

IT Security Lab Report

by

Anuschka Clasen
Thomas Maqua
René Neff

Institute of Computer Science IV
Work Group IT Security
Supervised by Dipl. Inf. Matthias Wübbeling
Examined by Prof. Dr. Michael Meier
Rheinische Friedrich-Wilhelms-Universität Bonn

April 30, 2015

ABSTRACT

This report explores three modern technologies and their combined power to enhance usability and security: Google Glass, QR codes and NFC. Four distinct ideas which leverage these innovations are presented: Firstly, a TAN device that uses NFC authentication as a second factor. Secondly, encrypted QR codes to combat shoulder-surfing while browsing the internet in public. Thirdly, a method to store financial transactions in cryptographically signed QR codes. Lastly, a new approach to handle money withdrawal on ATMs which uses NFC authentication as well. An extensive concept that combines these four ideas is presented alongside its implementation: This includes two android applications - one to be used on a smartphone and one to be used on google glass, an online-banking website and a mocked ATM. Within the evaluation, improvements in usability and security in comparison to common alternatives are presented and discussed.



CONTENTS

1	INTRODUCTION	1
2	BACKGROUND	2
2.1	Security Goals	2
2.2	Security of Banking	6
2.3	Nearfield Communication	10
2.4	Quick Response (QR) Codes	15
2.5	Google Glass	17
2.6	Public-key cryptography	18
2.7	Transaction Numbers	21
2.8	Compact Transaction Data Format	24
2.9	Bluetooth	27
3	PROSPECT	30
3.1	NFC TAN Device	30
3.2	Encrypted QR Codes	33
3.3	Signed QR-Codes for secure transactions	34
3.4	Mobile Device ATM	35
3.5	Combined System Concept	40
4	CONCEPT AND IMPLEMENTATION	43
4.1	QR-Codes	44
4.2	Server	52
4.3	App	61
5	EVALUATION	70
5.1	Security	70
5.2	Usability	72
5.3	Summary and Conclusion	73
A	ADDITIONAL IMAGES	I
A.1	Online Banking Website	I
A.2	Mock ATM	VIII
B	BIBLIOGRAPHY	X

1 INTRODUCTION

Today's modern smart phones include more and more features, new wearable technologies are presented, e-commerce is booming new purchase opportunities arise. Nevertheless, introduction of new technologies still takes its time to change established forms of use. [29] Within this lab three arising technologies are used to create a more secure and user-friendly system.

Beginning with the introduction on necessary background knowledge, a roundup on security aspects in terms of security in general but also in the more specific case of IT security is given. Following this a description of two communication technologies is laid out, this is *Nearfield Communication (NFC)* on the one hand and *Quick Response Codes (QR codes)* on the other. Both technologies become popular and wider spread within daily life, as more and more smart phones are equipped with cameras and NFC capability. Speaking of new technologies, *wearables* are the new hype of the latest years, gaining a lot of public's interest with the presentation of *Google Glass*. [28] A short presentation of Google Glass within this lab report will provide useful information about the hardware- and software capabilities of this new wearable. The next section about *Public Key Cryptography* sums up important cryptographic background information, which will be needed in various parts in the later work. To make use of already accepted concepts, an introduction to Transaction Numbers and how they are used in today's banking systems follows suit. The background chapter closes with the description of official standards for financial transactions.

Backed up with this base of knowledge the four main separable ideas of this lab are presented. At first the idea of *signed QR codes* which will allow for a cryptographic signature within a QR code. Next is the idea of *encrypted QR codes* which can be used to conquer shoulder surfing and other attacks. With the *NFC TAN Device* an alternative to existing TAN generators and distribution is presented. Finally the *Mobile Device ATM* a new user-friendly but more secure idea on ATM cash withdrawal is introduced.

Besides the conceptual design each idea found its way into the four implementations presented within this lab report. The products of these implementations are: an online banking website, a mocked ATM (website), an Android smart phone application and a Google Glass application. Combined they provide a viable proof of concept, nevertheless each concept and its implementation is explained and presented within the *Concept and Implementation* chapter.

With a final evaluation regarding improvements in security and usability this lab report concludes.

2 BACKGROUND

This chapter aims to provide an extensive background analysis on various topics which are important for this lab's main work. At the beginning, it will expand on security aspects, both for IT in general and for banking in particular. Next, the two communication technologies *Nearfield Communication* and *Quick Response Codes* will be looked upon, followed by a description of the *Google Glass* mobile device. The *Public Key Cryptography* section sums up important cryptographic background information which will be needed at various parts in the later work. The next section will explicate *Transaction Numbers* and how they are used in today's banking systems. At the end, official standards for financial transactions will be examined to prepare a compact transaction data format.

2.1 SECURITY GOALS

2.1.1 CIA TRIAD

The FIPS Publication 199 describes three different main goals of security, also called objectives of security [75]. They are relevant for dealing with information over all as well as with information systems. Those three goals are commonly known as the CIA triad of information security, not because of any relation to the central intelligence agency of the United States but due to the initials of the goals: *Confidentiality*, *Integrity* and last but not least *Availability*. [20]

A look at the threats accompanying the named goals explains them as following: In case of loss of confidentiality an unauthorized disclosure of information occurred. Therefore the goal of confidentiality covers the protection of personal privacy and information against undesired access. Security objects are in particular stored or transferred data. In banking context any information about account balances as well as transfer, ownership and related are potential targets of attacks if confidentiality is not sufficiently guaranteed. All in all theft of information is one big violation of confidentiality. One very widespread solution to protect confidentiality of a system is encryption. To ensure information confidentiality, access controls and file permissions are other options to restrict the access to sensitive information. [16]

The second part of the CIA triad, integrity, concerns the modification or destruction of information and therefore their authenticity. Integrity therefore aims to leave everything as it should be because no unauthorized change is allowed [7]. Stored information also

needs to be protected against deletion or irrevocable, accidental or undesired changes. It is useful to distinguish two different forms of integrity into the system's integrity on the one hand and data integrity on the other one. System integrity is meant in case of any modification which concerns the proper operation of a system [7]. For example the correct functioning of a banking transaction or an ATM (Automatic teller Machine) is aimed with this security goal. The integrity of data is covered, if the displayed, stored or used data is safe against any unauthorized changes. In this matter the ATM not only has to work properly but the data has to be protected additionally.

Integrity violations can be caused by malfunctions (hardware as well as software), inadvertent user errors and of course malicious activities [68]. One example for hardware malfunction are bit errors which can cause the system to overwrite files. Data corruption can for example be a result of software bugs and can lead to inaccessibility of files. Another corruption problem are unreliable networks which modify data passed through [68]. One problem with integrity violations is that most systems with integrity issues often cannot detect the data modifications due to missing integrity assurance mechanisms. An example in the banking area for violated integrity is if someone is trying to send an online money transfer for 50 Euro, but actually sent were 500 Euro, due to tampered information.

Integrity can be protected by several means, for example using message authentication codes or digital signatures. Cryptographic techniques in general are a way to provide a system's integrity. Another protection possibility is to protocol every actions which are of further relevance for data or user. This recording has to be secure for revision itself. Digital watermarks are another measure, as well as using hash functions to compare the hash of original messages with the one someone receives.

The last letter of the CIA triad reveals its purpose more apparently: Availability ensures the timely and reliable access to information as well as to its use. In case of loss of availability it is not possible to access a system in the intended manner. Different cases can cause problems with a system's availability, for example simple failures of the equipment. Natural disasters like fires or earthquakes can lead to problems and loss of system capabilities as well. And of course intentional attacks as well as undiscovered flaws prepare the ground for Denial of Service. A way to protect availability is the redundant storage of resources. [43]

2.1.2 'FIVE PILLARS OF INFORMATION ASSURANCE' AND OTHER PROTECTION TARGETS

The U.S. Department of Defense (DoD) added two more goals to the CIA triad to form the 'Five Pillars of Information Assurance'. Information assurances, in this context, are measures that protect and defend information (and systems) by insurance of those pillars. [47]

The first additional goal is named *authenticity*, and describes the data or system as being authentic and correct. Therefore an authentic system is verified so its attributes are correct and not fabricated. Techniques to protect the authenticity of correctness of the identity has to be proven, for example via certification, keys or passwords. [20]

Non-repudiation as fifth pillar is not as easy to understand on first glance as the others. It is often also named *privacy* protection target and refers more to anonymity in this context. In the original source it is again described more as property than a goal, assuring the completeness of an analysis and that it cannot be repudiated. It describes the ability to correlate a recorded action with its originating individual or entity. Violations of this attribute may cause unauthorized manipulation of transaction logs making it more complicated or impossible to either identify or prove the initiator of a performed action. A more trivial example may be the multiple use of one user account by different users. As a result it is impossible to proof who was logged on in person at which time. [23]

In some other definitions of security goals the CIA triad targets are extended by the term *accountability*. Accountability in this context ensures that actions can be traced to the executing entity. Therefore it is guaranteed that all operations which are carried out by individuals, systems or processes can be assigned to their originator. As a consequence traceability is given. To require accountability it is appropriate to log actions and timestamps and to protocol any action of security relevance. A further possibility is the use of digital signatures. [43]

2.1.3 SECURITY GOALS AND PRINCIPLES

It is rarely possible to protect each and every defined security goal or pillar in the same way. Interactions between those can be problematic, as measures taken to further the goal of one pillar are often blind to the needs of another one. For example a very strict insurance of confidentiality can violate the availability of information and vice versa. As a result it is important to determine the most important protection targets with respect to the specific project. For that reason suitable technical or organizational measures have to be defined as well as action guidelines as security policies. Those measures, in order to ensure the security goals, aim to a certain security level which fits to the project's circumstances and requirements. As an example, password policies can be one possibility. In such a policy it could be defined that system level passwords have to be changed on at least a quarterly basis whereas user level passwords only must be changed every six months. [23]

To achieve a proper security level the National Institute of Standards and Technology (NIST) defined 33 principles for information technology. Those principles shall be considered during the development and operation of an information system to develop the security policies and create secure systems. Again not all principles apply to all systems at all times. One of the first principles states that security has to be treated as an integral part of the overall system design, but the strive for simplicity is another important principle. Moreover, the principles encourage the developer to restrict the security mechanisms to a necessary limit. NIST describes a mechanism as appropriate, if it is necessary to accomplish one or more goals of security. If the goals are supported adequately without the corresponding, feature the adding of it would create an unneeded system complexity. Following those principles if possible leads to secure system designs which protect the security goals which are of elementary importance for the project.

Some principles aim directly in the protection of one certain security goal, as for example principle 33, which ensures accountability using unique identities. [76]

2.1.4 AUTHENTICATION TECHNIQUES

Regarding authentication three basic distinctions are made between three main protection methods. Those methods are based on “knowledge”, “ownership” and “biometrical identifiers”. Techniques in the class of knowledge are for example passwords or PINs as well as cryptographic keys. The ownership of something can authenticate someone as well, like the use of a SIM-card or smart cards in general. In the category of identifiers authentication techniques using biometric identifiers are the most important factors. Examples are fingerprint scans or iris capture.

In case of authentication based on knowledge the user needs to know certain information to authenticate himself towards the system. One possible mechanism is the challenge response procedure (CR). The idea is to use an authenticity check to login, whereby the user first identifies himself via his user name, MAC address or something similar. The instance responds by sending a challenge - e.g. a random number - and the user calculates his response based on it. The instance again checks the calculated response and therefore if the user was in possession of the correct knowledge. This method is based on a pre-shared secret (the key to calculate the response). Variations of this technique are one time passwords also known as OTPs (RFC 2289) or time based OTPs, the so called TOTP (RFC 6238).

An example for ownership based authentication is the use of the “Elektronischer Personalausweis” (nPA) in Germany. With this smartcard passport the user is able to authenticate himself on terminals or online with the contact free RFID chip. The card can be used as digital signature, this possibility is named eID function. The user has to be in possession of the nPA to authenticate himself.

Authentication using biometric identifiers is consequently bound to one person. The chosen identifiers shall be tamper resistant and precise. Two different classes of identifiers exist: the physiological identifiers, which stay static in most cases, and on the other hand behavioral characteristics. The latter are dynamic and change depending on the action. Examples are keystroke models of the user, personal signature style or voice. Iris, fingerprints or face, by contrast, are considered as static characteristics.

Multifactor authentication combines elements of the previously described features. One commonly known example for two factor authentication is the log in process on mobile phones: the user owns the SIM card (category ownership) and knows the PIN (category knowledge). Of course the use of an ATM to get money is another example for two factor authentication, as the user needs to be in possession of his bank card and the knowledge about the PIN. [23]

2.2 SECURITY OF BANKING

In the recent decade internet banking became more popular and in the last years especially the number of mobile banking subscriptions increased. In 2013 a total amount of almost 500 million sessions have been counted for mobile and internet banking sessions altogether, whereas mobile banking more than trebled their share compared to 2012 from 360,000 subscriptions up to more than one million. The number of internet subscriptions shows nearly linear growth for the last decade, increasing by about one million subscriptions each year to a little bit more than ten million in 2013.

Together with the counted subscriptions and sessions the available services multiplied over the years. Nowadays not only inspecting the balance but also manage of savings, money transfer, open new accounts and furthermore is possible online. Going hand in hand with the increased use of online banking are new possibilities for fraud in the context of banking.

It is not easy to find reliable sources regarding fraud in the banking area for Germany, Europe or Worldwide. Internet banking fraud cases increased in the past year in the Netherlands for an instance with a peak in 2012 with 1,003 registered fraud cases overall [64]. Almost three million Euro have been lost due to those fraud. Phishing is stated as the most frequently used technique. Besides skimming, shoulder surfing, man-in-the-middle attacks and compromised computers are used techniques to attack someone's banking account, to name only a few. In this chapter the most common attacks to the security of banking are presented and discussed. [64]

2.2.1 MAN-IN-THE-MIDDLE ATTACK

To perform a *man-in-the-middle attack*, the attacker has to be able to both monitor and alter the communication of his victim and a third party (in our case a banking institute's website). This is usually done by compromising the victim's computer with malware, as nearly all banking websites require the communication to be *https* encrypted, thwarting all *man-in-the-middle attacks* which would be possible by hacking e.g. a router that is used for the communication of Alice with the banking website. [34]

An exemplary scenario in which the user Alice tries to transfer an amount of money to Bob, but is tricked into transferring money to Mallory instead, will be outlined in the following. This attack scenario requires Alice to use a *TAN* (see section 2.7) system that is vulnerable to man-in-the-middle attacks. The description will focus on the case of *indexed TAN lists*, as this technique is still widely used. A visualisation can be seen in Figure 1.

Alice is using an untrusted computer, for example in an internet café, which has been previously modified by Mallory. She installed a custom browser which is visually indistinguishable from a commonly used browser. However, this browser is modified to route all traffic over a computer which is controlled by Mallory, too, without reporting any warnings about insecure *https* connections to the user. Thus, the latter falsely assumes that the communication channel is secure.

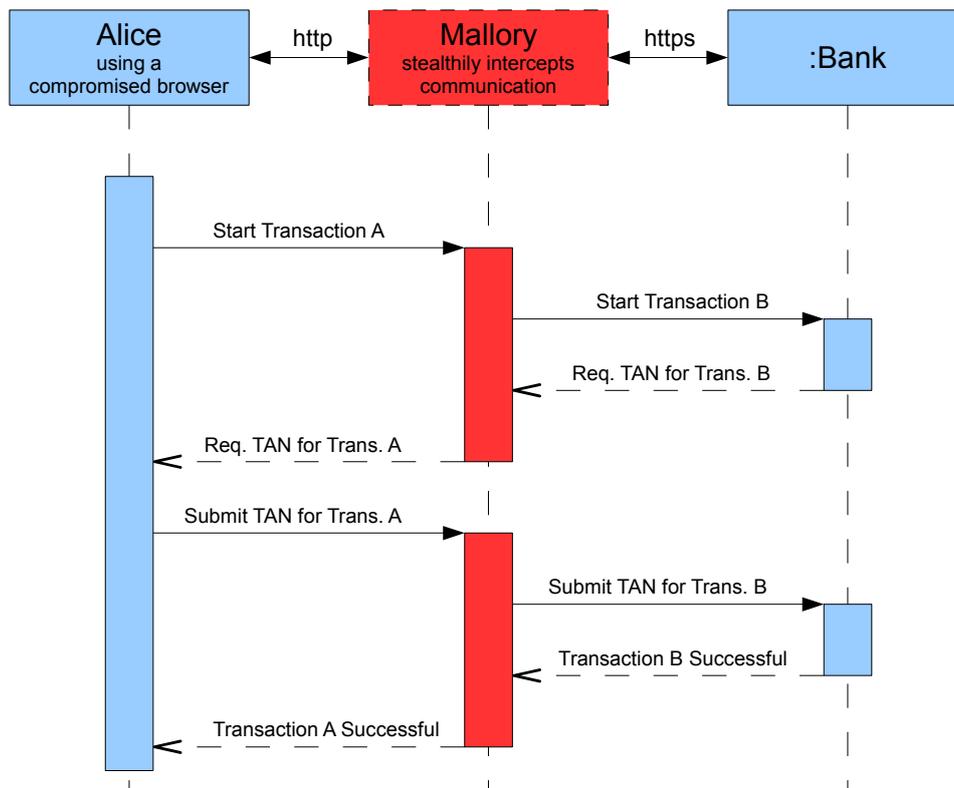


FIGURE 1: A sample man-in-the-middle-attack. The communication between Alice and her banking institute is intercepted and modified by Mallory.

Alice logs into her online banking website and fills out a formular with the intention to transfer money to Bob. Mallory simultaneously sends a request using Alice’s connection with himself as the recipient. He intercepts both responses, and forges a new one which will be sent to Alice. This new forged response is largely identical to the response to Alice’s request, but it asks for the *TAN* which is used in order to complete Bob’s request. Alice does not notice anything unusual, and it is not possible for her to detect the forgery of her response. Thus, she feels save to enter the requested *TAN* from her list. Bob intercepts her message to the website, and uses the *TAN* to complete his own transaction instead. Optionally, he even modifies the website’s content to decieve Alice into thinking that her transaction was completed successfully.

2.2.2 PHISHING

One well-known fraud attack is phishing. The name phishing is created from the words password and fishing and describes the method quite good: the fraudster fish for the passwords, credit card numbers or other sensitive data of the user like they would do for fish, using masses of mails or faked websites instead of rod and line. [72]

The information is typically gathered using an e-mail or website, but telephone calls, SMS, vicious software and social media advertisements are less used but nevertheless significant entry points for phishing. Using the stolen data criminals try to steal the user's money or buy products – mostly online – with the credit card information. The data and codes could also be used to steal and abuse someone's identity, for example to contact the bank or company in order to get access to the account or open new ones (with higher credit level).

A phishing attack is often separated into two levels: the contact level and the spoofing level, used together in most cases but sometimes alone as well [64]. During the contact level the victim is contacted, exploiting a relationship of trust and confidence. Examples are attackers who mime to be employees of the bank or fake websites which state to be the original banking website itself. Typically emails are used to contact the victim, as emails can be massively send and generated automatically. In the past those mails were easy to detect as phishing attack, because the text was not written in grammatically faultless form. Nowadays, with better translation engines, those mails are not as easy to distinguish from original mails of someone's bank. The mails either aimed to gather the information directly by asking for it – e.g. prompting to ask for out of security reasons – or led the user to websites, except malware was not attached to the mail already. The second level is the counterfeit website itself. The method is called spoofing. On those websites the victim is again asked to disclose information and sensitive data. [14]

A 2006 published study done by researchers of Universities of Harvard and Berkeley about the question why phishing works, illustrates users' problems to distinguish spoofing websites from real ones. Even in the best case, when the users are aware of the threat of phishing and actively trying to acknowledge fraud, good phishing websites could fool up to 90 percent of the study's participants. This underlines how much risk is given by phishing. [21]

2.2.3 SKIMMING

Skimming is a type of fraud and a special case of a Man-in-the-middle attack. This term summarizes the method of illegal electronically exploration of banking data on automated teller machines (ATMs), one kind of skimming is the capture of data from the magnetic stripe on the back of an ATM card. Another way of skimming is to attach cameras to the ATM to fraudulently capture the PIN numbers. The captured data can be put onto a manipulated or false card and money can be withdraw from the accounts all over the world. Skimming is not only limited to ATMs, instead any kind of cash register or card reader could be victim of a skimming attack. Even if the attackers were not able to gain access to someone's PIN it is possible to use for them, as for example online retailers do not ask for any PIN or other security codes. [27]

In 2010 the number of skimming attacks reached a record high rate in Germany. In the following years the amount of attacks decreased significantly but nevertheless the stolen money summed up to more than 35 million EURO in 2011 – only in Germany. 150.000 cards had to be frozen due to skimming. [63]

To skim – or scam – different areas on an ATM criminals can extend it with different devices. It is hard for the user to reduce the risk of skimming as ATM look different from bank to bank, why it is difficult to suspect anything unusual with the card reader or the area around. Even if the user’s hand shields the PIN area during the entering – and therefore reduce the risk of shoulder surfing – , an overlaid skimmer plate over the existing keypad as a method of PIN capturing and track the digits anyway. Fake card readers are another typical technique to capture information about a customer’s account data and to read out the information stored on the credit card. [27]

2.2.4 COMPROMISED COMPUTERS

Compromised computers can be affected by a virus, malware or corrupted system files. Those infections can occur due to successful phishing attacks or other emails or websites, which aim to infect the user’s computer. The aim of compromising computers is to gain control over the infected computers in order to use them in botnets, or – more relevant in the security of banking context – to obtain sensitive information like passwords, account information, TAN or access data in general. Mobile banking on smartphones or tablets is a risk in the same way as online banking on computers, a fact lots of users ignore or do not know. Only a third of mobile users are aware about security measures which are needed on mobile devices according to the Bundesamt für Sicherheit in der Informationstechnik (BSI). Attackers could use those vulnerabilities to send faked SMS with malware or get administrative access to the smartphone, to name only a few examples for attacks or fraud. [13]

Examples for malware and infected systems are widely spread. Some of the best known are man-in-the-middle attacks, as described in chapter 2.2.1. Man-in-the-browser attacks in contrary do not attack the data flow between computer and banking sever but manipulate the visual presentation of the user’s online banking website in the browser. If Alice as example user types in the address of the website into the browser of a compromised computer, a usual connection will be set up. The malware opens the correct website but displays manipulated content, asking for TANs, PINs or other credit card information. Such manipulations were used to lever out chipTAN techniques which seemed secure so far. [13]

2.2.5 SHOULDER SURFING AND OTHER FRAUD TECHNIQUES

The term shoulder surfing refers to someone watching over somebody’s shoulder to capture a password or PIN, while the user enters it. Alphanumeric passwords are vulnerable to shoulder surfing as well as graphical passwords [89]. In the banking context especially PINs are a common target of an attack, as they consist only of four digits. Typically the ten digits are arranged in the same way on a keypad, why the capture of hand movements could be enough to guess the PIN itself. Typical attacks in the area of cash dispensers are done by someone disrupting – often kindly – the user during the process of the money withdrawal after looking over his shoulder and getting the information about the PIN [64]. During the disruption the fraudster would try to

steal the bank card as well and therefore get all information needed to withdraw money from someone else's account.

Besides the capture of passwords over the shoulder surfing is a security risk as well if sensitive data is displayed in public. Examples are balances of banking accounts which a user wants to check in public as well as account information in general like account numbers. Additionally the term shoulder surfing is used to describe observation techniques which directly aim to get information. Therefore it is not necessary in every case to stand aside someone to capture the wanted information, binoculars or other vision enhancing devices could be used to do long distance shoulder surfing. [62]

Other vulnerabilities in context of banking are smartphones used for online banking in general. PINs or TANs are often stored on the device to prevent the user against forgetting it. MobileTAN methods often only base on the phone number as security aspect and could therefore easily be compromised if the smartphone is stolen. The authentication on banking apps is also often only based on one password – or the PIN itself – and the data stored on the phone could easily be gathered if the smartphone is compromised. To use publicly accessible computers or networks is another vulnerability if the user wants to do online banking in public. [13]

2.3 NEARFIELD COMMUNICATION

Near Field Communication (*NFC*) is a standardized communication technologies that enables radio communication between two devices within a small proximity to each other. Since 2011 NFC is becoming more and more popular within media and public perception [31], by now nearly 300 different smart phones support NFC [42]. Introduced by a summary on the development of NFC, this following part will give a technical overview on the concept of NFC and the NFC Data Exchange Format. Throughout emerging and already in use use cases are presented and finally some possible security problems are discussed.

Several standardization bodies already passed norms and standards to allow a broad use of NFC: ISO 18092 [38], ECMA 340 [74], ETSI TS 102 190 [26]. Those standards facilitate the radio-frequency identification (*RFID*) technology, which only describes the broad field of using electromagnetic fields for identification, as printed barcodes do too. Barcodes, as they are printed on almost every packed product in a supermarket, allow for a easy scanning and thereby automated processing for example at the checkout. [90] Yet they the product needs to be placed in the correct way in front of the scanner to function properly, with RFID only some kind of proximity needs to be arranged. Until 2002 RFID lacks a global standard, this changed when the NFC standard was developed by NXP and Sony. [54]

Today ongoing development is done by the *NFC Forum* a non-profit industry association which assembles over 100 different companies and institutions [50]. This led to a wide spread and still growing acceptance and availability of NFC [58].

NFC's bidirectional communication allows a maximum communication speed of 424 kbps over a distance of less than four centimeters [49]. Communication can be arranged between self-powered devices (active mode) as well as non self-powered devices (passive mode), such as smartcards [26]. Communication is done at a radio frequency of 13.56 MHz which is part of the international industrial, scientific and medical radio band (ISM), which can be used free of charge [44]. Additional communication speeds are 105 and 212 kbps, all of them are supporting different operating modes: Card Emulation Mode, Peer-to-Peer Mode and Reader / Writer Mode.

In passive mode a so called *tag*, which can be a sticker carrying a small NFC chip, can be powered by the initiating active NFC device. Necessary power is provided by an RF field generated by the active NFC device which therefore has to be in close proximity to the passive tag. Communication in such a scenario would be handled in the Reader / Writer Mode, a typical use case would be reading contact information from a business card. [49]

Similar to this use case NFC devices can emulate smart cards used for banking or access control, in such a scenario the Card Emulation Mode would be used. This mode allows for contact free interaction with already established infrastructure and still preserving expected security requirements. Card Emulation Mode is also done by interaction between an active and a passive NFC device. [49]

In contrast to those use cases a third way of interaction has been standardized: Peer-to-Peer Mode. It is used when interaction between two active devices, for example two smart phones, is desired. [49]

Message exchange is done in the NFC Data Exchange Format (*NDEF*) which specifies exactly how data has to be arranged when stored or sent by a NFC device or communication with one [48] [53]. A single NFC message, for example the content of a NFC tag, can be divided into single records, called NDEF records, which then hold a header and payload part, as shown in figure 2.3. [39]

The header is separated into three parts, leading with a part of header flags which define the following header fields. Those fields specify type, length, other record attributes and finally provide a identifier, as shown in figure 2.3. Within the first byte of the header, the header flags, it is specified whether this record is part of a continuous message. Additionally it can be stated if the record's payload was spread into multiple parts, so called *chunks*; a single bit indicates if this record starts, continues or ends such a chunked payload. If multiple records are present an identifier is needed to arrange and select them, in such a case an identification (ID) field is present and provides each record with a unique ID. This is the case if a longer payload is sent. If only a short record is sent a single bit indicates that this record is complete and thereby a single data record. Further more it can be specified if the following payload is a small one and therefore additional payload length fields can be omitted, in such a case the short record flag is set. Finally three bits of the first byte within the header are used to select one of eight predefined payload types:

The empty type is regularly used for new tags and devices as every NDEF compatible device has to have at least one record. [48]

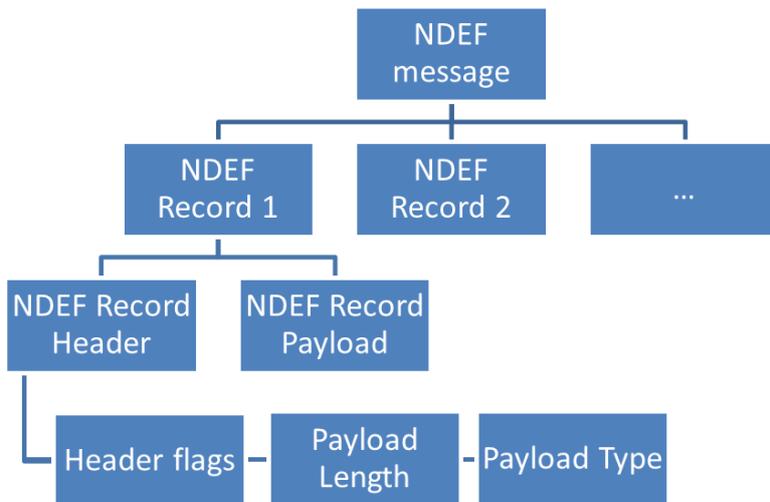


FIGURE 2: Structure of a single NDEF message.

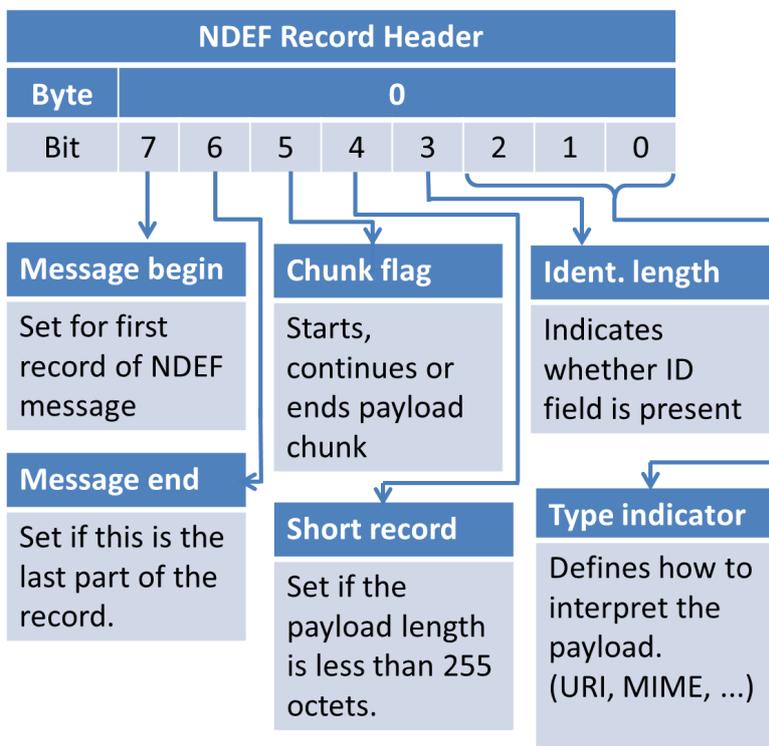


FIGURE 3: First byte of a NDEF header.

Byte	1
Type length	Holds length in octets of the Type field. An unsigned 8-bit, which can be zero for some Type indicator cases.
Byte	2
Payload length	Holds length in octets of the actual Payload field (app. data). One up to four unsigned 8-bit, length may vary if a short record was defined .
Byte	3
ID length	Length of the ID field.
Byte	4
Payload Type	Type which actual gives information about the payload if it can be defined by RTD type.
Byte	5
Payload ID	An identifier which allows to identify multiple payloads to be associated or referenced to.

FIGURE 4: Bytes 1-X of a NDEF header.

TABLE 1: NDEF predefined payload types

Bit value	Type	Description
0	Empty	Has to be used if the following record is empty and has therefore no type or payload.
1	Well-Known	Used if a pre-defined type specified in the NFC Forum RTD specification is used. [51]
2	MIME media-type	Set if an Internet media type as defined in RFC 2046 is used. [46]
3	Absolute URI	Set if an URI as defined in RFC 3986 is used. [84]
4	External	Used if a value based on rules in the NFC Forum Record Type Definition specification is used. [51]
5	Unknown	Used if type is unknown. In consequence the type length must be zero.
6	Unchanged	If this record is part of a chunked payload and not the leading part this type is set, type length must be zero again.
7	Reserved	Reserved by the NFC Forum for future use.

Source: [48]

With his paper [45] Collin Mulliner presented some possible attacks on NFC tag reading phones and pointed out some, by now fixed, bug within the handling of read tag data. As t Overall his research conducted NFC tags may hold malicious data and NFC could be used to create a NFC worm. More on a technical point of view Ernst Haselsteiner and Klemens Breitfuß propose that eavesdropping a NFC communication can be done with in one meter range even if one of the two communicating devices is a passive one, for to active components this extends to ten meters [33]. To counter such attacks a secure channel for communication is proposed, as NFC communication is not encrypted or checked for integrity of exchanged data. By using modern NFC (debit) cards this is taken into consideration, as they use especially designed techniques to comply with the EMV specifications while they exchange data [24].

2.4 QUICK RESPONSE (QR) CODES



FIGURE 5: QR Code which includes the URI for *qrcode.com*

A QR Code is a two dimensional, or matrix, barcode designed by Denso for Toyota to mark Products in their stock [60]. As shown in figure 5 it consists of a square white field where smaller black squares placed in a grid. The structure of a QR Code is explained in Section 2.4.1. The coding includes a error correction and is variable in how many percent of faulty reading can be corrected, as described in Section 2.4.2. Also it can include up to 2953 byte of data, depending on its size, which is described in Section 2.4.3. All information about the definition of QR Codes can be found in ISO/IEC 18004:2006 at [37].

2.4.1 STRUCTURE

As shown in figure 6, in three corners of the QR Code a Position pattern is placed, to help the reader to identify the correct orientation of the QR Code. For bigger sizes of QR Codes more position pattern will show up in the big data part in the lower right to help recognizing the right orientation. Further there are two lines of alternately black and white dots between the tree position patterns to define positions inside the matrix grid. In the blue parts of the shown QR Code the error correcting level and the data mask pattern is described which is shown in Section 2.4.2 and 2.4.3. The red squares specify the version of the shown QR Code which is explained in Section 2.4.3. At last there are the yellow data parts where the data and the error correction is stored. While the error correction is described in section 2.4.2 here is shown how many data can be stored with which encoding. If the numeric encoding is choosen, it is possible to store up to 7089 characters and each 3 are encoded in 10 bit. In alphanumeric encoding 2 characters are encoded into 11 bit, while it is possible to store 8 bit encoded data with up to 2953 Byte

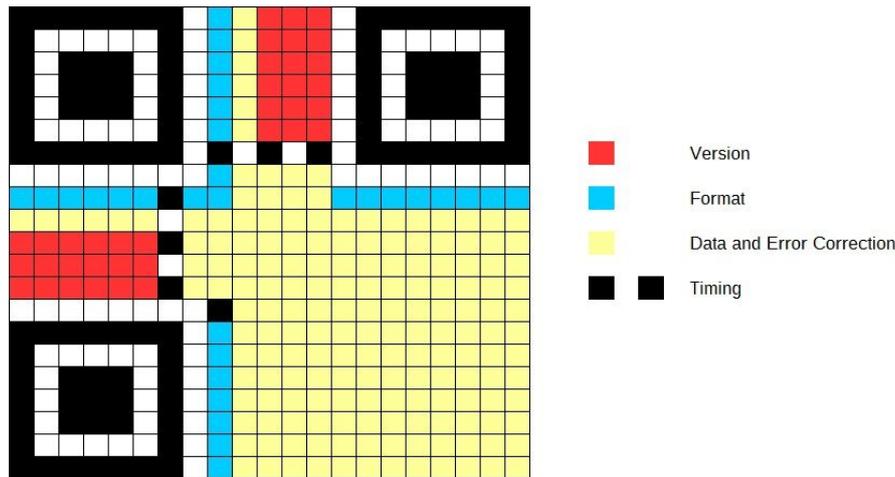


FIGURE 6: General Structure of a 21×21 pixel Version 1 QR Code

of data. It is even possible to encode kanji, Chinese characters which are used in the Japanese language, with up to 1817 characters in 13 bit per character. [37]

2.4.2 ERROR CORRECTION

For error correction Reed-Solomon error correction is used with 8 bit long codewords and four different correction levels. There are the following correction levels:

1. level L: About 7% or less errors can be corrected.
2. level M: About 15% or less errors can be corrected.
3. level Q: About 25% or less errors can be corrected.
4. level M: About 30% or less errors can be corrected.

The Reed-Solomon error correction describes a way to detect and correct multiple symbol errors. If t symbols are added to the source symbols t errors can be detected and $t \text{ DIV } 2$ errors can be corrected [83]. Because of the error correction it is possible to decode a QR Code with a percentage of misread bits correct. To provide this ability it is necessary to sacrifice a part of the storage capacity. [37]

2.4.3 CAPACITY

The data capacity increases by size of the QR Code, as it increases its version number. It starts with version 1 at 21×21 pixel and rises for each version number increase of 1 for 4 pixel. So version 2 is 25×25 pixel and version 40 177×177 pixel which is the maximum size of a QR Code. This maximum size is able to hold the in Section 2.4.1 described number of bytes. So it is possible to save 2953 byte of binary encoded data in a 177×177 pixel QR Code which have to be presented quite big to read it properly, because of limited resolution in display and limited resolution of pictures taken of the QR Code. In the following table the level of a QR Code in connection of modules at this level and its capacity in bytes at each of the four correction levels is shown. [37]

version	modules	L	M	Q	H
1	21X21	17	14	11	7
10	57X57	271	213	151	119
20	97X97	858	666	482	382
30	137X137	1732	1370	982	742
40	177X177	2953	2331	1663	1273

TABLE 2: Version, size and capacity in byte for some correction level

2.5 GOOGLE GLASS



FIGURE 7: Google Glass

Google Glass, as shown in figure 7, is an optical head-mounted display, with a resolution of 640x360 pixels, which is built in a pair of glasses with a bigger frame as usual. Inside the frame is a 570mAh battery and a full Android device, which runs up to 8 hours. In contrast to an Android smartphone it has only Wi-Fi and Bluetooth and needs an Android device to run applications. An application can read all sensors of the Google Glass and show anything on the display. Besides an accelerometer and a gyrometer to gather data about head and body movement it has a microphone for voice commands. Furthermore, it has a light sensor and a proximity sensor. The 5 MP camera can take videos with 720p and save everything on its 16 GB flash. Besides controlling the Google Glass with the Phone App or its touchpad on the right frame, it is possible to control it with head movements and eye tracking. Applications for Google Glass are developed in Java with the API which is provided in Android's wear editions. [30] Google provides its own development editor "Android Studio". [1]

2.5.1 TECHNICAL SPECIFICATIONS

- **CPU** OMAP 4430 SoC dual core
- **Memory** 2 GB RAM
- **Storage** 16 GB Flash
- **Display** Prism projector, 640x360 pixels
- **Sound** Bone conduction transducer
- **Input** Voice command through, microphone, accelerometer, gyroscope, magnetometer, ambient light sensor, proximity sensor
- **Controller input** Touchpad, MyGlass Phone App
- **Camera** 5 MP, 720p videos
- **Connectivity** Wi-Fi 802.11b/g, Bluetooth, micro USB
- **Power** 570 mAh Lithium-ion battery
- **Weight** 43g

2.6 PUBLIC-KEY CRYPTOGRAPHY

The following part will present public-key cryptography, also known as asymmetric cryptography, which was invented to solve the problem of distribution of necessary keys and also to provide a way of verifying authenticity [70]. Furthermore, a short outlook on the use case of public key infrastructure, a system build upon the features of public-key cryptography is given.

Asymmetric cryptography uses two cryptographic keys, this was an all new concept presented by W. Diffie and M. Hellman in their seminal paper "New Directions in Cryptography" [22]. Former known symmetric cryptography, is based on a single shared secret used for both encryption and decryption. Their idea focuses on using two different keys, while a private key should be used for decryption, a (different) public key should be used for encryption. Nevertheless they were unable to present an algorithm capable to fulfill their requirements, this was achieved some years later by R. Rivest, A. Shamir and L. Adleman, who presented the so called *RSA* algorithm. [32]

In addition to using two separate keys for encryption and decryption, asymmetric cryptography allows communicating parties to have no former direct communication to start an encrypted transfer. In contrast to symmetric key cryptography this separation allows for mitigating the difficulty of exchanging keys. This is because, a public key only permits encryption of to be exchanged information, it can be publicly posted and shared with everybody. Public directories of such keys can then be queried by the sender to receive a desired public key of the intended receiver. Messages encrypted by the sender with the desired recipient's public key can again be shared with everybody but are ultimately decryptable by the holder of the corresponding private key.[66]

To achieve these separation between private and public key, asymmetric cryptography is based on so called trapdoor one-way functions. One-way functions, to be more precise *Cryptographic hash functions*, need to satisfy three properties:

CRYPTOGRAPHIC HASH FUNCTIONS [70]

PREIMAGE RESISTANT: It should be hard to find a message with a given hash value.

COLLISION RESISTANT: It should be hard to find two messages with the same hash value.

SECOND PREIMAGE RESISTANT: Given one message it should be hard to find another message with the same hash value.

One-way functions used with asymmetric cryptography comply to the standard definition of a one-way function, also used with symmetric cryptography, but in one aspect: Given the correct key to the trapdoor they no longer hold the property of preimage resistance. [70] Providing this key will ultimately allow decryption to receive the former encrypted input.

Different tools make use of public-key cryptography: SSH, GPG (as implementation of OpenPGP) and also Secure Socket Layer as defined with TLS just to name a few [3] [82].

The most common system which facilitates the advantages of public-key cryptography is the so called public key infrastructure (*PKI*) as used with Transport Layer Security (*TLS*) the successor of the more commonly known Secure Sockets Layer (*SSL*)[82] [66]. *TLS* is mostly used if a secure connection to a website is desired, this can be identified if a connection is established using the Hypertext Transfer Protocol Secure (*HTTPS*).

Public Key Infrastructure is a system of hardware, software, people, policies, and procedures that are needed to create, manage, distribute, use, store, and revoke digital certificates. In contrast to unchecked public keys within public directories, within the *PKI* system keys and the identity of its holder are checked by so called Certificate Authorities (*CA*)).

In common *HTTPS* communication a hybrid cryptosystem is used to secure transfer of data., as shown in figure 2.6. Connecting to a *TLS* secured website the website's certificate is requested, it contains the servers public key. This certificate is checked against a list of trusted *CAs* and if found to be valid a session key is created. This symmetric session key then is encrypted with the servers valid public key and send back to the server. The server is able to decrypt the session key with his private key and therefore can now communicate with the client in private.[66]

By this one can be sure to be not part of an ongoing man-in-the-middle attack as presented in 2.2.1.

Sequence Diagram SSL Handshake

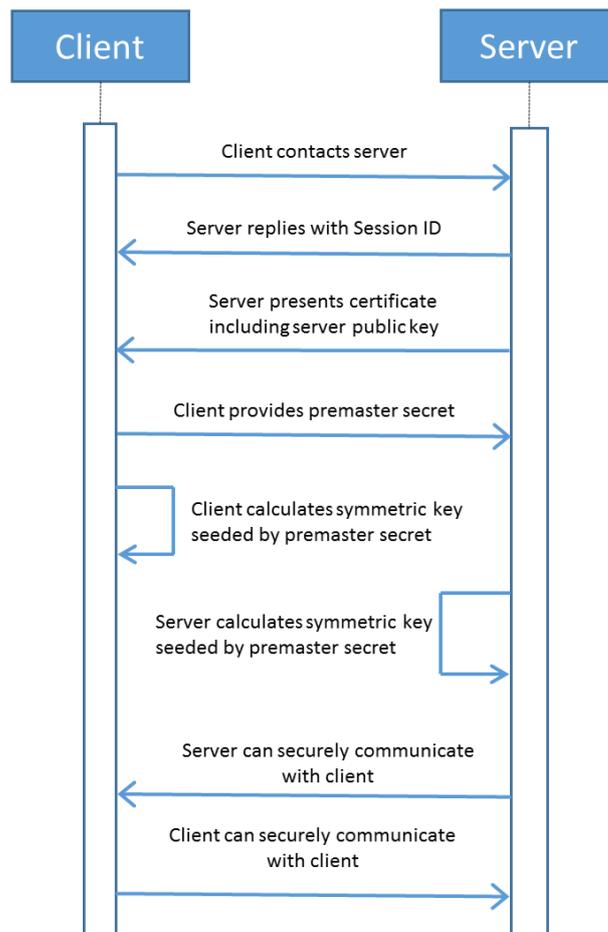


FIGURE 8: Sequence diagram: SSL Handshake [66]

2.7 TRANSACTION NUMBERS

A Transaction Authentication Number (*TAN*) is a secret that is shared by two parties, in order to provide authentication for (trans)actions. It is usually used in addition to conventional authentication methods like personal passwords, and therefore a form of two-factor authentication.

In a banking context, these two parties are usually a banking institute and a customer, where the latter uses *TANs* to authenticate himself when performing a financial transaction.

2.7.1 TAN SYSTEMS

There are many different *TAN* systems currently in use. Though all of them ultimately aim to fulfill the security goal of *authenticity* (see section 2.1.2), they differ in the actual implementation. Below, the most common ones will be outlined and their effectivity will be discussed. This includes their effectivity against the two attack vectors most relevant to *TANs*: Phishing and man-in-the-middle attacks, as described in section 2.2.

TAN LISTS

A *TAN list* contains a set of *TANs* which is generated by a banking institute and handed to a customer in physical form, typically requiring the latter to identify himself with his passport or a similar document. When the customer submits a transaction, he has to enter one of the unused *TANs* from that document. Each *TAN* can be used only once, and after all *TANs* have been used, a new list has to be obtained. [15]

There are multiple security problems regarding *TAN lists*. They are very prone to phishing attacks, as every obtained *TAN* enables an attacker to perform a transaction of his own desire. Furthermore, they offer no protection against man-in-the-middle attacks. An exemplary attack is described in section 2.2.1. [15]

INDEXED TAN LISTS

Indexed TAN lists (also called *iTAN*) aim to reduce the risk of phishing attacks for *TAN* lists by indexing every *TAN*. Instead of asking the customer to enter any *TAN*, he has to enter a specific one using this system[71, 15]. Therefore, if an attacker obtains one or even a few *TANs*, it is improbable (but not impossible) that he will be able to use them. [15]

The other discussed problems with *TAN lists* however remain, and this method is generally deemed to be unsafe. [15]

TAN GENERATORS

TAN generators are physical devices which can be used to generate *TANs* on demand. There are many different techniques that define how the *TAN* generation process works. The most notable ones are:

eTAN An *eTAN generator* is equipped with a display and a keypad. To generate a *TAN*, the banking website asks the user to enter a specific control number, usually the account number of the recipient, into his *eTAN generator*. The latter will then generate a *TAN* which is only valid for a transaction to the given account number, and only for a short amount of time.[15]

eTAN is secure against phishing attacks and provides a good amount of safety against man-in-the-middle attacks, depending on the concrete implementation and the attention of the user.[15]

sm@rtTAN In a different way from the *eTAN* method, a *sm@rtTAN* (also called *chipTAN*) *generator* does not have a keypad. Instead, the user has to insert his bank card into the generator in order to get a *TAN*. *TANs* generated this way have to be used in the same order in which they were generated, but they are not bound to either a specific transaction or a time span. Thereby, this method is vulnerable to both phishing and man-in-the-middle attacks.[15]

sm@rtTAN plus A variation of this method known as *sm@rtTAN plus* (also called *chipTAN manuell*) combines *sm@rtTAN* and *eTAN* by requiring the user to enter a control number and his bank card. This method provides even better security than the *sm@rtTAN* method, as every transaction number is bound to a specific transaction.[15]

sm@rtTAN optic Another modification, named *sm@rtTAN optic* (also called *chipTAN comfort*), replaces the keypad of a *sm@rtTAN plus generator* with optical sensors. Instead of having to enter the control number by hand, the user points the sensors to a flickering image on the banking website, which contains an encoded version of the control number. This improves the usability and allows longer control numbers, enabling the device to display a short summary of the transaction, thus strengthening the method against man-in-the-middle attacks.[15]

MOBILETAN

mobileTAN (also called *mTAN* and *smsTAN*) works by transmitting uniquely generated *TANs* per SMS every time the customer needs one. These *TANs* are only valid for the single transaction they were created for. To provide additional security, the SMS containing the *TAN* also contains some information about the associated transaction.[15]

The security level provided by *mobileTAN* is generally better than that of *TAN lists* and *indexed TAN lists*, as the use of two separate communication paths thwarts man-

in-the-middle attacks. However, there are still some security issues: With the rise of smartphones, it is generally possible that both your pc and smartphone are compromised, which would circumvent the protection against man-in-the-middle attacks.[15]

In some cases it is even possible to completely bypass *mobileTAN* with a so called *SIM swap fraud* (also called *SIM swop fraud*): After obtaining the victim's phone number, the attacker requests a new *SIM* card (this process is called *SIM swap*) while pretending to be his victim. Once he manages to trick or bribe the cellphone company into fulfilling this request, he is in possession of the new *SIM* card, while the existing one gets deactivated. Thus, he can read all SMS which are send to the victim, rendering *mobileTAN* useless.[86]

Another reason against *mobileTAN* is that some banking institutes charge their customers for every send SMS.[15]

PUSH-TAN

pushTAN works analog to *mobileTAN*, but uses a smartphone application as the second communication channel instead of SMS. This eliminates the costs for SMS and prevents *SIM swap fraud*. Compromised or stolen smartphones however remain an issue. To minimize this threat, *pushTAN* requires that your smartphone is not rooted.[55]

PHOTO-TAN

Similar to *sm@rtTAN optic*, *photoTAN*, works by displaying an image on the banking website, which transmits the transaction details. Instead of a dedicated *TAN generator*, the user takes a photo of the image using an app on his smartphone. Furthermore, the image does not flicker as it does with *sm@rtTAN optic*, instead a coloured bar code is shown. A bank card is not needed for this method.[15]

As with all techniques that use a smartphone, the security of *photoTAN* depends on the security of the phone. If it has not been hacked or stolen and the user is attentive to the displayed transaction details, *photoTAN* is safe against both phishing and man-in-the-middle attacks, as the generated transaction authentication numbers depend on the scanned image, and thereby on the transaction details.[15]

QR-TAN

QR-TAN is another method that uses images on the banking website in combination with smartphones to authorize transactions. Here, the transaction details are transmitted via a *QR-Code* (see section 2.4), which is cryptographically signed by the banking institute. A smartphone app decodes the *QR-Code* and displays the transaction details to the user. If he decides that everything is correct, there are two possible next steps: For some implementations of *QR-TAN*, if the smartphone is connected to the internet, it can send a confirmation message to the server, completely eliminating the necessity of *TANs* in

the first place. Alternatively, if no internet connection is available on the smartphone, a regular *TAN* will be generated.[41, 65]

QR-TAN provides the same amount of security as methods like *photoTAN* as the underlying mechanism is the same.[41]

2.7.2 TAN GENERATION

Depending on the specifics of the used *TAN* system, different methods for the generation of *TANs* exist. Unfortunately, no reliable sources on this topic could be found, so two plausible methods will be presented below.

RANDOM NUMBERS

In cases where the *TAN* is created by the agency which will have to verify it's correctness, generic random numbers can be used. This is for example the case with *mobileTAN* and *TAN lists*: If the banking website creates the *TAN* and sends it to the customer, it will know whether the *TAN* entered by the latter is correct. However, it has to be ensured that the random number generator upholds current security standards: An attacker must not be able to guess *TANs*.

MATHEMATICAL-ALGORITHM-BASED TANs

Sometimes, for example when using *sm@rtTAN optic*, both the *TAN generator* and the banking institute share information about the specific transaction. If the former is distributed with a secret key which is known by the latter, it is possible to combine the transaction details and the key in a defined manner, generate a hash sum of it, and use the result as a *TAN*. This process can be performed by both the generator and the banking institute, so they can both check the validity of the *TAN*. For additional security, one can include the timestamp of the transaction request into the calculation. This ensures that it is not possible to reuse *TANs* for identical transactions.

2.8 COMPACT TRANSACTION DATA FORMAT

At several points in this work, it will be necessary to store transaction details in heavily limited data formats, for example *QR-Codes* (see section 2.4). A possible use case would be the ability to display transactions and a cryptographic signature in a *QR-Code*, which could then be used by *TAN* generators to provide a *TAN*.

Thus, it has to be determined which data is part of a transaction and how much memory space is needed to transmit this data.

The European Payments Council (*EPC*) is in charge of "the coordination and decision-making"[79] of the European banking industry. It supports the Single Euro Payments Area (*SEPA*), which currently consists of 34 member states.[79]

Specifically, the *SEPA* issues the *Sepa Credit Transfer Scheme Rulebook*, which describes the requirements and schematics for interbank credit transfers in detail. Thus, this document determines the components of a financial transaction, such as the International Bank Account Number (*IBAN*) of the account of the originator, the amount of the credit transfer in euro, and many other parameters. Altogether, there are six different datasets, each consisting of a different set of attributes:[67]

- **DS-01** Customer to Bank Credit Transfer Information
- **DS-02** Interbank Payment Dataset
- **DS-03** Reject or Return Credit Transfer Dataset
- **DS-04** Bank to Customer Credit Transfer Information
- **DS-05** Recall of Credit Transfer Dataset
- **DS-06** Answer to Recall of Credit Transfer Dataset [67]

Customers are only exposed to *DS-01* (Customer to Bank Credit Transfer Information) and *DS-04* (Bank to customer Credit Transfer Information). Thus, all attributes that could possibly get exposed to a user can be determined by the superset of the attributes of *DS-01* and *DS-04* (See table 3 and 4). It is important to note, that some of these 20 attributes are redundant, e.g. it is no longer necessary to provide a Business Identifier Code (*BIC*) since February 2014, as the *IBAN* is sufficient to uniquely identify an account.[87]

TABLE 3: Attributes of *DS-01* (Customer to Bank Credit Transfer Information)

Attribute	Description
AT-01	The IBAN of the account of the Originator
AT-02	The name of the Originator
AT-03	The address of the Originator
AT-04	The amount of the credit transfer in euro
AT-05	The Remittance Information sent by the Originator to the Beneficiary in the Credit Transfer Instruction
AT-07	The Requested Execution Date of the instruction
AT-08	The name of the Originator Reference Party
AT-09	The identification code of the Originator Reference Party
AT-10	The Originator identification code
AT-20	The IBAN of the account of the Beneficiary.
AT-21	The name of the Beneficiary
AT-22	The address of the Beneficiary
AT-23	The BIC code of the Beneficiary Bank
AT-24	The Beneficiary identification code
AT-28	The name of the Beneficiary Reference Party
AT-29	The identification code of the Beneficiary Reference Party
AT-41	The Originator's reference of the Credit Transfer Transaction
AT-44	The purpose of the credit transfer
AT-45	The category purpose of the credit transfer

Source: [67]

TABLE 4: Attributes of DS-04 (Bank to customer Credit Transfer Information)

Attribute	Description
AT-02	The name of the Originator
AT-04	The amount of the credit transfer in euro
AT-05	The Remittance Information
AT-08	The name of the Originator Reference Party (optional)
AT-09	The identification code of the Originator Reference Party (optional)
AT-10	The Originator identification code
AT-20	The IBAN of the account of the Beneficiary
AT-21	The name of the Beneficiary
AT-24	The Beneficiary identification code
AT-28	The name of the Beneficiary Reference Party (optional)
AT-29	The identification code of the Beneficiary Reference Party (optional)
AT-41	The Originator's reference of the Credit Transfer Transaction
AT-42	The Settlement Date of the credit transfer (optional)
AT-44	The purpose of the credit transfer (optional)

Source: [67]

The business requirements for these attributes are laid down in section 4.6 of the rulebook[67], which refers in some cases to the ISO 20022[36]. They describe for example the maximum character to be used in AT-05 (The Remittance Information sent by the Originator to the Beneficiary in the Credit Transfer Instruction) and the possible values for AT-44 (The purpose of the credit transfer). However, there are many attributes, for example AT-02 (The name of the Originator), where no maximum length is provided.

In order to be able to transmit as much useful information with as few bytes of data as possible, redundant or irrelevant information has to be stripped. Though a selection of important attributes itself should not be part of this section, it allows to omit a lengthy deduction of memory requirements for attributes which will not be used in this paper. Thus, the results of these decisions which will be presented in section 4.1 in more detail will be used here. The selection was loosely based on other scenarios where the amount of information that can be transmitted is limited, for example *mobileTAN* (see section 2.7.1).

The following attributes were deemed as dispensable: AT-03 (The address of the Originator), AT-08 (The name of the Originator Reference Party), AT-09 (The identification code of the Originator Reference Party), AT-10 (The Originator identification code), AT-22 (The address of the Beneficiary), AT-23 (The BIC code of the Beneficiary Bank), AT-24 (The Beneficiary identification code), AT-28 (The name of the Beneficiary Reference Party), AT-29 (The identification code of the Beneficiary Reference Party), AT-41 (The Originator's reference of the Credit Transfer Transaction) and AT-45 (The category purpose of the credit transfer).

For the remaining nine attributes, only for the names (AT-02 and AT-21), not maximum length could be found in any official documentation. However, 30 bytes should suffice to uniquely identify a person by its name. An IBAN (AT-01 and AT-20) consists of 34 alphanumeric characters, resulting in 34 bytes (though it would be possible to

compress it even further, if needed). The maximal allowed amount of money that can be transferred (*AT-04*) is 999,999,999.99€, which needs a `uint64_t` to be stored, resulting in 8 bytes. The Remittance Information (*AT-05*) consists of up to 140 characters, where all allowed characters are part of the American Standard Code for Information Interchange (*ASCII*), thus needing no more than 140 bytes. Dates (*AT-07* and *AT-42*) can be stored as Unix Timestamps, requiring 4 bytes each. The purpose of the credit transfer (*AT-44*) consists of four character purpose codes, thus needing 4 bytes.[67, 36]

The result can be seen in table 5. Added up, all information require a total sum of 288 bytes. The implications of this result will be discussed in section 4.1.

TABLE 5: *Memory requirements for transaction details*

Attribute	Description	Bytes
AT-01	The IBAN of the account of the Originator	34
AT-02	The name of the Originator	30
AT-04	The amount of the credit transfer in euro	8
AT-05	The Remittance Information sent by the Originator to the Beneficiary in the Credit Transfer Instruction	140
AT-07	The Requested Execution Date of the instruction	4
AT-20	The IBAN of the account of the Beneficiary	34
AT-21	The name of the Beneficiary	30
AT-42	The Settlement Date of the credit transfer	4
AT-44	The purpose of the credit transfer	4
	Total	288

Source: [67, 36]

2.9 BLUETOOTH



FIGURE 9: *Bluetooth logo (bluetooth.com)*

Bluetooth was invented 1994 by Ericsson originally as a wireless alternative to RS-232 connections. It exchanges data over short distances using radio transmissions. Operating in the ISM band at 2.4 to 2.485 GHz Bluetooth technology using a spread spectrum, frequency hopping, full-duplex signal at a rate of 1600 hops per second. [12]

The IEEE standardized Bluetooth as IEEE 802.15.1 but no longer maintains the standard. Now its managed by the Bluetooth Special Interest Group (SIG), which oversees development of specifications, manages the qualification program, and protects the trademarks. [11]

This technology is called Bluetooth after the tenth-century king Harald Bluetooth who united dissonant Danish tribes, as its designed to unite different communication

protocols into one standard. The Bluetooth logo is a bind rune merging the runes for H and B, the initials of king Harald. [77]

The connection attempt between two Bluetooth enabled devices is called pairing. A Device can be connected to up to seven other devices, either as server or as client. These short-range ad hoc networks are known as piconets. These piconets are established dynamically and automatically as paired Bluetooth devices enter or leave the radio proximity of the other devices. Also each device can be part of several piconets. These connections between the Bluetooth devices can handle data and voice transmissions simultaneously. [6]

Bluetooth uses adaptive frequency hopping to reduce interference between wireless technologies sharing the 2.4 GHz spectrum. This technology detects other devices in the spectrum and avoids the frequencies they are using while hoping through the 79 1MHz intervals at a rate of 1600 hops per second. [6],[56]

2.9.1 RANGE

The range of Bluetooth connections vary depending on class or radio used in the specific implementation. There are three classes of Bluetooth radios:

Class	Range	permitted Power
Class 3	up to 1 meter	1 mW
Class 2	up to 10 meters	2,5 mW
Class 1	up to 100 meters	100 mW

TABLE 6: Rang and permitted power of bluetooth classes [6]

Also the effective range varies due to material coverage, antenna configurations, other devices using the same wireless spectrum, and other effects.

2.9.2 TECHNICAL DETAILS

There are several profiles which are definitions of possible applications and specify general behaviours that devices use to communicate with other Bluetooth devices. Every connection must choose one of the profiles, which list is to long for this document, but is shown here: [59] The actual version of Bluetooth is 4.2, which was introduced in 2014. The history of Bluetooth and the features added over time is shown at: [10] There are different protocols, of which each device have to implement in its Bluetooth stack to be able to use Bluetooth. The list of these and its descriptions is would be to long for this protocol, but for further research these informations are shown here: [10] Also there is a baseband error correction in Bluetooth, which protects individual packets, depending on packet type, either 1/3 rate forward error correction (FEC) [78] or 2/3 rate is used. Additionally packets with cyclic redundancy check (CRC) [18] will be rtransmitted until acknowledged by automatic repeat request (ARQ) [61].

2.9.3 CONNECTIONS

Any Bluetooth device, which is in discoverable mode, transmits the following information on demand:

- Device Name
- Device class
- List of services
- Technical Information like manufacturer, clock offset or Bluetooth version

Any device may perform an inquiry to find other devices to connect to, and any device can respond to such inquiries. To use a service provided by a device a pairing or acceptance by its owner is required, but the connection can be established without. Every device has a unique 48 bit address, but these are not shown in the inquiries, instead names are used, the user sets. These names appear when another device scans for discoverable devices.

The reason for pairing is, that it can be wanted to connect to specific devices automatically, but not to all, because of private information which can be opposed if a device connects to every bluetooth device. This problem is solved by a process called bonding, and a bond is generated through a process called pairing. This process can be triggered by a user or automatically when connecting to a service where the identity of a device is required for security purposes. Before Bluetooth 2.1 there was only Legacy pairing where each device must enter a PIN Code, which can be every 16 byte UTF-8 string. With Bluetooth 2.1 the Secure Simple Pairing (SSP) was introduced which uses a form of public key cryptography to provide more security and an easy pairing. Before Bluetooth 2.1 encryption was not required and could be turned of and on at any time, after 2.1 it is required for most services. [12]

3 PROSPECT

This chapter provides detailed information about the four ideas underlying the TART project. First the assumptions about NFC TAN devices are presented. Contained in this section are among others explanations about two-factor authentication based on NFC. Advantages of those are discussed as well as possible application areas in the field of banking.

The second section of this chapter focuses on encrypted QR codes and their benefits. In a similar manner section three concentrates on the assets of signed QR codes. Additionally this section considers besides advantages and related work the limitations of the technique.

By combining the advantages of the named three concepts idea four sums them up to improve the withdrawal of money on local ATMs. The section examines related work in detail as well as the current used cash withdrawal process. Moreover the advantages of this project's approach are discussed.

In the last and fifth section of this chapter the presented ideas are summed up and presented in context of two big sample use cases. Those use cases shall facilitate the understanding and show some of the many possible applications.

3.1 NFC TAN DEVICE

This part will present an idea to authenticate transactions by facilitating the users' NFC capable debit card. This solution aims to improve the overall security in the context of online banking, ATM cash withdrawal and other banking operations. Furthermore it tries to dismiss the need for additional designated hardware, such as physical devices and old-fashioned, with all their specific drawbacks as presented in chapter 2.7.

With online banking the need for techniques to authenticate a user in a highly usability friendly manner arise. Besides this, every presented solution must satisfy certain security criteria and thereby be hard to unknowingly intercept, mislead and overall attackable as described in chapter 2.7.

Most commonly every banking user is at least equipped with a debit card and a PIN. An online banking user will have to remember an additional set of information to login into his online banking account and will have some sort of mechanism to authenticate single banking operations such as a single transaction.

As several solutions presented in 2.7 show, additional physical elements or hardware such as TAN lists or a smartTAN device are provided to the user to perform those single operation authentications. For mobileTAN, pushTAN and photoTAN only a most commonly available smart or mobile phone is needed, but they all come with drawbacks as presented in chapter 2.7.

With the presented idea a combination of traditional and modern parts is used: On the one hand a (NFC) debit card, already available to each user, and on the other hand a more and more common NFC capable smart phones as shown in chapter 2.3. Combining those two elements allows minimize the adjustment a user has to make when thinking of an offline over the counter transaction within his local bank. In a offline scenario a user has to provide his request and his debit card and a signature to authenticate his request. A similar authentication is done in the presented idea.

Sequence Diagram NFC two-factor authentication

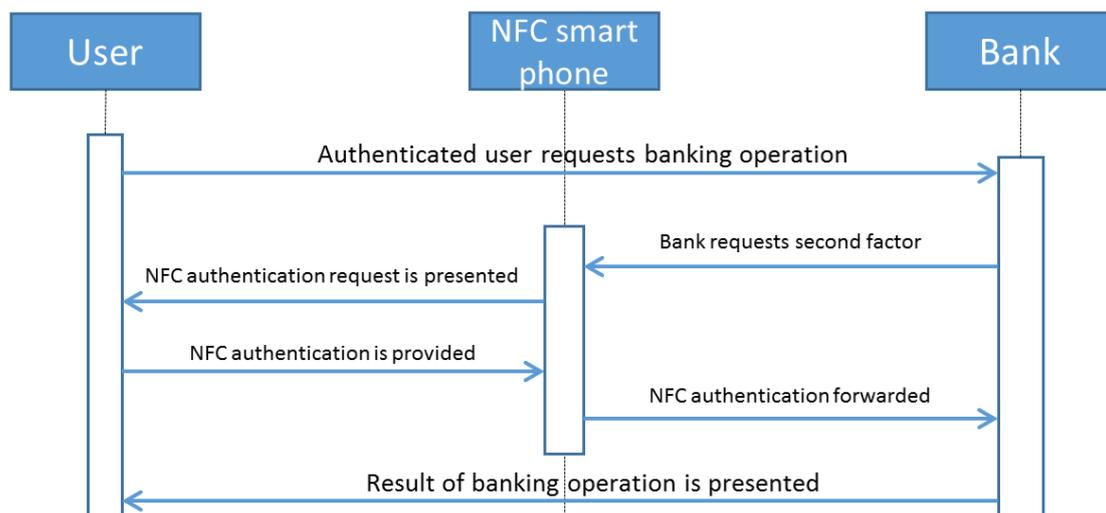


FIGURE 10: Sequence diagram: NFC two-factor authentication

As shown in figure 3.1, the first step to request some banking operation online, a user is still required to login in to the online banking website. Requesting his banking operation he is asked to provide a transaction authentication number, this authentication request including the operation details is then presented on his smart phone. After acknowledging the transaction he can authenticate it by providing his NFC capable debit card. A possible representation of the operation details is shown in figure 3.1 He has then completed a full two-factor authentication, as he provided the necessary knowledge to login in to the online banking website and also provided a secure element by acknowledging the transaction with his NFC debit card.

The advantages of this lies in the fact that users have their smart phones usually at hand, as well as their bank cards. Additionally, the smart phone can be used as a TAN generator for multiple bank accounts. Another benefit is that NFC works even through the wallet without taking the card out.

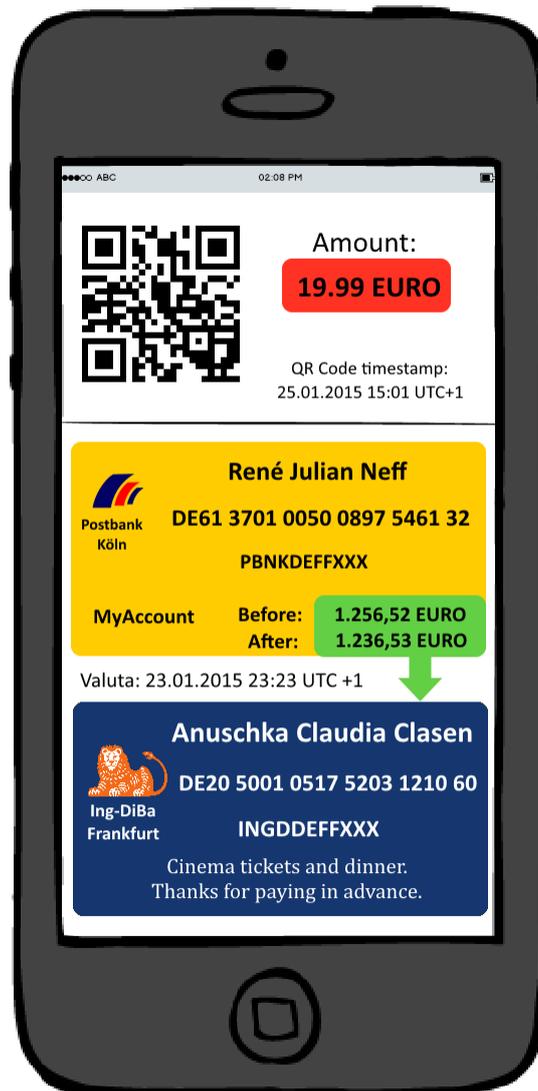


FIGURE 11: *Mockup of a bank transfer operation presented on an NFC smart phone*

All designated hardware TAN generators share a common problem: as an extra device only usable to generate TANs they have to be carried around in case of mobile online banking and are thus likely to get lost, forgotten or stolen. Additionally, the bank cards need to be inserted into the generator. As a result the user has to take the bank cards out of his wallet and it is not rare to forget cards in the generators or on the desk.

3.2 ENCRYPTED QR CODES

To read critical Information in a public accessible space always has a problem, others can read it too. So some attacker could read anyones debit details or other privat informations which are read on some kind of screen like a smart phone, a tablet or a notebook just by taking a look about the users shoulder. To avoid this it is possible to encrypt all critical informations that should not be available in public. There starts the next problem, as any user who are allowed to read them should be albe to do so without a big expense of time, with no possibility for not authorized users to read it too.

Here starts the idea of Encrypted QR Codes, to transmit information between a display and a Google Glass as describete in Section 2.5 over QR Codes as explained in section 2.4. The provider of the information encrypts these information with GNUpg as described in section 2.6 and uses the public key of the adressed reciever of this specific information. So it is ensured only the right receiver is able to decrypt the information and read it. To verify the informaion comes from the sender, the user believes it come from, it can include a signature of the sender too, as it is described in section 3.3. The transmission between sender and receiver should always be encrypted, but if this described strategy is used all information stay secured even if the transmission itself is not additionally secured in any way.

But how to read the transmitted information in public properly? Packed in QR Codes every device with a camera can read it, and if it is a head mounted display like a Google Glass it is possible to decrypt informations using the users private key and verifying the sender using its public key. Now the information can be read on the prism display with minimized risk anybody is reading it too. If it is a big chunk of information it must be seperated in more QR Codes than one, which then have to include a serial number to get information in right order if the camera finds the QR Codes in a mixed order. Such a transmission system can be used for example to read debit informations from a banking website or to transmit a mobile TAN number with minimized risk to get stolen.

3.2.1 HOW IT WORKS

In the proof of concept for this strategy the situation appears that the user tries to get his debit information from a banking server over a not trustworthy network in a public accessible space on a display which can read by everybody passing by. So the debit information will be signed by the server with the private key of the bank and encrypted with the pbulic key of the user. This encrypted information will stored in one or more QR Codes, debending of the size of the informations. The QR Code with the encrypted and signed information will shown on a display of a mobile device like a notebook, and be seen by the Google Glass camera which is worn by the user. There the QR Code will be detected, photographed and analyzed by an app on a to the Google Glass connected smart phone. If it has got the format the sender will be verified and the informations

decrypted on the connected smart phone. After decrypting the information is shown on the Google Glass display, so only the user can read it.

3.3 SIGNED QR-CODES FOR SECURE TRANSACTIONS

In this section, a technique to transmit transaction details via *QR-Codes*, which includes a cryptographic signature of said details, will be discussed. This technique would include two components: One to generate the images on the server, and one to decode them on a mobile device.

3.3.1 SCOPE OF APPLICATION

The main goal is the combination of said technique with the proposed *NFC TAN Device* (see section 3.1): This would result in a method similar to *QR-TAN* (see section 2.7.1): When a user wants to make a transaction, he is shown a *QR-Code* including the transaction details. These are taken as a seed for the *NFC TAN Device*. This process would be immune to phishing and man-in-the-middle attacks, as an attacker could not generate a correct signature for a faked *QR-Code*. Additionally, it would benefit from the protection of the *NFC TAN Device*, with the result that an attacker would need the victim's bank card, the associated pin, and, depending on the implementation, the victim's smartphone to circumvent the protection. In this case, he would be able to withdraw money from an ATM anyway. However, this method does not protect the user against the risks associated with a compromised smartphone.

Another possible use would be the ability to print out transactions which have been carried out in the past in a forgery-proof way. This could be useful if someone wants to store proof of important transaction on a non-electronic medium for security reasons.

As a third application, it would be possible to use signed *QR-Codes* which include a timestamp and a unique identifier i_1 as a secure way to enable an observer to identify the device on whose display the *QR-Code* is shown. These *QR-Codes* would be valid for a defined amount of time t , and have to be regenerated after this time. An observer is able to verify whether the cryptographic signature is correct and whether the timestamp is not older than t , which proves to him that the device is correctly identified by i_1 , if it has not been attacked within t . This holds only if an attacker is not able to compromise both this device and another device with the identifier i_2 , causing the device i_1 to display the *QR-Codes* of i_2 . A real world application could be *QR-Codes* which are shown on *ATMs*, as they are used in 3.4. While it is possible to circumvent this protection, it is still more secure than most other forms of identification, which can be for instance painted over or forged in a similar way. Thus, it is a form of authentication of the device to the observer.

3.3.2 LIMITATIONS

As for all data transmitted by *QR-Codes*, the amount of transferrable data is limited. Though *QR-Codes* can theoretically contain up to 2.88 KiB of information (see section 2.4), it would be preferable to make do with far less, as it is easier to correctly detect *QR-Codes* with fewer pixels and better error correction. As described in 2.8, about 288 B would be needed to convey the most important transaction data. Further space would be needed to include a cryptographic signature and a header. This suggests that while it probably will be possible to fit all needed data into a *QR-Code*, trade-offs will be necessary.

Furthermore, the system depends on the correct verification of the signature. If an offender manages for example to provide an incorrect public key to the mobile device, or manages to compromise it altogether, the system would be broken.

3.3.3 RELATED WORK

QR-TAN (see section 2.7.1) includes a method to transfer transaction data over *QR-Code* and is overall fairly close to the main goal described in section 3.3.1. It is, however, not clear whether all implementations of *QR-TAN* include a cryptographic signature. While the *LIOTP QR-TAN Procedure*[65] explicitly mentions one, the *182direkt* bank[35] does not.

Neither of those seems to be designed for a use of the *QR-Codes* for other purposes than *TAN* generation.

3.4 MOBILE DEVICE ATM

In the fourth and last approach of this work the aim is to improve another customer-banking-interaction: The withdrawal of money on ATMs. In the common way of money withdrawal three entities are involved, the user – in this well-known as Alice – and an ATM as well as the corresponding banking server. Figure 12 explains the current process as a sequence diagram.

At the beginning the ATM displays a welcome screen until Alice inserts her bank card. Thereafter the ATM asks her for her PIN, which Alice enters. This information is used by the ATM to authenticate Alice and to check for her account at the server. If the account is sufficient the banking server sends back the 'ok' to the ATM. The machine is then able to ask the user about the amount of money he wants to withdraw. Alice as user chooses the amount on the ATM, which afterwards sends the transaction request to the banking server. After the execution of the transaction the server answers with a transaction complete information, as a result of which the ATM returns the card together with a request to the user to return the card. Once the card is received the money can be output, again accompanied with an information on the ATM screen to receive the money. As the final step Alice Doe withdraws the money and the ATM displays again the welcome screen, since the transaction is completed.

Sequence Diagram Current Cash Withdrawal at an ATM

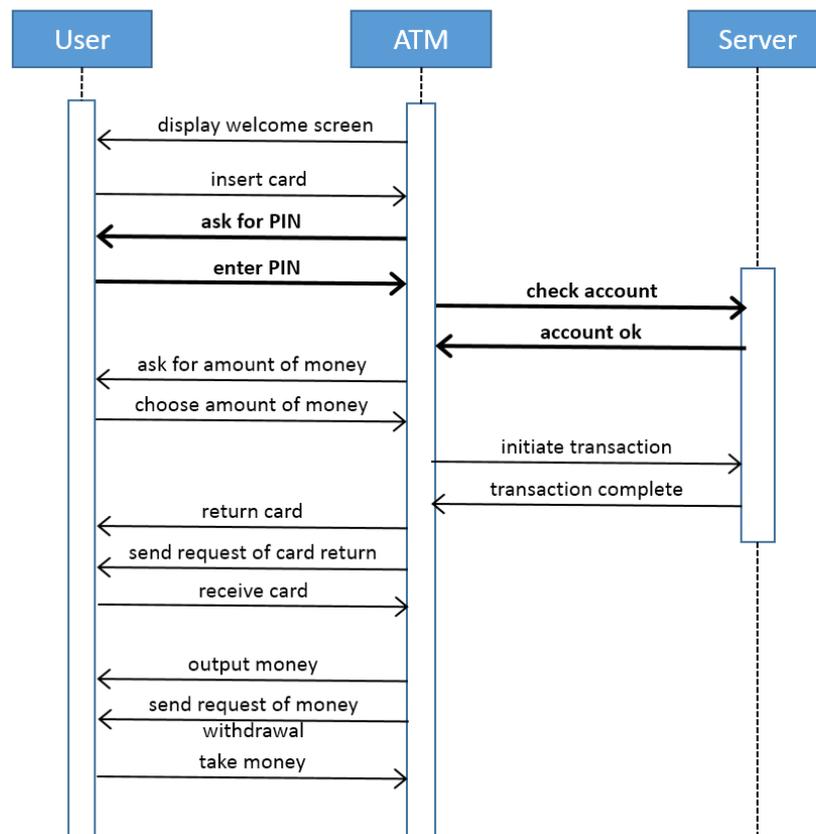


FIGURE 12: A sequence diagram showing the process of money withdrawal at ATMs most commonly used.

The aforementioned approach is widely used. Vulnerabilities are nevertheless easy to detect and often used, as described further above in chapter 2.2. The skimming of ATMs, incidents of fraud by man-in-the-middle-attacks on such machines, repeatedly happens. Especially fake keypads or card readers are typical examples of such attacks, aiming to read the customer's PINs or bank card information. The user is forced to trust the security preparations of the ATM alone. The authentication process of the user is based on a two-factor authentication method, but nevertheless he has to give both information to one single unit: the ATM. If the ATM is infiltrated or infected – in short manipulated – both levels of authentication are given to the attacker. An attack on an ATM is therefore worthwhile because both values can be captured within one offensive.

Another problem is the loosening or forgetting of the user's bank card in the area of the ATM, because the user has to receive the money and the bank card in short time – too overstraining t once for some. Occasional critic on the actual ATM money withdrawing system is the long processing time. As the user needs to authenticate himself in every level directly in front of the machine and has to make decisions about the amount of money, denomination of it and to insert and take back his possessions, the overall time

to do this is naturally not insignificant. As a consequence longer cues especially at peak times may occur and frustrate the customers.

3.4.1 RELATED WORK

To increase the level of security during the money withdraw process, as well as to optimize the processing performance, several new approaches have been presented and discussed in recent years. In addition to the named aimed-for improvements the including of latest technology was a purpose.

In 2012 the NCR Corporation introduced a way to withdraw money from an ATM in about ten seconds. In this approach the customer is not forced his card out of the wallet as the authentication process is moved to his smartphone and works without it. The customer only needs to be in possession of an Android or iOS smartphone with a camera built in and to launch the application NCR developed. This application administrates all bank accounts of the customer, if he authenticated himself with the correct PIN. The banking card is no longer necessary to log in. In the launched application the user picks the account he wants to withdraw the money from and chooses the amount he wishes to get. Therefore he touches the corresponding dollar figures which represent the different amount possibilities. If those decisions are made the customer is encouraged to use the smartphone's built-in camera within the application using the scan button. Displayed on the ATM he wants to withdraw the money from is a QR code. This QR code is used to confirm the transaction and to dispense the money if it is scanned within the application. Consequently, the log in on the application of the customer's smartphone could be done outside the bank or in the queue in front of the ATM, whereas the named ten seconds transaction time could be improved. [69]

NCR states that it not necessary to purchase new hardware for the ATM, because only the ATM software needs to be updated to display the mentioned QR codes. The approach is therefore easy to implement and to use on a big scale. Another main argument raised by NCR is the improvement of safety. Because the bank card is not used with the ATM to withdraw the money nor the ATM's keypad is used to type in the PIN, skimming attacks could be prevented. [69]

Only limited access about the security policy of the application is given. Obviously the authentication security level decreased, due to the reduction of the two factor authentication process in the common withdrawal approach using bank card and PIN on the ATM to the only one factor authentication on the user's smartphone. The application can be launched on any Android or iOS device and is not tied to a specific smartphone. Therefore the ownership factor is eliminated. An attacker who is in possession of the PIN (and presumably he needs information about the account number additionally) would own all necessary information to have complete access to the money. To gather knowledge about the PIN phishing attacks, over the shoulder surfing or a successful prior skimming attack could be used. It might be assumed that the common approach and the one presented by the NCR would co-exist – at least for a while. As a consequence skimming attacks to customers using the usual withdrawal method would make it easier for the criminals to get access to the money. The would not have to replicate the bank

cards any more but use the NCR application to get any money they want only with the gathered information.

Of course the smartphone itself is another vulnerability factor, which, however, cannot be estimated because too less information about the security of the app is given. One possible problem is the easier shoulder surfing on smartphones compared to ATM keypads why PINs could easier be spied. The QR codes itself are another vulnerability to the system. If the displayed code on the ATM is manipulated – either by manipulated software or a fake notice to use the printed QR code instead due to technical problems – the money would be withdraw on any ATM the attackers have chosen. The user is not able to check if the displayed QR code refers in any way to the ATM he is standing in front of. It is evident that the NCR approach is not as secure and safe as it states and therefore does not improve the common approach satisfactorily. [69]

Basically the same approach was presented by FIS and Wintrust Financial Corporation in late 2013 on the ATM, Debit and Prepaid Forum in Las Vegas [4]. The firm Diebold chose a different but similar approach to address the 'millennium generation' [91]. The first stage of authentication is similar with almost same security issues. One difference is that the user has to scan the QR before he is able to use the application on his phone, what eliminates the improved interaction time as no steps of the process can be done without standing in front of the ATM. The mentioned vulnerability of faked or manipulated QR codes is reduced as the Diebold's withdrawal method adds another step to verify that the user is physical next to the ATM: To confirm the withdrawal a onetime PIN is sent to the user's smartphone application which he has to enter into the ATM. [91]

Even with this improvements the overall method is vulnerable to attacks and fraud, as a user only must know the PIN to declare his ownership of the account. Only the QR code vulnerability is better than in NCR's approach.

An older but different idea was installed on several Spanish banks in spring 2011. It is based on contactless ATMs and tries to prevent skimming and fraud by enabling the user to only tap cards or NFC phones to withdraw money instead of inserting the cards into the ATM. The problem with this approach is that skimming is still possible as the NFC readers on the machines should as easily be manipulated or changed than the card readers before. The article states nothing about the security policy of the NFC phone technology or other security measures, but calls the technique 'fully secure'. [5]

3.4.2 THIS WORK'S APPROACH

As mentioned above the current system as well as so far presented improvements lack of security in different ways. Especially fake keypads or card readers are typical examples of attacks, aiming to read the customer's PINs or bank card information. The user is forced to trust the security preparations of the ATM alone – or, as in the named other approaches, the security levels of his phone.

To avoid this, the NFC technology presented in chapter 3.1 can be used to separate the two factors of authentication. The first development stage of this work was to separate

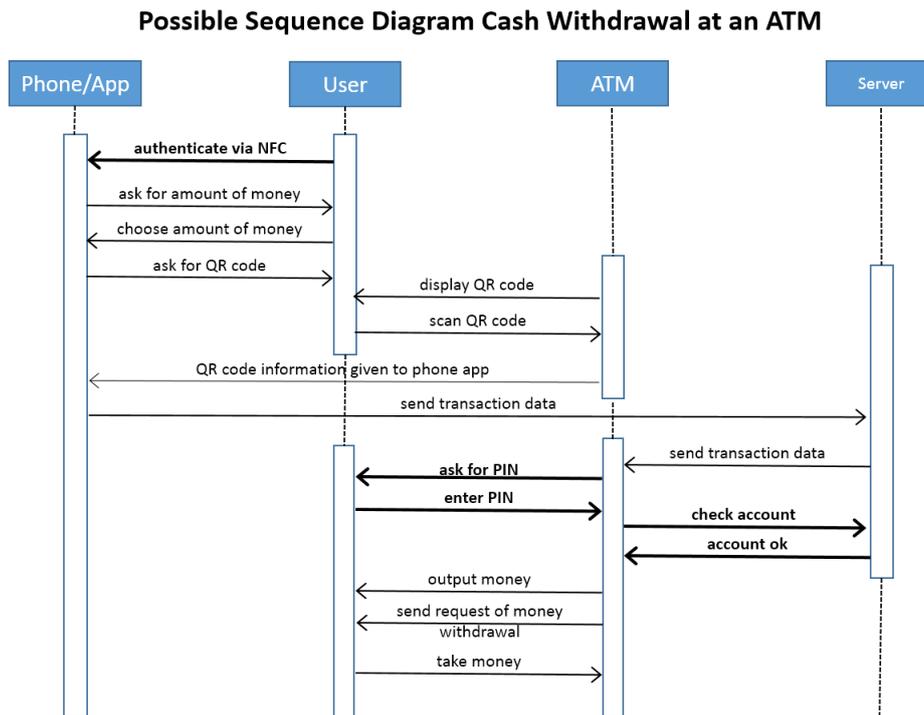


FIGURE 13: A sequence diagram showing the process of money withdrawal at ATMs with the approach presented in this work.

only the authentication process physical from the ATM as given infrastructure and move it to the user's mobile phone, similar to NCR's or Diebold's methods. As a result the source of the problem would have been displaced by more or less the same vulnerability. Both information values would have stayed in one place – although the place changed. Based on this considerations one must assume that the security level could not have been considerably enhanced.

Under the new approach the two factor authentication process is separated into two different parts. The possession based authentication part, more precisely the user's ownership of credit or bank card, is switched to the mobile phone. The well-known example persona Alice would use her card to authenticate herself. But instead of giving the card away (and under the control of the ATM) she uses the NFC technique as authentication method. This is an improvement to the related work because not the customer's PIN has to be known by him but he has to be in possession of the bank card. As the NFC technique works even through the wallet it is not necessary to take it out (and therefore to lose it). The authentication can be done in beforehand, for example in the privacy of the own home or car. Afterwards an application can be used to choose the amount of money the user wants to withdraw.

Only when it takes the next step Alice needs to be in front of the ATM. She scans the displayed QR code of the ATM, to locate and verify it. Subsequently after the withdrawal information is send from the phone to the banking server and then to the ATM. But

instead to withdraw the money immediately the ATM asks Alice for her PIN to verify herself.

Consequently the second factor of the two-factor authentication is still on the ATM and separated from the ownership factor. Figure 13 shows the method as a sequence diagram. The added level and separated authentication factors are highlighted.

Common skimming attacks to the ATM would only give the attackers the information about the PIN but not about the corresponding account. The same holds true for the phone application, as only the card information is stored and processed there. A successful attack should therefore aim at both device – a disproportionately higher level of effort. Additionally the transaction time can be improved as the first authentication factor as well as the selection of the money amount can be done in beforehand.

3.5 COMBINED SYSTEM CONCEPT

To unite and summarize the aforementioned concept ideas the following paragraph emphasizes the application possibilities as well as an overall use case. The previous sections clarified the use of signed and encrypted QR codes and furthermore the way NFC technology of modern smartphones can improve the usability and security aspects of authentication. Besides an approach to withdraw money from an ATM was presented and its assets and drawbacks illustrated. Those features commonality is the possible assignment to the banking sector, as all ideas aim to improve both security and usability in several related tasks.

In the presented work the following approach was considered to satisfy the user's needs: From the user's perspective two main application technologies are used, a mobile app on the one hand and an improved website view on the other. Two different use cases are covered by this project which are discussed in the following paragraphs.

Alice as sample user is in habit of an NFC enabled debit card as well as a smartphone and a computer. Sometimes her friend Bob lends her his Google Glasses. In the first of the scenarios Alice wishes to withdraw \$100 from the local ATM near the train station, which is usually strongly attended by other customers. In the second scenario Alice would like to check the transaction details of her banking account online during traveling by train. As there are many passengers on the train Alice does not want her details to be spied on. Afterwards she wants to transfer money to Bob, which forms the second part of the use case two.

The first use case is illustrated by an activity diagram in figure 14. Alice starts the authentication by holding her NFC enabled debit card near to her smartphone. The NFC triggers the start of the TART mobile App on her phone and she is automatically logged in. She is now able to check the credit balance on the phone screen. If the amount is high enough to allow the withdrawal of \$100 she presses the *ATM* button on the screen. On the next screen she is asked to choose the amount of money she wants to withdraw. After selecting \$100 the app waits for information about the local ATM she wants to withdraw the money from. Therefore she could scan the QR code displayed on the

selected ATM near the train station either using the smartphone camera or with Bob's Google Glasses, if she has got them with her.

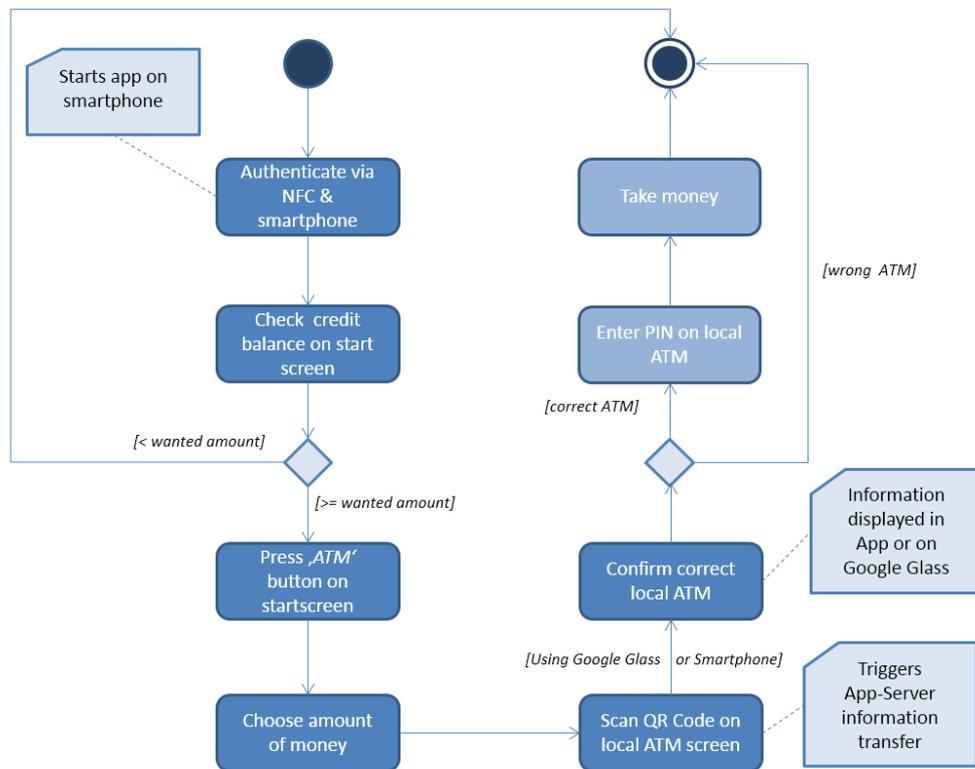


FIGURE 14: Activity diagram illustrating the TART cash withdrawal process. Actions concerning smartphone and/or Google Glass are highlighted in dark blue, ATM interactions in light blue.

Both scanning possibilities trigger the App to Server communication process to exchange the needed information about customer, wanted amount and identification of the ATM. Depending on the chosen device the ATM identification information is shown on smartphone display or Google Glass. If the correct ATM is selected Alice enters her PIN on the local ATM and thereby confirms the cash withdrawal. After taking her money the transaction ends.

Since it is possible for Alice to do most of the needed authentication and amount choosing on her phone, she is not forced to those steps in front of the teller machine. Instead she is able to prepare the withdrawal in beforehand for example during waiting in the queue. This increases usability as well as it decreases time consumption.

In the second use case Alice wants to publicly check her transactions as well as transfer money to her friend Bob. Using her laptop on train makes her vulnerable against *over-the-shoulder-attacks*, what she wants to avoid. Using TART this vulnerability is minimized since the transaction information are not clearly given on the screen but encoded in a QR code.

To check the transaction details Alice simply - after logging in on the banking website and selecting old transactions in the menu - the corresponding transaction QR code. This can be done either by smartphone or Google Glass. The sensitive transaction data is therefore not shown on the big and easily seen laptop screen but on her phone or even better the Google Glass display. This maximizes the privacy level.

Transferring money in a secure and usable way is also done on the TART website. Instead of carrying around a TAN device or TAN list, using the not always trustful mobileTAN (cf. chapter 2.7) she is able to authenticate and confirm the transfer only using her NFC enabled debit card and her smartphone (and possibly Google Glass) in addition to the website. This increases usability and safety, discussed in detail in chapter 5.

Her activities start again on the online website, logging in and selecting a new transaction. After inserting the relevant transaction details she is asked to enter a TAN to authenticate and confirm. Therefore she first authenticates herself using smartphone and NFC enabled debit card, as described above for the ATM cash withdrawal use case. Afterwards she scans the QR code depicted on the website using smartphone or Google Glass camera. The app decodes the code using the private key of Alice and displays the TAN on smartphone or Google Glass screen. After entering the TAN into the text field on the website she confirms the money transfer.

Both use cases combine several of this project's techniques. Encrypted and signed QR codes are used as well as the NFC authentication and the ATM mobile device approach. The ways those ideas were transformed into concepts and implementations is considered in the following chapter.

4 CONCEPT AND IMPLEMENTATION

In this chapter the concepts and implementation attempts of the TART project are presented. The chapter includes three sections. Section two and three deal with the the website and server aspects of the projects in first case and the app implementation and conception in the latter. The first section concerns encrypted and signed QR codes. Since both are used on website side as well as by the app the considerations regarding QR codes are split from the rest and presented in thus section. This facilitates the understanding of the implementation and use in different parts of the presented project.

Figure 15 illustrates the whole TART project and its elements as well as their relations. In the upper half of the diagram the server and website part is depicted. The server provides the two websites: *ATM* and *Online Banking*. Both websites use a database, characterized in the centre of the diagram.

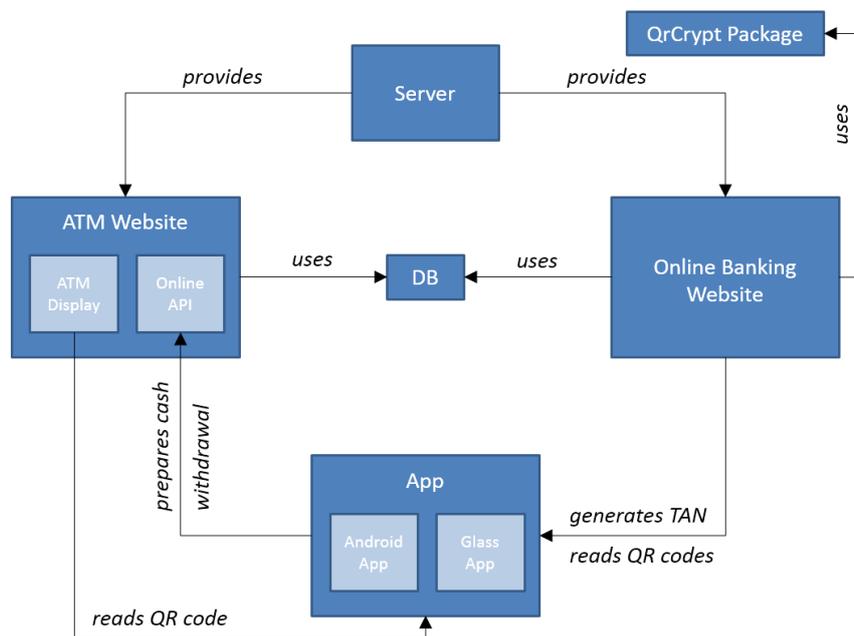


FIGURE 15: Overview about the elements and relations of TART project

The ATM website is composed of an ATM display part and the online API. The first one is needed to display and create QR codes containing the ATM identification information

which are presented to the App. The latter part prepares the cash withdrawal and deals with the communication.

The online banking website uses the QrCrypt package (discussed in detail in chapter 4.1) to create QR codes. Those codes are also used by the app to generate TANs or display sensitive information.

The app elements of TART project are divided into an android app part for smartphones and a Google Glass app. The app regarding project parts are explained in the last section of this chapter. The smartphone app project are hereafter referred to as *TART mobile* and the Glass concerning as *TART Vision*.

4.1 QR-CODES

Most of the presented approaches regard at least one of the following two techniques: encryption and signing of QR codes. The developed app uses QR codes as well as the website, wherefore the encryption and signing of QR codes are elementary aspects of the presented work. For that reason, this section starts with the detailed explanation of the QR concept, which was evolved as a basis for the implementation. In that regard, conceptual decisions respecting the public-key cryptography and the programming language to be used are justified *inter alia*. The implementation itself is presented and discussed in the second part of this section.

4.1.1 CONCEPT

In chapter 2.1 several security goals have been presented. Those goals have to be assured to avoid a lack of banking security, which could lead to divers attacks such as *Man-in-the-Middle-Attacks* or *Phishing*. The resulting vulnerabilities concern QR codes in this work's approaches in different ways.

The signing of QR codes as described in chapter 3.3 protects against the two before mentioned attacks, because an attacker is not able to generate a correct signature during the creation of a faked QR code. Therefore, the signing assures the *authenticity* of the code. In contrast encrypted QR codes ensure the *confidentiality* and *integrity* of the data given by the code (cf. chapter 3.2).

As encrypted and signed QR codes are both required within different parts of this project, it was therefore natural to create a library supporting both aspects. On this account a PHP library was built dealing with signed QR codes as well as encrypted. Due to the fact, that at least regarding the website aspects of this work PHP as programming language is relevant, PHP was chosen as language for those library as well. As a server-side script language it is well suited for web development [88]. For dependency management and to deal with the library an implementation in form of *Composer* packages was used (cf. 4.2) [17].

GPG was opted as public-key cryptography tool. It is a free implementation of the OpenPGP standard (Short for *Pretty Good Privacy*, cf. 2.6) and fulfils the in chapter 2.6

discussed properties using cryptographic hash functions. GPG (short for *Gnu Privacy Guard*) enables the user to encrypt data and communication and to sign those, too [80]. Thereby it fits perfectly the aforementioned needs.

In beforehand of the implementation it was important to evaluate the performance of Google Glass in dependency to different sizes and correction levels of QR codes (cf. 2.4).

To ensure recognize a QR-Code automatically with a Android device it is important to know in which size it is readable with which error correcting level and at which size it is sure to read it with high possibility. For ensuring this there has to be a measurement of how good which code is readable at which size, but first the framework have to be defined. The QR-Code will be shown on an ATM and should be readable by Google Glass while the user is standing in his usual position. The size of the ATM Display and its hight about the floor can vary, but at the ATM at VR-Bank Rhein-Erft eG in Swisttal-Heimerzheim has a Display of 34,4cm to 22cm which starts 125cm about the floor. The User stands 25 cm from the wall and looks on the display in a 90 degree angle to ensure best readability for the Google Glass camera. So the distance between the camera and the display is approximately 50cm. The size of the tested QR-Code is 21cm to 21cm, to be sure it can be shown on the display without a overlapping display frame. Also the app should read a QR-Code shown on a pc display. To be sure it will be readable on small notebook displays it is tested which QR-Codes can be read at a size of 15 cm with the same distance as at the ATM measurements. The results of this measurements can vary on different Display and other surrounding conditions like the light environment. The following table shows the QR-Code size which means its version corresponding to the error correction level and shows if the QR-Code was recognized correct.

Error Correction	L	M	Q	H
Version and size				
v18, 15cm	yes	yes	yes	yes
v20, 15cm	yes	yes	yes	yes
v22, 15cm	yes	yes	yes	yes
v24, 15cm				
v26, 15cm				
v22, 21cm	yes	yes	yes	yes
v24, 21cm	yes	yes	yes	yes
v26, 21cm	yes	yes	yes	yes
v28, 21cm	yes	yes	yes	yes
v29, 21cm	no	yes	yes	yes
v30, 21cm	no	no	no	yes
v32, 21cm	no	no	no	no
v34, 21cm	no	no	no	no
v36, 21cm	no	no	no	no
v38, 21cm	no	no	no	no

TABLE 7: This shows the relation of QR-Code version, its size and error correction level compared to the ability to read it with the given circumstances

4.1.2 WEBSITE

As it could be used in other projects as well, the source code dealing with the creation of signed and/or encrypted QR codes has been separated from the rest of the project and is publicly available on github as a PHP package. This package is then imported into and used by the server-side part of our implementation.

Amongst other the package consists of an interface `Mask` and different implementations of this interface. Furthermore the class `QrCrypt` is the main part of the package, dealing with both signing and encryption as well as the building and saving of QR codes. Both parts are discussed later on in detail. The `Composer.json` file, which is also part of the package, is needed for the package manager. It describes (amongst other things) what the package is designed for (allowing the creation of GPG encrypted or signed QR codes) and PHP version is needed (at least 5.6.3). To build the QR codes the *Endroid QR Code* library is required and used [25].

The purpose of the previously mentioned interface `Mask` is to describe the kind of information the QR code shall depict. Therefore different implementations of that interface exist to create concrete masks which can be used by the `QrCrypt` class. Contained in the `Mask` interface are two different functions.

The function `getId()` returns an unique identifier for the mask. It should be only used for this implementation of mask and be up to five characters long, to describe the mask type appropriate. To avoid misinterpretation it must not contain any colons.

The second function which is part of `Mask` is `toString()`. That method is needed to generate the string which will be encoded into the QR code. In consequence it serializes the information contained in the mask into a string.

Both functions are elementary features of each `Mask` implementation. In this work four different implementations of the interface were necessary: Firstly, `TransactionMask` and `PlainMask`, which are part of the `QrCrypt` package. Secondly, `AtmIdentifierMask` and `KeyMask`, which are not part of the `QrCrypt` package itself, as they are only useful in our implementation and would not be of much use to other programmers.

`PlainMask` is probably the easiest implementation of `Mask`. Its purpose is output the string which was given by input, for example to deliver the amount of an transaction. The `getId()` function returns 'plain' to identify this type of mask.

A `TransactionMask` is much more complex compared to the first mentioned. Analogous to the return of the `getId()` function of `PlainMask` the `TransactionMask` `getId()` gives 'trans' as an answer. More importantly this implementation contains several attributes and functions.

As `TransactionMask` is used to transform the attributes of a transaction into a string which is needed to build or interpret encrypted and/or signed QR codes containing it. Therefore selected items of the set of attributes defined within the *Sepa Credit Transfer Scheme Rulebook* (which have been discussed in particular in chapter 2.8) are used in this mask. Those attributes have been selected because of their importance for transactions. As for instance the attribute *AT-23* - the BIC of the beneficiary bank in an customer

to bank credit transfer - is not essential for transactions or TAN generation, it is not included in the source code. Important for the transfer are on the contrary the IBAN and name of the originator as well as the amount of money he wants to transfer. This amount has to be given in Euro cents, why the attribute type can be integer. Other attributes contained in `TransactionMask` are the remittance information sent by the originator sent to the beneficiary in the credit transfer instruction and the requested execution date of the instruction. Furthermore the IBAN and name of the beneficiary, the purpose of the credit transfer and the settlement date of it, too. All those attributes are named according to the *Sepa Credit Transfer Scheme Rulebook* to simplify the understanding. In comparable techniques, as SMS TAN or other, more or less the same attributes have been used, which had an additional impact on the selection.

The only obligatory attribute in this selection is the amount. Additionally the attribute `tan` was implemented, containing a corresponding TAN to confirm a transaction.

`TransactionMask` encompasses a couple of functions to check the validity of the attributes where appropriate. Those checks test whether a TAN consists of suitable characters for instance, do a general test regarding attribute type or verify if the amount is within a valid range (following *Sepa Credit Transfer Scheme Rulebook*, cf. 2.8). The functions are all named in style of `isValid...` with regard to the class diagram depicted in figure 16.

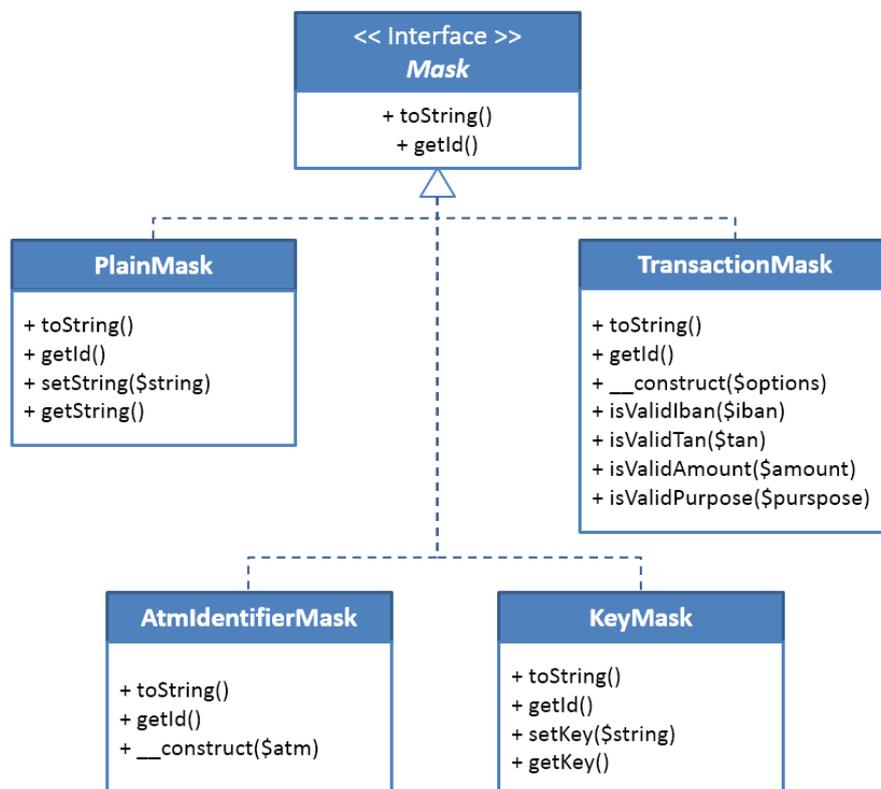


FIGURE 16: Class Diagram depicting the relationship with interface `Mask`

To create a new instance of `TransactionMask` the function `__construct($options)` is needed. `$options` in this connection is an associative array containing the attributes of the transaction. This function considers each element of the list to check for the need of throwing exceptions and is automatically invoked if a new instance of the class is created.

The parsing of the available information into a string which can be encoded into a QR code is done by the `toString()` function. It links the list's elements with an separator, set in a constant. A tabulator was chosen as separator since it is never part of any element and separates them properly.

The other two implementations of `TransactionMask` refer to key handling and the identification of local ATMs. In the former case in addition to the `TransactionMask` functions `toString()` and `getId()` (the latter returning 'key') two new functions are part of this mask. The functions' `setKey()` and `getKey()` task is hereby to transfer the GPG keys. This mask is similar to the `PlainMask`, as input is equal output in both cases.

The last mask, named `AtmIdentifierMask`, also implements `Mask` and parses the available information into a string that is encodable into a QR code. As those mask is intended to deal with the ATM identification, the parameters `id` and `time` are needed. The function `__construct($atm)` creates a new instance of this transaction mask. Again, like `$options` in the corresponding function of `TransactionMask`, `$atm` is an associative array containing the relevant attributes.

To display the dependencies of the interface `Mask` and the implementing masks the class diagram in figure 16 shall be seen. It clearly shows the corresponding public functions.

The QR code building is made using the `QrCrypt` class. If a new code shall be generated a new instance of this class has to be set up, giving a mask as parameter. Consequently, a mask instance has to be created in beforehand. All attributes and functions of `QrCrypt` are depicted in figure 17. The most important aspects are discussed in the following paragraphs. To understand the structure of `QrCrypt`, it is appropriate to consider the general data format of QR codes in this work first.

The structure of a QR code is as follows: `QCR:{$mask_id}:{$mode}:{$payload}`. `QCR` identifies a QR code as one of this project's. The `$mask_id` depicts the used mask type, referring to the value set by `getId()`. `$mode` furthermore determines whether the `$payload` is signed (s), encrypted (e), both (x) or unmodified (n).

The `$payload` is reliant on the type of mask. In case of an instance of `PlainMask` or `KeyMask` it corresponds to the text itself. A `$payload` of an instance of `AtmIdentifierMask` consists of an ID, an time stamp and an description of the ATM. In case of an `TransactionMask` instance the `$payload` includes the corresponding attributes which have been discussed above, separated by tabs.

In `QrCrypt` the submitted mask payload is saved at first using the `__construct` function. The functions `setEncrypted` and `isEncrypted` refer to the encryption. This applies in the same way for `isSigned` and `setSigned`. The directory in which the QR code shall be saved can be set and checked using the functions `setDirectory` and `getDirectory`.

QrCrypt
<ul style="list-style-type: none"> - \$encrypted - \$signed - \$directory - \$mask - \$pack - \$minVersion - \$errorCorrection - \$errorCorrectionMap - \$gpg - \$qrCode
<ul style="list-style-type: none"> + __construct(Mask) + save(\$filetype , \$filename) + display(\$filetype) + encode() + isEncrypted() + isPacked() + setPacked() + isSigned() + setSigned() + getDirectory() + setDirectory() + getMinVersion() + setMinVersion() + getErrorCorrection() + setErrorCorrection(\$minErrorCorrection) + addEncryptKey(\$fingerprint) + addSignKey(\$fingerprint) + setSize(\$size) + setPadding(\$padding) + stripgpg(\$string)

FIGURE 17: Class *QrCrypt* with its attributes and functions

The hand-over of the keys is done with the functions `addEncryptKey` for encryption and `addSignKey` for signing using `$fingerprint` as parameter. Size and padding (the latter referring the width of white border surrounding the QR code) can be set by the accordingly labelled functions.

`Encode` is the function which produces the string for which shall be contained in the QR code actually. It assembles the pieces to a string according to the structure mentioned before. With the function `Display` this string is transformed and packed into a QR code, which is saved by the `Save` function. Hereby, some validation checks regarding data type, directory and suchlike are made and appropriate exceptions thrown if required.

The attributes mentioned in the `TransactionMask` context are added together about the size of 288 byte, as calculated in chapter 2.8. The chosen key size in this project is 1024 bit. Since the size of the strings for signed and/or encrypted QR codes is minimum in the range of this key size, the total size of the attributes does not matter so

much. Nevertheless not too many attributes could be added to the class, to fit the size limitations.

In this section the website regarding aspects of the QR source code was presented. The app-side differences and implementations are discussed in the following section. Those discussion deals especially with the rearward use. This means among other things the decryption and reading of the information contained in QR codes and the corresponding masks.

4.1.3 APP

In following section the counterpart of the before explained signed and encrypted QR code creation will be laid out. This implementation is used within the Android applications created and presented in detail in chapter 4.3.1.

There are three packages involved dealing with the encryption and handling of encrypted and signed QR codes: *TARTCommon*, *TARTMobile* and *TARTVision*. As described in section 3.5 there are several use cases in which QR codes are scanned, implementation-wise most aspects of handling the scanned information are similar. Only final minor changes in dealing with extracted information from a scanned QR code, e.g. IBAN or private key, at the most latest moment of QR code interpretation are necessary. This is mostly with regard to display and interaction capabilities of Google Glass in contrast to an Android smart phone. Avoiding code replication in both implementations, a common code basis for handling QR code interpretation was designed: the package *TARTCommon*.

The hereafter description will follow the program flow which occurs when a QR code is handled within the *TARTMobile* and *TARTVision* application. A class diagram of the involved classes is shown in figure 18.

Given a scanned QR code, which was created via the implementation described in section 4.1.2. Scanned data, is passed over at the construction of an instance of *QrCryptDecoder*. A special decryption handler equipped with the user's private key can be added, this will in the later process allow for decryption of encrypted data within the QR code. As the former is optional, one has to provide a platform depending instance of a *QrCryptMaskFactory*, which can either come from the package *TARTMobile* or *TARTVision*.

A *QrCryptDecoder* instance will provide a public method *getMask()*, a call of this method will create a concrete instance of *QrCryptMask*. Concrete within this context then an instance of *AtmIdentifierMask*, *KeyMask*, *PlainMask* or *TransactionMask* which all derived from the *QrCryptMask*. Additionally, the scanned data's payload will be decrypted and / or its signature will be checked if necessary, further methods will therefore be able to operate on plain data.

The actual characteristic of the *Masks* differ in two ways one is in the handling of the payload. A *PlainMask* will provide only a simple *getString()* method which will return the transferred payload as string. In contrast to this a *TransactionMask* will provide a suitable range of methods to directly access relevant transaction information, for exam-

ple a `getOriginIBAN()` method returning the sender's IBAN. The second difference is achieved through the different `QrCryptMaskFactory` implementations, these determine which *Activity* the associated *Masks* will return for further actions. This system allows for handling payloads within both projects in the same way. But, while omitting code replication and allowing simple editing in one place, it is still possible to direct program flow in different ways depending on the underlying platform.

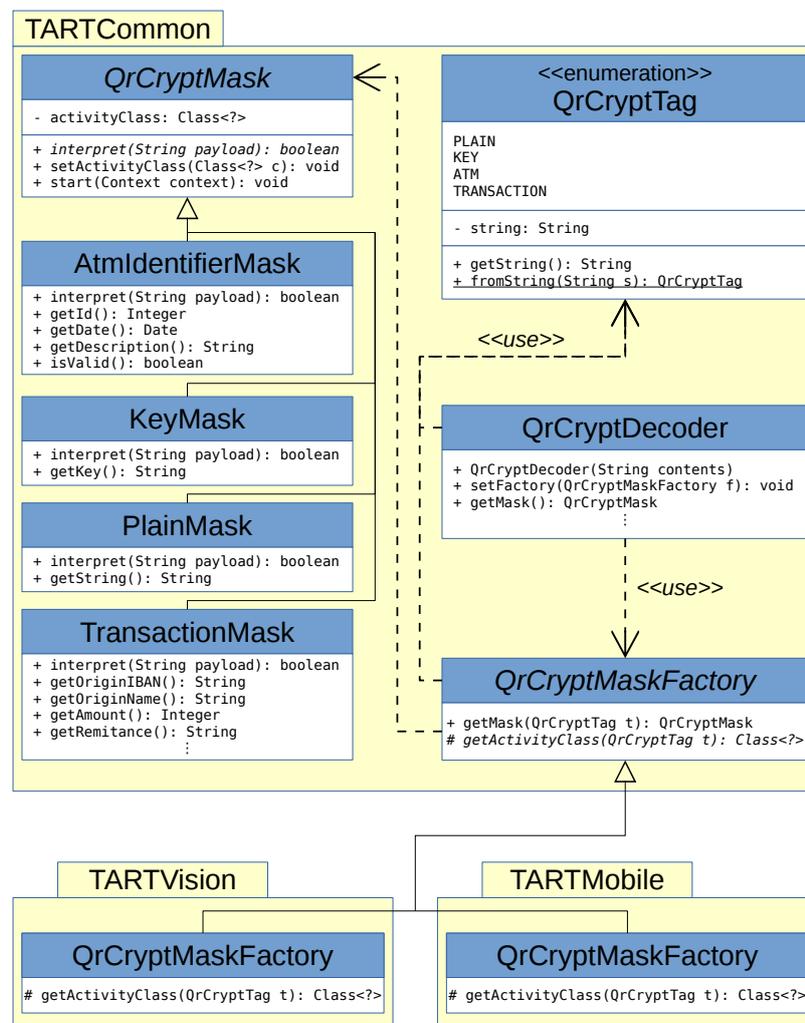


FIGURE 18: Class diagram of TARTCommon package and the corresponding classes within the TARTMobile and TARTVision application

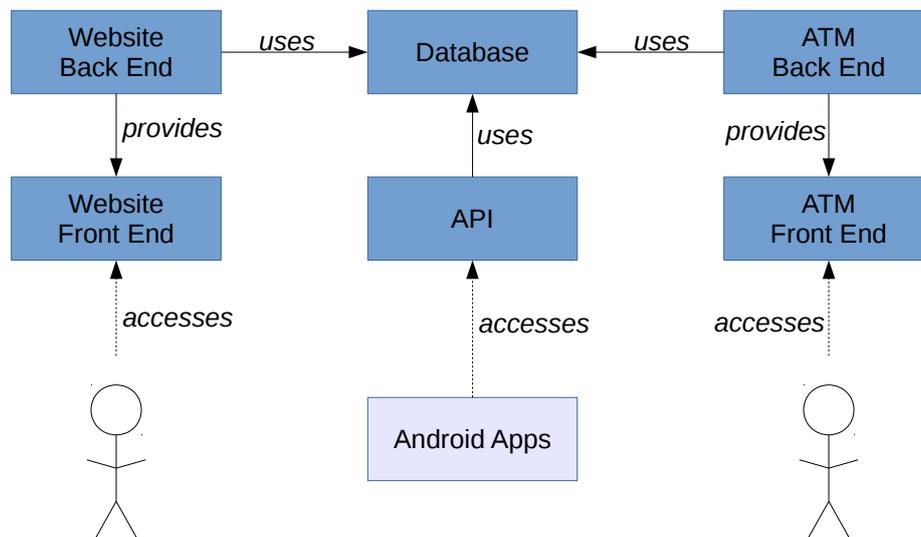


FIGURE 19: A diagram of the different components which are to be designed in this section. The development of the Android applications will be discussed in section 4.3.

4.2 SERVER

The developed project depends on two services in addition to the mobile applications: An online banking website and an ATM. As it is not feasible to use an actual ATM, it was decided to mock one by only implementing the user interface, without any actual capabilities to withdraw money beyond a visual indicator. This section covers the concept and implementation of both services and describes the Application Programming Interface (*API*) which will be used by the mobile applications to communicate with them.

4.2.1 CONCEPT

Figure 19 depicts the individual components which were needed in order to provide an online banking website and a mocked ATM: Both the online banking website and the mocked ATM consist of a tightly coupled back end and a front end. It is the job of the former to manage data and provide it to the latter, which deals with the presentation to the user. Both back ends are connected to a shared database, which stores information about the accounts and transactions. In addition, an *API* which can be used by the Android applications had to be designed (see section 4.2.2).

While it was clear that the online banking website should be implemented exactly as its name states - a website - there were multiple options for the implementation of the mocked ATM:

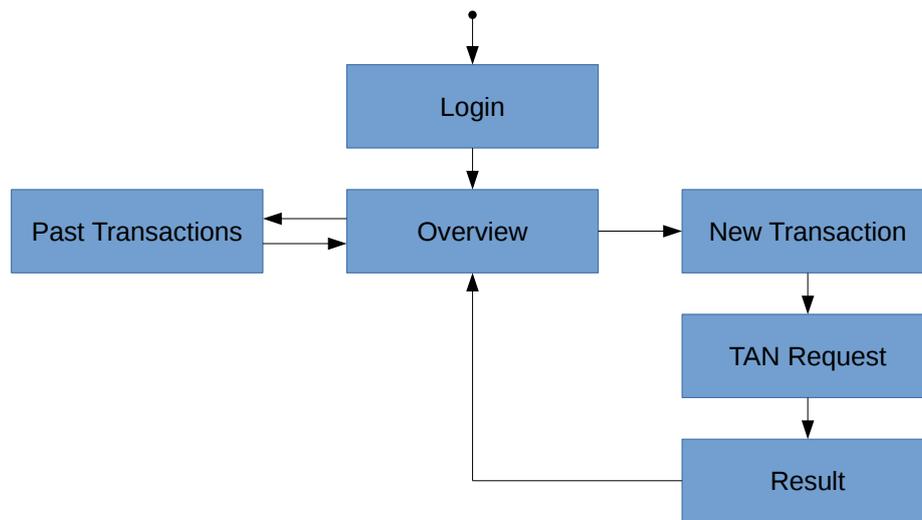


FIGURE 20: An overview of the online banking website

- Both front and back end are part of a local application which runs on the ATM's machine itself and is connected to the database.
- Both front and back end are part of a website, which would be displayed using a browser on the ATM's machine and is connected to the database.
- The front end is a local application which runs on the ATM's machine and is connected to the back end, which is a dedicated application running on a server. Only the latter is connected to the database.

For simplicity reasons, the second option was chosen. This resulted in the development of a single website which serves the online banking website, the website which gets displayed on the ATM's machine and also the API. While this would not be a practical solution for a real-world application, it prevented a considerable amount of code duplication and simplified the development by reducing the number of applications which had to be built.

Despite this bundling, the online banking website and the ATM website are separate concepts, and shall be explained successively below.

ONLINE BANKING

The online banking websites primary focus is to make an exemplary use of the QrCrypt package (see section 4.1) and to provide a counterpart for the NFC TAN Device (see section 3.1). Thus, a minimal website that makes sensible use of these components suffices. Figure 20 shows the planned overview of the banking website.

In total, the online banking website consists of 6 pages: A new visitor arrives at the login page. Once he authenticated himself, he will be shown an overview of his account.

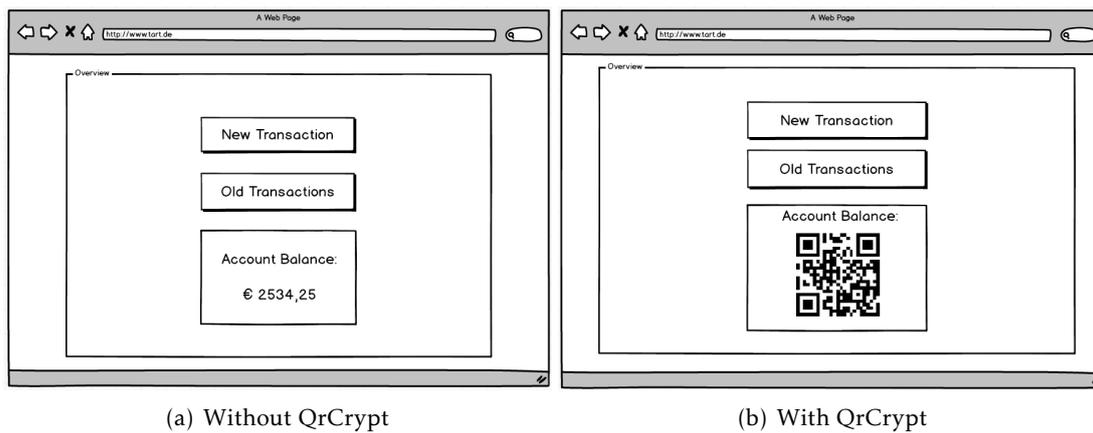


FIGURE 21: *Mockups of the overview page*

From there he can either choose to view his past transactions, or make a new transaction. In order to complete the latter, he will be prompted for an TAN. Once he submits his TAN, he will be shown whether the TAN was correct and thereby whether his transaction has been successfully stored or not.

As stated above, one major goal of the online banking website is to showcase the QrCrypt package. Thus, sensitive information such as the accounts balance or transactions should not be shown as plain text, but rather as a signed and encrypted QR code. However, this would complicate the usage of the website in cases where the user does not need the additional protection, for example while he is at home using his private computer. Thus, the website was designed to provide two different modes: One with the QrCrypt package enabled, and one which does not use QrCrypt except for the TAN request (as there is no sensible plain text alternative for this). A user can choose the mode he wishes to use at the login page.

The individual pages of the website will be discussed below in short detail.

Login

This page mainly consists of a login form. In addition to the usual user name and password fields, there is a checkbox which is used to choose one of the two modes mentioned above.

Overview

For the overview page, two different versions exist. One using the QrCrypt package, shown in figure 4.21(b), and one without, shown in figure 4.21(a). Both include a link to the list of past transactions and a link to create a new one. Additionally, the current balance is shown. Depending on the active mode, this is done in plain text or as a QR code.

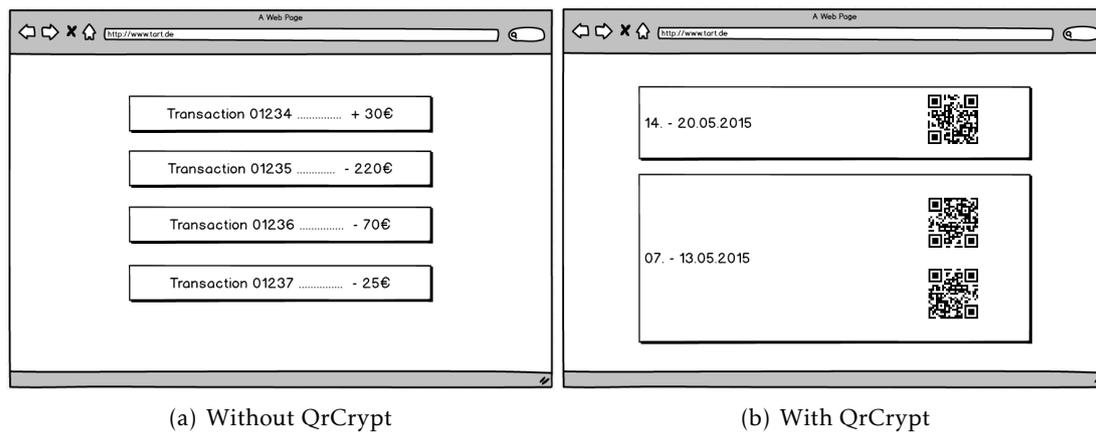


FIGURE 22: *Mockups of the past transactions page*

Past Transactions

On this page, past transactions from or to the current account will be shown. With the QrCrypt package disabled, this is a simple list of transactions, as depicted in figure 4.22(a). With the QrCrypt package enabled, QR codes which include the transaction details will be shown instead (figure 4.22(b)). As it would be very hard to scan through this list in order to find a specific transaction, they are grouped weekly. This makes finding transactions a lot easier, while an observer would only be able to make a rough guess about the frequency of transactions.

New Transaction

This page displays a form in order to begin a new transaction. The fields are:

- Recipient IBAN
- Amount
- Purpose Code
- Remittance Information

Only the first two fields are mandatory. Once the form is submitted, the fields are checked for validity. If the recipient IBAN is found in the application database, the amount is a valid number, the purpose code consists of up to 4 letters and the remittance information does not exceed 140 characters of a limited character set, the user is redirected to the TAN Request step (see below). If not, the form is displayed again and an error message is shown. These restrictions are due to the SEPA guidelines (see section 2.8).

TAN Request

On this page, a signed and encrypted QR code, which includes transaction details and the randomly generated TAN, is shown with an input field below (see figure 23). As stated above, this QR code is shown irrespectively of the mode chosen on the login page. The user should scan this code using one of the Android applications and enter it in the



FIGURE 23: A mockup of the tan request page

input field. This requires the user to have his private key scanned beforehand, as the QR code is encrypted. After submitting the form, he is redirected to the Result page.

Result

The Result page simply states whether the TAN was correct, and hence whether the transaction was successfully stored or not.

ATM

The ATM part of the website simulates the display of actual ATMs, its use case and the general procedure is explained in section 3.4. Each ATM is identified by an id, which is included in the identifying QR code along with a short description and a timestamp of the generation of said QR code. Using the API (see section 4.2.2), a user can start a withdrawal at a given ATM. From this point on, he has a limited time frame to enter his pin and complete the withdrawal, for security reasons. There are only two pages needed for this part of the website:

Landing Page

A mockup of the landing page is shown in figure 24. It displays the QR code which includes the signed details about the ATM and is constantly regenerated. Next to it, there is an input field which allows a user to enter his PIN. Once a customer enters his PIN, the server should do the following checks: First, it has to determine whether a user is currently withdrawing money, i.e. whether a user started a withdrawal within the time frame mentioned above at this ATM. Then, the entered pin of that user is compared with the one that was entered into the ATM. If both checks succeed, the withdrawal was successful and the next page is shown. Otherwise, an error gets displayed.

Successful Withdrawal

This page is displayed after a user successfully entered his pin on the landing page. It

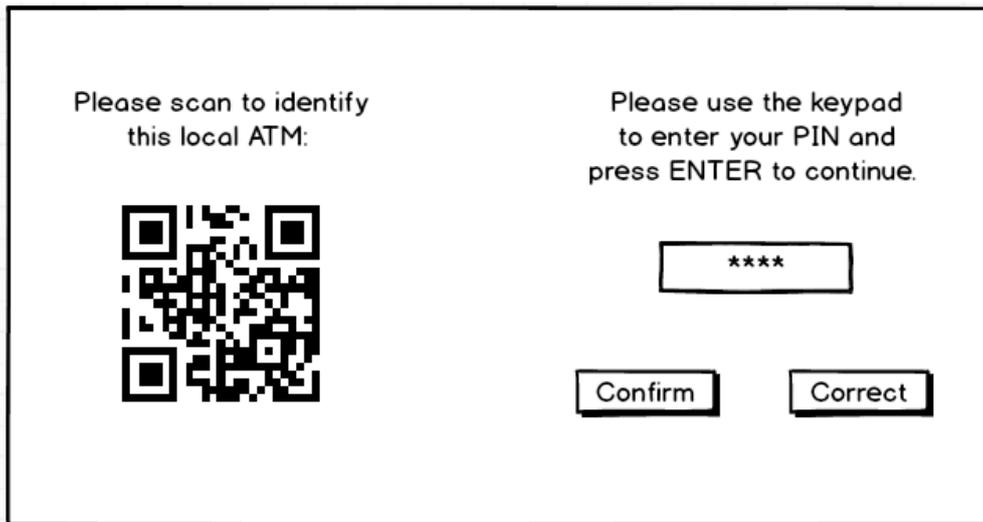


FIGURE 24: A mockup of the ATM page

is merely a visual identification that the withdrawal was successful. In a real world application, the ATM should hand out the selected amount of money here.

4.2.2 IMPLEMENTATION

The actual implementation of the website (which includes the online banking website, the mocked ATM and the API) will be described below. First, an overview of the development stack will be given, followed by the database layout. Then, the three parts of the website (online banking, mocked ATM and API) will be discussed separately.

DEVELOPMENT STACK

As the website is primarily a proof of concept and does not need to handle huge amounts of traffic, the choice of the development stack was mainly decided by familiarity with existing tools. Thus, the application was built with *PHP*[57], using the *Laravel*[40], framework.

Laravel is an open-source Model-View-Controller (*MVC*) framework written in *PHP*. [40] It utilizes *Composer*[17], the de facto standard dependency manager for *PHP*, which allowed the *QrCrypt* package (see section 4.1) to be bundled as a separate *Composer* package.

While a complete description of *Laravel* is out of scope of this paper, the request cycle as seen in figure 25 will be explained briefly: When a user requests a page, *Laravel's router* examines the URL and chooses a *controller* which will handle the request. The controller can interact with various *models*, which are abstractions of database entries. It then creates a view, which is essentially a template for an *HTML* document, and fills it with data. This view is finally transmitted back to the user and gets displayed in the browser. This allows for a clean separation of the online banking website, the mocked

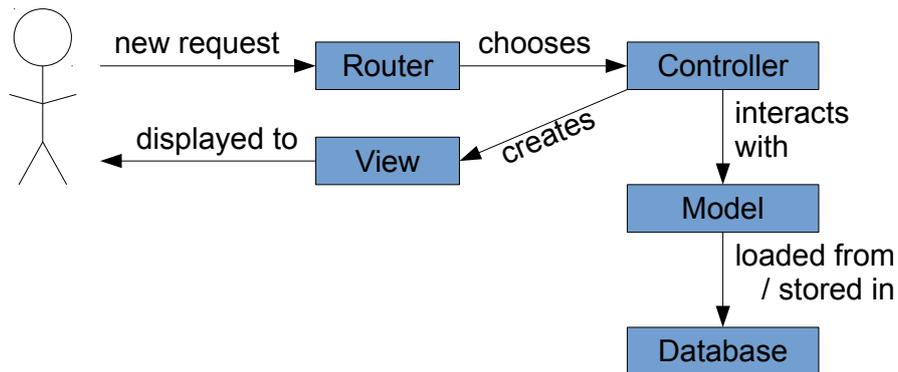


FIGURE 25: *The request cycle of Laravel*

ATM and the API: While they share the database, models and router, they each have their own set of controllers and views.

DATABASE

As shown in figure 26 there are three database tables which are of importance for this implementation: users, transactions and atms, each one with its own model class (see above).

Each user is identified by a unique id. He has an email address which is only used at the login form, in combination with his password. In this implementation, each user has only one banking account, hence his accounts information (IBAN, current balance, credit card pin, the security token and the fingerprint of his public key file) are directly saved to the users database too.

Transactions are also identified by a unique id. Originator and beneficiary of the transaction are referenced by their id. The remainder of the fields are used to store the rest of the transaction details which are used by the QrCrypt package, with the addition of a flag which specifies whether the transaction has already been executed.

Likewise, the atms table uses an id field for unique identification. It furthermore includes a description, which is embedded in the QR code shown on the ATM display. Lastly, it stores information about the last user which started a withdrawal at this ATM: A timestamp of the beginning of the withdrawal, the chosen amount, and a reference to the user himself.

ONLINE BANKING

As stated above, the online banking website can be used in two modes: One makes extensive use of the QrCrypt package, while the other one provides the relevant information as plain text instead (with the mentioned exception of the TAN request QR

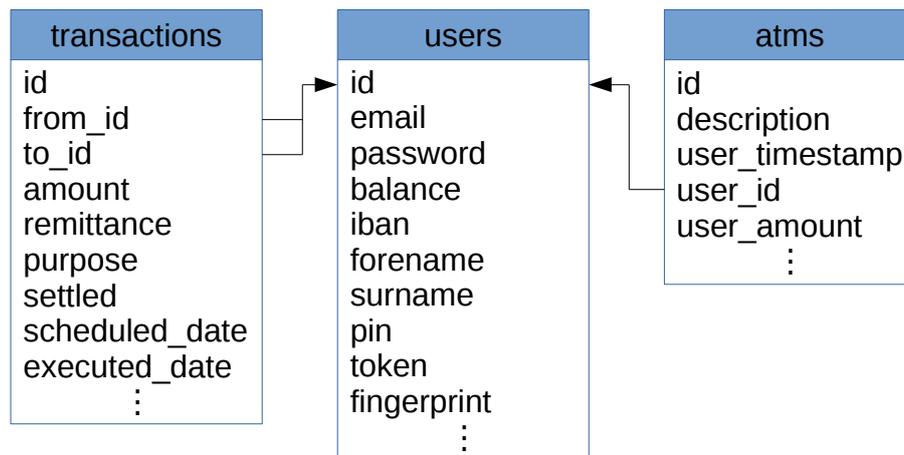


FIGURE 26: A short summary of the database setup. Each table has a few additional fields which are only used by Laravel's object relational mapping engine.

code). To provide a clean and consistent pattern which can be used to present sensitive information according to the chosen mode, the `Presenter` class was designed (see figure 27). It serves as an interface to the `QrCrypt` package. For every implementation of `Mask` (see section 4.1), an implementation of the abstract `Presenter` class has been provided. They are used to create a new instance of said concrete `Mask` (for example by passing the contents of a Laravel model to the `Masks` constructor) or to create a view instance which holds the same information in a human readable form. The abstract `Presenter` class owns the method `present()`, which checks for the active mode, and returns a view instance that can be included into the response to the user:

If the QR code mode is used, it generates the QR code using the `QrCrypt` class and the mask which is provided by its own concrete implementation. The URL of this image is then provided to a special view, which holds the correct HTML data to present this image. This view is then returned. If the QR code mode is not used, it simply returns the view instance which is provided by its own concrete implementation instead. This method separates the correct presentation from the presented information, and bundles the correct invocation of the QR code package in one place.

Apart from that, the implementation did not provide any challenges worth mentioning. Screenshots of the final implementation can be seen in the appendix (section A.1.2), as they do not hold major differences from them presented mockups.

MOCKED ATM

The implementation of the mocked ATM did not hold any major challenges. The concept (see section 4.2.1) was extended by two elements: First, a visual indicator was added that shows when the QR code should be reloaded. In addition, there is now a keypad

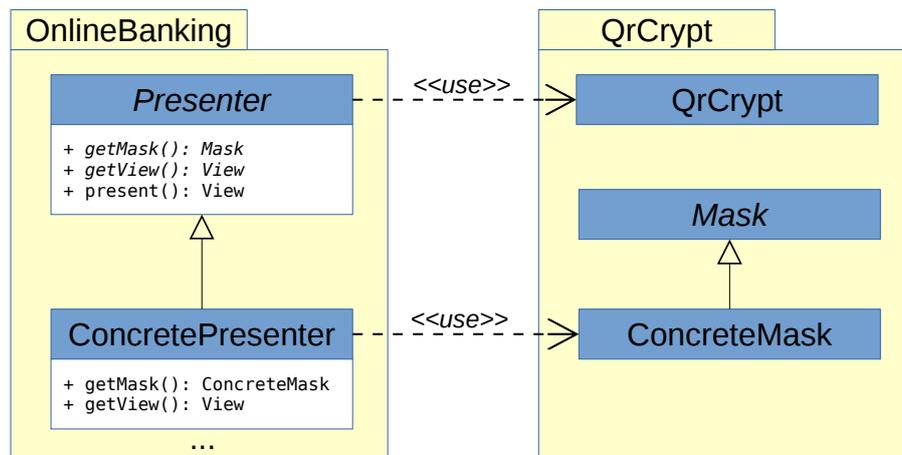


FIGURE 27: A class diagram of the presenter class

included on the website. This allows the usage on a device which has a touchscreen, but no physical keypad. The result can be seen in figure 28.

API

In addition to the two web pages, the server has to provide an API to be used by the mobile applications. For the same reasons as mentioned above in relation to the implementation of the mocked ATM, this API is provided by the back end of the two web pages, too.

In total, there are three API actions which can be called by the mobile application:

- `/atm/{id}/start` starts the withdrawal at the ATM identified by `{id}` (see section 4.2.2). The amount has to be specified as a POST parameter.
- `/atm/stop` stops a currently running withdrawal at any ATM.
- `/balance` returns the current balance of the associated account

All of the above require authentication using an IBAN and the correct token, which have to be supplied as POST parameters. All of them return a JSON response which at the least includes an indicator of success.

Additionally, a fourth API action exists, purely for maintenance reasons:

- `/key/{id}` returns a QR-Code which includes the private key of the account identified by `{id}`. In a real world application, this should under no circumstances be publicly accessible. Instead, the generated image should be mailed to the owner of a newly created account per mail. He would then be able to scan it with the TARTMobile application, which would in turn allow him to use the forecited API actions.



FIGURE 28: *The ATM in use*

4.3 APP

Within this section detailed information about the development of two Android applications is given. At first a small round up about the conceptual ideas of the applications is presented. In the following the used development environment is stated while further information about used libraries discussed.

APP DESIGN

As presented with in section 3.5 the designed system will use two Android devices: On the one hand there is Google Glass, on the other hand there is a smart phone. Both devices should be able to perform certain task: scanning QR codes, decrypting PGP messages, scanning NFC tags, connecting to a web server and nevertheless display information to the user in addition to accept certain input of the user. As development time with in a lab is limited, a focus on fast development is key. This includes focusing on one new development stack rather than two, the Google Glass is partnered with an other Android smart phone. This allows for reuse of code as certain parts within the Google Glass application could be shared with the smart phone application, refer section 4.1.3.

Android smart phone are rather divers as there exist different Android (API) version within the Android hardware family. Google Glass API is available with Android version 4.4 which corresponds to the API version 19. To be fully compatible the smart phone therefore is set to be of at least version 4.4, too.

Further requirements become clear in respect to chapter 4.1 and the idea presented with the NFC TAN device, refer to section 3.1. This is the need for a NFC-capable smart phone equipped with a camera capable of scanning rather detailed QR codes.

While implementing it was noticed that Google Glass' camera is rather limited in its use to scan generated highly dense QR codes, 4.1.2. With this limitation the need for a data transfer between the smart phone and Google Glass arise. As one hardware-wise given fact Google Glass supports Bluetooth, which excels at the task of communication wireless within near range.

The available Google Galaxy Nexus produced by Samsung meet all of the former stated requirements. For this project two smart phone have been updated to CyanogenMod 11 Snapshot M11 [19] which is a modified version of Android 4.4.4 KitKat. This open-source operation system is based on the Android mobile platform, beside many other modifications it supports root access to simplify development.

Implementation-wise the development for Google Glass and an Android smart phone, both within the same Android API level are rather similar. Nevertheless certain differences in displaying information and how the user can provide input differ. With the missing touch screen and only a small touch pad, no NFC-hardware and only a rather limited camera on Google Glass, two separate Android applications are developed within its project. A Google Glass application called *TARTVision* and a smart phone application called *TARTMobile*. Both share some common code but most interaction parts are to distinct to be handled with in one single application.

4.3.1 DEVELOPMENT ENVIRONMENT AND USED LIBRARIES

For developing an Android application two major platform are available: Android SDK, Eclipse IDE bundled with the Android Development Tools (ADT). The former presented applications are developed with Eclipse IDE. With the availability of well documented development guides and former familiarity with Eclipse, is chosen for development.

For the decryption and checking of valid signatures SpongyCastle OpenPGP [73], a Android implementation of Bouncy Castle [81] is used. While certain methods to access the camera are provided with the Android API, no decoding function for QR codes exists within these. Therefore to decode QR codes ZBar [92] an open-source barcode reading library is used.

Both libraries are rather complex and require thoughtful implementation as documentation from the libraries projects is limited to non-existing.

4.3.2 NFC AUTHENTICATION

This section of the implementation chapter will present the concept and actual implementation of an NFC authentication used within the TART project. Within the concept chapter, the specific details about the NFC authentication, which NFC tag is used and which data is stored will be discussed. In the implementation chapter thereafter a detailed view on the actual implemented code is given.

CONCEPT

As presented within chapter 3.1, an NFC debit card can be used to provide the user with an additional token to perform a full two-factor authentication. The TARTMobile application (chapter 4.3.1), which is designed for an NFC capable Android smart phone, will leverage this to enhance security and usability of the following presented concept.

For the purpose of an additional token a secret hard coded read-only set of information stored on an NFC tag, as presented in chapter 2.3, can be used. Within this proof-of-concept implementation, a none-cryptographic-enhanced tag is used as it suffices to present the the overall concept. The actual tag used is a NXP NTAG203 chip complying with the ISO 14443A standard, a single tag can store up to 137 byte [8]. This limited storage capability still suffices to hold up a costumer's IBAN and name (refer to chapter 2.8), further more a ten symbol long security token can be added. The length of the chosen token is kept short due to the proof-of-concept scenario, it should hold more entropy n a real world scenario. As presented in chapter 2.3, an NFC tag can be formatted to be compatible with the NDEF format, which allowed the storage of multiple data sets on a single tag. Within the TART project, the costumer's IBAN, name and personal token are stored as individual NDEF records of one single NDEF message. Each record is set to be of type "well known" (bit value: 0x01) and of record type "RTD TEXT" (bit value: 0x54), as can be seen in figure 29. The used tags have been formatted and filled with information via an NFC capable smart phone (refer to 4.3.1) and the "NFC Tool" Android App [52].

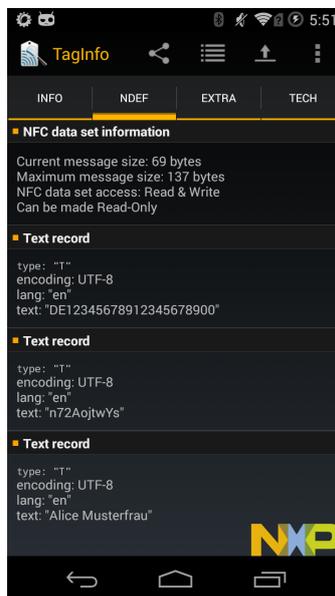


FIGURE 29: NFC records within a single NFC message to be used as two-factor-token.

Providing correct and well formatted information via an NFC tag, a user can authenticate himself to the TARTMobile application as he is bearer of his personal security token (his debit card) to access his account. This is done via the login process of the application, which can be initiated via two scenarios. One is to use the autostart abil-

ity of TARTMobile by simply bringing his NFC tag into the smart phones NFC range. Another scenario is in place when the user launches the application and is prompted with a request to provide his personal NFC tag. While this will only provide one factor, in this case the physical object in possession of the user, an additional factor has to be provided by the user to perform a complete two-factor-authentication (refer to 2.1.4). In case of an ATM cash withdraw (refer to 3.4) within the TART project knowledge of the user's PIN is required, too. As for an online transaction via the TART website (refer to 3.3) a user has to provide his login credentials as second factor.

IMPLEMENTATION

The following part describes how the former concept was implemented in the TART-Mobile application. To be able to use the NFC hardware of an Android device, specific permissions have to be requested within the application's `AndroidManifest.xml`.

Stating these permission and feature requests does not check whether a specific feature is present but rather informs about the application's desire to use a feature [85]. The permission request is also necessary to be able to access stated hardware or system resources. Within the `AndroidManifest.xml` of TARTMobile, a standard launch intent is set to start the `MainActivity` which will request the user's NFC debit card to login. Another intent is also set, this intent will check any NFC tag read by the Android smart phone and check whether it is NDEF formatted and contains data in the NDEF text format. Data read from a presented NFC tag, while in the login screen and also via the NDEF intent, has to be checked if it contains the three former defined data fields.

The `processIntent` function of the `MainActivity` class takes the NDEF message and passes it to the `readText` function. The `readText` function expects the NDEF message read by a NDEF intent, it will then separate it into its three record parts. These parts are then decoded as they are UTF-8 encoded and have a language code as well. Read information of IBAN, token and user name is used to login the user via the `login` function of the `User` class. If a different than the already logged in user's NFC tag is provided, the current user is logged out immediately to avoid false accounted actions when using multiple debit cards.

```

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.tartmobile"
  android:versionCode="1"
  android:versionName="1.0" >
  ...
  <uses-permission android:name="android.permission.NFC" />
  ...
  <uses-feature android:name="android.hardware.nfc"
    android:required="true" />
  ...
  <application
    <activity
      android:screenOrientation="portrait"
      android:configChanges="orientation"
      android:name=".MainActivity"
      android:label="@string/app_name" >
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category
          android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
      <intent-filter>
        <action
          android:name="android.nfc.action.NDEF_DISCOVERED" />
        <category
          android:name="android.intent.category.DEFAULT" />
        <data android:mimeType="text/*" />
      </intent-filter>
    </activity>
    ...
  </application>
</manifest>

```

Listing 1: *AndroidManifest.xml* file of TARTMobile

4.3.3 COMMUNICATION BETWEEN SMART PHONE AND GOOGLE GLASS

Within the following section, it will be described how the data communication between the TARTMobile and TARTVision application is implemented. Prior to this, the concept of this communication is laid out, this includes the chosen communication channel as well the designed communication protocol. To begin with, the need for a communication between the TARTMobile smart phone and the TARTVision Google Glass application is explained through their use case scenarios.

CONCEPT

TARTMobile is the main usage component within the TART project, nevertheless with TARTVision an easy and modern way to handle one's banking processes was developed. One of the key features of TARTVision is its QR scan feature which is used while doing an ATM cash withdraw and while accessing the TART online banking website. Within the TART online banking website different transactions and as well TAN requests can be displayed via a QR code, refer to section 4.2.1. These QR codes are encrypted and signed as described in section 3.2 and 3.3. While in the ATM cash withdrawal scenario the displayed QR code is only signed, it can as well be scanned with Google Glass and the desired cash out can be processed.

As stated in section 3.5, for PGP decryption the private key of the specific user is needed. This private key has to be scanned by the user, it can therefore be mailed to the user in a printed out QR code, as described in section 3.5 and 4.2.2. As for checking the signature of a signed QR code, as one is used at an ATM, the needed public key is already provided with the TARTVision and TARTMobile applications. Given a user has correctly added his private key to the smart phone application, it still has to be transferred to the Google Glass application. Additionally, if a user starts a cash withdraw with TARTMobile, he can within the withdraw process, choose to switch the current control flow to TARTVision. Changing the control flow allows to prepare a withdraw action in advance, then while moving towards the ATM the user may put his smart phone back to his pocket. Though while then arriving at an ATM, the user has not to take out his smart phone again as he can henceforth handle his actions via his Google Glass using TARTVision.

This motivates for the need of a way to communicate information from TARTMobile to the TARTVision application. Within this transfer a private key or all necessary information to handle an ATM cash withdraw is exchanged. In both cases the data amount to be transferred is rather small and bounded by the size of the used cryptographic keys. As presented within section 2.5, Google Glass can communicate via Bluetooth and this suffice for the data transferred refer section 2.9, this communication channel is used. With this another requirement is added to the capabilities of the used Android smart phone, as already presented in 4.3.

The data which is transferred is formatted in a special fashion, as a self-designed protocol defines.

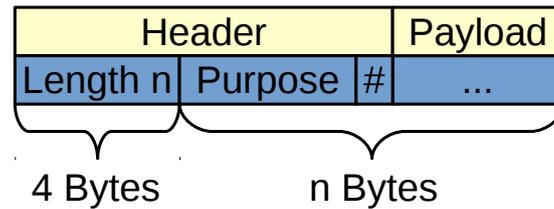


FIGURE 30: *Self-designed communication protocol.*

Figure 30 shows the features of the self-designed transfer protocol within the TART project. The header starts with a four byte long length field which holds the length of the following message including its purpose tag and delimiter. Following this static length field a purpose field with variable length holds a describing tag of the message's purpose. Separated by a #-symbol the actual payload completes the message. An example message formatted according to this protocol can be seen in figure 31.

Header		Payload	
42	SCAN #	DE12345678912345678900#n72AojtwYs#100	

FIGURE 31: *Example of a SCAN-message, which hands over control flow to Google Glass. The message is formatted to be conform with the self-designed communication protocol.*

With this example one of the two use cases is presented, here the transfer of control flow within an ATM cash withdraw transaction. To perform a valid ATM transaction the user's device needs to provide three essential pieces of information, this is the user's;

- IBAN,
- token,
- and desired amount.

This data will be used for authentication when using the online API, as described in section 4.2.2, to initiate or stop the cash withdrawal. A Bluetooth transfer for an ATM cash withdrawal can be initiated within the ATM cash withdraw use-case as described in section 3.5. The ATM cash withdrawal functionality can be selected from the home screen of the TARTMobile application. Furthermore, a user can start the ATM cash withdraw process by scanning an ATM's QR code, he than can also select to proceed with his intend via Google Glass and hand over control flow via Bluetooth. Besides these two ATM scenarios there are again two key transfer scenarios in which a user is able to initiate a Bluetooth transfer. This is first of all when selecting the *Send to Glass* button within the *PGP Key* settings accessible via the home screens menu. A second way to transfer a key to Google Glass via Bluetooth can be initiated when actually scanning

a private key via the TARTMobile QR code scanner. With this the conceptual details about the use of Bluetooth within the TART project are provided and a closer look to the actual implementation is given within the next section.

IMPLEMENTATION

As the TARTMobile application itself can be fully used without Google Glass, the existence of a Bluetooth adapter available to the application is only checked at the start of a Bluetooth transfer. Nevertheless, the application's `AndroidManifest.xml` states the requirements for Bluetooth in the same fashion as described for the NFC requirements, refer 4.3.2. With the initializations of a Bluetooth adapter in Android one has to provide the name of the to be paired device. As for this application only a connection to a Google Glass device is desired from all available Bluetooth devices a connection is only setup if the specific device's name contains the string *Glass*. This selection of to be paired device suffices for a proof-of-concept implementation but should be more sophisticated in a real world scenario. After the initializations of a connection between to Bluetooth devices, separate applications can communicate using the already established connection. To select a corresponding application on the counterside a application specific *Universally Unique Identifier (UUID)* has to be set. [9] With this the prerequisites, for a successful Bluetooth communication, are explained, the control flow while transferring data via Bluetooth within the TARTmobile and the TARTVision application is shown in figure 32.

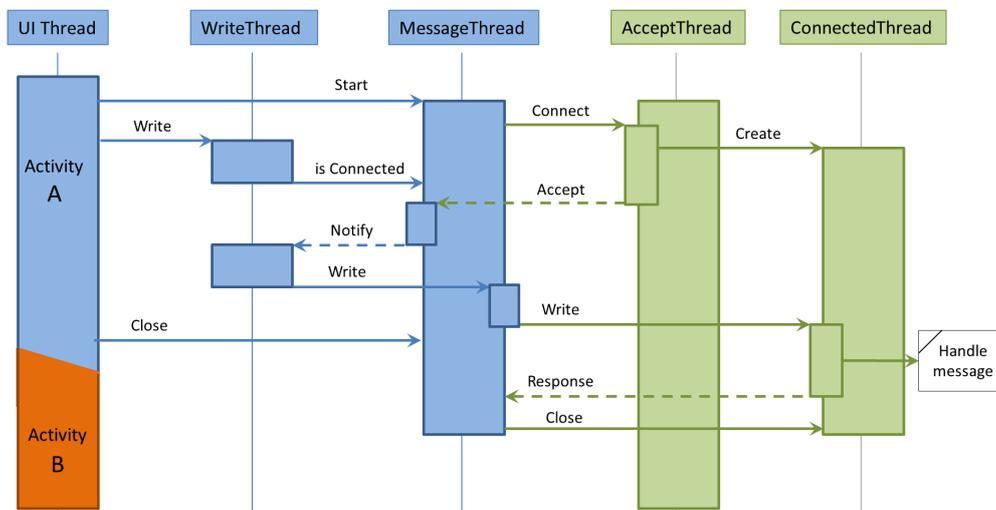


FIGURE 32: Sequence diagram of Bluetooth communication within the TART project. Blue threads run on the TARTMobile application, green threads are part of the TARTVision application

The following will explain the sequence diagram in figure 32, it is separated into two different colors which indicate the actual device on which a specific thread is executed. Blue being the TARTMobile application whereas green threads are run on the TARTVision application.

If an *Activity* wants to send a message, for example the message shown with in figure 31, control flow will be within the *User Interface* thread to the left.

This is the thread which handles all display and input activity to and from the user, it must therefore not be blocked for any longer time. If this thread gets blocked nevertheless, the Android system will pause die currently running application and will prompt the user with a "Application Not Responding" dialog. [2] As different methods, provided by the Android's Bluetooth classes, are blocking methods, which eventually could cause such a blocking, they can not be run in this thread. To counter this, the transfer is handled within different threads, it is initiated via the *MessageThread*. This thread will call different Android methods to start the Bluetooth device and connect to the desired counterpart. Within this thread, methods will then use the UUID and set selector "Glass", as stated earlier, to initiate a communication between the corresponding TARTVision application on the user's Google Glass. To transfer data at first a connection has to be initiated on which then data can be transferred.

The connection request by the TARTMobile application's *MessageThread* is handled within the TARTVision application's *AcceptThread*. This thread will then create a data transfer socket and hand this single socket over to be the *ConnectedThread* of the TARTVision application. This is done to allow multiple requests to the *AcceptThread*, which otherwise would be blocked by a single data transfer request. With the transfer socket created and delegated to the *ConnectedThread* a positive response can be send back to the TARTMobile application, as the data transfer request was accepted.

While creating a connection to the TARTVision application, a write request, which will hold the actual to be transfered data, can be made. The write request is done by the *User Interface* thread, this thread will start up a new *WriteThread* instance. This threaded instance will try to transmit its data via a socket to the *ConnectedThread* on the TARTVision counterside. Therefore it requests a transfer socket with the *MessageThread* of the TARTMobile application, this socket can and will only be available if the former discussed data transfer request was accepted and thereby a transfer socket was created. Conclusively the *WriteThread* has to wait until it gets notified by the *MessageThread* to be handed over the required transfer socket.

With then all prerequisites complete the actual data can be transferred. It will be received by the TARTVision application's *ConnectedThread* which will hand of further message handling to different methods, depending on the purpose field within the received message. As Bluetooth connection can get lost, for example by leaving the receiver's data receive radius, a response to the TARTMobile *MessageThread* is only given if the complete message was received. While developing, non-closing or early closing sockets caused several problems while communicating, introducing the self-designed protocol and waiting for an response from the receiver's side provided a suitable solution. This is determined by the message length introduced with the self-designed protocol, refer to section 4.3.3. With this response and a close request by the *User Interface* thread a transfer socket can be gracefully closed. This completes a full message transfer between the TARTMobile and TARTVision application for example to hand over control flow to the TARTVision application during a ATM transaction.

5 EVALUATION

This chapter regards the evaluation of the developed concepts and system. Therefore it discusses the security improvements of the presented project in a first step. In a second section the usability aspects of the present work are discussed briefly.

The chapter concludes with a summary. In this section a conclusion is done, reflecting the work done. The major challenges of the project are considered. Ideas and suggestions for future work based on this work are presented additionally.

5.1 SECURITY

One of the main driving forces of this work was the improvement of security in the banking context. This was to be achieved through latest technology, such as Google Glasses and NFC technique. This section discusses the eventual benefits and improvements of the project. The analysis conducted in the same order as the ideas have been presented in chapter 3 to clarify the explanations.

The first presented approach dealt with NFC authentication. In the presented work an app was developed which enables the user to authenticate himself using NFC tags. By way of illustration the technique was used in the context of banking. The features developed android app include the display sensitive information as well as the reading of signed or encrypted QR codes. Since those and other functionalities are personal and contain sensitive banking information the the access should be restricted and secured.

To use NFC as authentication provides an easy and secure way of authentication. As the user has to be in possession of the smartphone with the app on the one hand and in particular of the debit card the first factor of a two-factor authentication process is given. Within the app the only information given directly without any further confirmation or authentication is the account balance.

When considering the concept of the app the easy access to the named information was extensively discussed. Due to the fact, that those information is open to anyone in possession of a debit card (if it is his own or not) using a local bank statement printer at the present moment, the drawback of this one-factor authentication was considered as insignificant vulnerability. It could be a possible aspect of future work, to think about additional security aspects and their vantages (cf. chapter 5.3). However, the presented approach is as secure as the current solutions, but enables additional security

applications as mentioned in the following and increases the usability for the user, as discussed in chapter 5.2

Any other information or action which can be initiated by the app requires additional authentication. This can be the logging in (using password and user name or similar) on the web page or the entering of a PIN into an ATM. In any case a second factor of knowledge on the one hand and in particular another device has to be used. This increases the security level significant, since attackers have to be in possession of the card as well as the knowledge about the additional login information and use two different devices. This minimizes the possibility of automated attacks complicates assaults.

The next idea presented in this work regarded the encryption of QR codes. Those encrypted codes were amongst other used in the project to improve the privacy of transaction details. Those are typically depicted directly on online banking web pages or account statements at the moment. With the TART approach those information is encrypted and therefore only readable if the user is inhabit of the correct key. This adds another security level to this application area.

One of the main advantages of those encrypted transactions which nevertheless can be decrypted easily with the TART android app is the following: The storage of transaction statements (in digital or analogue) can easily be safeguarded. Even printed transaction statement lists can be read out only by authenticated users, which are in possession of the key. Nonetheless the reading is usable and prompt possible.

Another application possibility of encrypted QR codes in this work is TAN generation respectively the secure transmission of those. If an user wants to transfer money from his account to another one's using online banking, he is typically asked for a TAN to verify the transaction. In the presented approach those asking includes the displaying of an encrypted QR code. Those code is only readable if the user is in possession of his private key. In consequence the TAN itself is secured and only comprehensible to him. Compared to TAN lists this is a magnificent security improvement. Even mobile or SMS TAN can be easily read out by attackers, as chapter 2.7 emphasized. The security level is comparable with TAN generators but reformed their lack of usability and portability.

Since the used QR codes are not only encrypted but also signed by the bank several advantages for the user are generated. Due to the signing the bank institute is able to add a signature to any information given to the user. This complements the certificate given by https and give the user a new level of trust. By this the user can be sure that for instance his entered transaction details were given to the bank and not captured by a man-in-the-middle attack or similar attacks. Additionally the user is able to verify the information on a second device like his smart phone or the Google Glass by scanning the displayed QR code.

The signed QR codes could not only be used in the context of banking but also to sign tickets with QR codes, which can be easily read out by the user, because of the underlying cryptographic technique. Other applications are conceivable.

Last but not least the ATM withdrawal approach: This approach resorts to different aspects of the TART project. First, the app and NFC authentication process and as second signed QR codes. The latter is displayed on the local ATM and enables the user

to check the apparent authenticity of the ATM as well as to verify himself to be standing in front of it. As the QR code displayed on the ATM changes after a certain length of time he can be sure the QR code is currently valid. This fact is new compared to the related work presented in chapter 3.4.

Another major new development given by the QR code displayed on the ATM which is signed by the bank itself is that the user is able to check the authenticity of the ATM. In the current applications this was not possible to him; he had to trust the integrity blindly. Thereby another level of security is given to the user and enables him to detect manipulation. As the QR code displayed on the ATM changes from time to time and it is digital and not printed the manipulation of the code more complicated for attackers in addition. In preliminary research it was not possible to find any comparable approach of signed QR codes to authenticate ATMs, making the TART approach in this application unique for the time being.

The last enhancement allowed by the TART ATM approach is the two-factor authentication process not only supported by two different factors but also by two different devices. In the current applications of cash withdrawal the user is forced to trust the ATM. He is charged to hand over the ATM not only his possession factor (the debit card) but also the PIN. If the ATM is manipulated both needed authentication factors are given to the attackers in one step. TART resolves those security gap by separating the factors onto different devices.

The information stored on the debit card is only given to his smart phone, more precise to the app. The needed information for the ATM is only communicated directly to the server, without use of the input possibilities of the ATM. However, the PIN is the only information presented directly to the ATM. It is quite more complicated for attackers to obtain both information in one single attack. Therefore the number of skimming attacks could be reduced or made less profitable.

5.2 USABILITY

The presented work dealt with different security issues of online banking. Besides the usability of online banking, money withdrawal and other application areas have been part of this approach. In this section some key elements concerning usability of TART are discussed and highlighted.

As mentioned in several parts of this paper the cash withdrawal with TART accelerates the process in some ways. By preparing the withdrawal on the smart phone in beforehand and without the need of physical proximity to the ATM, the main part of withdrawing money can be lead away from the bank and ATM. Queues in front of ATM can be avoided as users only have to scan the QR code displayed on the ATM and take their money in the bank branch.

Another increase of usability is enabled due to the fact, that the user is not forced to take his debit card out of the purse. The authentication based on NFC is possible even if

the wallet is brought only near by the smart phone. Therefore the risk of loss or oblivion of debit cards is minimized.

Similar usability improvements apply for the authentication in the app in general, as the card does not necessary have to be taken out of the purse. The NFC technique is significantly easier to be done than entering long passwords or PINs on small smart phone screens, additionally. It increases the speed of the process as well.

The usability is provided as well in context of TAN generation. Since TAN generators are usually large and unwieldy and in particular uncomfortable to be carried around, the TAN regarding aspects of TART increase usability in several ways.

5.3 SUMMARY AND CONCLUSION

The main goal of the presented work was to use three arising technologies to create a more secure and user-friendly system. The field of banking was chosen as application area and to show the advantages of the technologies. The technologies NFC, encrypted and signed QR codes and Google Glass have been integrated in a first step into four ideas. The pros and cons of those ideas as well as the therewith associated challenges have been discussed and evaluated.

The four ideas were summed up in two major use cases, underlining the benefits of the ideas for user and security. Additionally to the conception of the ideas and hereby converting them into to concept of one huge TART project the implementation was successfully done. The QR aspects as well as the server and website regarding and additionally the functionally rich app system form a fruitful proof of concept. The system is completely implemented and working.

The most compelling elements of the presented TART project include the possibility of ATM verification. Thereby the user is enabled to check the integrity of the ATM. In similar way another level of trust can be added to online banking, as signed and encrypted QR codes containing transactional data allow the user to check the signature of the bank on a second device. The idea of encrypted QR codes is hereby new and innovative.

This projects relied on existing and established security libraries when implementing the encryption and signing of QR codes. Those libraries and techniques are generally accepted guidelines. Due to this the presented system is reliable and easy to extend.

To prove the possible integration of Google Glass into security aspects of banking was another task of this work. Especially regarding over-the-shoulder-attacks this technology facilitates the satisfaction of the users' need for privacy. Furthermore, the integrated communication of Google Glass with the app using *TART Vision* eases the handling of QR codes significant.

The most challenging aspects of this work was to deal with android development. The construction and operation of android apps was quite complex and needed more time than calculated before. The integration of external libraries under android was a tremendous challenge leading to different decisions concerning the selection of the

development tool. Under Android SDK it was almost impossible to integrate the libraries in the intended way, why Eclipse was used development environment instead.

On the basis of the presented work several future ideas can be considered. One main aspect on the next development stage should be the integration of cryptographic NFC instead of the actually used NFC tags. The current solution was suitable for proving the concept, but replaced by the more secure NFC version in future.

Another possible improvement could be the automation of the TAN input. In the current development the generated and decrypted TAN has to be entered manually by the user into the input mask on the web page. This could be automatized based on the app.

To conclude this work was challenging but nevertheless resulting in a working system, proving the developed concepts. It could be shown that the integration of Google Glass, NFC and encrypted as well as signed QR codes is possible to increase the security level of online banking. Based on this foundation derivative work is possible and preferable.

A ADDITIONAL IMAGES

A.1 ONLINE BANKING WEBSITE

A.1.1 MOCKUPS

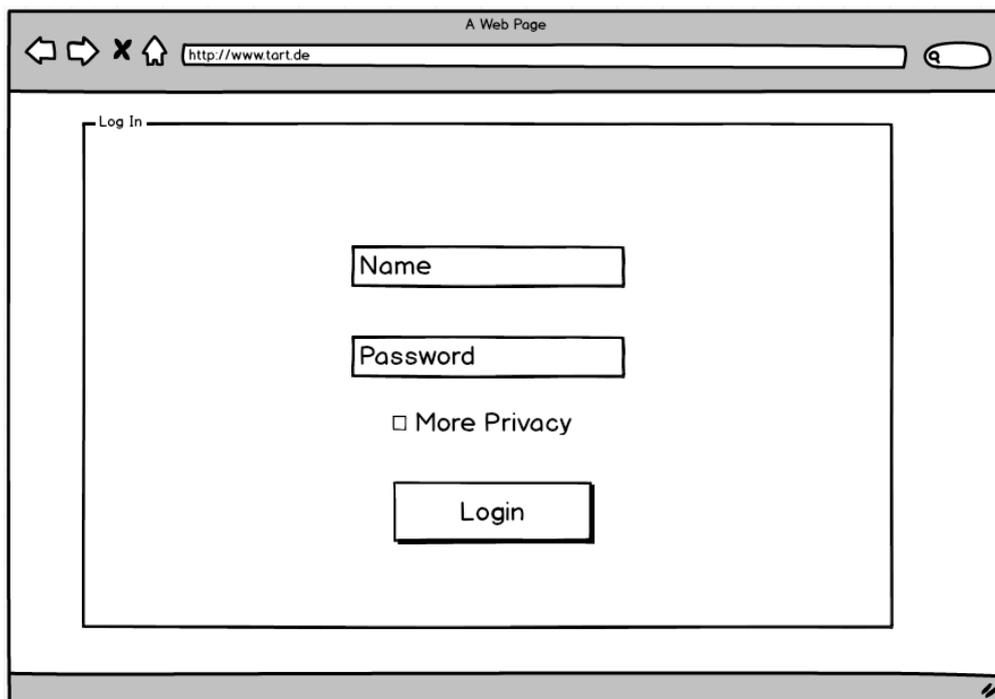


FIGURE A.1: *Mockup of the login page of the online banking website*

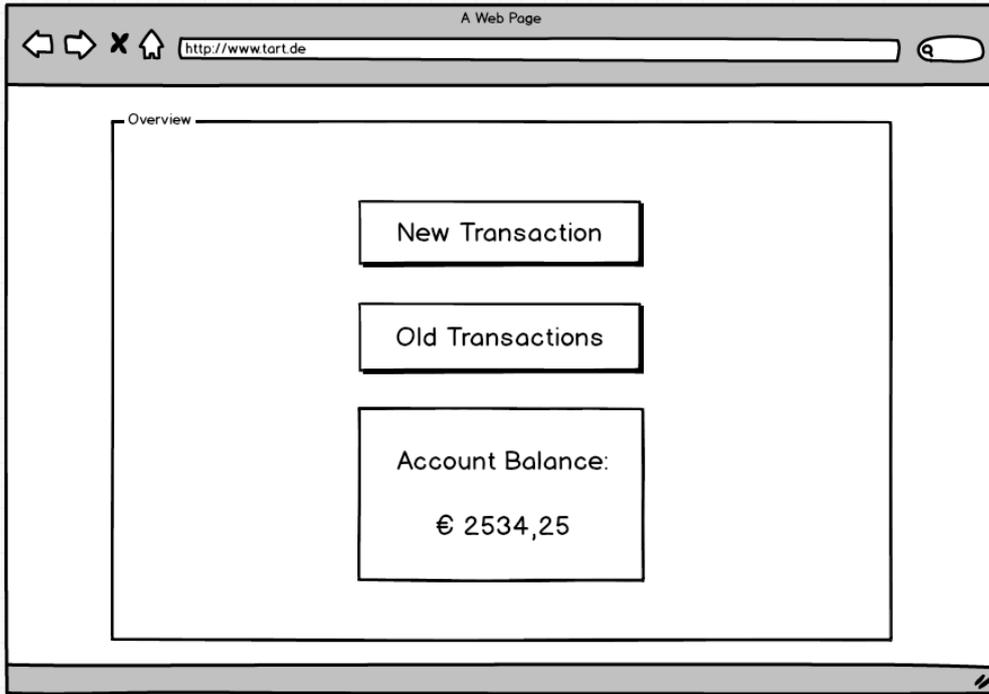


FIGURE A.2: Mockup of the overview page of the online banking website, without QrCrypt

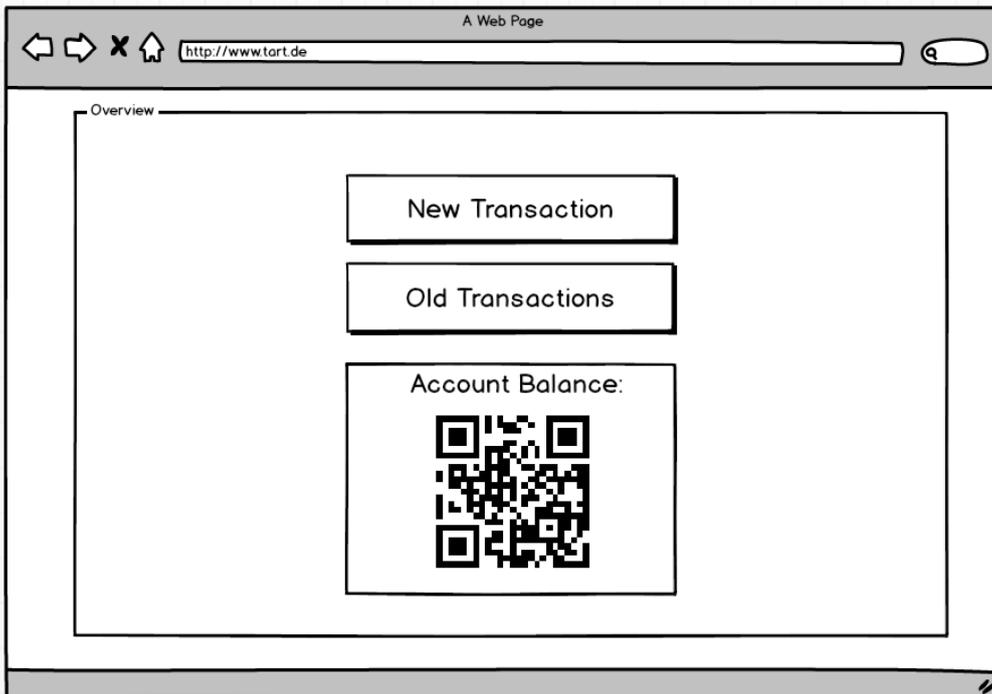


FIGURE A.3: Mockup of the overview page of the online banking website, with QrCrypt

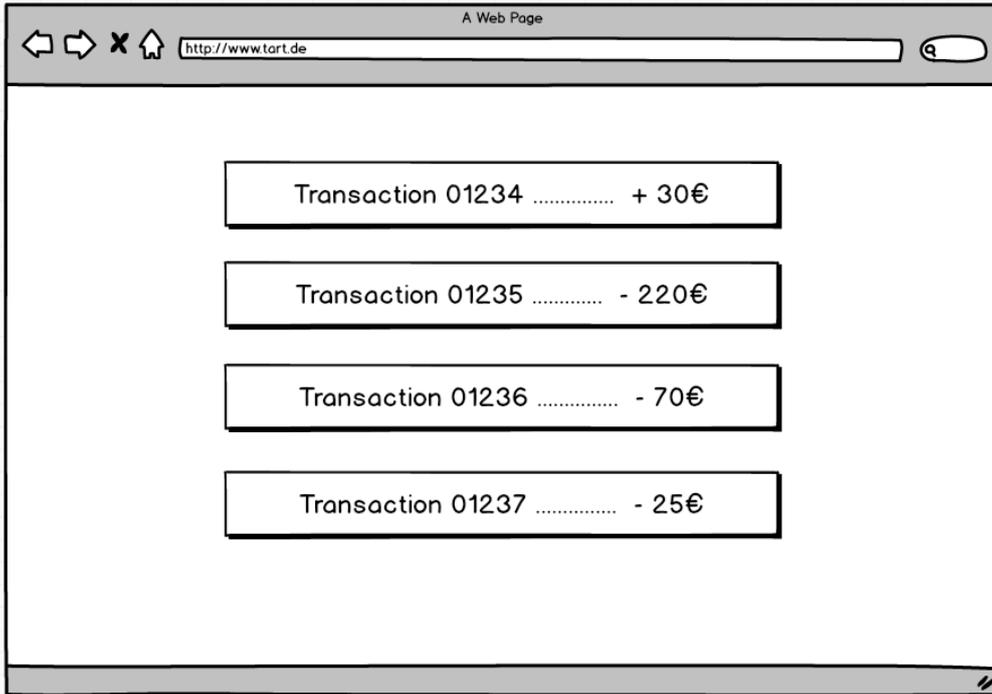


FIGURE A.4: Mockup of the past transactions page of the online banking website, without QrCrypt

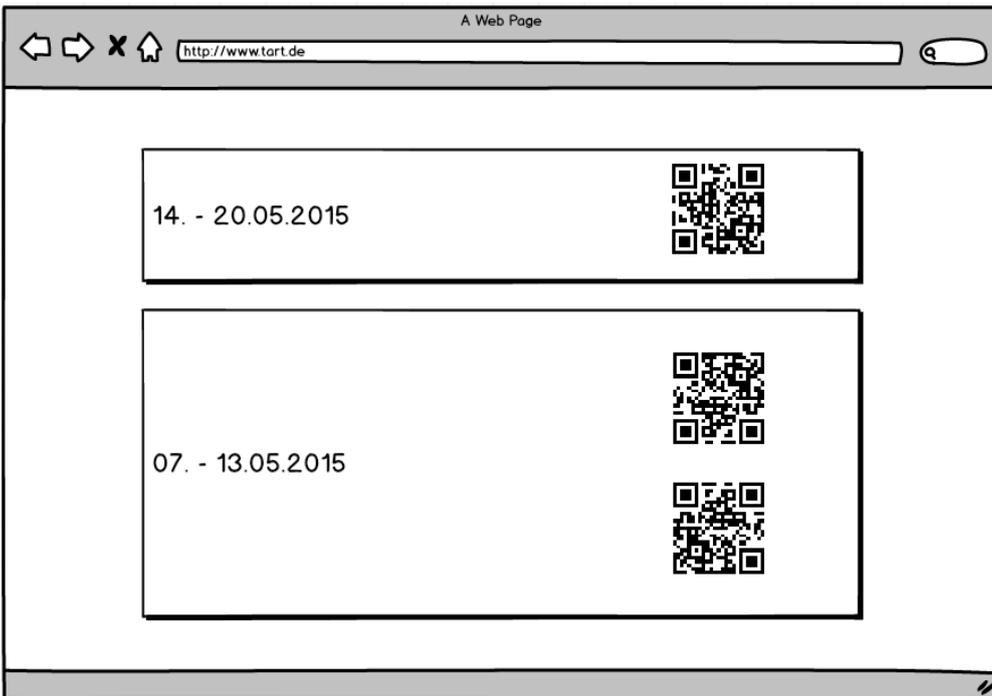


FIGURE A.5: Mockup of the past transactions page of the online banking website, with QrCrypt

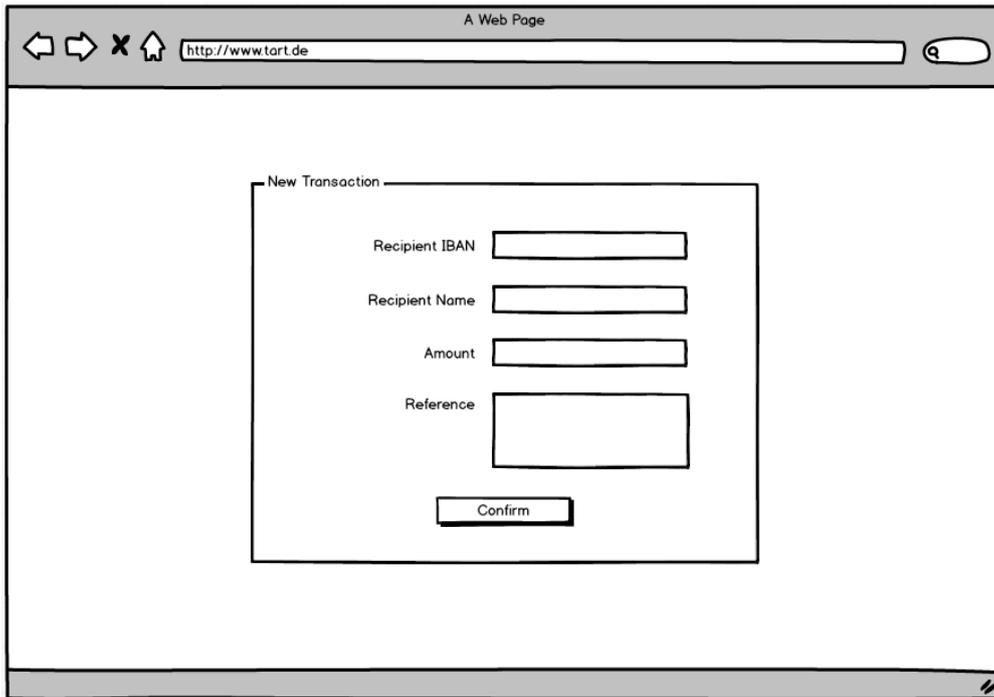


FIGURE A.6: *Mockup of the new transaction page of the online banking website*

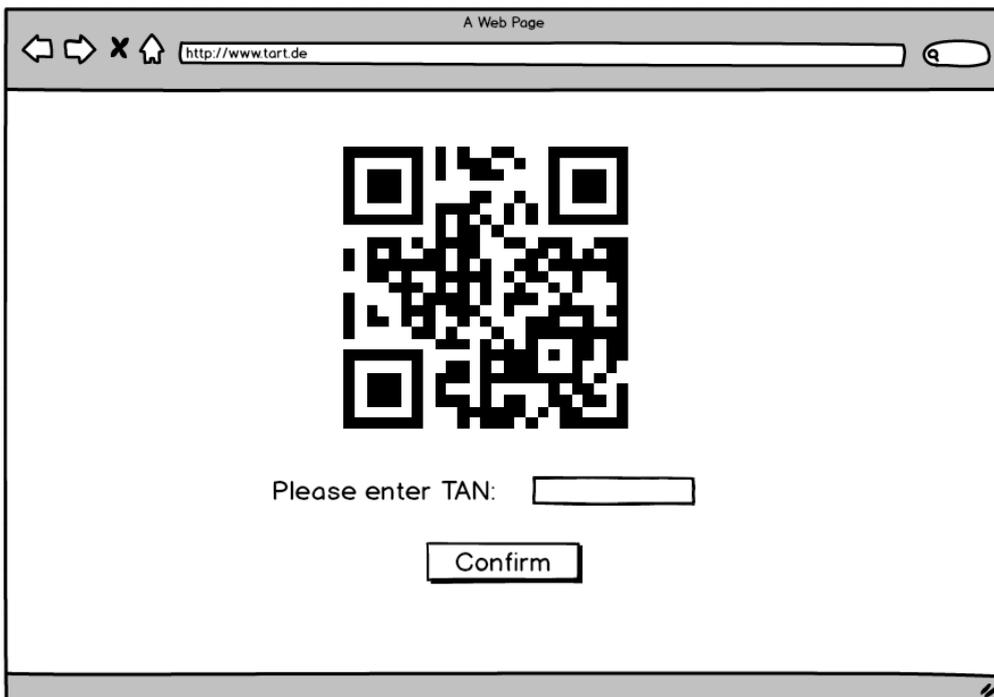
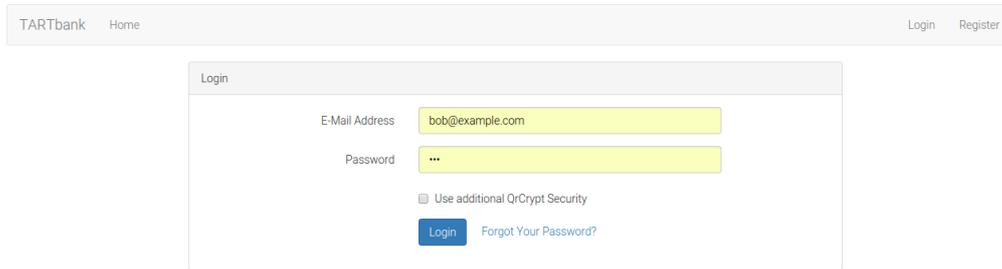


FIGURE A.7: *Mockup of the TAN request page of the online banking website*

A.1.2 SCREENSHOTS



The screenshot shows the login page of the online banking website. At the top, there is a navigation bar with "TARTbank" and "Home" on the left, and "Login" and "Register" on the right. Below the navigation bar is a "Login" form. The form contains two input fields: "E-Mail Address" with the value "bob@example.com" and "Password" with three dots indicating a masked password. Below the password field is a checkbox labeled "Use additional QrCrypt Security" which is unchecked. At the bottom of the form are two buttons: "Login" and "Forgot Your Password?".

FIGURE A.8: *Login page of the online banking website*



The screenshot shows the overview page of the online banking website. At the top, there is a navigation bar with "TARTbank" and "Home" on the left, and a dropdown arrow on the right. Below the navigation bar is a "Welcome" section with the text "This is the main page." and two links: "Transactions" and "New Transaction". Below the welcome section is a "Balance" section with the value "2,774.34€".

FIGURE A.9: *Overview page of the online banking website, without QrCrypt*



FIGURE A.10: Overview page of the online banking website, with QrCrypt

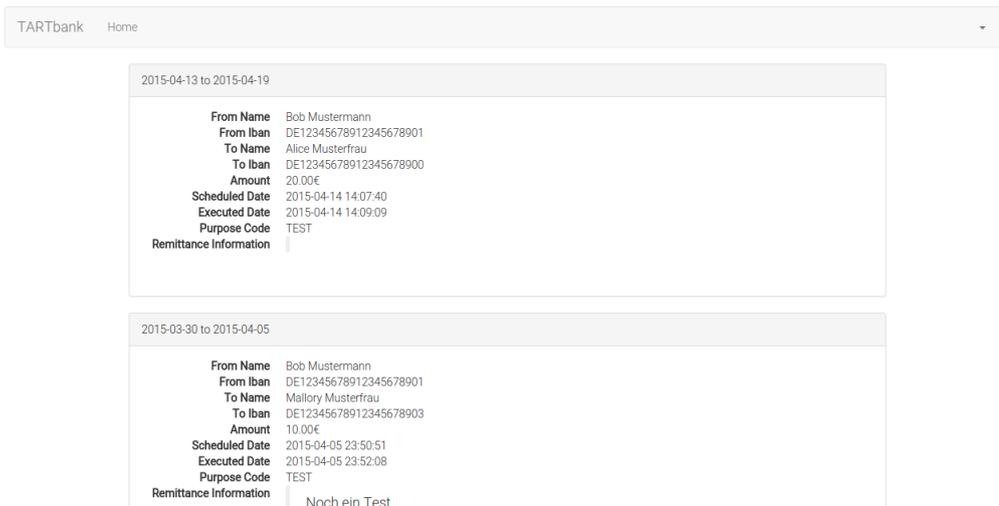


FIGURE A.11: Past transactions page of the online banking website, without QrCrypt

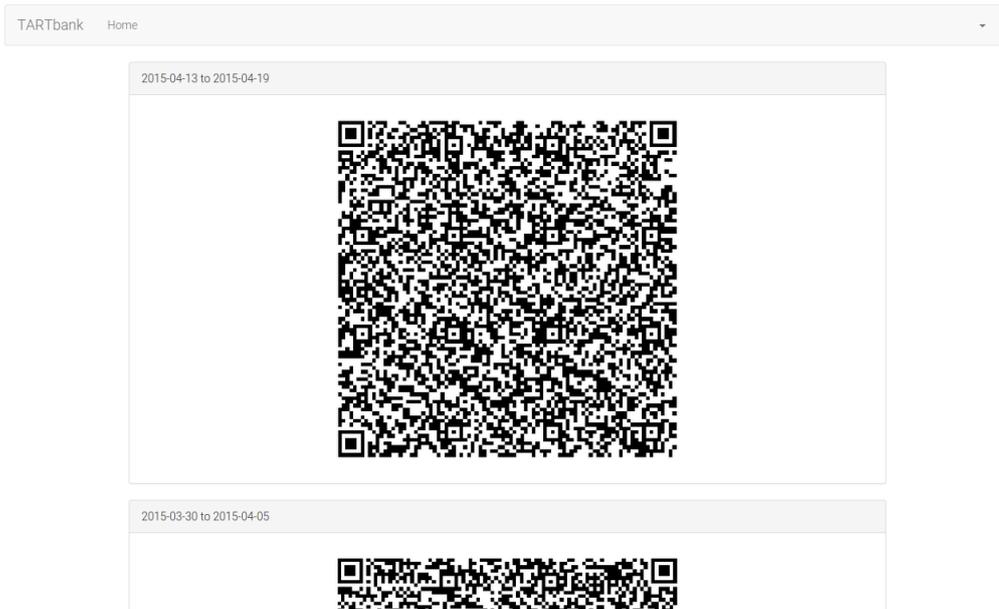


FIGURE A.12: Past transactions page of the online banking website, with QrCrypt

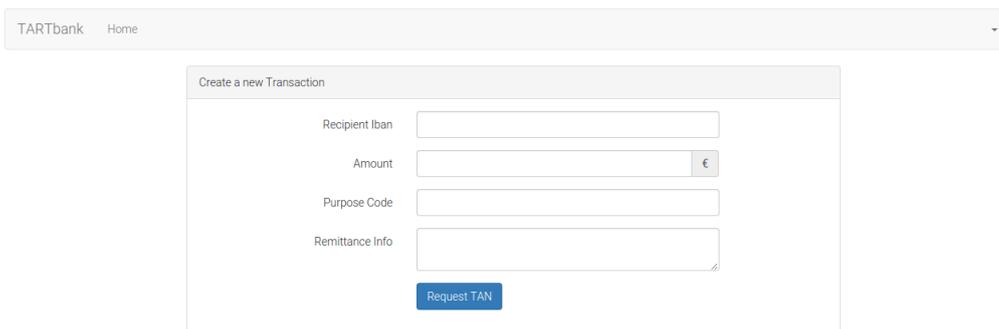


FIGURE A.13: New transaction page of the online banking website

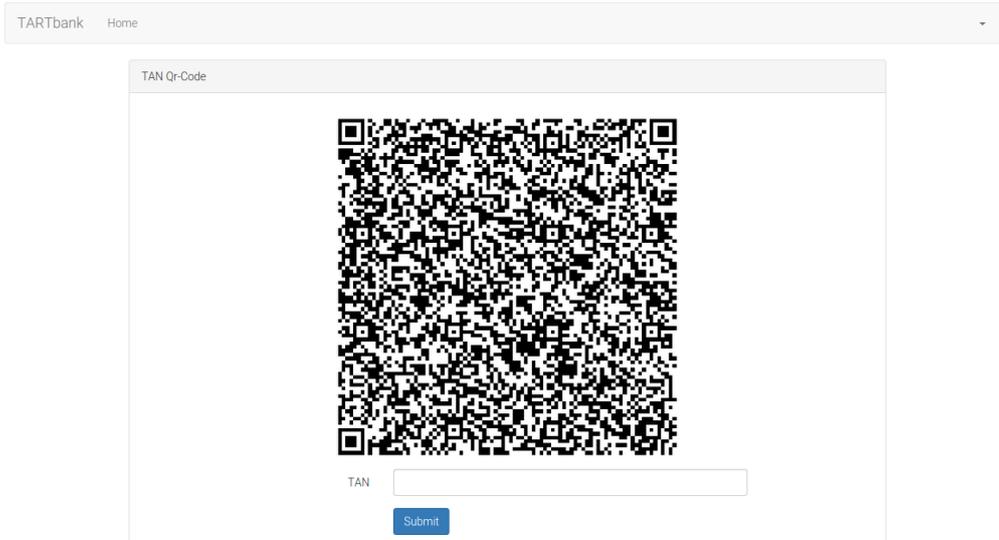


FIGURE A.14: TAN request page of the online banking website

A.2 Mock ATM

A.2.1 Mockups

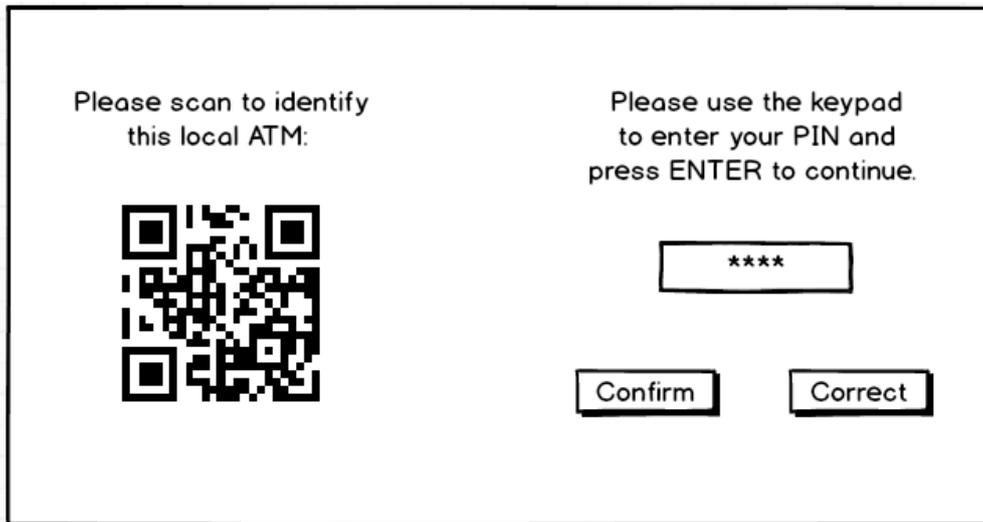


FIGURE A.15: The mockup of the ATM page

A.2.2 SCREENSHOTS

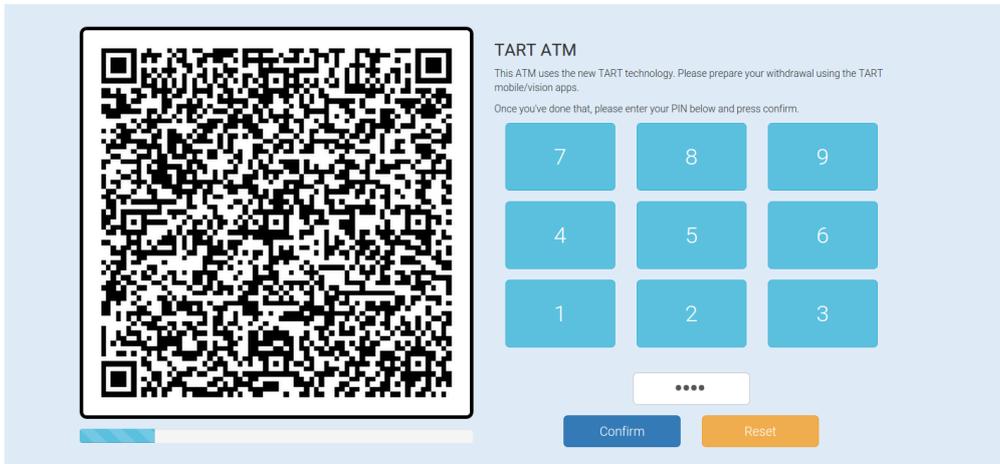


FIGURE A.16: *The final implementation of the ATM page*

B BIBLIOGRAPHY

- [1] *Android*. Feb. 1, 2015. URL: <http://www.android.com>.
- [2] *ANR - Android Developers*. Apr. 2015. URL: <http://developer.android.com/training/articles/perf-anr.html>.
- [3] Yaacov Apelbaum. *User Authentication Principles, Theory and Practice*. Technology Press, 2008.
- [4] *ATM cardless cash access: Why the QR code matters (a lot) to FIs*. 2013. URL: <http://www.atmmarketplace.com/articles/atm-cardless-cash-access-why-the-qr-code-matters-a-lot-to-fis/>.
- [5] Dan Balaban. *Spanish Bank Installs 'First' Contactless ATMs*. Apr. 2011. URL: <http://nfctimes.com/news/spanish-bank-installs-first-contactless-atms>.
- [6] *Basics | Bluetooth Technology Website*. Apr. 2015. URL: <http://www.bluetooth.com/Pages/Basics.aspx>.
- [7] M. Bedner and T. Ackermann. *Schutzziele der IT-Sicherheit*. Datenschutz und Datensicherheit, vol. 5/2010, 2010.
- [8] *blobboc-NFC-Etiketten*. Mar. 2015. URL: <http://www.conrad.de/ce/de/product/646781/>.
- [9] *Bluetooth - Android Developers*. Apr. 2015. URL: <http://developer.android.com/guide/topics/connectivity/bluetooth.html>.
- [10] *Bluetooth History | Bluetooth Technology Website*. Apr. 2015. URL: <http://www.bluetooth.com/Pages/History-of-Bluetooth.aspx>.
- [11] *Bluetooth Technology Special Interest Group*. Apr. 2015. URL: <http://bluetooth.org>.
- [12] *Bluetooth Technology Website*. Apr. 2015. URL: <http://bluetooth.com>.
- [13] *BSI für Buerger: Welche Gefahren begegnen mir im Netz? - Online-Banking*. Jan. 2015. URL: https://www.bsi-fuer-buerger.de/BSIFB/DE/SicherheitImNetz/OnlineBanking/GefahrenUndSicherheitsrisiken/Gefahren_Sicherheitsrisiken.html?notFirst=true&docId=3606116.
- [14] *BSI für Buerger: Welche Gefahren begegnen mir im Netz? - Phishing*. Jan. 2015. URL: https://www.bsi-fuer-buerger.de/BSIFB/DE/GefahrenImNetz/Phishing/phishing_node.html.

- [15] *BSI für Bürger: Sicherheit im Online-Banking*. Jan. 8, 2015. URL: <https://www.bsi-fuer-buerger.de/BSIFB/DE/SicherheitImNetz/OnlineBanking/SofunktioniertDasOnlineBanking/Sicherheit/PIN-TAN-Schutzverfahren.html>.
- [16] Terry Chia. *Confidentiality, Integrity, Availability: The three components of the CIA Triad*. Jan. 2015. URL: <http://security.blogoverflow.com/2012/08/confidentiality-integrity-availability-the-three-components-of-the-cia-triad/>.
- [17] *Composer*. Apr. 2015. URL: <https://getcomposer.org/>.
- [18] *CRC: A Paper On CRCs*. Apr. 2015. URL: <http://www.ross.net/crc/crcpaper.html>.
- [19] *CyanogenMod*. Apr. 2015. URL: <http://www.cyanogenmod.org>.
- [20] Glenn S. Dardick. *Cyber Forensics Assurance*. Longwood University, Nov. 2010. URL: <http://ro.ecu.edu.au/cgi/viewcontent.cgi?article=1076&context=adf#>.
- [21] Rachna Dhamija, J.D. Tygar, and Marti Hearst. *Why Phishing Works*. Apr. 2006. URL: <http://escholarship.org/uc/item/9dd9v9vd#page-1>.
- [22] Whitfield Diffie and Martin E. Hellman. *New Directions in Cryptography*. URL: <http://www.cs.virginia.edu/cs588/diffiehellman.pdf>.
- [23] Claudia Eckert. *IT-Sicherheit*. TU München, 2013. URL: http://www.sec.in.tum.de/assets/lehre/ws1314/itsec/itsec_gezeigt.pdf.
- [24] *EMV FAQ*. Smart Card Alliance: Jan. 2015. URL: <http://www.emv-connection.com/emv-faq/#q18>.
- [25] *Android QR Code*. Apr. 2015. URL: <https://github.com/android/QrCode>.
- [26] *ETSI TS 102 190*. Jan. 7, 2015. URL: http://www.etsi.org/deliver/etsi_ts/102100_102199/102190/01.01.01_60/ts_102190v010101p.pdf.
- [27] Wesley Fenlon. *How does ATM skimming work?* Jan. 2015. URL: <http://money.howstuffworks.com/atm-skimming1.htm>.
- [28] *Gartner's 2014 Hype Cycle for Emerging Technologies Maps the Journey to Digital Business*. Apr. 2015. URL: <http://www.gartner.com/newsroom/id/2819918>.
- [29] *Gartner's Hype Cycle*. Apr. 2015. URL: <http://www.floor.nl/ebiz/gartnershypecycle.htm>.
- [30] *Google Glass*. Jan. 9, 2015. URL: <http://www.google.com/glass>.
- [31] *Google Trends - Websuche-Interesse: nfc - Weltweit, 2004 - heute*. Jan. 7, 2015. URL: <http://www.google.com/trends/explore#q=nfc>.
- [32] B. Lurz und J. Wohlrab H. Herold. *Grundlagen der Informatik*. Addison-Wesley Verlag, 2007. ISBN: 9783827373052.
- [33] Ernst Haselsteiner and Klemens Breitfuß. *Security in Near Field Communication (NFC) - Strengths and Weaknesses*. 2006. URL: <http://rfidsec2013.iaik.tugraz.at/RFIDSec06/Program/papers/002%20-%20Security%20in%20NFC.pdf>.

- [34] *How to defend yourself against MITM or Man-in-the-middle attack*. Oct. 7, 2015. URL: <http://hackerspace.lifehacker.com/how-to-defend-yourself-against-mitm-or-man-in-the-middl-1461796382>.
- [35] *Informationen zum QR-TAN Verfahren*. Jan. 11, 2015. URL: <https://www.1822direkt.de/service/tan-verfahren/qr-tan/>.
- [36] *ISO 20022 PAYMENT GUIDE*. Federation of Finnish Financial Services, Nov. 20, 2012. URL: http://www.fkl.fi/en/themes/sepa/sepa_documents/Dokumentit/ISO20022_Payment_Guide.pdf.
- [37] *ISO/IEC 18004:2006*. 2011. URL: http://http://www.iso.org/iso/catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43655.
- [38] *ISO/IEC 18092:2013*. Jan. 7, 2015. URL: http://www.iso.org/iso/catalogue_detail.htm?csnumber=56692.
- [39] Brian Jepson, Don Coleman, and Tom Igoe. *Beginning NFC*. O'Reilly Media, Inc., 2014. ISBN: 9781449372064. URL: <https://www.safaribooksonline.com/library/view/beginning-nfc/9781449324094/ch04.html>.
- [40] *Laravel - The PHP Framework For Web Artisans*. Apr. 2015. URL: <http://laravel.com>.
- [41] *LinOTP QR-TAN | secure Online Banking | optical TAN procedure - LSE Leading Security Experts GmbH*. Jan. 9, 2015. URL: <https://www.lsexperts.de/linotp-qr-tan-procedure.html>.
- [42] *List of NFC phones*. Jan. 7, 2015. URL: <http://www.nfcworld.com/nfc-phones-list/>.
- [43] Michael Meier. *Intrusion Detection effektiv!* Springer Verlag, 2007.
- [44] *Mikrocontroller.net - Allgemeinzuteilung*. Jan. 7, 2015. URL: <http://www.mikrocontroller.net/articles/Allgemeinzuteilung>.
- [45] Collin Mulliner. *Vulnerability Analysis and Attacks on NFC-enabled Mobile Phones*. 2009. URL: http://www.mulliner.org/collin/academic/publications/vulnalysisattacksnfcmobilephones_mulliner_2009.pdf.
- [46] *Multipurpose Internet Mail Extensions - (MIME) Part Two: Media Types*. IETF - Network Working Group, Nov. 1996. URL: <https://www.ietf.org/rfc/rfc2046.txt>.
- [47] *National Information Assurance (IA) Glossary*. Committee on National Security Systems, 2010. URL: http://www.ncix.gov/publications/policy/docs/CNSSI_4009.pdf.
- [48] *NFC Data Exchange Format (NDEF) - Technical Specification*. NFC Forum, July 24, 2006. URL: <http://www.eet-china.com/ARTICLES/2006AUG/PDF/NFCForum-TS-NDEF.pdf?SOURCES=DOWNLOAD>.
- [49] *NFC Forum - About the Technology*. Jan. 7, 2015. URL: <http://nfc-forum.org/what-is-nfc/about-the-technology/>.
- [50] *NFC Forum - Our Members*. Jan. 7, 2015. URL: <http://nfc-forum.org/about-us/our-members/>.

- [51] *NFC Record Type Definition (RTD) - Technical Specification*. NFC Forum, July 24, 2006. URL: [http://www.cardsys.dk/download/NFC_Docs/NFC%20Record%20Type%20Definition%20\(RTD\)%20Technical%20Specification.pdf](http://www.cardsys.dk/download/NFC_Docs/NFC%20Record%20Type%20Definition%20(RTD)%20Technical%20Specification.pdf).
- [52] *NFC Tools by wakdev*. Mar. 2015. URL: <https://play.google.com/store/apps/details?id=com.wakdev.wdnfc>.
- [53] *Nokia - Understanding NFC Data Exchange Format (NDEF) messages*. Jan. 7, 2015. URL: [http://developer.nokia.com/community/wiki/Understanding_NFC_Data_Exchange_Format_\(NDEF\)_messages](http://developer.nokia.com/community/wiki/Understanding_NFC_Data_Exchange_Format_(NDEF)_messages).
- [54] *NXP.com - NFC Everywhere*. Jan. 7, 2015. URL: <http://www.nxp.com/techzones/nfc-zone/overview.html>.
- [55] *Online-Banking mit pushTAN - So geht smartes Banking - Berliner Sparkasse*. Jan. 9, 2015. URL: <https://www.berliner-sparkasse.de/privatkunden/banking/pushtan/vorteile/index.php?n=/privatkunden/banking/pushtan/vorteile/>.
- [56] Hiroyuki Yomo Petar Popovski and Aalborg University Ramjee Prasad. *Strategies for adaptive frequency hopping in the unlicensed bands*. Apr. 2015. URL: <http://kom.aau.dk/~petarp/papers/DAFH-AFR.pdf>.
- [57] *PHP: Hypertext Preprocessor*. Apr. 2015. URL: <http://php.net>.
- [58] Engadget Primed. *What is NFC, and why do we care?* URL: <http://www.engadget.com/2011/06/10/engadget-primed-what-is-nfc-and-why-do-we-care/>.
- [59] *Profiles Overview | Bluetooth Development Portal*. Apr. 2015. URL: <https://developer.bluetooth.org/TechnologyOverview/Pages/Profiles.aspx>.
- [60] *QRcode.com DENSO WAVE*. Jan. 8, 2015. URL: <http://qrcode.com>.
- [61] *RFC 3366 - Advice to link designers on link Automatic Repeat reQuest (ARQ)*. Aug. 2002. URL: <http://tools.ietf.org/html/rfc3366>.
- [62] Margaret Rouse. *Shoulder Surfing*. Jan. 2015. URL: <http://searchsecurity.techtarget.com/definition/shoulder-surfing>.
- [63] *Rueckgang der Fallzahlen bei Skimming-Delikten in Deutschland*. Bundeskriminalamt, Sept. 2012. URL: http://www.bka.de/nm_241002/SharedDocs/Downloads/DE/Presse/Pressearchive/Presse_2012/pm120918_-_BundeslagebildZahlungskartenkriminalitaet2011_templateId=raw_property=publicationFile.pdf/pm120918_-_BundeslagebildZahlungskartenkriminalitaet2011.pdf.
- [64] *Safe Internet Banking*. Jan. 2015. URL: <https://www.safeinternetbanking.be/en>.
- [65] Peter Schill. In: *IT-Banken & Versicherungen* 3 (2013), pp. 22, 23. URL: <https://lsexperts.de/presseausschnitt/items/interview-mit-lse-zum-thema-online-banking-ist-mobil8db7.pdf>.
- [66] Jörg Schwenk. *Sicherheit und Kryptographie im Internet: Theorie und Praxis*. Springer Vieweg Verlag, 2010. ISBN: 9783658065430.

- [67] SEPA CREDIT TRANSFER SCHEME RULEBOOK. Conseil Européen des Paiements AISBL, Nov. 25, 2014. URL: <http://www.europeanpaymentscouncil.eu/index.cfm/knowledge-bank/epc-documents/sepa-credit-transfer-rulebook-version-8/epc125-05-sct-rb-v80/>.
- [68] Gopalan Sivathanu, Charles P. Wright, and Erez Zadok. *Ensuring Data Integrity in Storage: Techniques and Applications*. Jan. 2015. URL: <http://www.fsl.cs.stonybrook.edu/docs/integrity-storagess05/integrity.html>.
- [69] SMART ATM USES QR CODES INSTEAD OF CARDS TO DISPENSE CASH. June 2012. URL: <http://www.digitaltrends.com/cool-tech/smart-atm-uses-qr-codes-instead-of-cards-to-dispense-cash/>.
- [70] Nigel Smart. *Cryptography: An Introduction (3rd Edition)*. 2014. URL: <http://cs.umd.edu/~waa/414-F11/IntroToCrypto.pdf>.
- [71] *sparkasse.at TAN | Help*. Jan. 8, 2015. URL: <https://netbanking.sparkasse.at/help/Security/TAN>.
- [72] Horst Speichert. *Praxis des IT-Rechts*. GWV Fachverlage GmbH, 2007. URL: http://link.springer.com/chapter/10.1007/978-3-8348-9205-8_12.
- [73] *Spongy Castle*. Apr. 2015. URL: <https://rtyley.github.io/spongycastle/>.
- [74] *Standard ECMA-340*. Jan. 7, 2015. URL: <http://www.ecma-international.org/publications/standards/Ecma-340.htm>.
- [75] *Standards for Security Categorization of Federal Information and Information Systems, FIPS PUB 199*. National Institute of Standards and Technology, Feb. 2004. URL: <http://csrc.nist.gov/publications/fips/fips199/FIPS-PUB-199-final.pdf>.
- [76] Gary Stoneburner, Clark Hayden, and Alexis Feringa. *Engineering Principles for Information Technology Security (A Baseline for Achieving Security), Revision A*. National Institute of Standards and Technology, 2004. URL: <http://csrc.nist.gov/publications/nistpubs/800-27A/SP800-27-RevA.pdf>.
- [77] *Tech History: How Bluetooth got its name*. Apr. 2015. URL: http://www.eetimes.com/document.asp?doc_id=1269737.
- [78] *The Error Correcting Codes (ECC) Page*. Apr. 2015. URL: <http://www.eccpage.com/>.
- [79] *The European Payments Council*. Jan. 6, 2015. URL: <http://www.europeanpaymentscouncil.eu/index.cfm/about-epc/the-european-payments-council/>.
- [80] *The GNU Privacy Guard*. Apr. 2015. URL: <https://www.gnupg.org/>.
- [81] *The Legion of the Bouncy Castle*. Apr. 2015. URL: <https://www.bouncycastle.org/>.
- [82] *The Transport Layer Security (TLS) Protocol Version 1.2*. IETF - Network Working Group, Aug. 2008. URL: <https://tools.ietf.org/html/rfc5246>.
- [83] *The Ubiquitous Reed-Solomon Codes*. Jan. 17, 2015. URL: http://www.eccpage.com/reed_solomon_codes.html.
- [84] *Uniform Resource Identifier (URI): Generic Syntax*. IETF - Network Working Group, Jan. 2005. URL: <http://www.ietf.org/rfc/rfc3986.txt>.

- [85] <uses-feature> | *Android Developers*. Apr. 2015. URL: <http://developer.android.com/guide/topics/manifest/uses-feature-element.html>.
- [86] *Victim's SIM swop fraud nightmare - South Africa*. Jan. 7, 2015. URL: <http://www.iol.co.za/news/south-africa/victim-s-sim-swop-fraud-nightmare-1.385531#.VK0UC3X503I>.
- [87] *Wegweiser zu SEPA. Informationen für Kontoinhaber ohne Bankleitzahl*. Deutsche Bundesbank, July 1, 2012. URL: http://www.bundesbank.de/Redaktion/DE/Downloads/Aufgaben/Unbarer_Zahlungsverkehr/SEPA/wegweiser-zu-sepa.pdf?__blob=publicationFile.
- [88] *What is PHP?* Apr. 2015. URL: <http://php.net/manual/en/intro-what-is.php>.
- [89] Susan Wiedenbeck et al. *Design and Evaluation of a Shoulder-Surfing Resistant Graphical Password Scheme*. May 2006. URL: <http://www.clam.rutgers.edu/~birget/grPssw/venice.pdf>.
- [90] *Wireless technology - Radio silence*. The Economist Newspaper Limited, July 2007. URL: <http://www.economist.com/node/9249278>.
- [91] Victoria Woollaston. *Next-generation cash machines set to replace bank cards with facial recognition*. July 2013. URL: <http://www.dailymail.co.uk/sciencetech/article-2365166/Next-generation-cash-machines-set-replace-bank-cards-facial-recognition.html>.
- [92] *ZBar bar code reader*. Apr. 2015. URL: <http://zbar.sourceforge.net/>.