camera and the displayed QR code on the device. The door station captures a digital image of what is currently viewed (i.e. taking an image snapshot which includes a picture of the handheld device).



**Figure 4.1:** Insecure and secure channels in system layout

This means that the data to be sent via the unsecure channel, must be somehow protected against any malicious attacker who could learn about the system from this data. This means that no sensitive data or security parameters can be sent via this channel. Also, since the data that is sent is in the QR code itself, this means that the content of the QR code cannot reveal any of these parameters or sensitive data, which in turn means that the actual content must securely encrypted before it is rendered as a QR code.

There are several encryption algorithms that could potentially be used. Several of these algorithms are evaluated in this thesis based with regard to various aspects. The major types of encryption algorithms are symmetric key encryption and asymmetric key encryption. They are both very strong but are most useful for different purposes.

#### 4.4.2.2 Symmetric keys

Symmetric keys, or *secret keys*, are desirable due to their performance in encryption and decryption when small keys are used as in Figure 4.2. Symmetric keys are already in use in protocols such as TLS when encrypting the content to be sent over the TLS tunnel. In a scenario such as that shown in Figure 4.1, if the door station is to decrypt any data sent from the handheld device, then the door station and the handheld device need the same symmetric key, which they must either have as a pre-shared key or have negotiated. Since these devices communicate via an insecure channel, the only secure way for them to communicate is via the server. This communication will be safe since these two channels are secured TLS.

Whether or not the QR decoding takes place at the door station, the content of the QR code must be encrypted in such a way that the service which performs the validation is able to decrypt it. Since the handheld device generates the QR code, it can encrypt the content with a secret key which it has negotiated with the validation service. Assuming that the server in the figure does the validation and there already is a secure channel between the handheld device and the server, they can negotiate secret keys can be used for encrypting and decrypting the QR code's content.

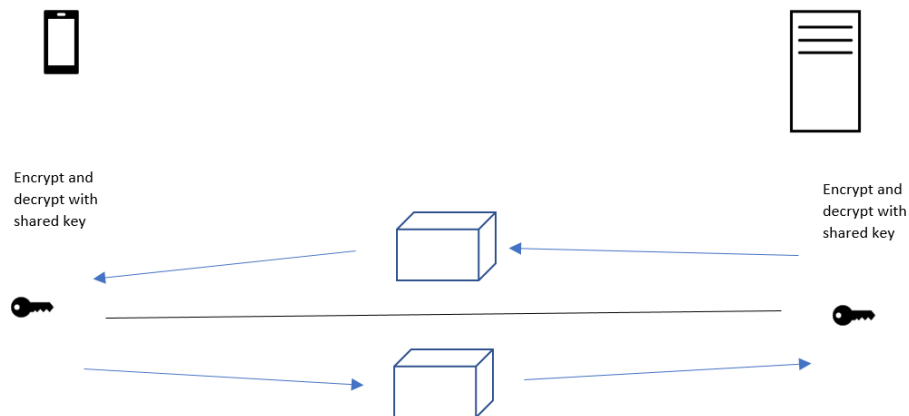


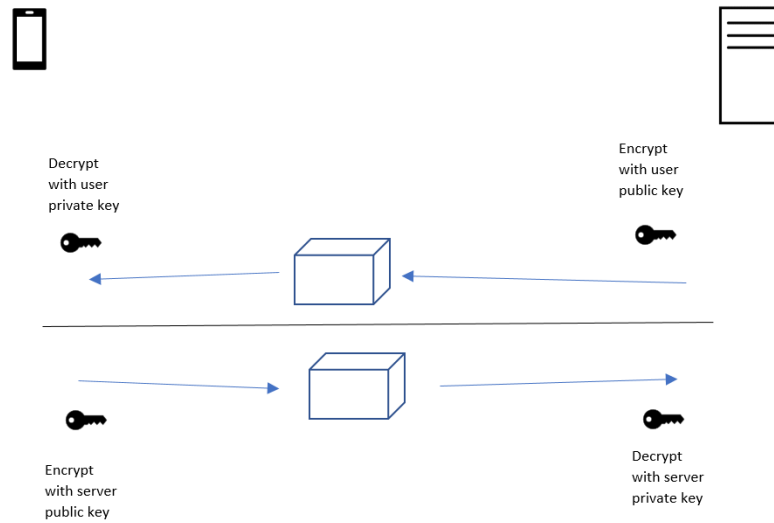
Figure 4.2: Symmetric keys encryption and decryption

#### 4.4.2.3 Asymmetric keys

Using asymmetric keys is a widely used method for distributing symmetric keys securely as well as ensuring authenticity of data. In the case of the problem in Figure 4.1, if symmetric keys are used, then anyone with the key can decrypt the content in the QR code. Since QR decoders are available for most smartphones today and also available online, all that is needed is a copy of the QR code, which is easy to get from the unsecure channel (for example, someone might see the code that is being display). Given such a copy of the QR code if some knows the secret key they could decrypt the data present in the QR code.

Asymmetric keys avoid this problem by having the mobile device encrypting the content with the server's public key (assuming that this server performs the validation of content). When the decoded QR content reaches the server, only it has possession of the private key which it can use to decrypt the content (as shown in Figure 4.3). This increases the security of the physically unsecure channel. Even if someone does get in hold of the QR code, they have little chance of decrypting the

content. Additionally, if a nonce is included in the content, then the server can prevent replay attacks.

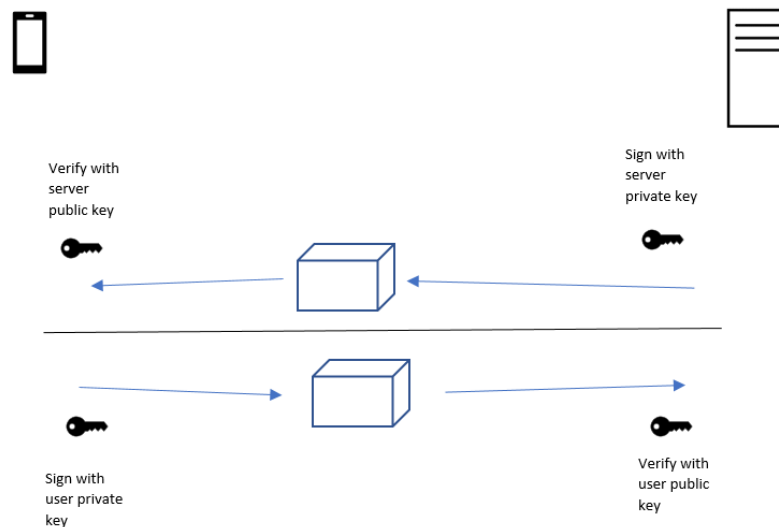


**Figure 4.3: Asymmetric keys encryption and decryption**

#### 4.4.2.4 Sign and verify

As mentioned earlier, asymmetric keys can not only be used for encryption but also for ensuring data authenticity. This authentication can be done by reversing the use of the keys for encryption in order to realize signing and to verify the data. In this process the sender signs the data with its private key and a receiver who possesses the sender's public key, can verify the signed data. See Figure 4.4.

Signing and verifying data improve security in this system since users who connect need to be authenticated before using the service. Details of this authentication procedure will be described in Section 4.4.2.5.



**Figure 4.4: Signing and verifying procedure**

#### 4.4.2.5 Challenge device

When a new mobile device wants to use the service, it is important that the user is authenticated before continuing the process. If no authentication check was implemented, the connection between the server and a mobile device might not be necessary but would then require the mobile device having the capabilities to generate QR codes with content that the server still must be able to recognize in a secure way. This will be explained in detail in the next section.

The authentication method used is based upon having the user sign a nonce, created by server and sent to user (encrypted with the user's public key), and then send this back to server (encrypted with the server's public key). The server can use the original nonce, the received signed nonce, and the user's public key to verify the signed nonce which completes the challenge of the device. This process will be shown later in Figure 4.7.

### 4.5 Complete design

In this section, there will first be an example of the case of a simplified validation procedure which uses the components described so far, but in an unsecure way. This is described in Section 4.5.1. In Section 4.5.2, a secure version of the validation design will be thoroughly described.

#### 4.5.1 No authentication example

In the previous section, it was mentioned of the case when no authentication check was implemented. If the mobile devices were not required to connect to the server, just be used as plain hardware device that provides QR codes to a camera, then there would be a less secure situation when the validation procedure begins. If we assume that the validation requires a nonce, a user's guid, a user public key, and a timestamp. Then the QR code created by a user could contain the user's guid, timestamp, and nonce, all encrypted with the server's public key. However, the server's public key must still be accessible for the user, hence it must be built into the mobile application. Assuming that is the case, the user encrypts the content using the server's public key and forms the QR code which is subsequently displayed to the door station. Later when server gets this QR code, it first decrypts the code using its private key and learns the user's guid, nonce, and timestamp. The server can then check the nonce table in the database to see that this nonce it is not already there, if it is missing it adds it and continues, otherwise it rejects it as a replay attack. If a QR code has a limited time during which it is valid, then the timestamp also needs to be checked. In the case of a simple validation, this could be done right now, since it all three content parameters are available. However, in this case no authentication was done, hence this QR code could have be sent from anyone, since the door station provided the QR code; therefore, the server has no way determining if this was provided by a genuine user or just a screenshot of a genuine code, provided by an attacker. This motivates the use of authentication prior to the validation process.



#### 4.5.2 Secure and optimized validation procedure

The example in Section 4.5.1 makes it clear why authentication is required to have a secure validation procedure. One method to improve the security from the previous example would be to use encryption-then-apply a Message Authentication Code (MAC). This would mean that the encryption is done as before but a MAC is appended to the ciphertext before storing the content to the code. This helps the server determine if the content in the received code has been compromised along the way.

Although this minor improvement would help in some cases, it will not solve the problems of all of the possible scenarios in this project. One scenario is that 2-factor user authentication is required (QR code + PIN code), hence some traffic between the server and mobile device occurs; therefore, this communication must be secured. As the intention is to implement as much application security as possible, we need to use a challenge in the authentication. This does not mean that the user is authenticated, just that the mobile application is approved to use the service.

Figure 4.5 shows the complete system design. It shows an overview of the main components, where the arrows show what the components can connect to.

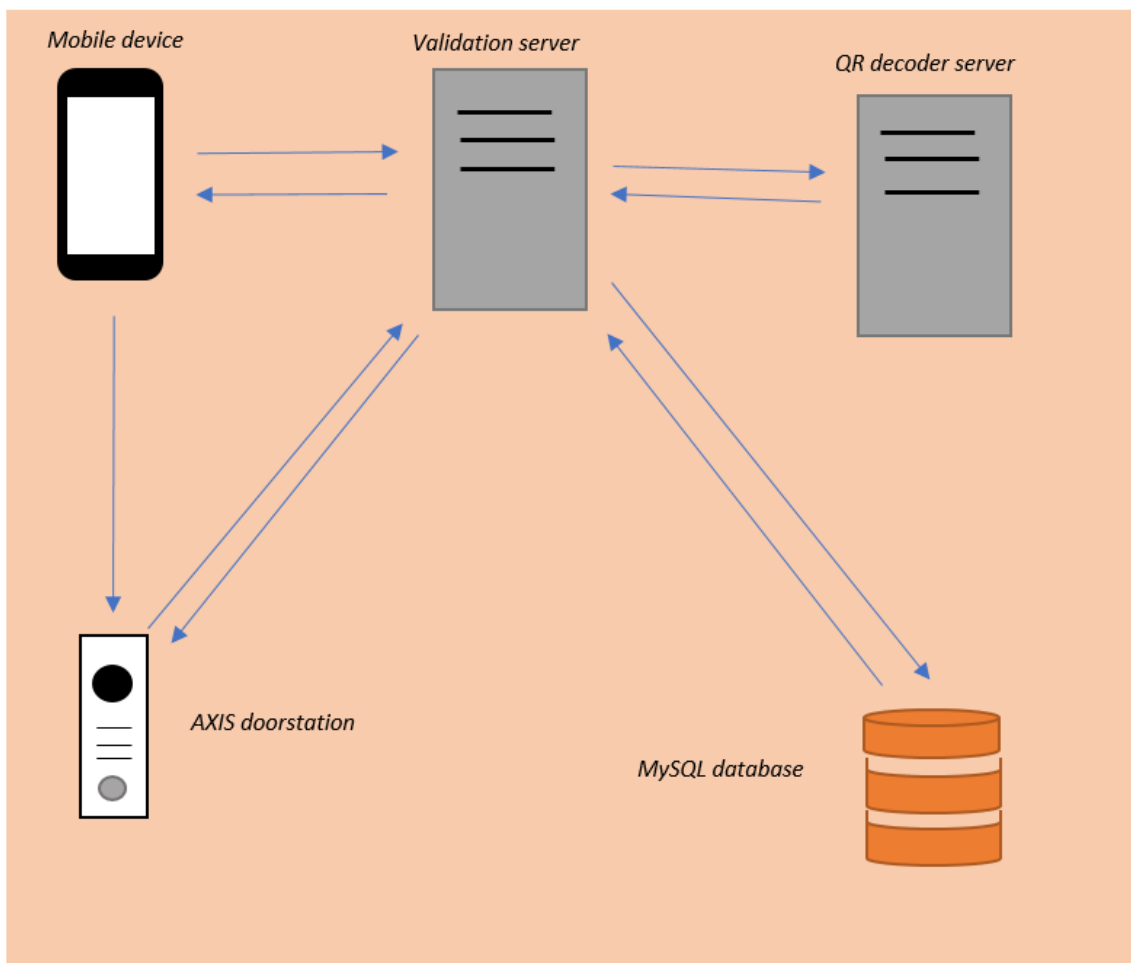


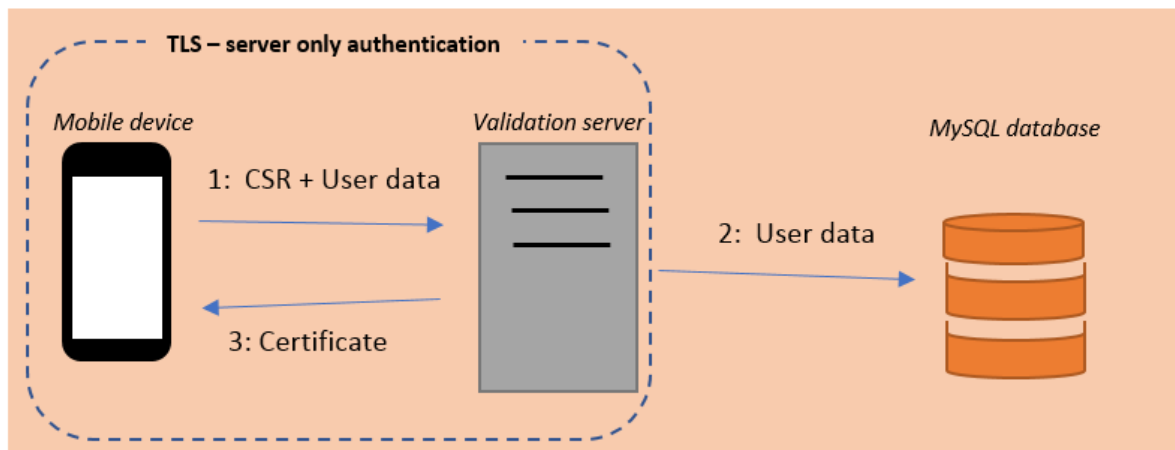
Figure 4.5: System component overview

First, no matter how secure or insecure the validation is considered to be, the first requirement is that a user be registered at the service before we can begin validation. This means that the database needs to contain the relevant user information for each genuine user. In order for the database to be populated with this data, the data first needs to be supplied to the system. For this reason a registration function was developed, rather than having a few hardcoded users in the database.

Before registration, the mobile device does not contain a certificate for use with the TLS connection. This means the initial connection will be server-only authenticated. Later, when user is involved in a validation process, it connects with a mutual-authenticated TLS connection, meaning that this first registration should supply the user with a certificate.

The complete registration process is shown in Figure 4.6 where (1) is the user sending registration information to the server. This information is used as parameters for a certificate, which is held by a certificate signing request (CSR), and also some user information to be (2) stored in the database along with the public key for the generated certificate. Then the certificate is returned to the user, this user can now connect to the service with mutual-authenticated TLS and thereby use the service.

The advantage of having the server create and supply the certificate to a user is not clearly shown in this project setup because the mobile device will register and then immediately get the certificate. The real-world scenario would be that a user registers at the service and then some administrator will process this registration request before determining that this request was from a valid user (i.e. employee at the company).



**Figure 4.6:** Registration procedure

After registration is complete, the user can initiate the validation process at the service. This is done immediately after the user connects with the service. This process begins by the mobile device sending its *user\_guid* to the server. A general description of the procedure is given in Section 4.4.2.5 and shown in Figure 4.7. A more detailed described implementation of the challenge is shown in Figure 4.8. In this figure, we have the following notations:

- $USER+$  public key of user's mobile device
- $N_{Server}$  nonce created by server
- $E_R$  encryption of resource R
- $S_R$  signature of resource R
- *Verify* function that verifies signature

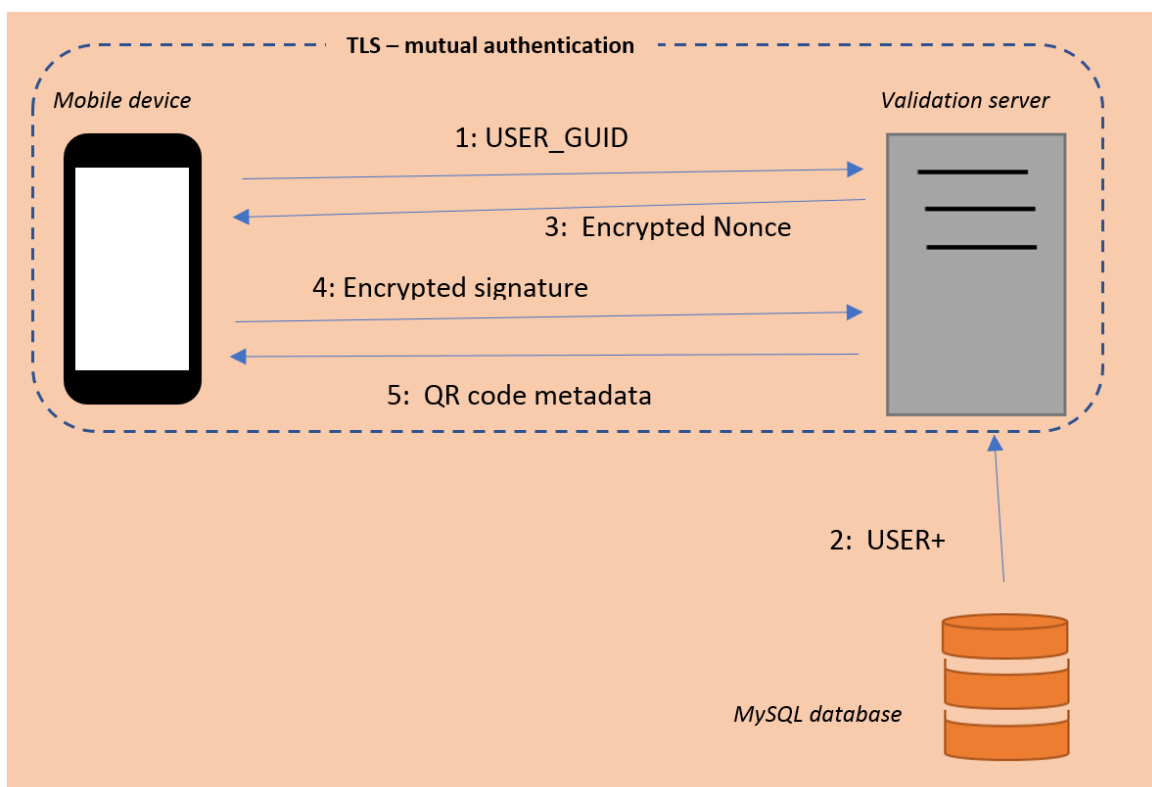
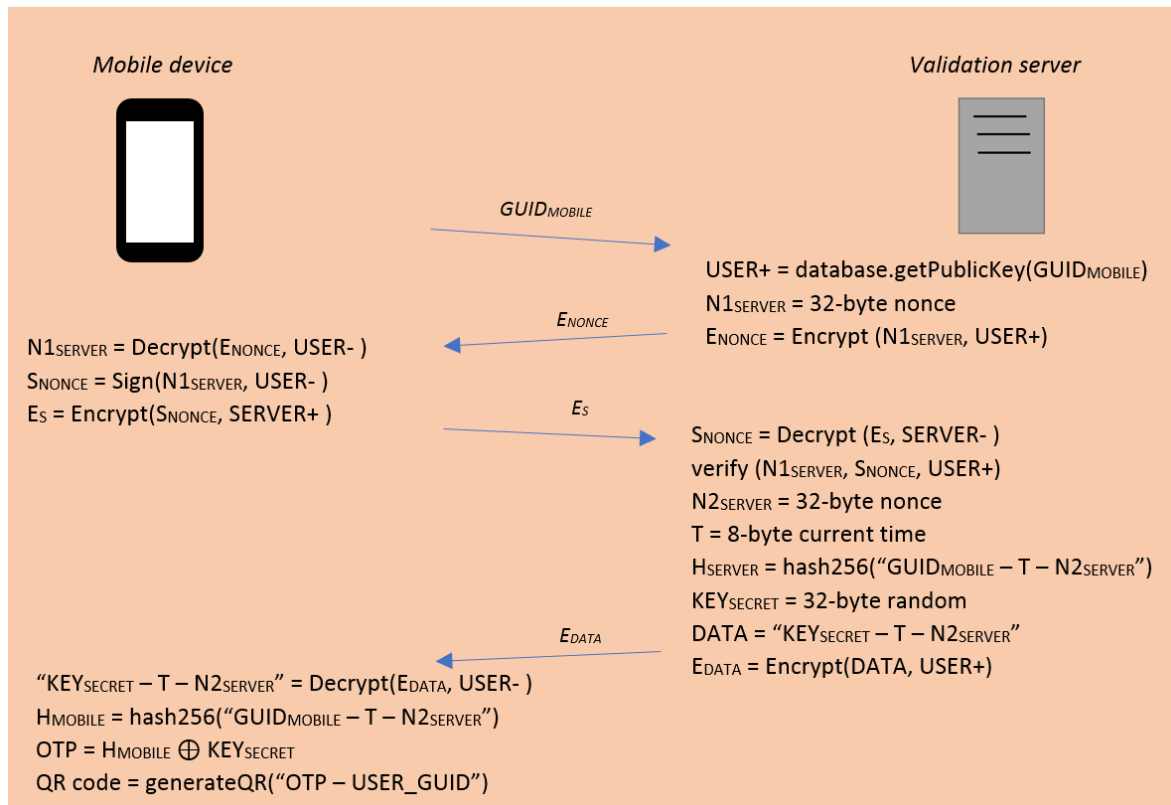


Figure 4.7: User authentication to the service



**Figure 4.8: Detailed flow of challenge between mobile device and validation service**

The main idea for keeping the application is to use encryption for the data transmissions between mobile device and server. The choice of encryption was based on the type of security it provides, if authenticity and integrity are included, how flexible it is, and how it will affect the QR codes.

Part of the research done in thesis project and the result of that research, was to understand the reasons for implementing different encryption methods at different places in overall system.

The first thing to notice about the challenge in the Figure 4.8 is that *asymmetric* keys were used for encryption up to the point where the QR code was generated. This method was chosen because the connection between the mobile device and server is already over a secure channel (TLS) and because this channel can be considered “abstract” and independent of the data, unlike the unsecure “physical” channel between the mobile device and door station. This lead to a decision to simplify the QR code’s complexity (i.e., reducing its content); however, this could only be done by storing only the absolutely relevant data and encrypting this data in a way that does not result in a large encrypted output.

The application logic that provided this security has the server compute a secret key (i.e., a symmetric encryption key) and some metadata, then encrypt this (using asymmetric encryption) and send the encrypted result to the mobile device. Both the server and the mobile device can compute a hash of the metadata. Next, the mobile device will use the secret key it got from server, to encrypt the hashed metadata forming a One Time Pad (OTP) by XORing.

In the event of an eavesdropper seeing the QR code and decodes it, they will only see the result of  $hash \oplus key$  which is useless without either the unencrypted hash or the key. Moreover, even assuming that the eavesdropper does know the key, hence can compute  $(hash \oplus key) \oplus key = hash$ . However, in this case the eavesdropper would learn the hash of the sensitive metadata but not the actual data. This demonstrates that learning the key is useless since once the key is generated and used, it is only used once, hence the resulting pad used only once.

So far, the connection discussed, has been between the channel between the mobile device and the validation server. At this point, when the challenge of the mobile device is complete, the user will holding up their mobile device to the door station for reading of the QR code. This will lead to a second connection in this case between the door station and validation server.

We will assume that a connection has already been setup between the door station and validation server when the door station is physically setup. This connection will remain open as long as the application in door station is running and the server thread is alive. Maintaining this connection instead of the door station connecting each time data needs to be transmitted (as the mobile device does), reduces the resources needed for initiating a connection. Moreover, it avoids the delay of having to re-establish the TLS tunnel. The door station is configured to handle the exception that occurs if this connection dies, by trying to connect until it establishes this tunnel again.

Figure 4.9 shows the scanning a QR decoding procedure in progress. The user holds up their mobile device while (1) the door station takes a snapshot (i.e., captures an image) and (2) sends it to the validation server. In step (3) the server forwards this image to the QR decoder server which does the decoding and (4) returns either a successfully decoded result or an error message. Depending on this result, the validation server will either send an error message right away to door station (5a) or initiate a validation process of the decoded content (5b).

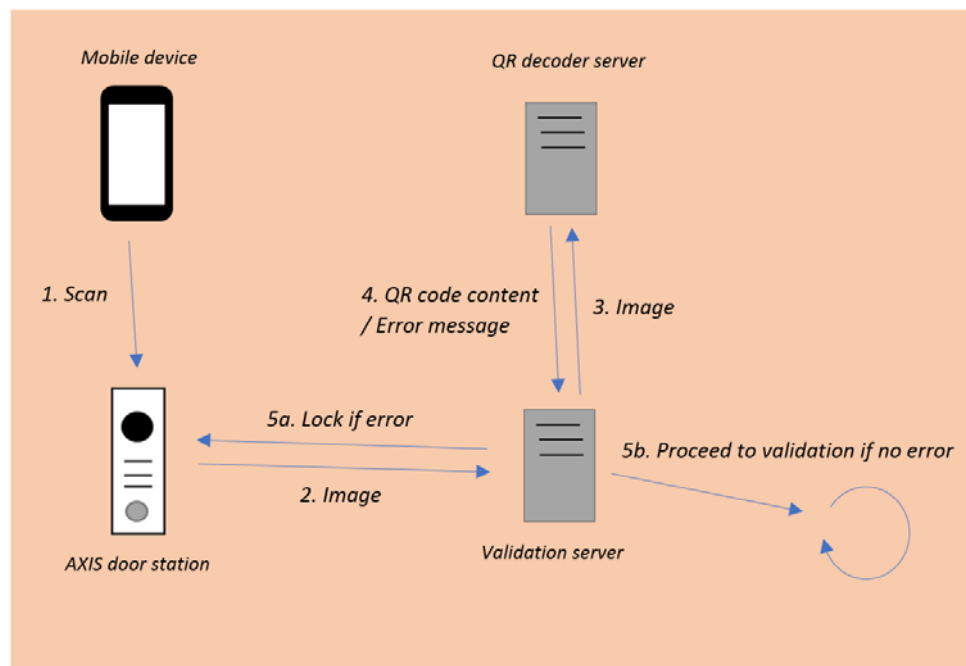
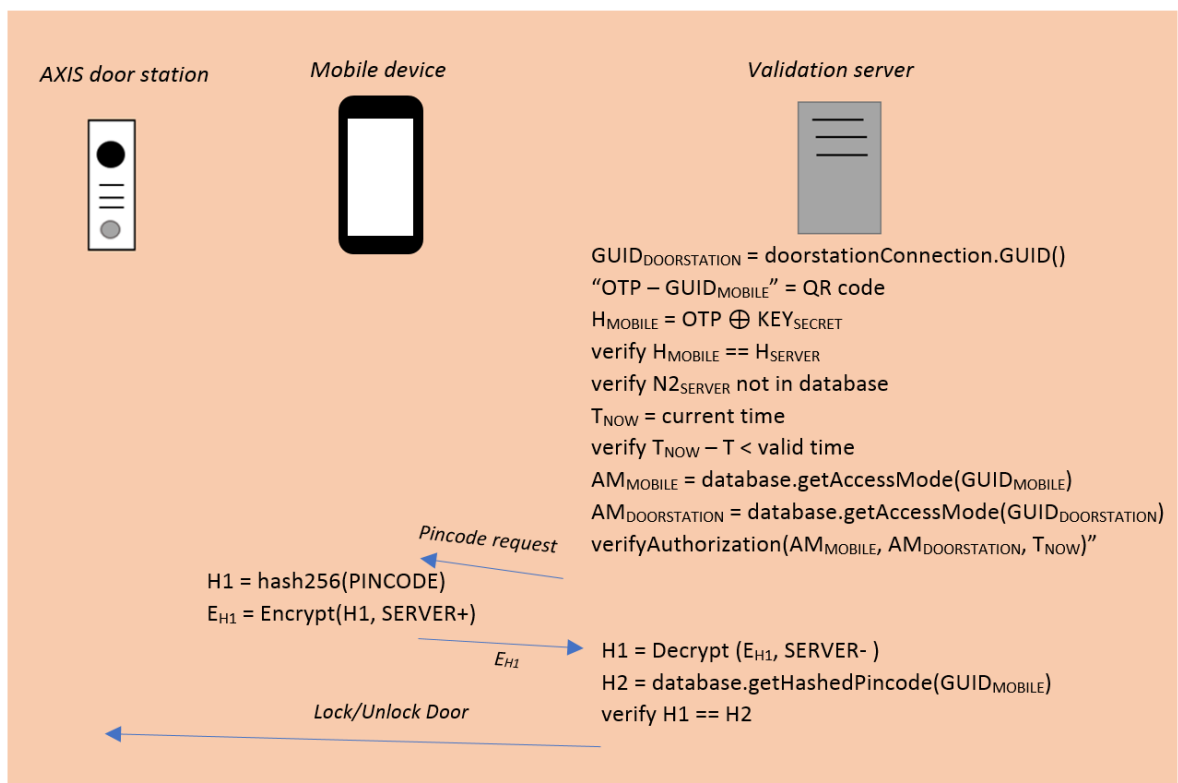


Figure 4.9: Scanning and QR decoding procedure

Assuming the QR decoding is successful, the next step is the actual validation of the QR code content. This is shown in Figure 4.10. For this figure to make sense, some modifications are made from previous annotations and names as well as a few clarifications of functions:

- USER\_GUID has changed name to GUID<sub>MOBILE</sub>
- GUID<sub>DOORSTATION</sub> is the guid that a door station will supply to the server when it connects to it. This guid will be accessible by calling the function *doorstationConnection.GUID()* in the server thread that maintains the connection with the door station.
- Access mode is defined to have three different values:

Strong authentication	requires PIN code
Normal authentication	requires PIN code
Low authentication	requires PIN code at certain times



**Figure 4.10: Validating content of a QR code**

For further clarification, refer to Figure 4.11 and Figure 4.12 which illustrate the content to be stored in the QR code and then hash validation at the server (respectively).

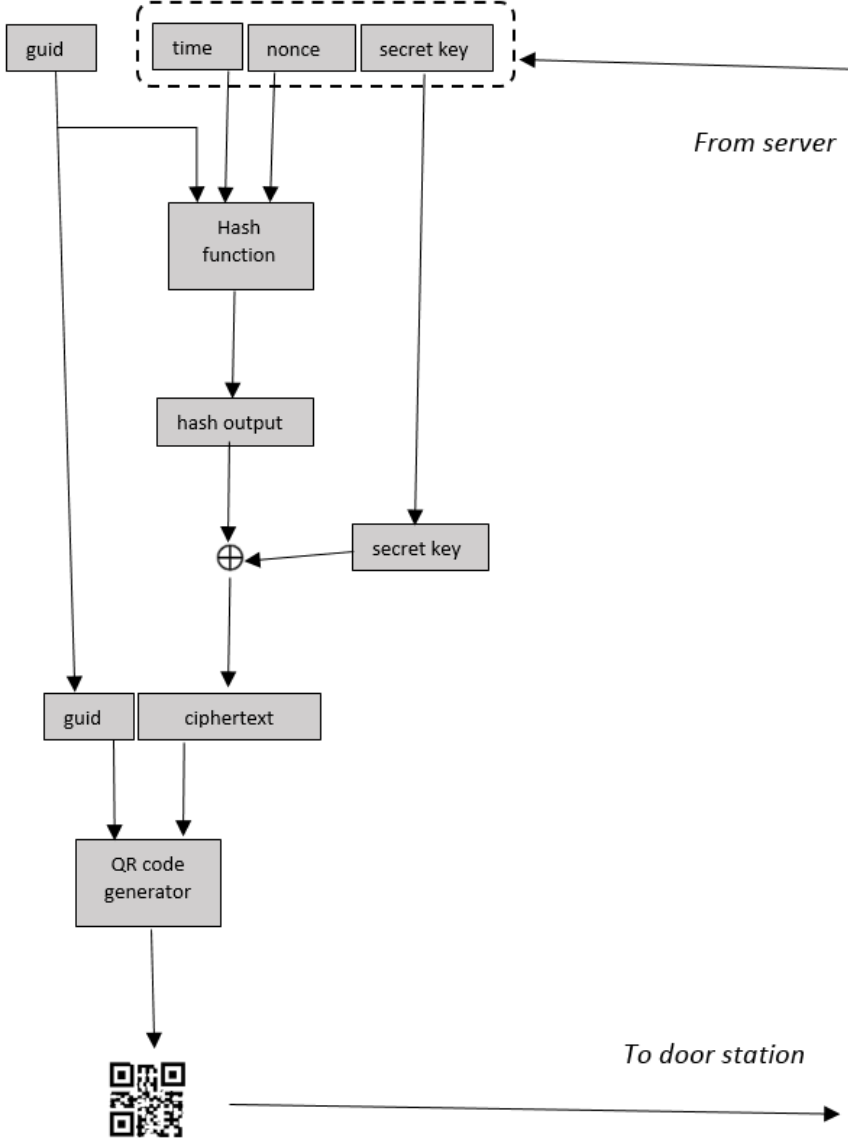
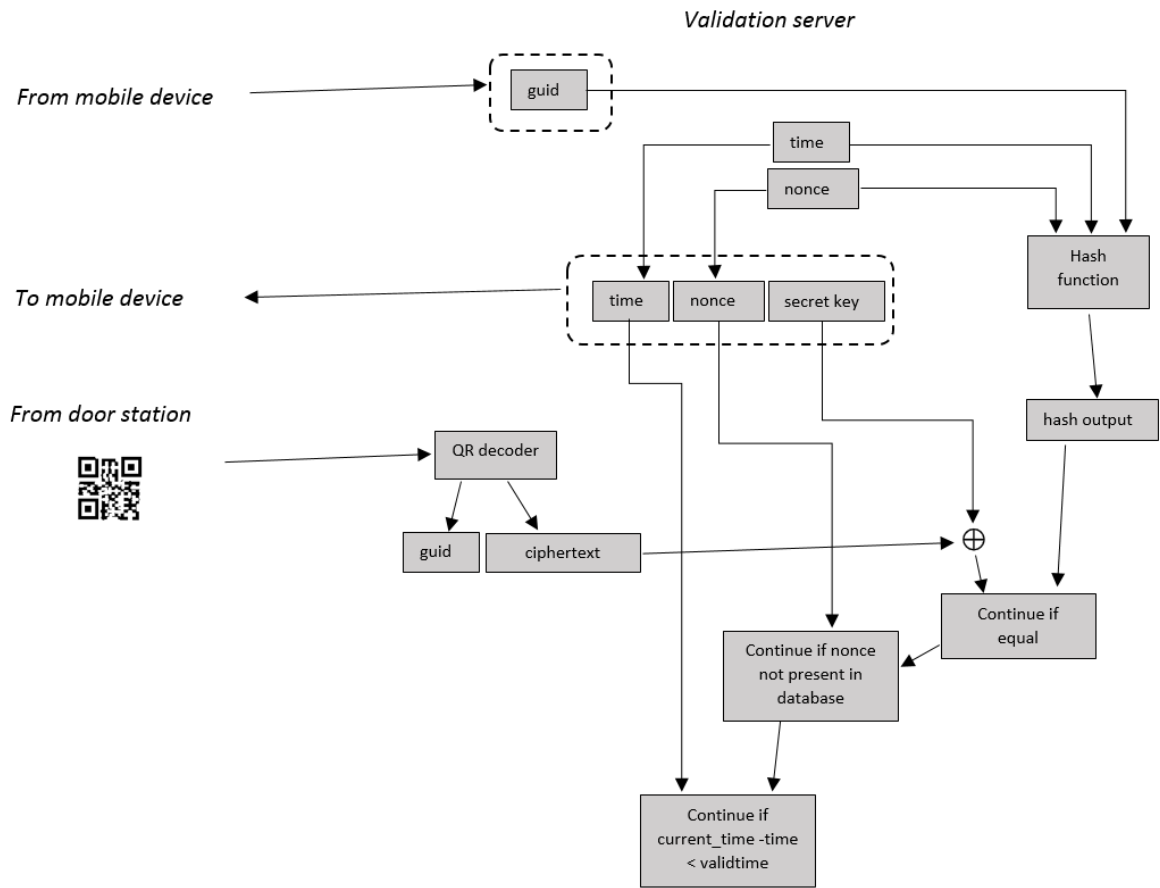


Figure 4.11: QR code generation in mobile device

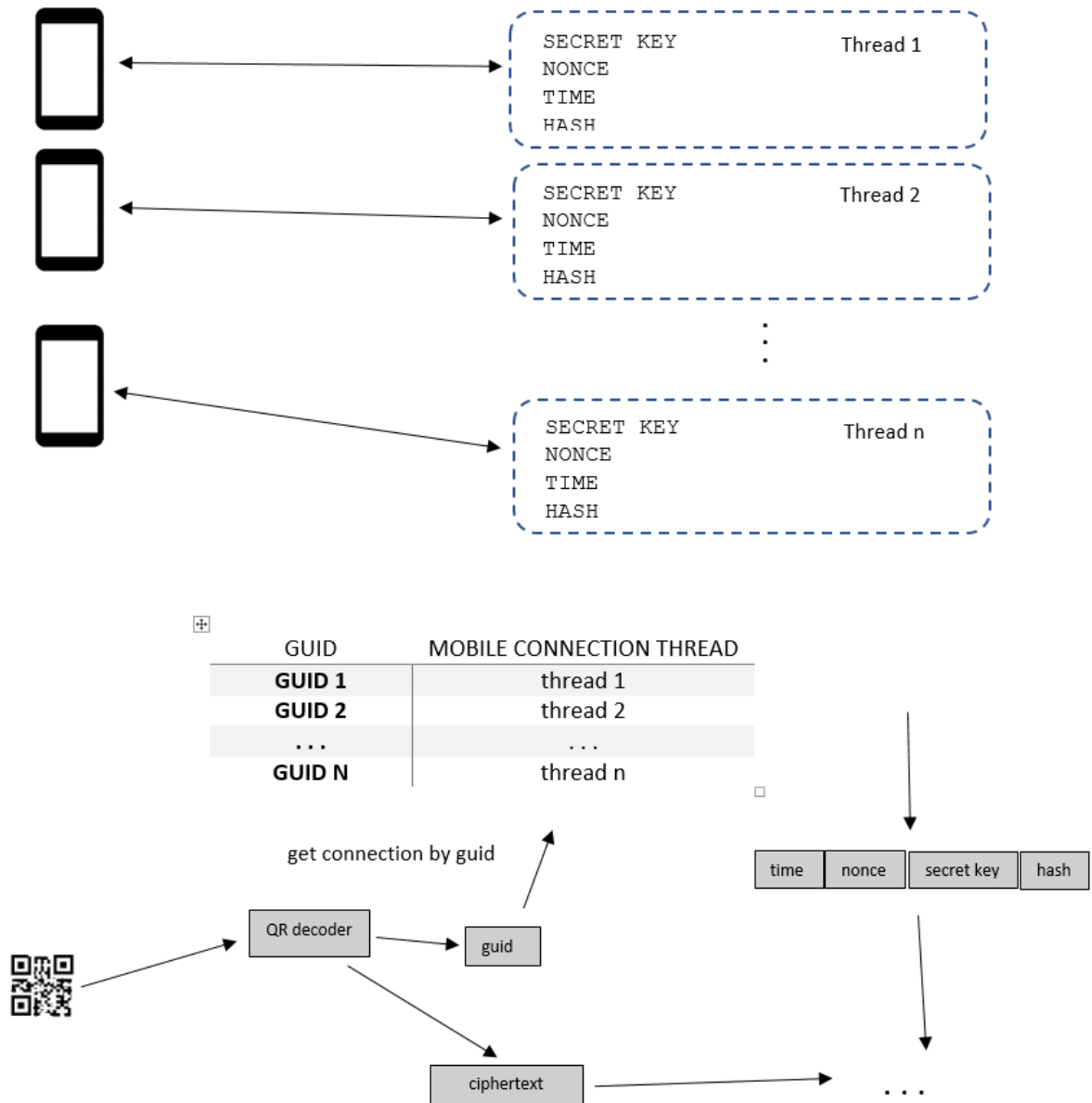


**Figure 4.12: QR code validation in server, by the hash values**

Since the server now has the decoded content from the QR decoder server, the validation server can begin its validation steps. First, it will decrypt the content which is done in the same way as encrypting, since XOR was used together with the OTP. Therefore, decrypting procedure requires only the secret key ( $KEY_{SECRET}$  in Figure 4.10) which can be obtained by the active mobile connection. During the creation of this secret key, it was stored in the connection thread, for easily access. Finding the correct connection thread is trivial since the QR code's content also includes the mobile device's guid, which is mapped to a specific mobile's connection. This is shown in Figure 4.13.

The decryption outputs a hash value, which was created earlier by the mobile device from the same parameters that the server used when it created its hash value. The first step in verification is to ensure these hash values are equal, because if they are not equal, something has occurred with the QR code between mobile device and server. If they are equal, then the next steps will utilize the nonce and timestamp parameters created and stored in the mobile connection thread.





**Figure 4.13: Mobile connection management with validation parameters**

Verifying that the nonce is valid, is done by checking it does not exist in the nonces-table in database for this guid. When the verification completes successfully, the current nonces is added to the database, leading to this QR code becoming invalid from this point on.

If the nonce was considered valid, then the time parameter is verified to be within the configured valid time interval for a QR code. Since the timestamp is saved in the mobile connection thread among the other parameters, there is no need for the mobile device to put its timestamp on the code. This avoids the problem that occurred with an earlier design that required accurate time synchronization to perform validation. This turned out to be problematic since there could be time differences among server and devices. This earlier design is shown in Figure 4.14. Another option would be to utilize a NTP server to ensure synchronization. In choosing the alternative design the time parameter is redundant on the mobile device, since it could be provided by the server and baked directly into the QR code.

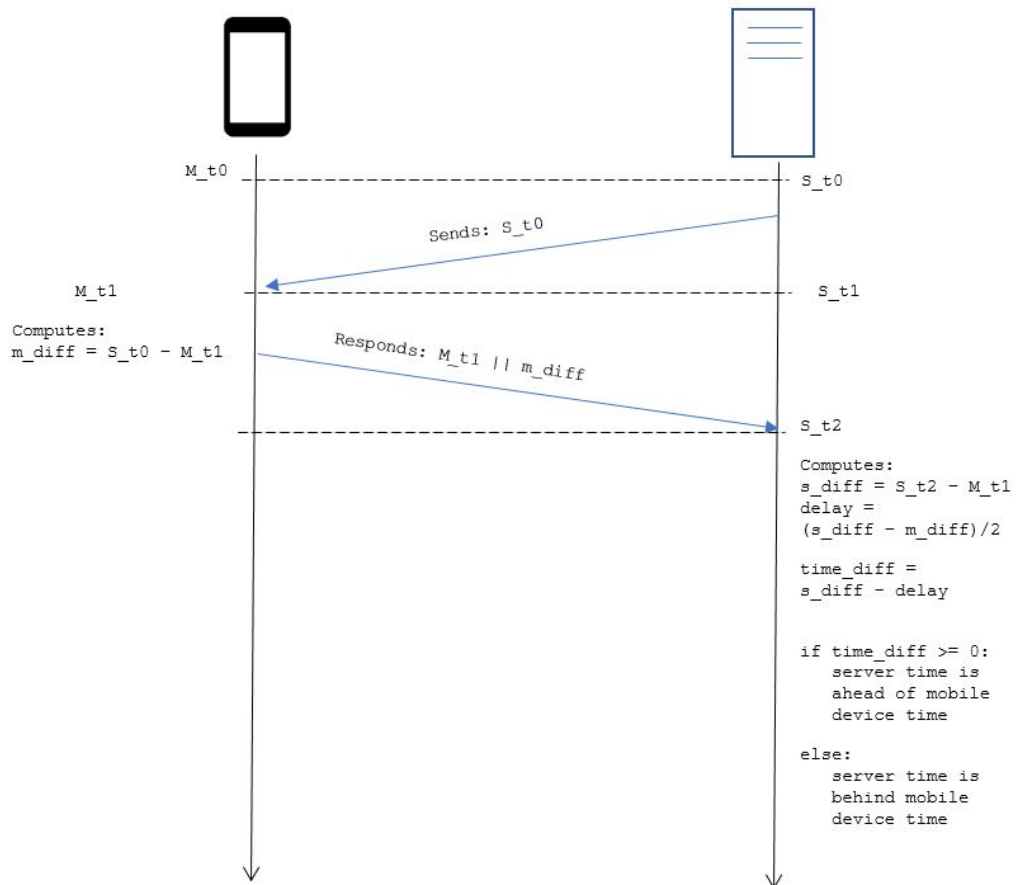


Figure 4.14: Time synchronization between mobile device and server

## 4.6 Bootstrapping service

A few different methods were explored to initiate the system (i.e., bootstrap the system). When the QR decoder was placed inside the door station, it could be started and then run as a background process independent of the Go – application which communicates with the server. This means that starting this Go application or setting it up as another background process would bootstrap the door station.

When the QR decoding was migrated to the backend part of the system, the door station still uses the Go application which could be initiated in the same way as previously. However, when a user arrives at the door station there is a difference in how a session is initiated. With the QR decoder inside a door station, it could simply detect the presence of a QR code itself and initiate a session. From a flexibility point of view, this is a great solution, but the door station always remains in the state of scanning for codes. In this case, any QR code, even those not related to the service, will be scanned and obviously their validation will fail, but still all these codes will be sent into the system. Moreover, capturing unnecessary information (that may be sensitive) is not good from an ethical point of view. Therefore, the best case would be that the arriving user initiates a session, which can be done by exploiting the hardware of the door station. In particular, the door station has a built-in button, which can be configured to cause the door station to take a snapshot. Programming the Go-application on door station to have a listener

for these images and forwarding them to the server is straight forward – and is part of the proposed solution.

The validation server can be constructed as a .jar file to be run directly from a Microsoft Windows console. This server can be configured to run on any server, but for this project, it will be run on the Windows laptop for simplicity.

Since the system is requires only registered users utilize the service, the mobile device's application must be used to first register, then receive the TLS certificate (described earlier) and store this user's information in the server's database, then connect to the server for challenge and generate a QR code.



## 5 Results and Analysis

In this chapter, the results are presented and analysed from the various experiments that were conducted. The metrics used for the analysis will be presented in Section 5.1 and the major results in Section 5.2. The analysis with regards to reliability will be presented in Section 5.3 and with regards to validity in Section 5.4. Finally, a discussion is presented in Section 5.5.

### 5.1 Metrics

In this section, the metrics will be described for all the areas for which results will be analysed.

#### 5.1.1 QR code

The metrics for analysing the QR code are size (in pixels), content length (in characters), and rotation (angles in degrees). All these metrics will be analysed with regard to *readability*, that is if a QR code is detectable and decoded successfully by the system (where the QR decoder is located in backend and not a door station).

##### 5.1.1.1 Device

The following results with regards to the mobile device refer to the Nokia 3 as the device. Although the HTC One mobile phone was also used in some cases, the main results in this chapter are with the Nokia 3 as mobile device.

##### 5.1.1.2 Size

The size of QR codes was measured in terms of pixels during the experiments. This is basically what is fed to the generator which creates and displays the QR code on the device's screen. Since the device had a screen width (which naturally is the minimum value of width versus height), of 720 pixels, the maximum tested size was 700 pixels.

##### 5.1.1.3 Content length

The content which is fed to the QR code generator (as well as size) was a factor which could affect the readability of the QR code and thus is an important metric to consider in the analysis. The minimum content length was chosen to be 50 characters because anything below that was unlikely to be sent from the device to the validation server. Similarly, the maximum content length was chosen to be 1000 characters, since the amount of data that was expecting to be stored was expected to be much less than 1000 characters.

##### 5.1.1.4 Rotation

The rotation space can be divided into several planes which are references for the rotation angles. Since rotation is measured between the mobile device and the door station, the door station was considered as the fixed unit inside the space which has X, Y, and Z axes. Therefore, rotations are considered to occur around these axes.

#### **5.1.1.5 Distances**

Analysing the distance from the door station at which the QR code is readable, can be difficult in a three-dimensional space. As with the rotation-analysis, this will be analysed with regards to X, Y, and Z axes.

### **5.1.2 Environment**

This section describes the metrics which correspond to lighting conditions. These metrics were often tested in combination with other metrics, in order to understand how light conditions, affect various other parameters.

#### **5.1.2.1 Ambient light**

Ambient light is the light around the actual measurement area, which in this case is everything around the door station and mobile device. Since these light conditions could potentially vary depending on the situation (i.e. outdoor light conditions), it was important to understand how this affects the area around the door station and mobile device.

#### **5.1.2.2 Screen backlight**

The screen backlight refers to the mobile device's screen which most likely has a variable brightness setting. Similarly, to ambient light, this brightness might affect the readability of the codes.

### **5.1.3 Encryption**

This section describes those metrics regarding the encryption parameters.

#### **5.1.3.1 Key size**

One of the major factors that contributes to the differences in the performance of different encryption schemes is the key size (in bits) of both the encryption and decryption keys. Different encryption techniques use different sized keys; hence this is an interesting metric to measure in the different experiments.

#### **5.1.3.2 Data**

The data refers to the content that is to be encrypted or decrypted. The different encryption techniques and implementations will treat the data contents differently, hence properties of the data (specifically its size in number of bits) will be analysed.

#### **5.1.3.3 Time**

One of the most important metrics to analyse is time. Nobody wants to wait for a long time at a door while the system takes a long time to authenticate you. Analysing the time required to perform encryption/decryption when using different techniques is therefore relevant.

## 5.2 Major results

This section describes and visualizes the result of every experiment that was described in Section 3.4. The metrics, described in previous section are referred to and shown in various figures. The results are divided into QR codes, encryption, and environment results and shown in this order.

### 5.2.1 QR code results

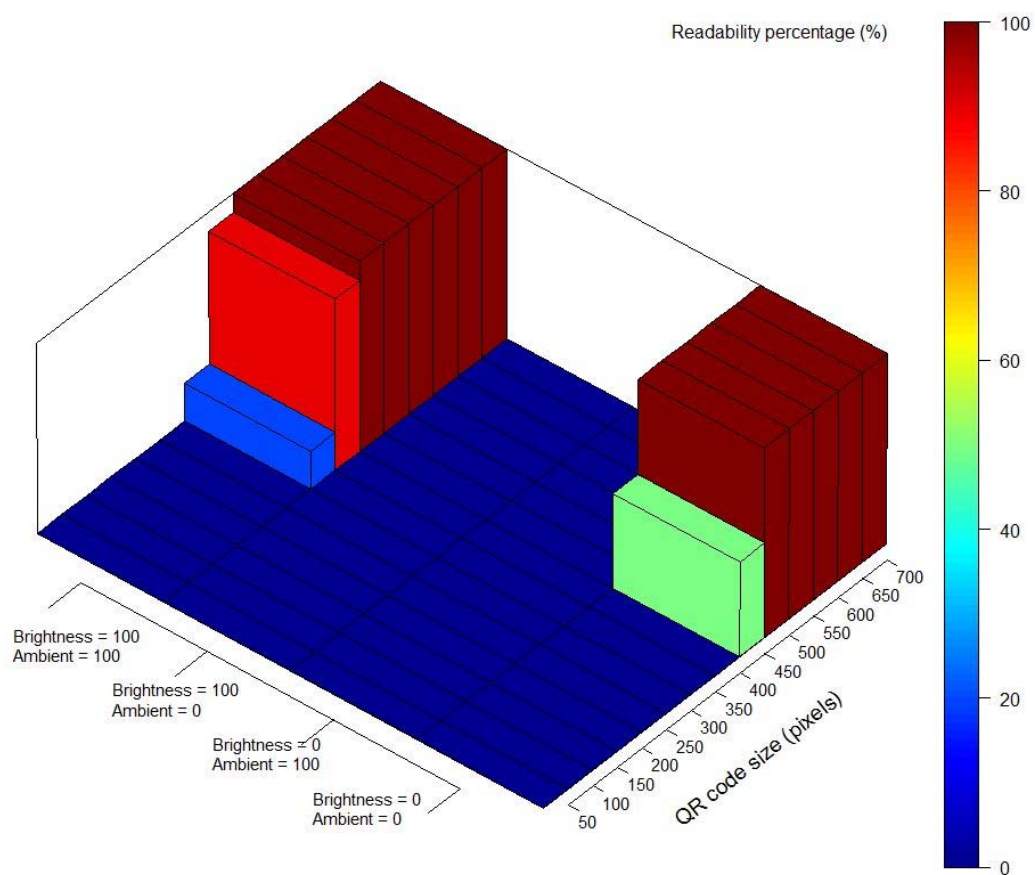
Section 5.1.1.2 describes the result of testing different QR code sizes, followed by the content length results and finally the rotation results.

#### 5.2.1.1 QR code size

Figure 5.1 shows result of experiment with various sizes of QR codes in units of pixels displayed on the mobile device. This is done under four different lighting conditions to see how these conditions affect the readability of different sized QR codes. A value of 100 denotes a very bright light\*, while 0 was darkest it could get in the test lab environment. A bright ambient light condition had every surrounding light lit. Similarly, the brightness value is given as a percentage of what can be set as the brightness of the mobile device.

---

\* This corresponds to a light source illuminating from behind of the mobile device at 800 lux per m<sup>2</sup>. Conversion equation of Lumens to Lux per m<sup>2</sup> [42] and table of standard light bulbs to Lumen values [43] where in this case a 60 Watt light bulb was used.

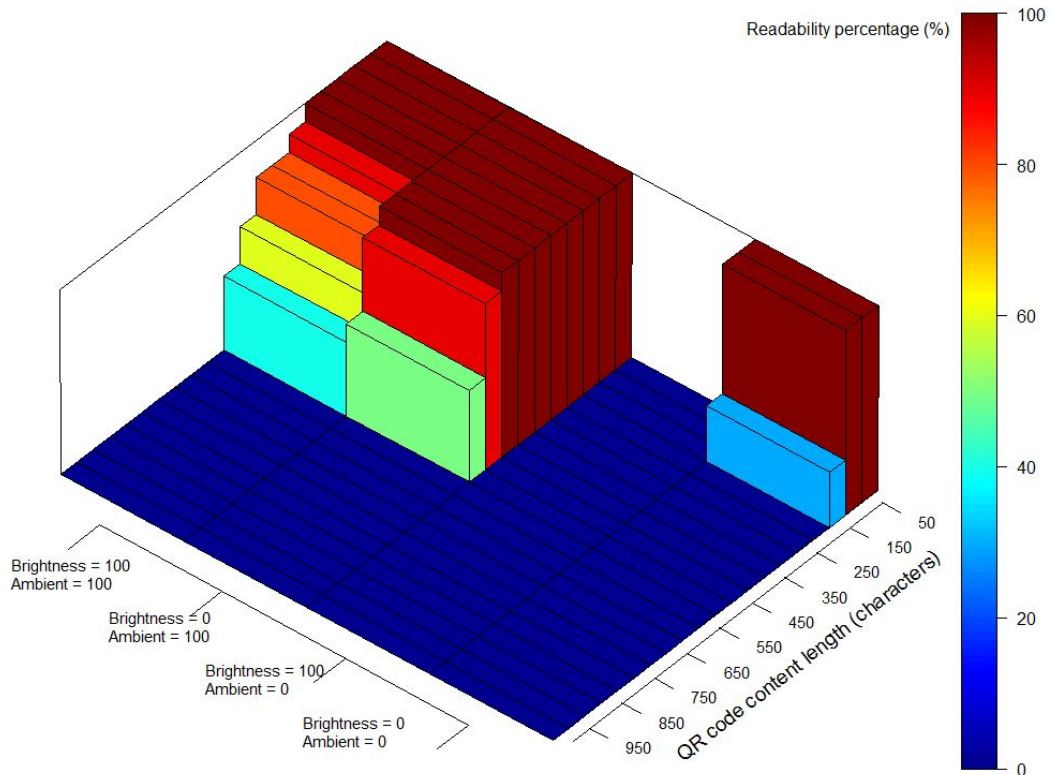


**Figure 5.1:** Readability of QR codes with different sizes in pixels, during four different light conditions.



### 5.2.1.2 Content length

Figure 5.2 shows a plot of readability in relation to QR code content lengths as well as different lighting conditions. This plot shows how longer content length generates much higher complexity in the generated image and increases the difficulty of reading the code correctly. As with the earlier size results, this figure shows the results of experiments under different lighting conditions.

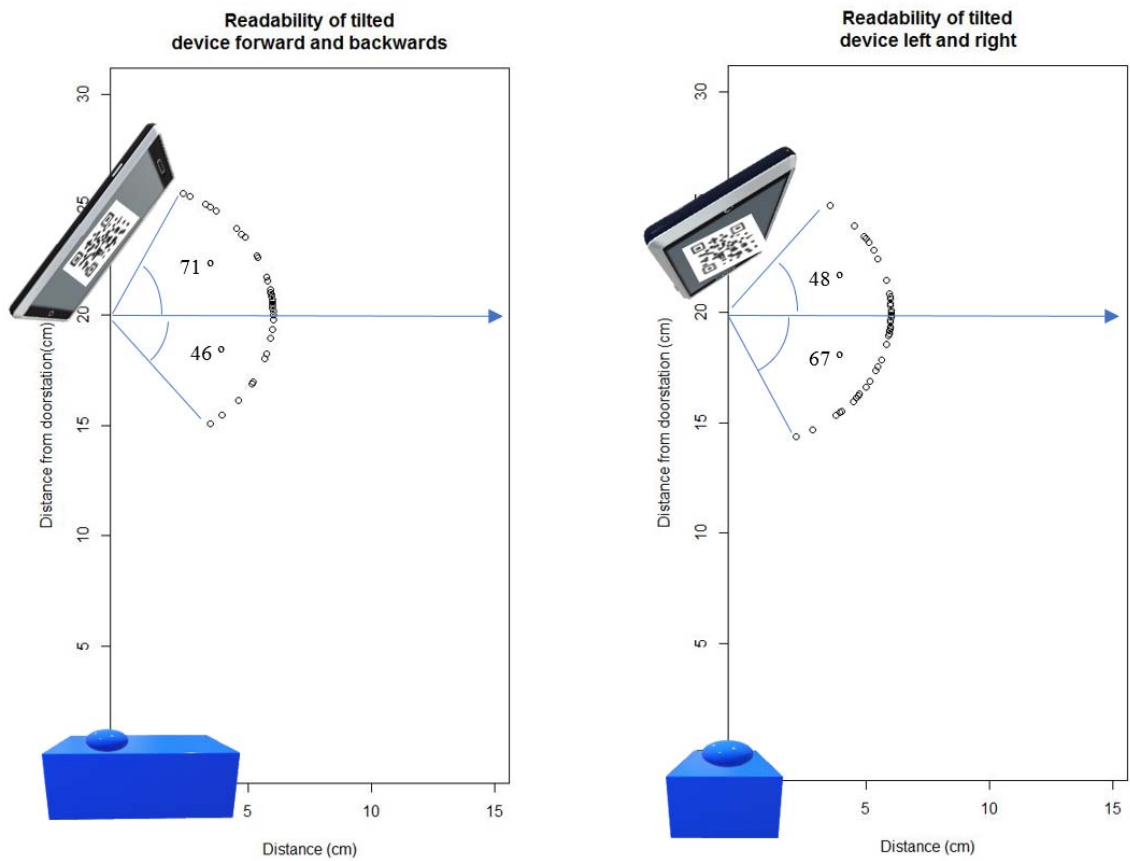


**Figure 5.2:** Readability of QR codes with different stored content lengths, during four different light conditions

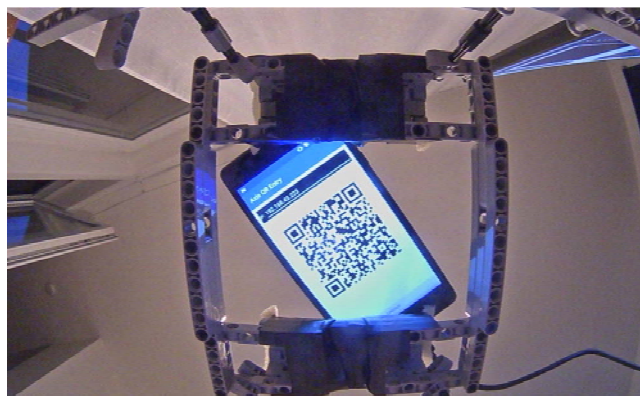
### 5.2.1.3 Rotation

The result of measuring angles is shown in Figure 5.3. The total number of measurements in this experiment was 100 data points, where 59 of these gave readable results of the QR code. The device was tilted both left/right and forward/backward in different ways which gave results: maximum left angle of 48 degrees and maximum right angle of 67 degrees. Tilting it forward gave the maximum value of the four available angles, 71 degrees, whereas backwards gave the minimum value, 46 degrees.

While rotating around the Z-axis the code was readable at every point of a 360-degree rotation. Figure 5.4 shows the snapshot at one point where the rotation around Z-axis was around 45 degrees.



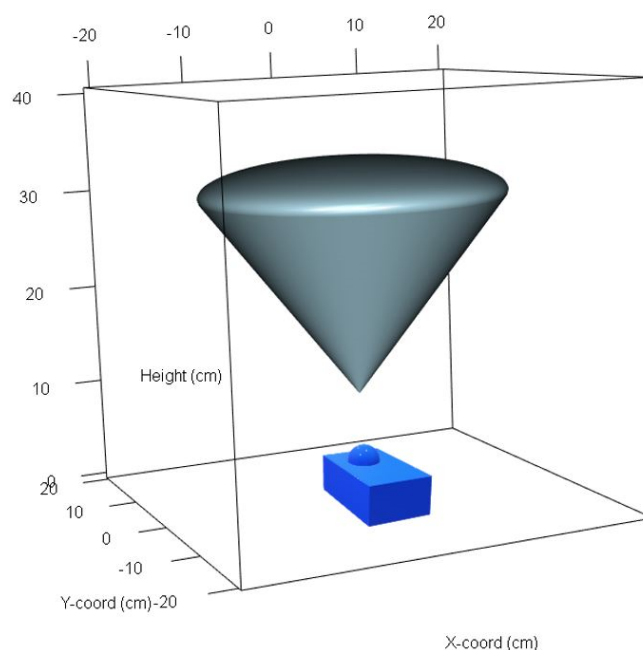
**Figure 5.3:** Readability of QR code on a tilted device. The plot on the left shows when device is tilted forward and backward along the doorstation while the right plot show tilting left and right.



**Figure 5.4:** Rotation along the Z-axis of the door station, this yielded a successful reading of the code

### 5.2.1.4 Distances

The result from measuring distances at which the QR code is detectable and readable, is shown in Figure 5.5. This plot is built upon measured values of 5 cm increments on each axis. This figure illustrates that the readable locations of the QR code are within this cone. Note that there is a minimal distance of ~10 cm and a maximal distances of ~30 cm.



**Figure 5.5:** Detectable and readable area around the door station, based on coarse measured values


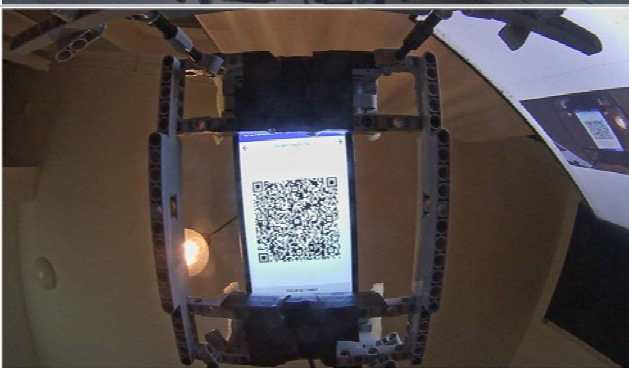
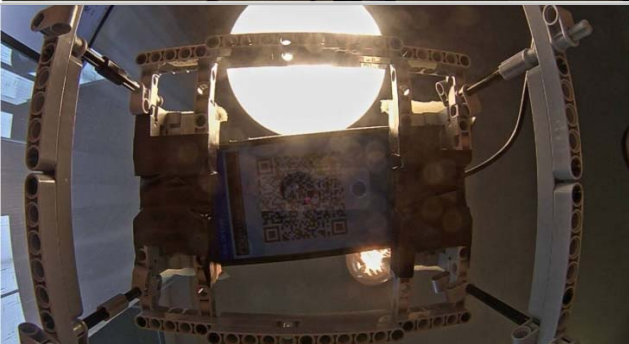
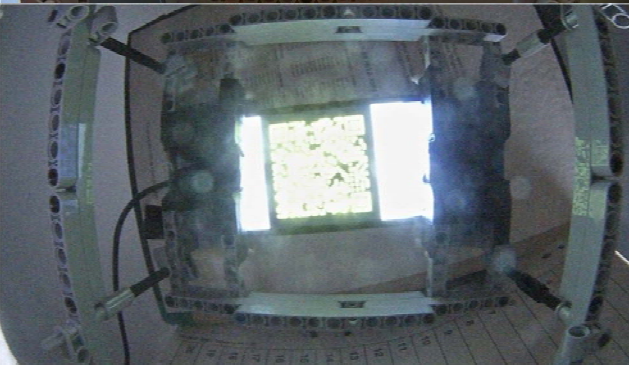
### 5.2.1.5 Colour results

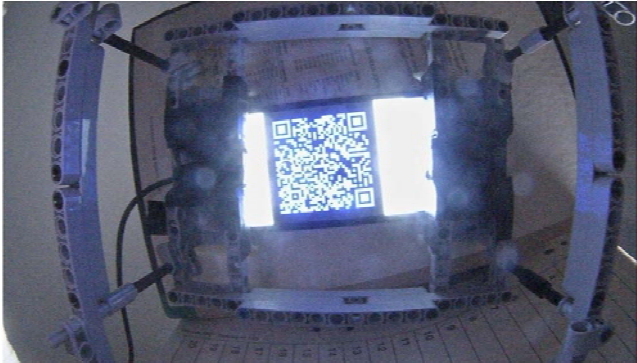

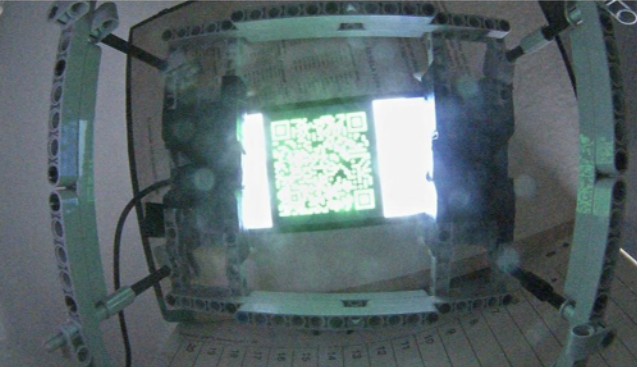

The results of testing with different colours are shown in Table 5-1, with image indexes 4 to 12. Only one of these gave a readable result, which was the last one of these images (i.e., with a blue foreground and white-grey background).





## 5.2.2 Environment


The light conditions gave a variety of results as was seen in previous metrics, but to give a visualization of what these results look like to the human eye see image index 1 to 3 in Table 5-1. These images will help the reader of understand the effect of different lighting conditions.

**Table 5-1: Image snapshots from various experiments, taken from door station viewpoint**

Index	Image description	Image
1	QR code with content length of 50 characters stored in it. It is shown with 100% backlight before the camera in an environment with 0% ambient light. QR reading resulted in failure.	
2	QR code with content length of 250 characters in an environment of almost highest ambient light possible in the environment. The decoding resulted in success.	
3	QR code with content of 85 characters. Ambient lighting was high while screen backlight was at minimum.	
4	Low ambient light with high screen backlight showing a QR code with yellow foreground and black background.	

5	Low ambient light with high screen backlight showing a QR code with blue foreground and black background.	
6	Low ambient light with high screen backlight showing a QR code with dark grey foreground and black background.	
7	Low ambient light with high screen backlight showing a QR code with green foreground and black background.	
8	Low ambient light with high screen backlight showing a QR code with purple foreground and black background.	

9	<p>Low ambient light with high screen backlight showing a QR code with red foreground and black background.</p>	
10	<p>Attempt of removing any screen light to enhance the QR code, in this case by having a complete black background while the foreground of QR code is dark but not black colour. The screen backlight is set to minimum here which shows the reflection of doorstation illuminating led circle.</p>	
11	<p>The background colour is set to dark grey here but because screen backlight is set to maximum, the background appears more white than grey.</p>	
12	<p>The standard setting of white and black colour of QR code is set but in a black background to minimize any unnecessary light, while still allowing the <i>quiet zone</i> of the QR code. Here with a QR content of 43 characters</p>	

13	Same situation as previous but with QR content of 85 characters.	 A photograph showing a QR code displayed on a small screen. The screen is mounted on a dark, complex mechanical structure, possibly a camera rig or a sensor assembly. The QR code is the central focus, appearing as a bright white square with black pixels on a dark background. The surrounding environment is dimly lit, with some structural elements visible as dark silhouettes.
----	--	---

### 5.2.3 Encryption results

In these experiments we first test of how different key sizes for encryption affect the input and output data. These experiments were conducted for asymmetric encryption and symmetric encryption. Furthermore, these same encryption techniques were tested regarding encryption and decryption time. When measuring the encryption and decryption time, each encryption/decryption procedure was repeated 10 times and the result is based on values calculated from the 75<sup>th</sup> percentile of these 10 iterations. This was an attempt to filter out infrequent outliers.

The execution time plots are shown on a logarithmic scale since the difference in performance for some parameter values were extremely large, hence they were unsuitable for plotting on a linear scale. This means also that the best fitted equations and  $R^2$  values that are calculated and shown for the plots, can be visualized a bit differently than if they would be shown in linear-based plots. All the  $R^2$  values are very close to 1, meaning the standard error values are very low, hence the equations are fitted very well to the data points. However, since the plots are based on logarithmic data, the deviations between data points and fitted equation lines might appear to be more than what is calculated and shown by the coefficients and the  $R^2$  values. Although the graphs are shown in logarithmic scale, all the equations have coefficients of linear type, which means they are in the form of  $y = \mathbf{kx} + \mathbf{m}$ , where  $y$  denotes the execution time,  $k$  and  $m$  denotes the calculated coefficients and  $x$  denotes the data input length.

#### 5.2.3.1 Asymmetric keys

Since different padding options can be applied in conjunction with the RSA algorithm, it is important to first know how this will affect the input data. Table 5-2 describes the maximum lengths of data that can be encrypted in a block. More generally, the equation to get these values is given by Equation 1 in Appendix B. For Optimal Asymmetric Encryption Padding (OAEP) paddings the exact values are 42 and 66 bytes respectively. For OAEP paddings, when the result from the equations were below 0 (meaning that the padding would take more space than the input), these cases are not relevant, hence they are denoted by a dash in the table.

**Table 5-2: Maximum data block lengths in bytes of what can be encrypted with the chosen padding and the used key size**

<b>Key size (bits)</b>	<b>PKCS1Padding (bytes)</b>	<b>OAEP with SHA1 (bytes)</b>	<b>OAEP with SHA256 (bytes)</b>
128	5	-	-
256	21	-	-
512	53	22	-
1024	117	86	62
2048	245	214	190
4096	501	470	446

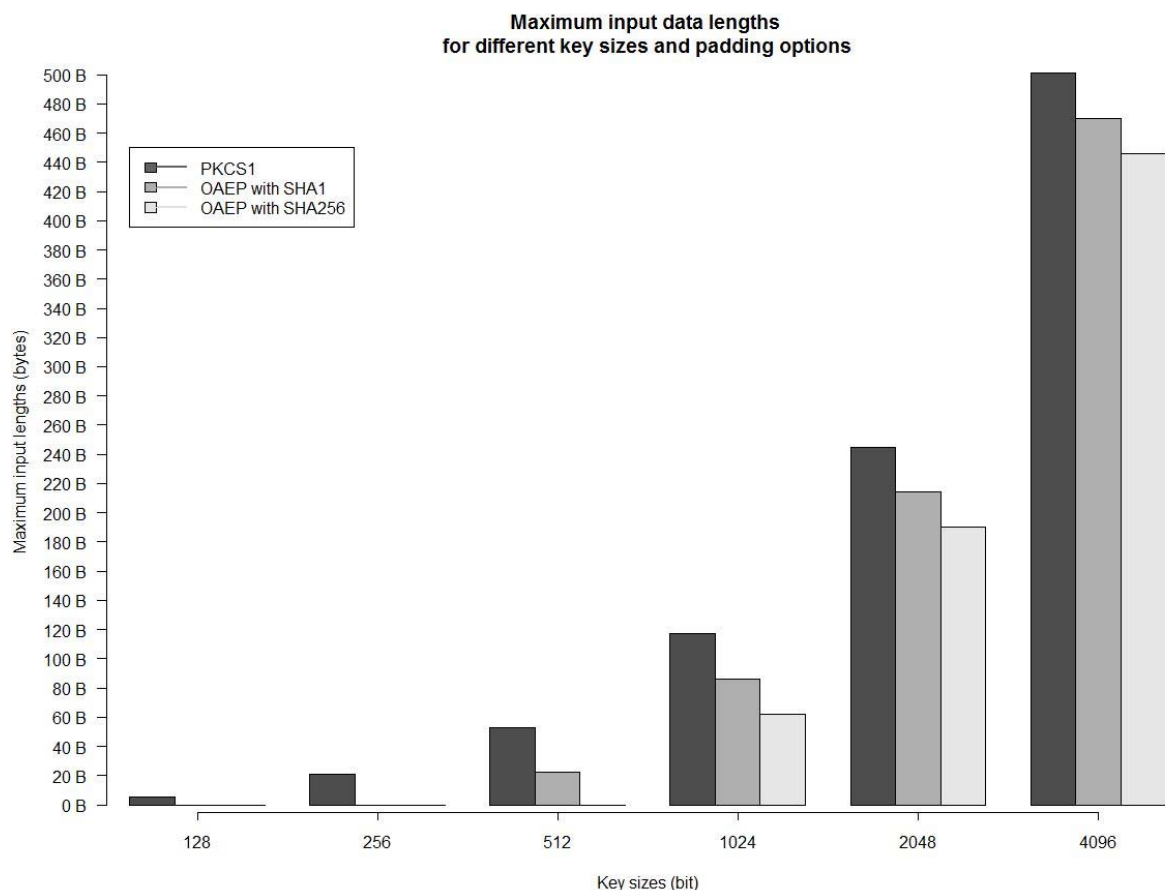


Table 5-3 shows some trivially simple results because *content length (bytes) = key size (bits) / 8* but clarifies for some key sizes which are compatible with the three padding options. Since there are no values for two respectively three of the OAEP padding options, the corresponding elements in this table is denoted by a dash as well.

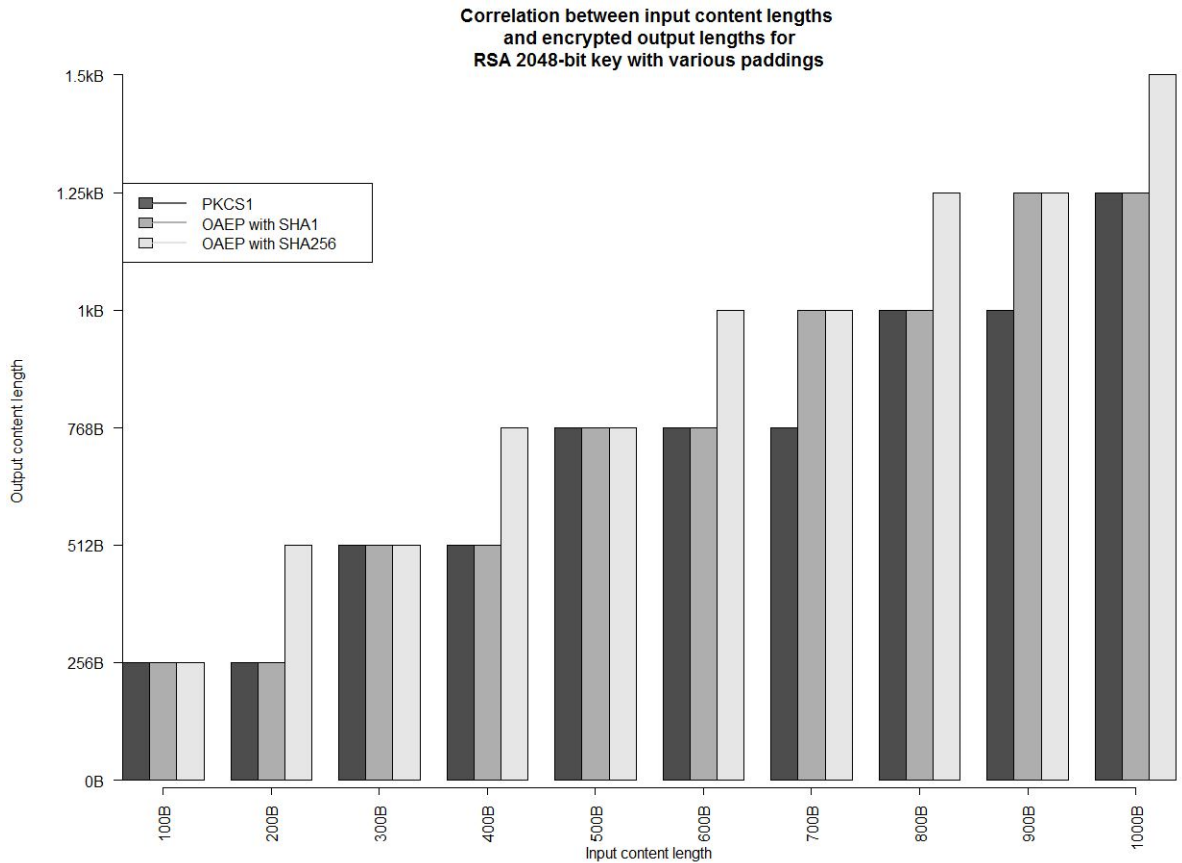
**Table 5-3: Resulted encrypted content lengths in bytes, for chosen paddings and key sizes**

Key size (bits)	PKCS1Padding (bytes)	OAEP with SHA1 (bytes)	OAEP with SHA256 (bytes)
128	16	-	-
256	32	-	-
512	64	64	-
1024	128	128	128
2048	256	256	256
4096	512	512	512

Figure 5.6 shows how the key sizes affect the input and output data in the asymmetric encryption tests.

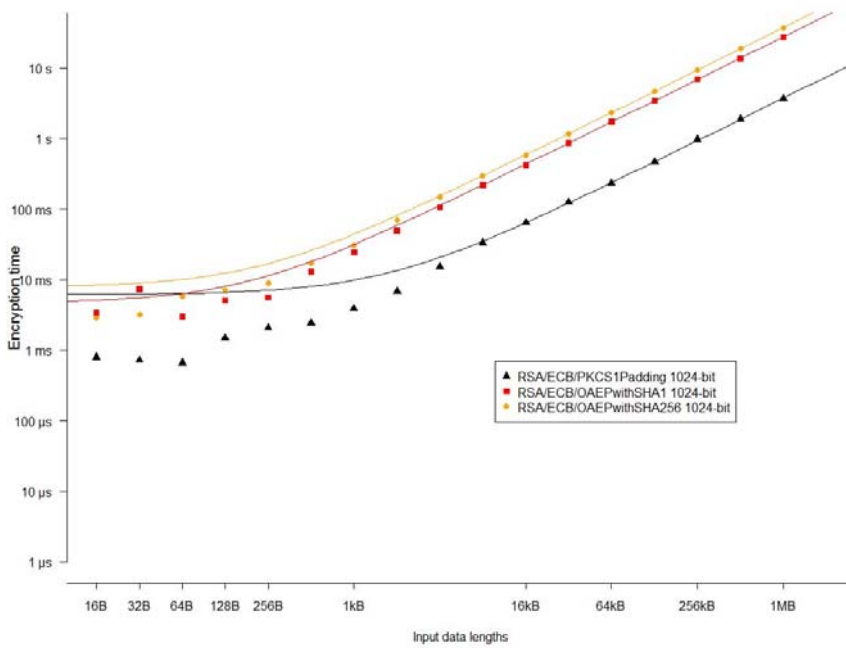


**Figure 5.6: RSA encryption which shows correlation of maximum input data with different key sizes and padding options.**



**Figure 5.7: Correlation between plaintext input length and encrypted output length for RSA encryption with 2048-bit key.**

Figure 5.8 shows the encryption time for an RSA algorithm with a key size of 1024-bits with different padding options and data lengths. These figures have best fit lines with their corresponding coefficients and  $R^2$  values. These fits can give somewhat strange results when applied to the whole dataset. The best fitted line which corresponds to PKCS1 padding (black) is clearly not fitted to the first data points in this image because it adapts to every point in the data set. Therefore, to concentrate on the first data points, Figure 5.9 shows a subset of the data up to 1 kilobytes where the best fit line fits this subset. Similarly, Figure 5.10 shows RSA decryption with the same parameters with a detailed subset of the plot in Figure 5.11. However, because decryption takes longer, the test was only conducted with content lengths up to 256 kilobytes.



**PKCS1 Coefficient**  
 $\text{time}(\mu\text{s}) = 6147 + 3.533 \times \text{Bytes}$

**PKCS1 R<sup>2</sup> value**  
 0.9996

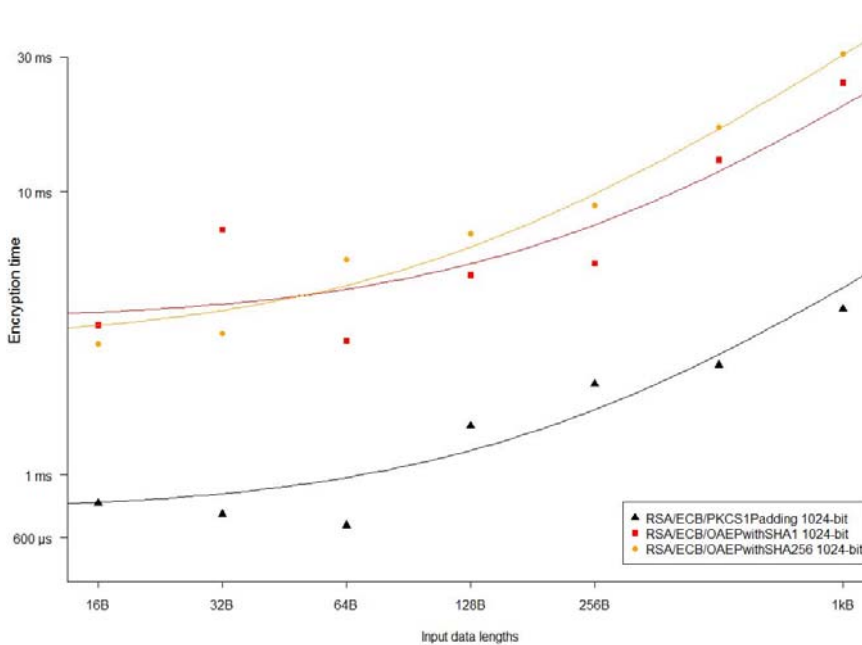
**OAEP SHA1 Coefficient**  
 $\text{time}(\mu\text{s}) = 4678 + 26.13 \times \text{Bytes}$

**OAEP SHA1 R<sup>2</sup> value**  
 1

**OAEP SHA256 Coefficient**  
 $\text{time}(\mu\text{s}) = 7810 + 34.43 \times \text{Bytes}$

**OAEP SHA256 R<sup>2</sup> value**  
 1

**Figure 5.8:** RSA encryption time with best fit lines with regards to whole data set. Note equation times are in microseconds.



**PKCS1 Coefficient**  
 $\text{time}(\mu\text{s}) = 737.6 + 3.761 \times \text{Bytes}$

**PKCS1 R<sup>2</sup> value**  
 0.865

**OAEP SHA1 Coefficient**  
 $\text{time}(\mu\text{s}) = 3490 + 16.27 \times \text{Bytes}$

**OAEP SHA1 R<sup>2</sup> value**  
 0.708

**OAEP SHA256 Coefficient**  
 $\text{time}(\mu\text{s}) = 2948 + 26.88 \times \text{Bytes}$

**OAEP SHA256 R<sup>2</sup> value**  
 0.976

**Figure 5.9:** RSA encryption time with best fit lines of subset of data. Input length up to 1 kilobytes. Note equation times are in microseconds.

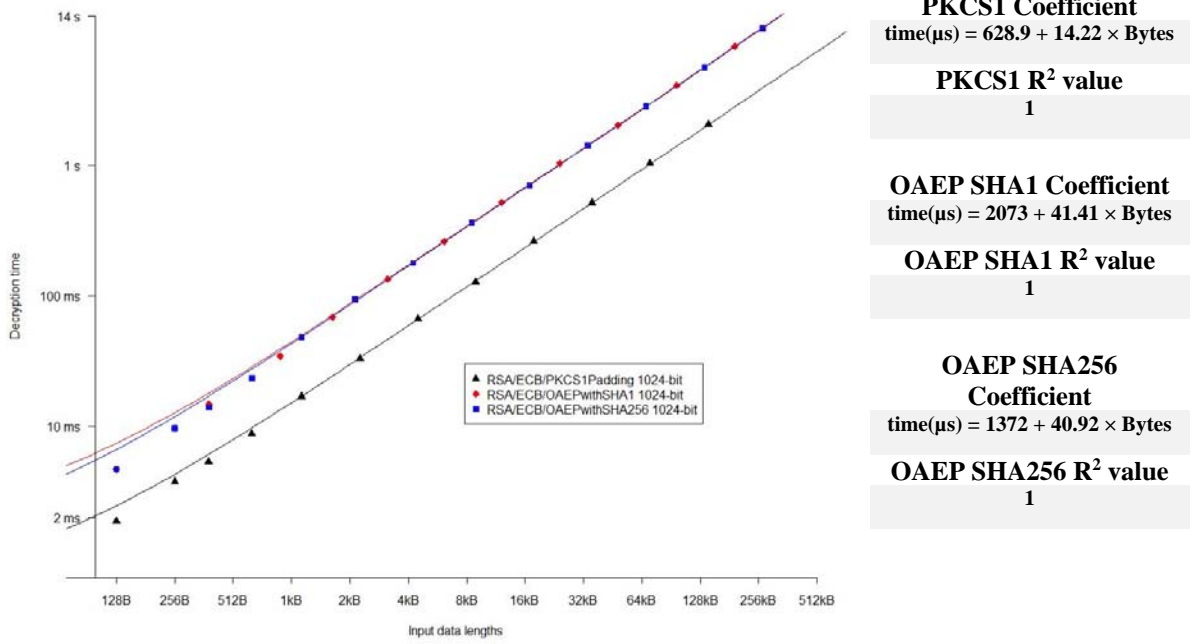


Figure 5.10: RSA decryption time with best fit lines for whole data set. Note equation times are in microseconds.

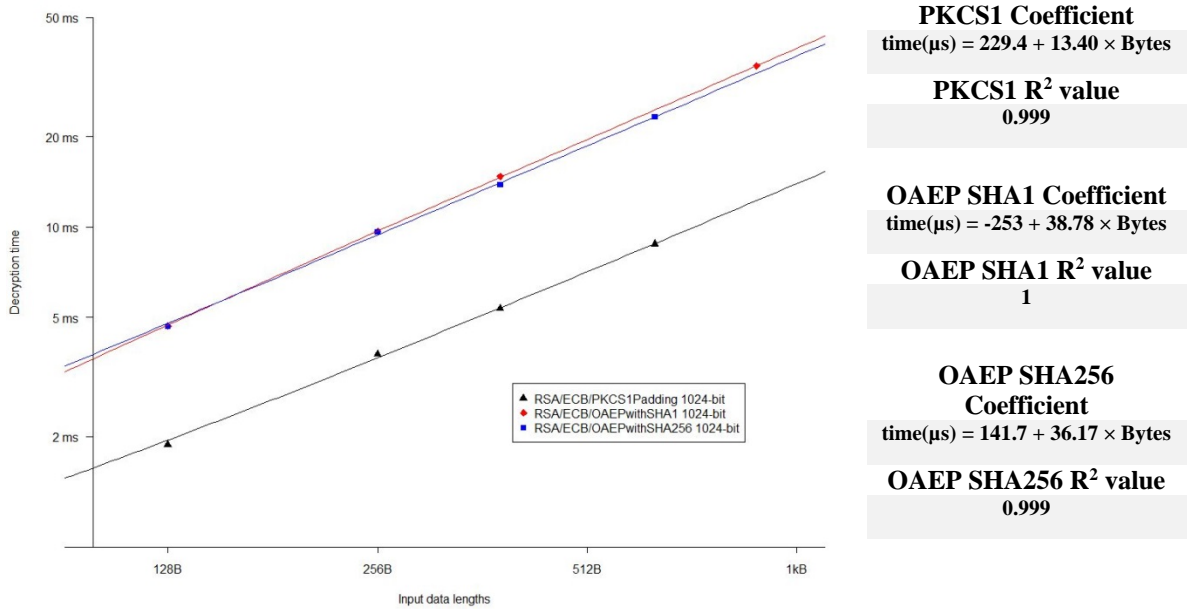
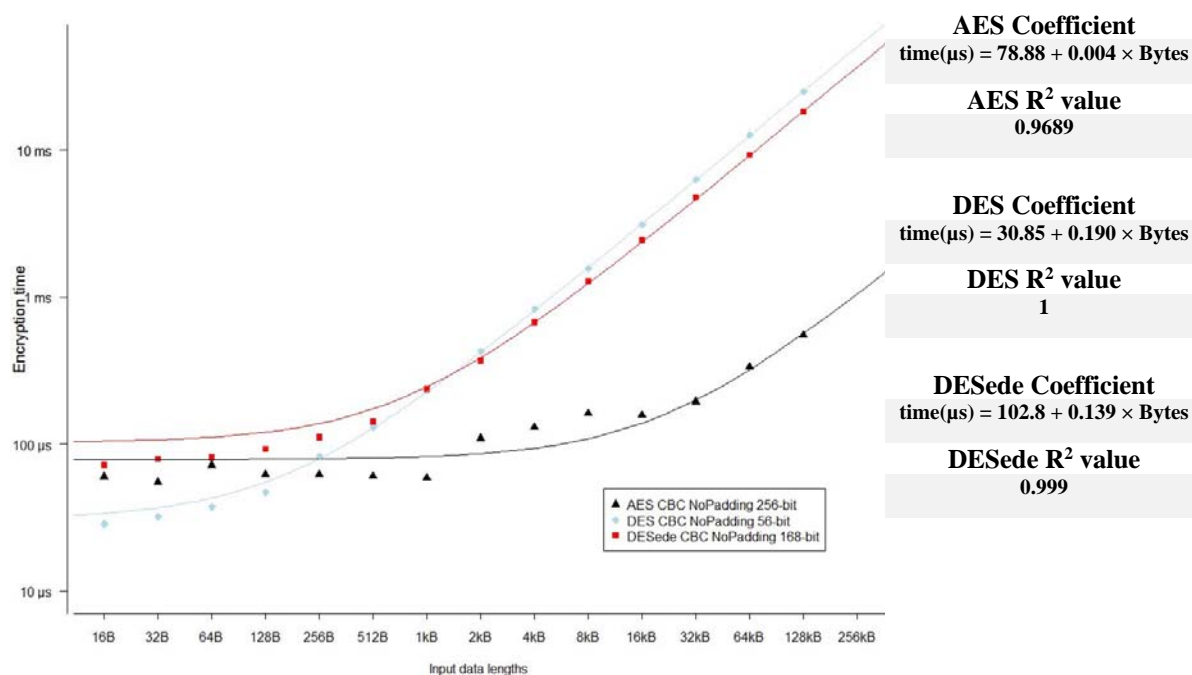


Figure 5.11: RSA decryption time with best fit lines, for data subset up to 1 kilobyte. Note equation times are in microseconds.

### 5.2.3.2 Symmetric encryption

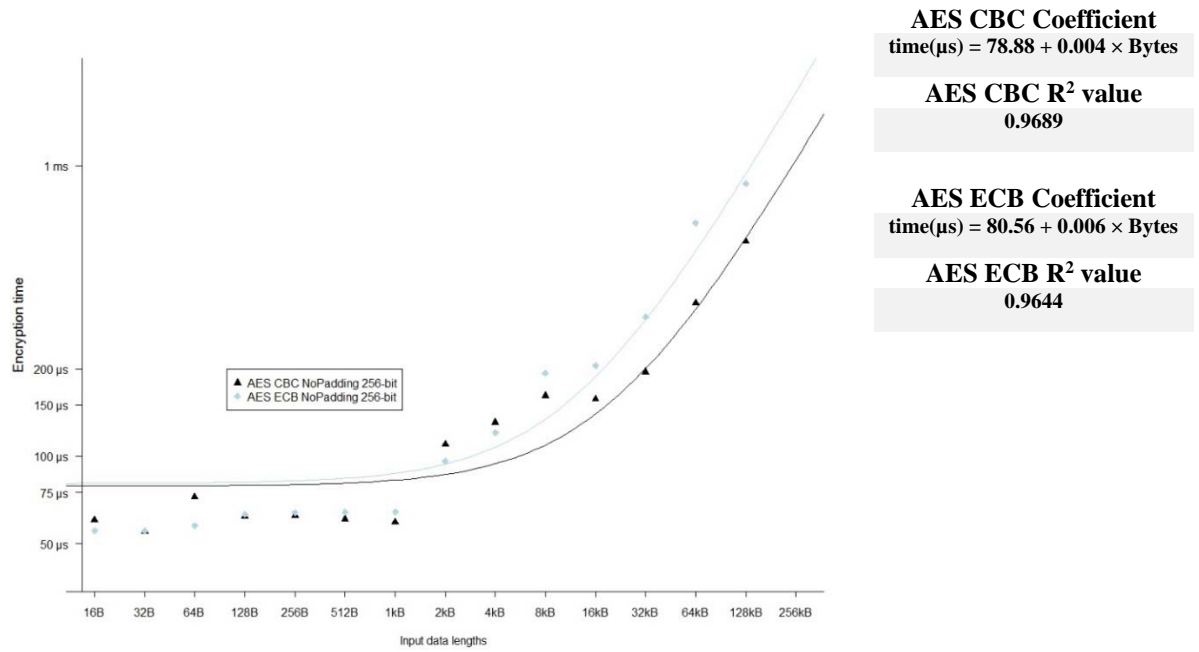
Figure 5.12 shows a comparison of the three symmetric key algorithms AES, DES, and 3DES for an encryption experiment with varied input data lengths, with regards to the encryption time. The best fit equations with coefficients show how the encryption times can be related to the input data. These equations have quite low standard error values which results in high  $R^2$  values.



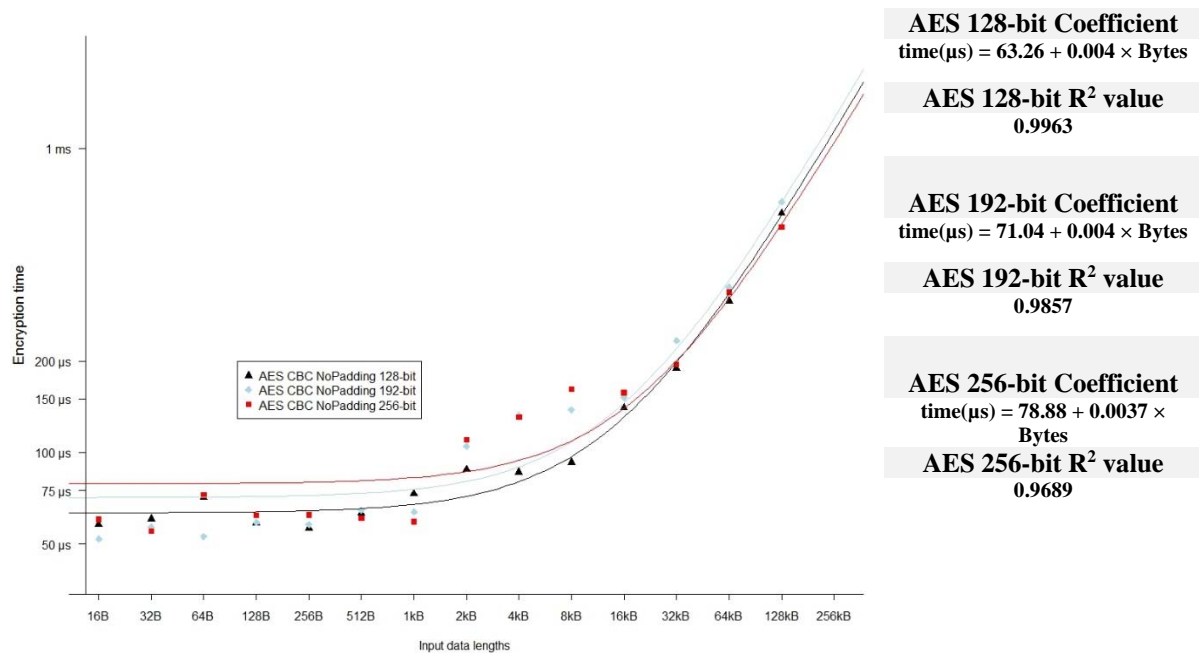
**Figure 5.12: Symmetric encryption times for three different algorithms with best fit lines. Note equation times are in microseconds.**

Figure 5.13 shows AES encryption with two different modes of operation: CBC and ECB. The best fitted equation and  $R^2$  values are quite similar to each other which is expected since it same algorithm running in just different modes.

Figure 5.14 shows AES encryption for various key sizes. Similar to Figure 5.13, the equations' coefficients and  $R^2$  values of this figure, are fitted with similar values which is expected since AES is still used.

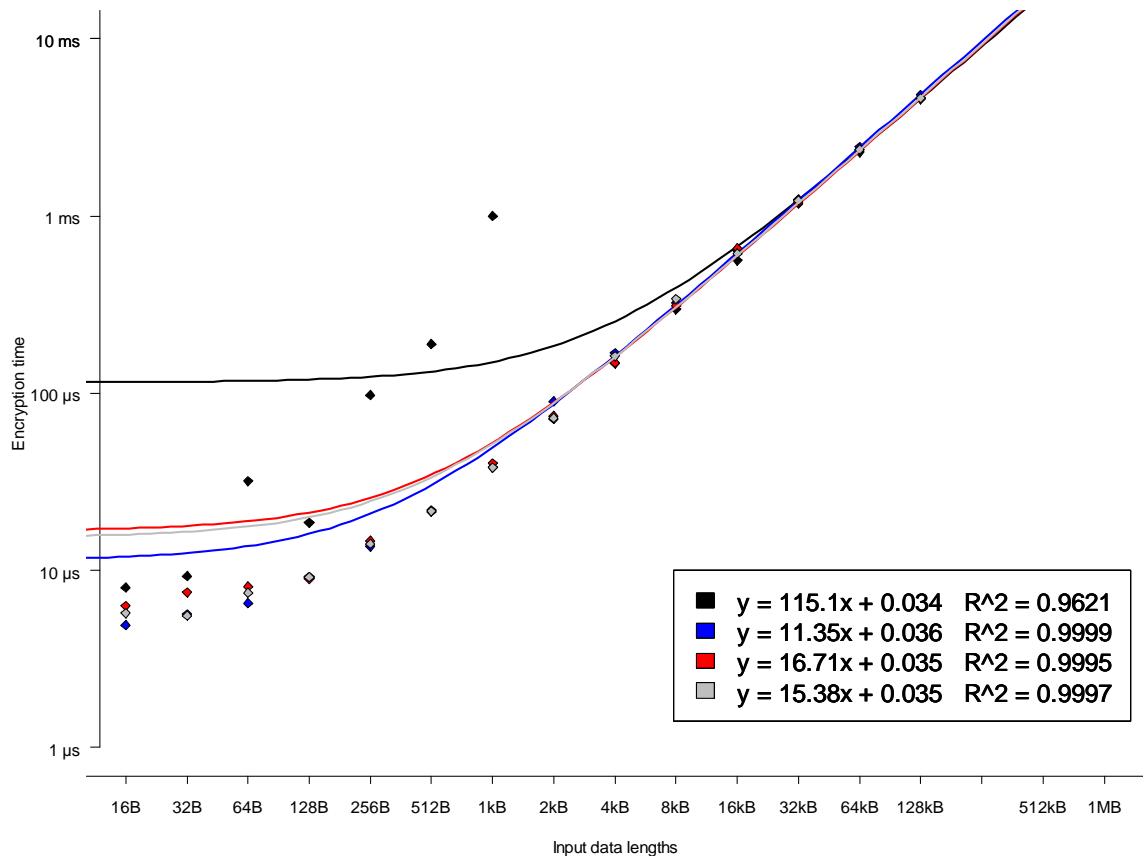


**Figure 5.13: AES encryption times for two modes with best fit lines. Note equation times are in microseconds.**



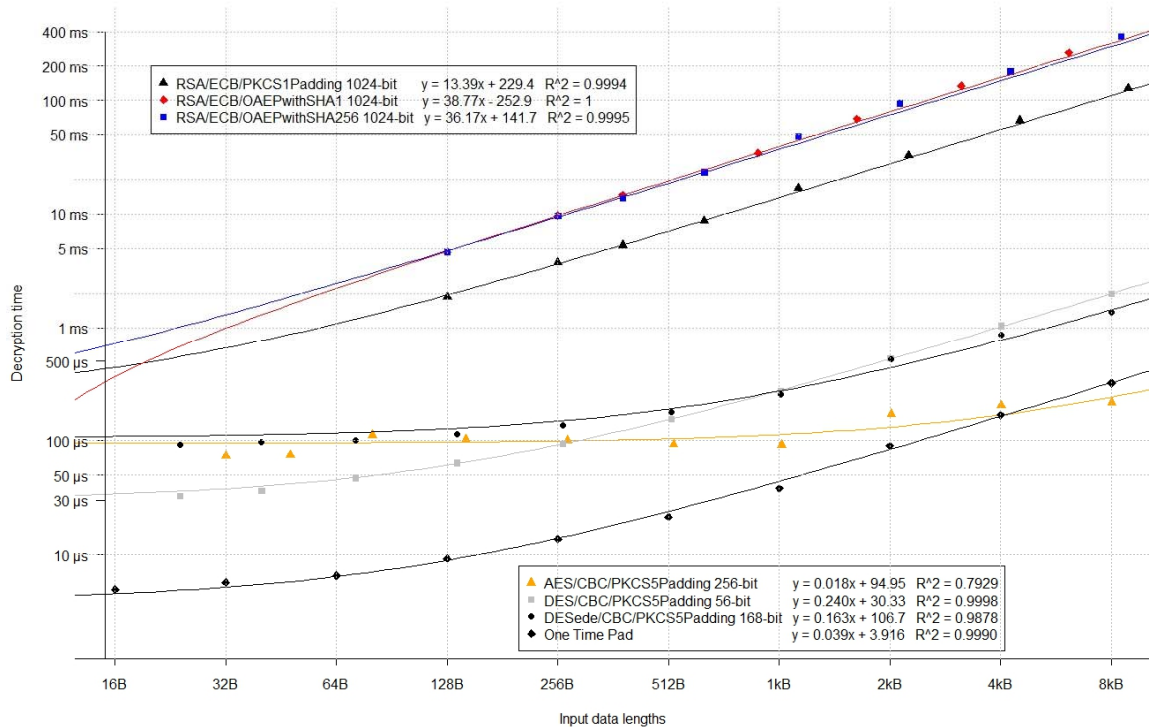
**Figure 5.14: AES encryption times for different key sizes, with best fit lines. Note equation times are in microseconds.**

Figure 5.15 shows only OTP encryption, which was repeated four times. These times are shown with lines of best fitted equations and  $R^2$  values. One of the test attempt resulted in higher values (black dots) which also results in a different equation for these points. Since these data points are spread out more than the other points, the  $R^2$  value is less since the fitted equation has more error.



**Figure 5.15:** OTP encryption during four different attempts. One of these attempts (black points and line) shows a large deviation in time. Note equation times are in microseconds.

Figure 5.16 shows a comparison plot of all algorithms and their different parameters, used in the decryption test. These values are only based on input up to 8 kilobytes to show the low and relevant input lengths. The calculated  $R^2$  values are all very close to 1 which is also noticeable by the plots where the best fitted lines do not divert much from the data points.



**Figure 5.16:** Decryption times of both symmetric and asymmetric algorithms, with best fit lines. Note that equation times are in microseconds.

### 5.3 Reliability Analysis

Here will be described the analysis of results regarding its reliability. Section 5.3.1 describes a thorough analysis of how light conditions affects in the results, Section 5.3.2 describes how reliable the detectable area results are and finally Section 5.3.3 describes the encryption results from a reliability perspective.

#### 5.3.1 Light conditions of QR code

When analysing the results of the readability tests with regarding light conditions, there are mainly two figures that are of relevance here: Figure 5.1 and Figure 5.2. The first of these, shows that with maximum light sources readability ranges from the largest of sizes (700 pixels), down to half (350 pixels), although the latter has a very low percentage of readability. In those cases where light sources are at a minimum (lowest values of both ambient light and brightness), readability ranges down to 450 pixels. When the brightness level is opposite the ambient light level (i.e. one is high the other is low), these light settings work in a destructive way as the result is unreadability. However, investigating the second figure where the size of the code stayed the same, but the content lengths were changing, the environment that had different values for ambient and brightness resulting in some readability. Comparing with image index 3 in Table 5-1, readability in this image must be low, since the door station's reflection is visible on the mobile device's



screen, possible causing some interference in the pixel colours. This means that different results occurred on at least one occasion of the same environment, in this case with full ambient light, lowest brightness, 700 pixels size, and around 50-100 characters of content. As an overall result, situations with unbalanced light can lead to unreliable readability and thus should motivate using balanced light settings in real-world scenarios of a future product, for instance using protecting constructions over the hardware cameras or mount the camera units in areas with a homogenic ambient light.

### 5.3.2 Detectable area

When focused on finding spots in the space around the door station where QR codes could be readable, it was a bit difficult to generate exact values of distances using analogue tools (in this case a ruler). Using this rough method of collecting this data is very unreliable, especially as repeated measurements give different results. Unfortunately, this experiment was not repeatedly sufficiently many times to assess finer resolution values, but the few attempts are sufficient for the purpose of this experiment.

### 5.3.3 Encryption effects

Conducting the experiments with different encryption methods to generate the content to be stored inside a QR code, was necessary in order to gain insight into what the effect could be when storing nearly random data rather than storing plaintext.

We look at asymmetric keys first. Figure 5.6 shows the maximum data to be encrypted into one block using different key sizes and paddings, with the curves showing that increasing key size also increases the maximum allowed content. However, there was a deviation during the tests attempts and there should not have been any deviation (since there is mathematical logic behind these values). The same logic is seen in Figure 5.7 which demonstrates the numbers in Table 5-2 and Table 5-3 in a staircase formation plot, just to give a hint that small increments in input data do not necessarily mean an increase in the size of the encrypted output.

Figure 5.8 clearly shows that with padding option **OAEPwithSHA1** the content with length 32 bytes took almost 10 milliseconds to encrypt, even though half and double this amount of content both took a fraction of that time (~2 milliseconds since log scaled), thus not following the expected curve. This means that something could have happened during that time, such as an interruption which caused the application thread to pause for a while. Figure 5.10 shows the case of decryption with RSA, where the reliability is better although it takes much longer to decrypt that to encrypt.

For the symmetric key experiments, each of the algorithms AES, DES, and 3DES were tested for encryption and decryption with different padding modes. The first thing to notice is that these algorithms perform much faster than RSA which is expected since their key sizes are *much* smaller. Looking at their execution times in Figure 5.16, the symmetric key decryption outperforms asymmetric key decryption by a large factor. This was quite noticeable already with lesser amounts of data. For this reason, we aborted collection of further data about RSA decryption already at content length of 8 kilobytes since the point was already proven.

Focusing on reliability, the main deviation the results concerned the test of OTP encryption (i.e. the basic XORing process). Although this was known to be simplest and fastest of all method, it had a major deviation with short content lengths, although every test was repeated 10 times. This deviation is not shown in Figure 5.16 and the best fit line is based upon the *good* results which followed this curve quite well. Conducting an OTP encryption test (encryption and decryption is basically same operation and ought to show similar times) of 4 attempts gave results shown in Figure 5.15 where a similar deviation is between 128 bytes and 2 kilobytes. Because this deviation occurred only once during the attempts, like the decryption test, it cannot be determined exactly why this deviation occur and is considered to be a bit unreliable.

## 5.4 Validity Analysis

When analysing the content lengths and sizes of the QR codes, from Figure 5.1 and Figure 5.2 with reference to Table 5-1, it is clear that the readability is near zero in some environmental settings. For instance, with varying brightness of the mobile device's screen the results range from an image that is unreadable to perfect results in other settings. Another situation is the rotation test which has results in Figure 5.3, all the angles are within 90 degrees, which is physically possible to get valid results from. If one of the measured values had in fact exceed 90 degrees, this would have been considered an invalid result, requiring that the entire test be repeated, since such a value is not possible (i.e. the mobile screen would be facing away from door station's camera).

The time taken for various algorithms to encrypt and decrypt data clearly shows that AES is faster than DES, which in turn is faster than 3DES. Knowing whether this is true in reality or just by this test, could be determined by comparing with other studies of similar experiments, and then a conclusion can be draw. Earlier studies, such as [44] [45], showed that AES is faster than both DES and 3DES. Moreover, 3DES takes longer time than DES since 3DES is basically DES three times, so naturally 3DES must take more time. However, AES does not use the same encryption logic based upon *feistel rounds* [46] that DES and 3DES use. Instead AES relies on matrix operations (explained in detail in [12]) which evidently makes it much faster.

When analysing Figure 5.16, it is noticeable that the asymmetric algorithms takes much more time to execute than the symmetric algorithms. The fastest one among the symmetric algorithms is OTP up to 4 kilobytes of input data. Since OTP is a very simple XOR operation, it outperforms complex encryption algorithms that has much more operations in their functionality. After 4 kilobytes of input data, AES takes over to be faster than OTP, despite the simplicity of the OTP operation. This might be since OTP is user developed (does not require any library to be implemented), while AES must be enabled by a library in Java, which might have better memory and CPU utilization than the own-developed test code for OTP analysis.

## 5.5 Discussion

This discusses the actions taken based upon the results and the analysis, in order to create a design with optimal parameters.

### 5.5.1 Preparation of the optimal system layout

Below is a list of compatible cipher transformations for the *javax.crypto.Cipher* class [47]. Some additional specifications are not mentioned by this list, such as the restricted key size of AES is not 128-bit but rather 256-bit, if an additional policy file [48] is downloaded and added to the Java path. However, this does not increase the block size which remains 128-bits and this must be considered during testing. Another specification, not mentioned, is that 3DES (DESede in list) can also be used with 112-bit key size since the encryption key (encryption-decryption-encryption) can be reused, reducing the total key size to  $56+56=112$  bits. What is affected here is the initial vector (IV) that remains 8 bytes, as it does for standard DES.

*"Every implementation of the Java platform is required to support the following standard Cipher transformations with the key sizes in parentheses:*

- *AES/CBC/NoPadding (128)*
- *AES/CBC/PKCS5Padding (128)*
- *AES/ECB/NoPadding (128)*
- *AES/ECB/PKCS5Padding (128)*
- *DES/CBC/NoPadding (56)*
- *DES/CBC/PKCS5Padding (56)*
- *DES/ECB/NoPadding (56)*
- *DES/ECB/PKCS5Padding (56)*
- *DESede/CBC/NoPadding (168)*
- *DESede/CBC/PKCS5Padding (168)*
- *DESede/ECB/NoPadding (168)*
- *DESede/ECB/PKCS5Padding (168)*
- *RSA/ECB/PKCS1Padding (1024, 2048)*
- *RSA/ECB/OAEPWithSHA-1AndMGF1Padding (1024, 2048)*
- *RSA/ECB/OAEPWithSHA-256AndMGF1Padding (1024, 2048)* "

*(Quoted from javax.crypto.Cipher manual page )*

The list of cipher transformations options is detailed with the algorithm as well as the mode used in the format *algorithm/mode/padding* (i.e. "RSA/ECB/PKCS1Padding"). The last three transformations are used in PKI (RSA) while all those above these are used in symmetric encryption (AES, DES, DESede).

In Android systems, applications are run under thread management with priorities, hence the applications cannot execute in the foreground all the time, as could be done in a OS that does process scheduling. Furthermore, applications in Android devices are also restricted memory-wise by the nature of design. Since smartphones are designed to be able to handle many running applications (some of which might be graphic-demanding or CPU demanding) while smartphones still do not have the same memory capacity as a typical home-computer has today [49]. As a result each application is restricted to a very limited memory footprint, often around 10-50MB [50]. When launching an application which will measure time on a very high resolution (in this case nanoseconds), there might (will) be asynchronously interruptions which negatively affect the accuracy of time

measurements. Such interruptions could be from a simple update I/O call to the kernel or a *malloc* call asking for more memory for the application. The latter is likely to be used frequently in measurements of an operation on a size-dependent variable. Consider Code Snippet 2 in Appendix A.

If the function to calculate 2 to the power of the input parameter, takes 50 ns, the value of `end_time` will be 150 ns instead of ~50ns. A common way of achieving close to the correct value is to repeat the test a large number of times. However, in memory-restricted devices, if the code is like that in the second part of the function (i.e., storing 256 arrays of some data to be analysed), then the large number of allocations will eventually cause malloc calls since each chunk is 1MB and storing 256 of these chunks is too much for the initial Android application. These calls will cause the application thread to be paused, just as in the first example, since the OS kernel will spend time to perform memory allocations.

#### **5.5.1.1 RSA keys**

Choosing an appropriate key size is very important regarding the server's implementation of decryption. If key size is too big, then decryption of a block will take too much time on the server. This can make the server vulnerable to becoming a victim of a Distributed Denial of Service (DDOS) attack. On the other hand, if key size is too small, finding the key might become too easy. Also, as Figure 5.6 shows, the key sizes for this sort of encryption were able to be smaller than what is usually preferred. This is because on the mobile device the libraries used for encryption allowed these key sizes, whereas the server implementation of the encryption libraries has a minimum of 1024-bit key size for RSA encryption.

If RSA is chosen for the QR codes in this type of systems, the key size is the major factor that affects the complexity of the generate QR code and hence its readability. However, a disadvantage with asymmetric encryption is that data to be encrypted cannot be done stream-wise as is the case for OTP. With the Java libraries, the input data must be segmented into blocks of specific maximum sized, according to Table 5-2. This puts some extra work on the developer who must manually must segment the data if too long for the specified maximum data block length.

#### **5.5.1.2 Symmetric keys**

Symmetric key encryption is more suitable for large quantities of data, than asymmetric encryption since it uses smaller key sizes. The Java library implementation of symmetric algorithms allows a program to just input the data that is to be encrypted, without any segmentation, even though the symmetric key encryption is block based.

#### **5.5.1.3 Ruling out bad readability conditions**

Looking at the results of the readability tests of the QR code, the light conditions and how a user should orient the mobile device towards the door station are quite important factors. Since readability was poor during conditions of opposite light settings (ambient versus screen brightness) such factors must be considered in the final design. The light settings could be designed to always work constructively, meaning that they should be adapted to balancing the environmental ambient light with the brightness of the mobile device's screen.

Modifying the codes with colours, did not have the effect that I hoped it would have, since some colours had better clarity during high brightness settings with low ambient light. This appeared to only be the case for the human eye, rather than the camera. Instead, removing as much of the white background as possible while keeping the necessary white areas (i.e. quiet zones) improved readability under conditions of low ambient light.

Regarding rotations, there is not much more to in this thesis than analyse possible methods. Skånetrafikens experience in using QR codes on mobile devices as a type of ticket solution, lead them to design their readers to have small soft contours, almost like cushions, which allows the user's mobile device to be placed with the correct alignment. Applying a similar solution would require a different design of the door stations; however, this is not relevant for the current door stations.

#### 5.5.1.4 Ruling out unsuitable encryption algorithms

Table 5-4 shows the first two prototype formats for the QR code, where both formats are built upon asymmetric encryption to ensure that only the validation server can decrypt the content. Signing the content ensures the server that the QR code came from a legitimate sender and thus enhancing their authenticity.

However, many problems arose with both these proposed solutions. The first solution had an IP field which was for the purpose of determining the correct corresponding mobile connection for which the QR code was generated. Unfortunately, this would only work in a NAT-less network where all the IP addresses are globally routable. Another issue was that the fields *Username* and *Password* were considered redundant, while the desire was that only relevant information should be stored inside a QR code. Since the username and password could simply be used for a login-session perhaps to the app for the first app-session, this information could be excluded from the content in the QR code.

**Table 5-4: Previous designs of QR code format**

<b>QR code format</b>	<b>description</b>
<b>SERVER+{IP:(Username,Password,GUID):Timestamp:Nonce:Signature}</b>	8-byte IP, 16-byte username, 16-byte password, 16-byte GUID, 8-byte timestamp, 16-byte nonce, 256-byte signature, encrypted with 2048-bit server public key
<b>SERVER+{GUID:Timestamp:Nonce:Signature}</b>	16-byte GUID, 8-byte timestamp, 16-byte nonce, a 256-byte signature, which is encrypted with 2048-bit server public key

RSA asymmetric key encryption was used as the algorithm with a 2048-bit key size in order to address current security standards (although a 4096-bit key might be considered as a standard nowadays). In the experiments conducted in this thesis project, the maximum data capacity for one encrypted block is 245 bytes, 214 bytes, and 190 bytes (depending upon the padding option used). However, since the RSA keys for the user was also based on 2048-bit key size, a signature would be 256 bytes in size - which alone is above the limited payload size. This would cause two encrypted chunks each of 256 bytes, giving a total of 512 bytes that would need to

be stored inside the QR code. In order to easily manage the content, the output was encoded using Base64, which gives a final output of the following size:

$$4 \times \lceil \text{input}/3 \rceil$$

Therefore, in this case 684 bytes are output for 512 bytes of input.

Taking into account the results shown in Figure 5.2 of the readability tests for different content lengths, it is obvious that this format would yield zero readability. Therefore, these two formats for the QR code, proved not to suitable designs for the QR code, but the analysis of them gave pointers to the correct direction. A signature could be replaced with a pre-authentication method instead of baking it into the QR code, saving a lot of storage thus decreasing the code size. Assuming the case where the authentication check is performed *before* the generation of the QR code, then the rest of the components of the previous QR code formats would still be necessary and encryption must still take place. Having everything but the signature results in only one 256-byte encrypted chunk (344 bytes in Base64 encoding) and would be considered readable according to Figure 5.2.

### 5.5.2 Improved solution

The new system, which was described thoroughly in Chapter 4, was designed to minimize the QR code size in order to achieve high readability for as many environmental conditions as possible, while still being secure enough not to leak any sensitive information, despite the “unsecure channel” between the door station and the mobile device\*. Removing the asymmetric encryption was done to minimize the QR code, but cannot be removed without finding a replacement. The tests of symmetric encryption showed not only that the encryption speed was better but since a much shorter key size was used, a much smaller encrypted chunk was output. This motivated the use of symmetric encryption instead of asymmetric encryption for the QR code. However, knowing how TLS work, using symmetric keys cannot be done without negotiating a key in a secure way, meaning we have to send the symmetric key by using asymmetric encryption. Applying this type of approach lead to the idea of combining the sending of shared key with the authentication required by the server.

Given the decryption times of different algorithms (both asymmetric and symmetric) in Figure 5.16, there is clear that symmetric algorithms are much faster than asymmetric ones. It is also noticeable among symmetric algorithms that OTP is the fastest choice up to a data input length of 4 kilobytes, where AES intersects and becomes faster.

Having stated that OTP is the fastest for encryption with small amounts of content (< 4 kilobytes), it was still a question of whether it could be used in a secure way, since it has a very large downside of being easily broken if anything more than the ciphertext is known. For example, if the structure of either key or plaintext is known, then the other could be guessed in a brute force attack. So, the question was how to make such an implementation secure. The answer was using a hash. If the server and mobile device exchanged enough information, it is sufficient for the server to create a hash value of some data and then the mobile device computes the same hash parameters along the challenge, hence it creates the same hash. If this is not be secure enough, since a good cryptographic hash is very difficult to break,

---

\* Since that is the spot were attackers have the highest chance of learning things about the system.

adding OTP encryption on top of this with the shared negotiated key (also used in the challenge), the mobile device could perform an extremely fast encryption since the hash value only had to be 32 bytes (based on SHA-256), while producing a very strong QR code. Since the creation of a QR code only happens once in a mobile-server session, the shared key is only used once, hence OTP requirement of only using the encryption keys once is met. Even if an attacker does get in hold of the key in some way, decrypt the content yields only a hash. Likewise, if the hash somehow is learnt by the attacker, the key used is of no use since another key will be used for next session.

Figure 5.17 shows the overviewed new system layout with some detailed information and aspects regarding environmental aspects.

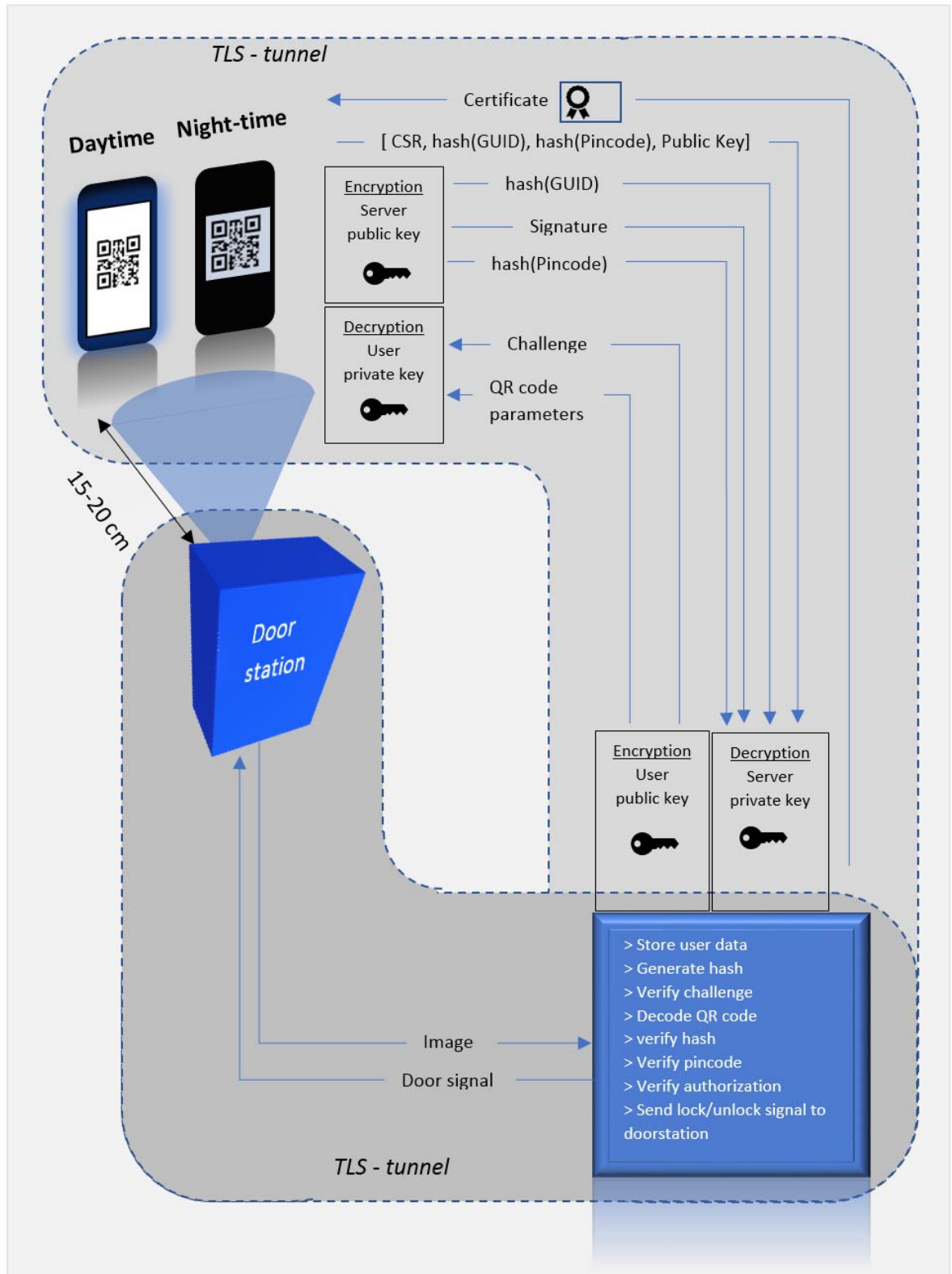


Figure 5.17: Optimized system layout



### 5.5.3 Evaluation and analysis of the optimized system

Table 5-5 shows the time taken with the improved security and performance. This table shows the system where 2-factor authentication (QR code + additional PIN code request and verification) was deactivated. Having this enabled would not generate measurable results since the time taken by the user's entry of the PIN code will dominate the time spent. Therefore, the times in the table correspond to the time from when the button on door station is pressed (i.e. causing the door station to take a snapshot), up to the point of which the door station receives a signal to unlock the door. We will refer to this total time as the *roundtrip time*.

In contrast, Table 5-6 shows the timing when the validation was unsuccessful, returning a signal to not open door. The roundtrip time for both successful and unsuccessful cases are quite similar but what has really changed is the time for the validation. There is a difference by a factor of almost 4, as the shortest time for unsuccessful validation was under one microsecond while the longest time for a successful validation was 46 milliseconds.

The time to decode a QR code, is almost the same time, around 180 milliseconds when a QR code is found and 120 milliseconds when no QR code was found.

It is worth mentioning is that the unsuccessful validation results are based on a readable QR code, where the validation failed because of the protocol (in this case the QR code had already been validated and hence the nonce had been used, since I used the same QR code for these attempts).

**Table 5-5: Time taken from when a QR code is scanned at door station to unlock signal being received**

Number of Attempts	Roundtrip time (ms)	Validation time (ms)
1	453.19	46.20
2	447.98	10.10
3	432.04	19.96
4	420.13	12.34
5	433.24	12.53
6	442.51	16.43
7	430.19	13.13
8	416.06	13.12
9	419.72	12.63
10	406.47	15.93

**Table 5-6: Time taken from when a QR code is scanned at door station to the remain locked signal being received**

Number of Attempts	Roundtrip time (ms)	Validation time ( $\mu$ s)
1	1092.48	14.211
2	447.56	1.184
3	431.00	1.184
4	371.10	1.974
5	444.30	0.790
6	407.06	2.368
7	403.56	2.368
8	434.93	1.973
9	423.81	2.369
10	439.66	2.368

#### 5.5.4 Comparison with MiFare access card

Current solutions that are used today are mostly smart card based, meaning that an employee is equipped with an access card issued to him/her by the company where he/she works. Using this access card, the employee can enter the building and different areas within the building. Additionally, this card is also associated with a specific PIN code which the employee might be requested to enter at certain times.

As the smartcard can compute a digital signature that can be quite complex such an access card system is thought to be very secure. Additionally, the employee does not need to remember or compute this signature. Moreover, the employee does not need to remember a password but does need to remember a four-digit PIN code that is inputted once in a while when using the card. Such an access card system replaces an earlier complex world of managing and remembering long and unique usernames and passwords for access systems with a simple world where all the complexity can be embedded into an access card with digital data and processing.

But with new solutions also come new problems and new ways to hack the solutions. Nowadays computers are fast enough to compute the necessary data to hack simple physical access cards. For instance, the well-known MiFare card is used in many different areas, such as public transportations and access to company buildings. There are multiple forms of MiFare but to list the most relevant types that are used for *corporate access* [51], these MiFare card families include DESFire, Plus, and SmartMX.

##### 5.5.4.1 MiFare DESFire

As the name implies, these cards implement DES and 3DES but also have support for (128bit) AES. Additionally, for data authenticity, they use an 8byte CMAC [52]. Additional improvements such as increased number of nonce bits, have generated

new versions of this card known as *MiFare DESFire EV1* and *MiFare DESFire EV2*.

#### **5.5.4.2 MiFare Plus**

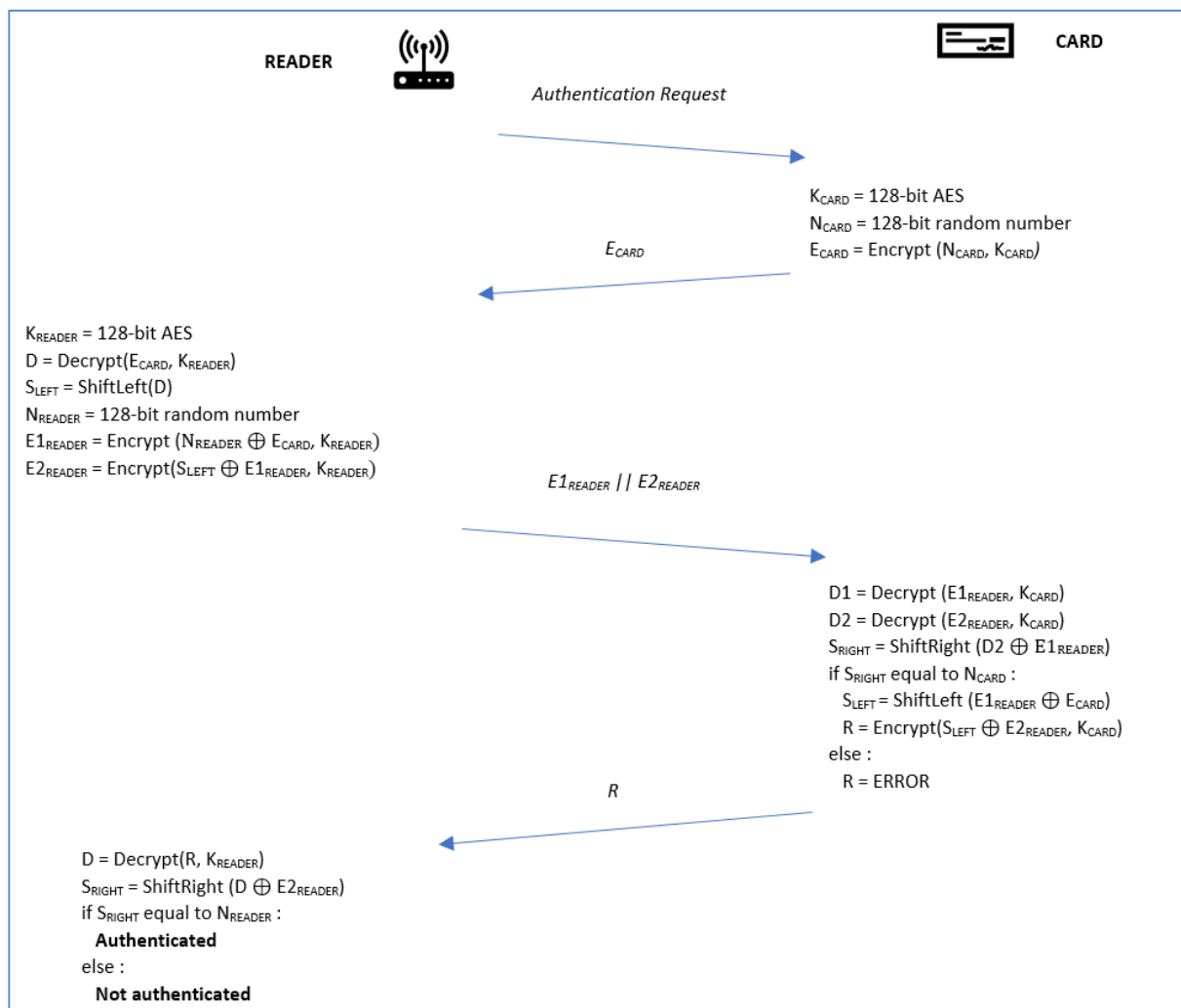
Plus is the next generation of the earlier *MiFare Classic* card which was a fairly easy task to hack [53]. This card is built upon a cipher of either 48-bit Crypto-1 or 128-bit AES keys.

#### **5.5.4.3 MiFare SmartMX**

The Mifare SmartMX family uses both symmetric and asymmetric cryptographic algorithms, specifically 112-bit 3DES and 128-bit AES for symmetric encryption and 32-bit RSA for PKI[54].

It is clear that the MiFare family consist of cryptographic technology that the company has put a lot of effort into, to bring the security up to a higher level. The main purpose of using encryption in cards like these is for the card and reader to authenticate themselves towards each other, hence the data to be sent from card to reader is from a genuine source and the reader is also genuine.

Looking deeper into the authentication protocol [55](Slide 17) of MiFare DESFire EV1, Figure 5.18 shows how this protocol is used. The reader sends an authenticate request to which card replies with an encrypted nonce. The reader creates its own nonce, decrypts the received nonce, and does some shifting operations as well as XORing, encrypts these results and sends the result back to card. The card can then decrypt these and verify the original nonce it sent, with the received nonce. To complete the authentication, if verification was successful, then the card applies shifting operations to the received nonce from reader, then sends a result back to reader which can verify its nonce.



**Figure 5.18: MiFare DESFire EV1 authentication protocol**

A problem that has been dealt with regarding access cards is the ability of a card to be cloned. An access card (as with many other cards) should be unique in order to ensure identification of a person. However, if the card can be copied it cannot be guaranteed to be unique. The solution to preventing cloning is to make the card tamper proof, which requires specific and very specialized hardware to be able to read the content inside the chip in the card or to ensure that this information cannot be read at all.

Analysing the authentication protocol of the EV1 card, there are some similarities and some differences compared with the protocol designed for this thesis project. Looking the EV1 protocol as an overview, it has a 4-way handshake to authenticate the card to the reader. This can be compared with the challenge implemented in this thesis project. The difference is that the EV1 protocol uses symmetric encryption for all of the communication, including the authentication. This is done because the cards already possess the keys to used and therefore there is no need for negotiation.

Similarly, the EV1 protocol use nonces for the authentication entity. Rather than using a signature to prove authenticity as I done in this thesis project, EV1 proves authenticity based on manipulation of the nonce, using bitwise shifting

operations. This is a much less computationally demanding operation and is therefore suitable for low power cards. The principle works the same when the reader can verify the incoming modified nonce with the nonce it sent, by modifying it and sending it back, hence the “signature”.

The 4-way handshake is sufficient for EV1 cards to authenticate themselves, before sending data, but this is not sufficient in my case. In this thesis project the mobile device renders the QR code which then is also verified by server. The difference between these two approaches is that a QR code’s content can easily be read (i.e. be decoded) by an QR code reader, such as that in smartphones today. In contrast, it is much more difficult to find a fast and simple method to decode the content stored inside a card. This means that the data to be stored inside the QR code must be secured sufficiently so that we prevent an attacker from learning about the information needed to attack the system from the QR code itself.



## 6 Conclusions and Future work

This chapter states some conclusions in Section 6.1. The limitations of the project are described in Section 6.2, the future work in Section 6.3, and finally some reflections in Section 6.4.

### 6.1 Conclusions

This thesis project has developed a proof-of-concept validation system using QR codes. The experiments conducted in parallel with the process of designing different prototypes led to the final system that works as intended. By utilizing different encryption techniques, different formats for the content inside a QR code and various environments, and of course within the limited resources available to the system, the resulting system meets the requirements for being developed into a potential future product.

One of the goals was to design a module that takes care of the QR decoding of an image taken by the camera on the door station. This goal was definitely met since the QR decoding was based on a Linux program which has good performance of jpeg images containing QR codes. Then in combination with the door station taking a screen shot which is sent to this QR decoder, then the system in general becomes very easy to implement.

The system does handle encrypted content and signatures which was another goal from the security aspect. Additionally, since the system modules are designed to handle multiple devices and no direct user information (such as names, phone numbers etc), then the system is scalable to handle multiple devices which corresponds to just one user.

The designed system does also have functionality to understand what devices that has unauthorized access and/or are not authenticated, which makes it very secure.

One of the goals that was not met is the development of an administration interface that makes it easy to access and manage database content and registrations for the systems. This was currently done in hardcoded ways and from testing point of views for quick results, but before this solution can become a product, the administration interface must be developed.

When designing systems that will co-operate with external hardware in many various modes such as managing multiple connections simultaneously, it is important to incrementally build them up from the simplest case possible and then make forward progress by adding modifications here and there. Thinking too complex from the beginning will slow down the process and introduce problems that could have been avoided when thinking differently.

Working in an area such as this that has the intention to solve problems and invent innovative solutions, it can be very productive to try to as much as possible think outside the box and investigating related products and solutions. This will help with the “dos” and “don’ts” keeping the developer(s) on the correct track from an early start.

I am very happy with what I have accomplished with this thesis project, but if I had the opportunity to redo it from scratch, I would have invested a small amount

of time to solve the situation which I did not have time to consider (as will be described in Section 6.3). While finding the correct protocol to use for the system already at the beginning would seem to be a great advantage, but having experimented with other protocols was definitely worth doing even though time consuming, as it gave an opportunity to understand why these other solutions might not be suitable for this project.

## 6.2 Limitations

In this project, there were sufficient hardware components available and software tools to help finalize the design. However, testing it was limited by my own test environment, consisting of a couple of door stations and mobile devices. Although one of the goals was to develop a proof-of-concept which does not require a large number of devices in this case, the limited number of devices tested might give biased results that could have been avoided by testing with many more devices.

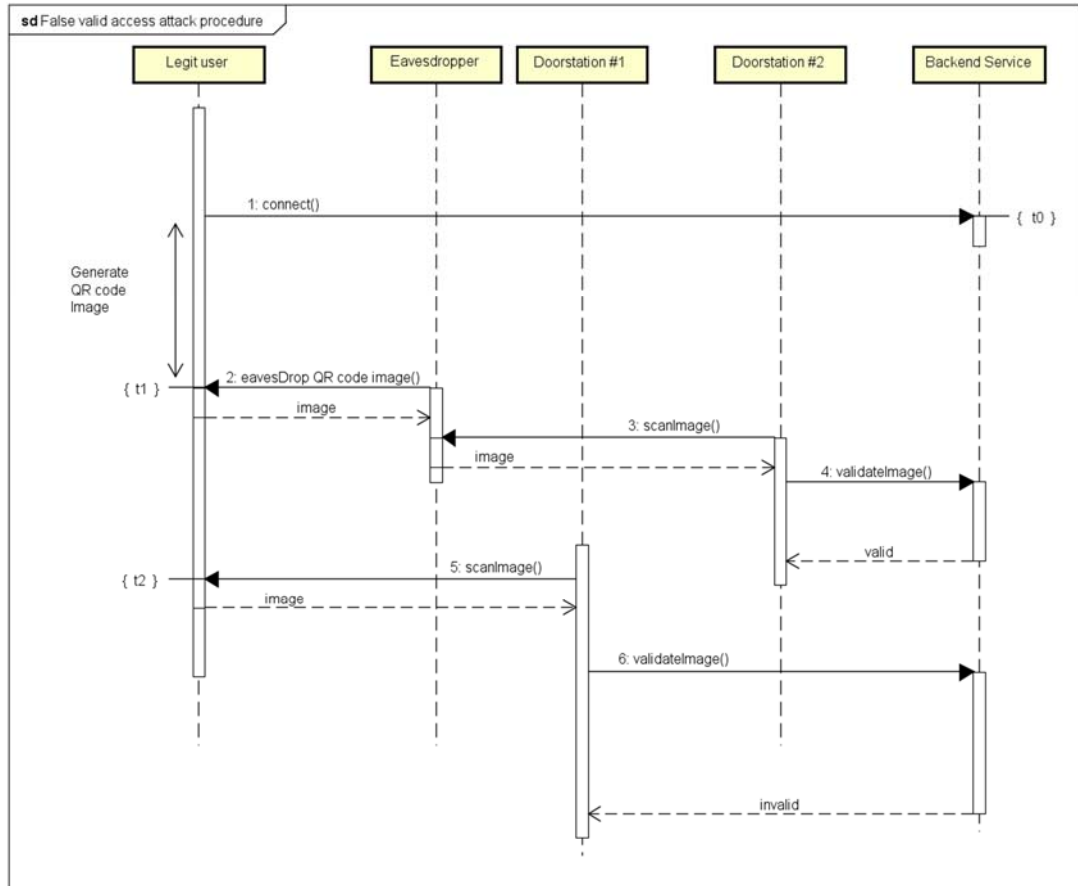
## 6.3 Future work

Due to the limitations of hardware, there is one situation of which could be considered a loophole in the system and has not been dealt with here. However, this is a problem that already exists today. Consider the simple case where someone finds your card that provides access to a certain service, which does not require visual authentication but only relies on this card. In this case this person can then use the card until the owner realises that they are no longer in possession of it and locks the card. Similarly, in this project, consider the situation where authentication is at its minimum, meaning the QR code is the only authentication needed to enter a door, i.e., there is no active 2-factor authentication. The difference from the example situation with the card, is that in this case, an eavesdropper only needs to get a hold of the QR code displayed on the mobile screen, then they can show this QR code to another door station and they will gain access. However, they can only do so for a limited amount of time.

The mobile device could be manipulated technically such that after the QR code has been generated, a snapshot of the screen could be sent to the eavesdropper. There are some programmatic ways to take a screenshot of the user's screen and the eavesdropper could get a copy of the QR code. But this must all occur within the validation time set by the validation service, so the possibility of such an attack should be very low. However, once the eavesdropper has a copy of the QR code he or she can still show this image to a doorstation (in this situation, assuming that the legitimate user is about to authenticate at another doorstation). This second doorstation might then be first to connect to the validation service and hence the validation service might respond with a signal to this second doorstation to open the door (this occurs when step 4 in Figure 6.1 is done before step 6). This can occur because the legitimate user has already connected their mobile device to the validation service before generating the QR code, hence the validation service can successfully lookup the user's mobile device in the connection pool of mobile connections and do a successful validation (of the copy) of the QR code. Later when the legitimate user performs a scan at the first doorstation and the doorstation sends the image to validation service, it will respond with an invalid signal because the validation server knows that this QR code has already been validated and thus will not treat it as valid again.



One solution would be to modify the door station such that it handles nonces that will also be used with the QR code. Then, a user that connects to the service, must also initiate the door station's part in the process, which now both supplies the nonce to the user (via some external screen or by voice for instance) and supplies the same nonce to the server along with the image. When the validation server then receives the decoded QR code content, it can cross-reference the stored door station's nonce in the code, with the actual door station's nonce delivering this QR code. If an attacker tries to copy the code as described earlier, the door station nonce in the QR code would be different than the nonce that the door station which the attacker is attempting to enter sends.



**Figure 6.1:** Sequence diagram of a situation where an eavesdropper could potentially gain access

While discussing the idea of the door station having the functionality of speaking to the user, the speaking could also be used for getting better readability in general. When a QR code becomes unreadable, the system could trigger the door station to inform user how to change the distance or horizontal/vertical alignment for a better chance of become readable. This could be based upon an image analysis of the received unreadable QR code where the analysis module can compute what kind of error there was in the image. Bad resolution could result in too close or too far and only partly readable QR code (having some algorithm that could decide this), could result in an alignment change in either horizontal or vertical mode.

## 6.4 Reflections

The economic aspects of this work refer to both the cost of manufacturing access cards but also the cost of eliminating the need of a paid employee sitting and manually verifying incoming people at the door station. On the whole, these two types of costs might not be the most significant ones, hence there might not be as much saved when using a system such as that described in this thesis. The social aspect is that this paid employee could be doing another job instead which he or she would contribute more to, rather than just sitting still most of the time. From an environmental point of view, the reduction in card manufacturing would definitely decrease the need for these cards and in the long term, reduce the plastic needed, and hence be better for environment.

The main aspect, according to me, is the ethical aspect since this system relies on a surveillance unit which has been modified to not only monitor people but now can monitor mobile devices as well. Since many people use their mobile devices for reasons other than just calls and SMS (sometimes everything else but these two main functions), means that there would be a possibility that other content could be displayed on the screen during the monitoring procedure by the door station. In this case whatever is displayed on the mobile device's screen is now captured as an image in the system for a short while. This is why the initial idea in this work was to restrict sending the whole screen's content into the network, but rather have the door station to do the QR decoding. Unfortunately, that did not seem to work optimally, hence the system prototype still captures the full screen content. Nevertheless, I expect that this could be changed in the future to limit as much as possible the collection of unnecessary data.

## References

- [1] “Control4.” [Online]. Available: <https://www.control4.com/solutions/products/door-stations>
- [2] “iVision+ | OPTEX.” [Online]. Available: <http://www.optexamerica.com/security-products/ivision-plus>
- [3] “AXIS A8105-E Network Video Door Station,” *Axis Communications*. [Online]. Available: <http://www.axis.com/se/sv/products/axis-a8105-e/support-and-documentation>
- [4] “AXIS A8004-VE Network Video Door Station,” *Axis Communications*. [Online]. Available: <https://www.axis.com/gb/en/products/axis-a8004-ve>. [Accessed: 13-Jan-2018]
- [5] “Timing Attacks on RSA: Revealing Your Secrets through the Fourth Dimension.” [Online]. Available: <http://www.cs.sjsu.edu/faculty/stamp/students/article.html>
- [6] P. Rogaway, “Nonce-Based Symmetric Encryption,” in *Fast Software Encryption*, 2004, pp. 348–358 [Online]. Available: [https://link.springer.com/chapter/10.1007/978-3-540-25937-4\\_22](https://link.springer.com/chapter/10.1007/978-3-540-25937-4_22). [Accessed: 09-Jan-2018]
- [7] DENSO WAVE INCORPORATED, “History of QR Code | QRcode.com | DENSO WAVE.” [Online]. Available: <http://www.qrcode.com/en/history/>. [Accessed: 13-Jan-2018]
- [8] “QR Code - Structure, Usage and Applications | Shield UI.” [Online]. Available: <https://www.shieldui.com/blogs/qr.code>
- [9] “QR Codes - Technical Limitations Of QR Codes.” [Online]. Available: <http://qrcode.meetheed.com/technical.html>
- [10] M. Riley and I. Richardson, “reed-solomon codes,” 1996. [Online]. Available: [https://www.cs.cmu.edu/~guyb/realworld/reedsolomon/reed\\_solomon\\_codes.html](https://www.cs.cmu.edu/~guyb/realworld/reedsolomon/reed_solomon_codes.html). [Accessed: 13-Jan-2018]
- [11] “Reed–Solomon codes for coders - Wikiversity.” [Online]. Available: [https://en.wikiversity.org/wiki/Reed%E2%80%93Solomon\\_codes\\_for\\_coders](https://en.wikiversity.org/wiki/Reed%E2%80%93Solomon_codes_for_coders)
- [12] U.S. National Institute of Standards and Technology, “Advanced Encryption Standard (AES),” Nov. 2001 [Online]. Available: <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>
- [13] H. Al-Anazi, B. Bahaa Zaidan, A. Alaa Zaidan, H. A. Jalab, and M. Shabbir, “New Comparative Study Between DES, 3DES and AES within Nine Factors,” ISSN 2151-9617, Mar. 2010 [Online]. Available: <https://arxiv.org/ftp/arxiv/papers/1003/1003.4085.pdf>. [Accessed: 13-Jan-2018]
- [14] B. Kaliski, “RFC2898, PKCS5 padding,” *Request For Comments*. [Online]. Available: <https://www.ietf.org/rfc/rfc2898.txt>. [Accessed: 13-Jan-2018]
- [15] Meri, “Secure Encryption in Java.” [Online]. Available: <http://meri-stuff.blogspot.com/2012/04/secure-encryption-in-java.html>. [Accessed: 09-Jan-2018]
- [16] B. Kaliski, “PKCS #1: RSA Encryption Version 1.5.” [Online]. Available: <https://tools.ietf.org/html/rfc2313>. [Accessed: 13-Jan-2018]
- [17] B. Kaliski and J. Jonsson, “Public-Key Cryptography Standards,” Feb-2003. [Online]. Available: <https://www.ietf.org/rfc/rfc3447.txt>. [Accessed: 13-Jan-2018]
- [18] T. Dierks and C. Allen, “The TLS Protocol Version 1.0,” *Internet Request for Comments*, vol. RFC 2246 (Proposed Standard). Jan-1999 [Online]. Available: <https://tools.ietf.org/html/rfc2246>
- [19] “ciphers(1): SSL cipher display/cipher list tool - Linux man page.” [Online]. Available: <https://linux.die.net/man/1/ciphers>
- [20] T. Apanasevic, J. Markendahl, and N. Arvidsson, “Stakeholder’s expectations: The case of mobile public transport ticketing in Sweden,” in *13th International*

- Conference on Mobile Business, (ICMB 2014) London, June 4-5, 2014*, 2014 [Online]. Available: <http://www.diva-portal.org/smash/record.jsf?pid=diva2:782595>. [Accessed: 20-Aug-2017]
- [21] Scandit, "Aztec Code: Exploring the Aztec Barcode Type," *Scandit*. 23-Sep-2014 [Online]. Available: <https://www.scandit.com/aztec-code-exploring-aztec-barcode-type/>. [Accessed: 14-Jan-2018]
- [22] "Skånetrafiken -." [Online]. Available: <https://www.skanetrafiken.se/sa-reser-du-med-oss/sa-koper-du-biljett/skanetrafiken-appen/>
- [23] "Nya attacker mot Skånetrafiken," *SVT Nyheter*. [Online]. Available: <https://www.svt.se/nyheter/lokalt/skane/skanetrafiken-pa-nytt-attackerat>
- [24] "IT-attack kostade fem miljoner," *SVT Nyheter*. [Online]. Available: <https://www.svt.se/nyheter/lokalt/skane/dataattack-kostade-fem-miljoner>
- [25] "Biljettbluffen – så lätt är det att lura Skånetrafikens nya system," *Sydsvenskan*. [Online]. Available: <http://www.sydsvenskan.se/2017-04-12/biljettbluffen-sa-latt-ar-det-att-lura-skanetrafiken>
- [26] SL, "SL - tickets." [Online]. Available: <http://sl.se/globalassets/taxehandboken/bilder-biljetter-kort-.pdf?id=561>
- [27] T. Loukusa, "Analys av säkerheten av RFID i inpasseringssystem," UPTEC IT 12 008, 2012 [Online]. Available: <http://uu.diva-portal.org/smash/get/diva2:547126/FULLTEXT01.pdf>
- [28] "SL byter ut korten – för att stoppa hackare," *Computer Sweden*. [Online]. Available: <https://computersweden.idg.se/2.2683/1.618301/sl-byter-ut-korten--for-att-stoppa-hackare>
- [29] BankID, "BankID," *Bankid*. [Online]. Available: <https://www.bankid.com/en/>
- [30] K. Annous, "Så skyddar du dig mot Bank-ID kapningar via SMS [Bedrägeri]." [Online]. Available: <https://artiklar.interlan.se/blogg/bank-id-kapningar-via-sms>
- [31] "Mobilbetalning | ICA," *ICA.se*. [Online]. Available: <http://www.ica.se/butiker/service-i-butiker/mobilbetalning/>
- [32] T. Apanasevic, "Challenges Related to the Introduction of Innovative Services in the Market: Mobile Payment Services in the Swedish Retail Industry," KTH Royal Institute of Technology, Stockholm, TRITA: 2015:06, 2015 [Online]. Available: <http://kth.diva-portal.org/smash/get/diva2:849380/FULLTEXT01.pdf>
- [33] P. Lambert, "SQRL: A new method of authentication with QR codes," *TechRepublic*, 22-Oct-2013. [Online]. Available: <http://www.techrepublic.com/blog/it-security/sqrl-a-new-method-of-authentication-with-qr-codes/>
- [34] Hotek, "QR Code Lock," *Hotek*, 2017. [Online]. Available: [https://www.hotek.nl/media/Product%20sheets/2017-17%20PS\\_Hotek%20QR%20code%20lock%202.0\\_EN.pdf](https://www.hotek.nl/media/Product%20sheets/2017-17%20PS_Hotek%20QR%20code%20lock%202.0_EN.pdf)
- [35] "LibeTech QR Code Door Lock," *JeremyBlum.com*. 21-May-2012 [Online]. Available: <http://www.jeremyblum.com/portfolio/libetech/>
- [36] "BeagleBoard.org - bone." [Online]. Available: <http://beagleboard.org/bone>
- [37] "ZBar bar code reader." [Online]. Available: <http://zbar.sourceforge.net/>
- [38] M. Security Solutions, "QR code based access control," 09-Dec-2015. [Online]. Available: <http://www.matrixsecusol.com/product-features.html>
- [39] "Wireshark · Go Deep." [Online]. Available: <https://www.wireshark.org/>
- [40] "R: The R Project for Statistical Computing." [Online]. Available: <https://www.r-project.org/>
- [41] T. Lewis, "IPsec VPN vs. SSL VPN: Is Your Remote Access VPN a Liability?," *Mirazon*. 26-Jan-2017 [Online]. Available: <https://www.mirazon.com/ipsec-vpn-ssl-vpn-is-your-remote-access-vpn-a-liability/>. [Accessed: 13-Jan-2018]
- [42] Rapidtables.com, "How to convert watts to lux (lx)," *Rapidtables.com*, 2017. [Online]. Available: <https://www.rapidtables.com/calc/light/how-watt-to-lux.html>. [Accessed: 14-Jan-2018]

- [43] Lampgrossen.se, “Jämförelsetabell från Watt till Lumen,” *Lampgrossen.se*, 2017. [Online]. Available: <https://www.lampgrossen.se/sida/jamforelsetabeller>. [Accessed: 14-Jan-2018]
- [44] “Comparison of encryption ciphers in Java,” 2012. [Online]. Available: <https://www.javamex.com/tutorials/cryptography/ciphers.shtml>
- [45] D. S. A. Elminaam, H. M. Abdual-Kader, and M. M. Hadhoud, “Evaluating the performance of symmetric encryption algorithms,” *IJ Network Security*, vol. 10, no. 3, pp. 216–222, May 2009.
- [46] M. Baches, “Block Ciphers - feistel function,” Saarland University, Literature, 2007 [Online]. Available: <http://web.cs.du.edu/~ramki/courses/security/2011Winter/notes/feistelProof.pdf>
- [47] Oracle, “Cipher (Java Platform SE 7 ),” *Oracle help center*, 2017. [Online]. Available: <https://docs.oracle.com/javase/7/docs/api/javax/crypto/Cipher.html>. [Accessed: 14-Jan-2018]
- [48] Oracle, “Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files for JDK/JRE 8 Download,” 2017. [Online]. Available: <http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html#close>. [Accessed: 10-Jan-2018]
- [49] Dell, “How Random Access Memory (RAM) affects performance | Dell Sverige,” 08-Mar-2017. [Online]. Available: <http://www.dell.com/support/article/se/sv/sebsdt1/sln179266/how-random-access-memory--ram--affects-performance?lang=en>. [Accessed: 10-Jan-2018]
- [50] Android Developers, “Manage Your App’s Memory | Android Developers,” 2017. [Online]. Available: <https://developer.android.com/topic/performance/memory.html>. [Accessed: 10-Jan-2018]
- [51] NXP Semiconductors, “Corporate access,” *MiFare*, 2017. [Online]. Available: <https://www.mifare.net/en/solutions/mifare-applications/access/corporate-access/>. [Accessed: 09-Jan-2018]
- [52] R. Poovendran and J. Lee, “The AES-CMAC Algorithm,” *Request For Comments*, Jun-2006. [Online]. Available: <https://tools.ietf.org/html/rfc4493.html>. [Accessed: 09-Jan-2018]
- [53] N. T. Courtois, K. Nohl, and S. O’Neil, “Algebraic Attacks on the Crypto-1 Stream Cipher in MiFare Classic and Oyster Cards,” University College London UK, University of Virginia USA, VEST Corporation France, 166, 2008 [Online]. Available: <http://eprint.iacr.org/2008/166>
- [54] NXP Semiconductors, “SmartMX for programmable, high-security, multi-application smart cards,” Netherlands, Jan. 2014 [Online]. Available: <https://www.nxp.com/docs/en/brochure/75017515.pdf>. [Accessed: 09-Jan-2018]
- [55] Tech4 Helper, “OpenCard hack (projekt chameleon),” 05:51:48 UTC [Online]. Available: <https://www.slideshare.net/lockeracid/opencard-hack-projekt-chameleon>. [Accessed: 09-Jan-2018]



## Appendix A: code snippets

```

void encrypt(String CIPHER_TRANSFORMATION, SecretKey SECRET_KEY,
byte[] PLAIN_TEXT){
    Cipher cipher = Cipher.getInstance(CIPHER_TRANSFORMATION);
    cipher.init(Cipher.ENCRYPT_MODE, SECRET_KEY);
    long start_time = System.nanoTime();
    cipher.doFinal(PLAIN_TEXT);
    long end_time = System.nanoTime();
    ...
}
...

```

**Code Snippet 1: Measure time for encryption test**

```

...
void doSomeWork(){
    long start_time = System.nanoTime();
    /*Asynchronous I/O interruption here - thread will be paused for 100 ns*/
    calculatePowerOfX(32);
    long end_time = System.nanoTime();
    ...
    start_time = System.nanoTime();
    doSomeMoreWork();
    end_time = System.nanoTime();
    ...
}

void calculatePowerOfX(int x){
    int y = 2^x;
}

void doSomeMoreWork(){
    byte[] bytes;
    List<byte[]> list = new ArrayList<>();
    for(int i = 0; i < 256; i++){
        bytes = new byte[1024*1024]; /*trigger malloc multiple times */
        ...
        list.add(bytes);
    }
}

```

**Code Snippet 2: running time affected by possible malloc calls**





## Appendix B: Equations

### Equation 1: Determine max payload size of an RSA encrypted content.

Determining encrypted data payload size depending on chosen encryption padding length in bits and key length in bits. The available paddings from cipher algorithms (that were supported by both sides of communications) are:

- PKCS1Padding
- OAEPWithSHA-1AndMGF1Padding (SHA length is 160 bits)
- OAEPWithSHA-256AndMGF1Padding (SHA length is 256 bits)

```
if algorithm == PKCS1:
    payload_size = (key_length/8) - 11;

else:
    payload_size = (key_length/8) - 2*(SHA_length/8) - 2;
```

### Equation 2: Determine QR content by encryption limits and content length

```
if content_length <= payload_size:
    return B64Encode(Encrypt(content));
else:
    chunk, rest = subContent(content, 0, payload_size);

    list.append(B64Encode(Encrypt(chunk)));

    while(rest > 0){
        chunk, rest = subContent(rest, 0, payload_size);

        list.append(B64Encode(Encrypt(chunk)));
    }

    return list;
```

TRITA-EECS-EX-2018:1